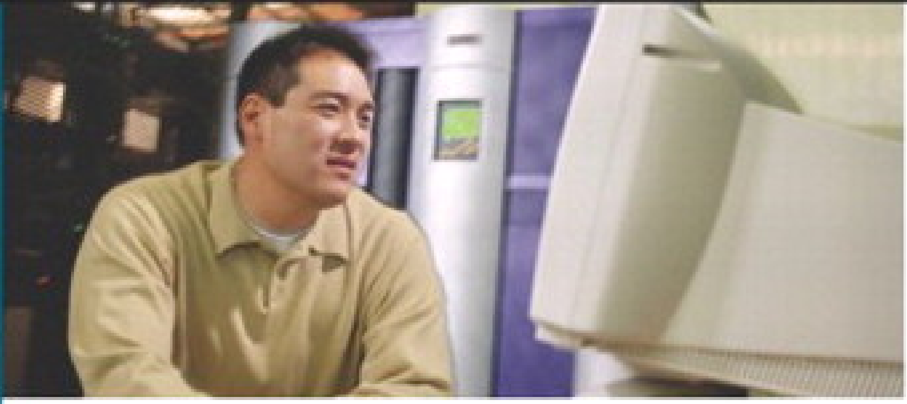
Cisco Systems



IP Telephony Self-Study

Cisco DQOS

Exam Certification Guide

ciscopress.com

IP Telephony Self-Study

Cisco DQOS Exam Certification Guide

Wendell Odom, CCIE No. 1624
Michael J. Cavanaugh, CCIE No. 4516

Cisco Press

Cisco Press
201 West 103rd Street
Indianapolis, IN 46290 USA

IP Telephony Self-Study

Cisco DQOS Exam Certification Guide

Wendell Odom and Michael J. Cavanaugh

Copyright © 2004 Cisco Systems, Inc.

Cisco Press logo is a trademark of Cisco Systems, Inc.

Published by:
Cisco Press
201 West 103rd Street
Indianapolis, IN 46290 USA

Printed in the United States of America 1 2 3 4 5 6 7 8 9 0

Library of Congress Cataloging-in-Publication Number: 2001097407

ISBN: 1-58720-058-9

First Printing July 2003

Warning and Disclaimer

Trademark Acknowledgments

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers’ feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through e-mail at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Publisher	John Wait
Editor-In-Chief	John Kane
Cisco Representative	Anthony Wolfenden
Cisco Press Program Manager	Sonia Torres Chavez
Cisco Marketing Communications Manager	Scott Miller
Cisco Marketing Program Manager	Edie Quiroz
Executive Editor	Brett Bartow
Production Manager	Patrick Kanouse
Development Editor	Ginny Bess Munroe
Copy Editor	Keith Cline
Technical Editors	Frank Knox and Tim Szigeti
Team Coordinator	Tammi Ross
Book Designer	Gina Rexrode
Cover Designer	Louisa Klucznik
Compositor	Octal Publishing, Inc.
Indexer	Larry Sweazy
Proofreader	Missy Pluta

About the Authors

Wendell Odom, CCIE No. 1624, is a senior instructor with Skyline Computer (www.skylinecomputer.com). Wendell has worked in the networking arena for 20 years, working in pre- and post-sales technical consulting, teaching, and course development. He has authored portions of over 12 courses, including topics such as IP routing, MPLS, Cisco WAN switches, SNA protocols, and LAN troubleshooting. He is author of the bestselling Cisco Press title *CCNA Exam Certification Guide*.

Michael J. Cavanaugh, CCIE No. 4516, has been in the networking industry for over 17 years. His employment with such companies as General Electric, Cisco Systems, and Bellsouth Communication Systems has allowed him to stay at the forefront of technology and hold leading edge certifications. His current focus is AVVID implementations, providing convergance consulting, professional services, and technical support. Michael's passion is learning the practical applications of new technologies and sharing knowledge with fellow engineers.

About the Technical Reviewers

Frank Knox, CCIE No. 3698 (Routing & Switching and SNA-IP), is currently the chief technology officer for Skyline Computer Corporation. Frank also participates in the business as a consultant and instructor in the areas of design, implementation, and customer training for all aspects of networking, including IP telephony. Frank has more than 35 years of networking experience with IBM, GTE, and Skyline Computer. During that time, he has worked in field service and support, product planning, education, and management. In addition, he has developed and taught several courses for the University of Dallas (Telecommunications MBA program). Frank has a master's degree in telecommunications from Pace University.

Tim Szigeti, CCIE No. 9794, is a member of the Enterprise Solutions Engineering design team at Cisco Systems. In this role, he works closely with customers and engineering to develop advanced, scalable, and tested solutions for the Cisco AVVID Network Infrastructure (CANI). Prior to this, he was performing technical marketing and product management within the Enterprise Management Business Unit, specializing in QoS Management.

Dedications

Wendell Odom: Mike Zanotto, or Mike Z as he's known throughout California and the world, has had a significant impact on my personal involvement with this book. Mike let me start our company's efforts to teach DQOS classes back in 2001—even when it didn't look like the class would be that popular—just because it might turn into more. Well, it turned out to be a great move for Skyline Computer, and for me. Mike's willingness to take chances like that has had a lot to do with Skyline's success over the years. As my boss, he makes it possible for me to have time to write without totally destroying all of my time for my family! As a friend, he helps keep me laughing, and not take the networking world too seriously. Mike Z, thanks for helping make this book possible!

Michael J. Cavanaugh: I would like to dedicate this book to my lovely wife KC and beautiful daughter Caitlin for their love and support through the years, but especially as I took my first steps in writing. I would like to thank Wendell Odom for giving me the opportunity to co-author this book. It has been an exciting and challenging experience. I would also like to thank all of the people at Cisco Press and the technical editors that made this book a reality.

Acknowledgments

My uncle Eulie used to work in the meat market in the small town I grew up in. I loved hot dogs—but he'd never let me see how they made them!! He always told me that if I liked them, then I really didn't want to know how they were made!

This book has taken a while to complete, with several distractions from all directions. The path we all took to complete this book was a little messy—like making hot dogs, I'm sure—and two individuals in particular made this book possible in spite of the interruptions and diversions.

Michael Cavanaugh, my co-author, worked tirelessly to finish several key components of the book. His vast practical skills have improved the book tremendously. Michael got to do some of the more challenging parts of the book, and under duress—Michael, thanks so much for making the difference!

Ginny Bess, the development editor for this book, got the opportunity to jump into the mix mid-project. While many people contribute to the success of any book, the development editor is the author's main contact for submitting and editing the content. When Gin came on board, we didn't miss a beat, with book development running very smoothly. When job responsibilities required a change in development editors mid-stream, we knew it was a risk, but Ginny came through and did an excellent job.

Chris Cleveland started the book development and gave us guidance as needed. Chris's primary focus is to make the author's life easier, and as always, he succeeded. Thanks for the usual stellar job!

Brett Bartow, executive editor for this project, happens to be an avid baseball fan, as am I. Brett's job requires that he be able to "hit major league curveballs"—for you non-baseball fans, that means he can hit a moving target with the best of them. Once again, Brett, your steady hand throughout the entire writing and editing process helped make this project a success. Thanks for staying on top of the positioning and business issues relating to the book.

Moreso than for most books, this book required some extra effort on the figures in the book. Amy Parker created several new icons for this book, with these icons representing some relatively complex concepts. Amy came through for us, taking my general comments and rough ideas, in some cases refining them, and in others coming up with whole new (and much better) ways to represent the concepts. Thanks much!

Finally, the production side of the business does not get as much notice, because the author (me) who writes these acknowledgements seldom works directly with them. Over the last few years, I've gotten to see more of their work, and believe me, I really do have the easy part of the job. I deliver Word documents and Powerpoint (rough) drawings—and all production does is somehow make this wonderfully polished book appear. Thanks for making me look good again, and again, and again!

As usual, the technical editors deserve most of the credit for making the content of this book robust and complete. For this edition, Tim Szigeti and Frank Knox did the technical editing. Tim's job requires that he help define and evangelize what Cisco views as "best practices" for QoS deployments. His practical experience and ability to communicate well helped us make this book a much more practical reference. Tim, thanks for your work and your patience in helping us resolve the occasional difference between what's on the exams and in the QoS courses, as compared with Cisco's suggested best practices!

Frank Knox, dual CCIE and someone I personally credit with getting me interested in the training world 15 years ago, also worked hard on the technical editing process. Frank's no-nonsense approach to editing, with an occasional nonsensical joke thrown in, both helped the manuscript improve, and kept me a little saner during my re-work of the chapters based on his comments. Thanks to you both for making this book so much better!

Ultimately, Michael and I are most responsible for the contents of the book, so any errors you find are certainly our fault. However, if you do think you found an error, the best way to get in touch to report the error is to go to www.ciscopress.com, click on the "Contact Us" tab, and fill in the form. When it's something that needs a look from the authors, the information gets to us expediently. If it's a problem that can be handled by the publisher, they can get to it even more quickly!

Finally, no section called acknowledgments could be complete without acknowledging a few others. My wife, Kris, regularly takes on all the load for practical stuff at home, instead of the usual majority of the load, when a book project comes down to the wire. This time around, we had three books on similar deadlines. As always, Kris, thanks for helping me when the timing is tight! And ultimately, in this book effort and all else, I must acknowledge Jesus Christ, my friend, intercessor, and savior!

Contents at a Glance

Table of Contents

Foreword

Cisco DQOS Exam Certification Guide is a complete study tool for the DQOS and QOS exams, allowing you to assess your knowledge, identify areas to concentrate your study, and master key concepts to help you succeed on the exams and in your daily job. The book is filled with features that help you master the skills to tune enterprise networks through use of QoS tools that enable high profile, mission-critical traffic to perform at an optimal level. This book was developed in cooperation with the Cisco Internet Learning Solutions Group. Cisco Press books are the only self-study books authorized by Cisco for CCIP and IP Telephony exam preparation.

Cisco and Cisco Press present this material in text-based format to provide another learning vehicle for our customers and the broader user community in general. Although a publication does not duplicate the instructor-led or e-learning environment, we acknowledge that not everyone responds in the same way to the same delivery mechanism. It is our intent that presenting this material via a Cisco Press publication will enhance the transfer of knowledge to a broad audience of networking professionals.

Cisco Press will present study guides on existing and future exams through these Exam Certification Guides to help achieve Cisco Internet Learning Solutions Group's principal objectives: to educate the Cisco community of networking professionals and to enable that community to build and maintain reliable, scalable networks. The Cisco Career Certifications and classes that support these certifications are directed at meeting these objectives through a disciplined approach to progressive learning. In order to succeed on the Cisco Career Certifications exams, as well as in your daily job as a Cisco certified professional, we recommend a blended learning solution that combines instructor-led, e-learning, and self-study training with hands-on experience. Cisco Systems has created an authorized Cisco Learning Partner program to provide you with the most highly qualified instruction and invaluable hands-on experience in lab and simulation environments. To learn more about Cisco Learning Partner programs available in your area, please go to www.cisco.com/go/training

The books Cisco Press creates in partnership with Cisco Systems will meet the same standards for content quality demanded of our courses and certifications. It is our intent that you will find this and subsequent Cisco Press certification and training publications of value as you build your networking knowledge base.

Thomas M. Kelly
Vice-President, Internet Learning Solutions Group
Cisco Systems, Inc.
May 2003

Introduction

Computing in general, and networking in particular, must deal with the issues relating to constrained resources. For computers, operating systems must find a way to equitably distribute the CPU time and memory among the various programs running on the computer. When the need for memory exceeds the available memory, the CPU spends more time performing memory management, moving data from memory to permanent storage, typically on a hard disk. Of course, the computer may be low on CPU resources at the same time, meaning the CPU has less available time to devote to overhead tasks like memory management. With only a small load on the computer, all is well. When the load exceeds the capacity of the CPU, memory, and other resources, a lower volume of useful work is accomplished, and the users get worse response time from the computer.

The competition for bandwidth is the classic battle for resources in networking. If the offered load sent into the network exceeds the available bandwidth, then the network must react by either discarding packets or queuing them in memory waiting for the bandwidth to become available. The packets that are queued experience more delay in the network than do packets that happen to be sent when the network is not congested. When consecutive packets experience different amounts of delay, then variable delay, or jitter, has occurred. So, while bandwidth may be the constrained resource for which many network attached devices compete, other side effects—delay, jitter, and loss—occur as a result.

Cisco calls the general topic of how to manipulate bandwidth, delay, jitter, and loss characteristics in a network *quality of service,* or QoS. The Deploying Quality of Service in Enterprise Networks Exam 9E0-601 (DQOS) tests your knowldege of QoS features and configurations covered in the course “Deploying Cisco QoS for Enterprise Networks” (DQOS). This book covers the topics on the DQOS exam, with some additional detailed explanations beyond what you find in the DQOS course. By going deeper, you can approach the exam with more confidence, while learning valuable information that will help you deploy QoS in real networks. This book also attempts to cover the exact breadth of topics found in the DQOS course, so it will keep you focused on what’s on the exam!

Similar to the DQOS exam, the Cisco Certified Internetwork Professional (CCIP) QoS exam 642-641 covers the topics in a particular course—in this case, the Cisco “Implementing Quality of Service” (QOS) course. Yes, the Cisco course called “DQOS” is indeed different than the course called “QOS.” The two courses have about 70 percent equivalent coverage. For those of you wanting to prepare for the CCIP QOS exam, Appendix B, “Topics on the CCIP QOS Exam,” covers most of the topics found in the course Cisco calls “QOS,” but not in the course Cisco calls the “DQOS” course.

This introduction discusses each of the two QoS related exams: what is covered on each exam and the reasons why you might want to pursue each.

Why Should You Take the DQOS Exam?

Most people that take the DQOS exam do so for one of two reasons. The first is related to the Cisco Channel Partner program, and the other relates to the general need to demonstrate knowledge based on attaining Cisco certifications.

Channel Partner IP Telephony Technology Specializations

The most popular reason for taking the DQOS exam relates to the Cisco Channel Partner program. Cisco calls their resellers and services partners *Channel Partners*. The way the program works is that Cisco moves more than 90 percent of their product sales, in dollar volumes, through their Channel Partners. Thus, Cisco is motivated to help themselves by working well with their Channel Partner community.

Cisco also focuses heavily on customer satisfaction. So Cisco uses both a carrot and a stick to motivate Channel Partners to certify their employees with different technology specializations, which helps ensure that the Channel Partner engineers know what they are doing for Cisco customers. For instance, to become a "Gold" partner, you need a certain number of points. To get the points, you need a certain number of *technology specializations*. To get the specializations, you need a particular mix of employees to certify in different roles—for instance, one role might be as a pre-sales engineer and another as a help desk customer service rep. To certify for a particular role, that employee must pass one or more certification exams, depending on the role.

Can the different Cisco Channel Partner roles, specializations, exams, and so on, become confusing? Sure. Suffice it to say that Channel Partners want to get the points needed to reach the next level of partnership with Cisco (Premier, Silver, and Gold, in order). Even if a Channel Partner does not want to make the next level of "partnership" with Cisco, they can use the fact that they have additional Channel Partner technology specicalizations when trying to win business.

The IP telephony Channel Partner technical specializations require the DQOS exam for several job roles. To achieve any particular technology specialization, a Channel Partner must have an employee pass the right set of certification exams for each job role. For the technology specialization called "IP Telephony—Revised," six job roles must be filled, with three of them requiring the DQOS exam. Table I-1 lists the job roles that require DQOS, as well as the certification exam requirements.

Table I-1 *IP Telephony Revised—Roles and Requirements for the DQOS Exam*

Role	Exams/Certifications Required
Systems Engineer	Cisco Certified Design Associate (CCDA)[a]
	Cisco Product Solutions Essentials IPT Exam v2.0 (SE0-286)
	Enterprise Voice over Data Design Exam (9E0-411 EVODD)
	Deploying Quality of Service in Enterprise Networks Exam (9E0-601 DQOS)
	IP Telephony Solutions Exam (SE0-261)
Field Engineer 1	Cisco Certified Network Professional (CCNP)*
	Telephony Fundamentals Exam (SE0-263)
	Cisco Voice Over Frame Relay, ATM, and IP Exam (9E0-431 CVOICE)
	Cisco IP Telephony Exam (9E0-402 CIPT)
	Deploying Quality of Service in Enterprise Networks Exam (9E0-601 DQOS)
Operations Specialist	Cisco Certified Network Associate (CCNA)*
	Deploying Quality of Service in Enterprise Networks Exam (9E0-601 DQOS)
	Cisco IP Telephony Troubleshooting Exam (9E0-421 IPTT)
Account Manager	Does not require the DQOS exam
Project Coordinator	Does not require the DQOS exam
Field Engineer 2	Does not require the DQOS exam

a. More advanced certifications can be subsituted. For instance, the person can be CCNP instead of CCDA or CCIE instead of CCNP.

In short, if you work for a Channel Partner, and you design, sell, or implement IP telephony solutions, then you will most likely be asked to certify in one of the job roles listed in Table I-1. In addition, because three of the four technical roles for the IP telephony technology specialization require the DQOS exam, the chances are you will need to pass this exam.

Cisco Qualified Specialist (CQS) Program

For any networker in any networking job, it helps to have knowledge and skills. Networkers can benefit from having "proof" that they know a set of technologies. Having the right certification on your resume can help you land a job, both at another firm and inside the same company. For those networkers who work with customers and clients, having the right credentials, in the form of certifications, can help convince the salesperson to convince the customer to hire your company for the consulting job. Having the right certifications can also make you more appealing to be hired by a Channel Partner, because you can add a simple exam or two, meet the requirements of one of the Channel Partner certifications and immediately help that Channel Partner meet their skills requirements.

Cisco offers a wide range of certifications, including a series of certifications in the Cisco Qualified Specialist (CQS) program. CQS certifications focus on one particular technology area, requiring multiple exams from that technology area. The goal of the CQS certifications is to let people prove their knowledge and skill about a particular technology, as compared to the Cisco Career Certifications, which cover a broad range of topics.

Three different CQS certifications require the DQOS exam. Unsurprisingly, these three CQS certifications all focus on IP telephony. Table I-2 lists the certifications along with the required exams.

Table I-2 *IP Telephony Revised—Roles and Requirements for the DQOS Exam*

Role	Exams/Certifications Required
Cisco IP Telephony Design Specialist	CCDA[a] Enterprise Voice over Data Design Exam (9E0-411 EVODD) **Deploying Quality of Service in Enterprise Networks Exam (9E0-601 DQOS)**
Cisco IP Telephony Support Specialist	CCNP* Cisco Voice Over Frame Relay, ATM, and IP Exam (9E0-431 CVOICE) Cisco IP Telephony Exam (9E0-402 CIPT) **Deploying Quality of Service in Enterprise Networks Exam (9E0-601 DQOS)**
Cisco IP Telephony Operations Specialist	CCNA* **Deploying Quality of Service in Enterprise Networks Exam (9E0-601 DQOS)** Cisco IP Telephony Troubleshooting Exam (9E0-421 IPTT)

a. More advanced certifications can be subsituted. For instance, the person can be CCNP instead of CCDA or CCIE instead of CCNP.

Every IP telephony-related CQS certification requires the DQOS exam. With the requirement for the DQOS exam for three of the four technical roles in the Cisco Channel Partner IP telephony technology specialization, pretty much anyone working with IP telephony or voice over IP will end up needing to take the DQOS exam, assuming that they want to be certified.

You may have noticed that the CQS exam requirements are very similar to the Channel Partner roles. In fact, the CQS requirements for the three certifications in Table I-2 are a subset of the requirements for a comparable Channel Partner certification. Table I-3 summarizes the differences.

Table I-3 *CQS Compared to Channel Partner IP Telephony Roles*

Channel Partner Role	CQS	Exams/Certifications Required by Channel Partner Role, but Not by CQS
Systems Engineer	Cisco IP Telephony Design Specialist	Cisco Product Solutions Essentials IPT Exam v2.0 (SE0-286) IP Telephony Solutions Exam (SE0-261)
Field Engineer 1	Cisco IP Telephony Support Specialist	Telephony Fundamentals Exam (SE0-263)
Operations Specialist	Cisco IP Telephony Operations Specialist	None

As you can see from Table I-3, there are few differences. In fact, if you fulfill the requirements for a Channel Partner IP telephony role, you will have accidentally become a CQS! Also, if you are adding certifications in hopes of getting a new job, you could take the extra step beyond CQS, meet all the requirements of one or more IP telephony roles, and make sure that any Channel Partners that you interview with know that you can already help them meet their Partner certification requirements!

For more information on the Cisco Channel Partner technology specializations, and the CQS program, refer to www.cisco.com.

The Cisco DQOS Exam 9E0-601

The DQOS exam consists of a one-hour exam administered at a proctored exam facility affiliated either with VUE (www.vue.com) or Prometric (www.2test.com). The exam typically includes approximately 70 questions.

Cisco lists the topics covered in the DQOS exam on their web site; the list is repeated in this introduction. Like many Cisco exams, the DQOS exam covers the topics in the Cisco DQOS course. In fact, the list of exam topics for the exam are actually taken from the list of objectives that begin each chapter in the DQOS course.

The exam topics are as follows.

QoS Overview

- List five benefits for implementing QoS in the enterprise networks.
- Describe how a converged network behaves without QoS.
- Correctly describe the QoS framework.
- Describe correctly what call admission control does.
- Describe the difference between Integrated Services and Differentiated Services.

Classification and Marking

- Explain the reason for classification and marking.
- Explain the difference between classification and marking.
- Explain class of service, IP Precedence and DiffServ code points.
- Configure QoS policy using modular QoS CLI.
- Explain the role of Network Based Application Recognition (NBAR).
- Classify and mark traffic.

Congestion Management

- Identify and differentiate between the different IOS queuing techniques.
- Correctly apply each queuing technique to the appropriate application.
- Describe the difference between IP RTP priority and low latency queuing (LLQ).
- Configure WFQ, CBWFQ, and LLQ.

Congestion Avoidance

- Explain how TCP responds to congestion.
- Explain tail drop and global synchronization.
- Identify and differentiate between the following IOS congestion avoidance tools: RED, WRED, FRED.
- Configure IOS congestion avoidance features.

Link-Efficiency Tools

- Explain the need for link-efficiency tools.

- Explain available LFI techniques including MLP interleaving and FR fragmentation using FRF.11 Annex-C or FRF.12.
- Explain Real Time Protocol header compression (CRTP) as a tool for improving link efficiency.
- Configure and monitor various LFI methods and CRTP.

Policing and Shaping

- Describe the difference between policing and shaping and how each one relates to QoS.
- Describe various mechanisms for policing, when to apply each, and how to configure them.
- Identify the various types of traffic shaping, their differences, and how to apply each.
- Configure the different types of traffic shaping.

Call Admission Control

- Correctly list five local CAC methods and their primary function.
- Correctly list two measurement-based CAC methods and their primary function.
- Correctly describe IntServ/RSVP and its main function.
- Given an enterprise network scenario, correctly determine which method(s) of achieving call admission control best meets the customer requirements.

QoS Management Tools

- Utilize QoS Device Manager to monitor performance, establish baselines and configure QoS policies.
- Utilize QoS Policy Manager to configure advanced QoS policies, scale policy deployment, upload/verify/roll-back policies, and deploy QoS policies by external time-based/event-based scripts.
- Configure Cisco Service Assurance Agent to measure key SLA metrics and monitor network performance between local and remote devices.
- Monitor and troubleshoot network performance with IPM and SMS.

QoS Design

- Design a converged multiservice network to provide proper QoS for voice, video and data traffic

NOTE The list of objectives was taken from Cisco.com at www.cisco.com/warp/public/10/wwtraining/certprog/testing/current_exams/9E0-601.html.

Interpreting the DQOS Exam Topics

The exam topics, like most exam topics listed by Cisco for other exams, use action words that follow a quasi-standard called "Bloom's Taxonomy of the Cognitive Domain." Bloom's taxonomy defines a standard for word usage for when educators create objectives for courses. Objectives written according to Bloom's Taxonomy define what the learner should be able to accomplish after taking the class.

So, when you look at an exam topic, look for the action word. If you want to see a description of Bloom's taxonomy, search the Internet, and you will find a lot of matches. My favorite quick list of terms is at http://chiron.valdosta.edu/whuitt/col/cogsys/bloom.html. The action word in the exam topic gives you a good hint about the level of knowledge and skill you will need to have before taking the exam. For instance, a course objective that uses the word "list" as the action word means that you should be able to list the features, but an action word such as "configure" means you should know all the related configuration commands, and how to use them. "Troubleshoot" might mean that you need to know what all the show and debug commands do for a particular topic.

For a specific example, under the section about Congestion Management, the last exam topic says "*Configure* WFQ, CBWFQ, and LLQ." So, you had better know the configurations of each of these and not just the concepts. Conversely, none of the four exam topics under the heading of CAC says anything about configuration, but rather the exam topics ask you to list, describe, and compare/contrast.

So, what does Bloom's taxonomy mean in terms of how you study for the exam? It means that you should focus on the action words in the exam topics and make sure you can do those things for the stated topics. In a perfect world, the exam questions would also follow the same convention. However, some questions that don't meet these guidelines will slip through into Cisco's set of exam questions. For instance, could you see a CAC configuration question on the exam, even though no stated exam topic uses the word "configure"? Yes, you could see this type of question. However, when you are trying to determine your strategy for studying, and you are choosing the topics to focus on, you should definitely interpret the meaning of the exam topics.

In addition, Cisco states that the posted exam topics for all of their certification exams are guidelines. Cisco makes the effort to keep the questions on the exam within the scope of the stated exam objectives, but doing this for every question and every exam is difficult. So, you could see questions that both fall outside the scope, and the depth, implied by the exam topics. However, if you follow the Cisco exam topic "guidelines," you should have a good understanding of the breadth and depth of topics on the exam.

Each time you start a new chapter, you should take the time to review the exam topics related to that chapter. Each chapter begins with a list of related exam topics. Table I-4 lists the chapters in this book and the major headings from the list of exam topics corresponding to each chapter.

Table I-4 *This Book's Coverage of the Major Headings from the DQOS Exam*

Book Chapter	DQOS Exam Topic Heading
1	QoS Overview
2	QoS Overview
3	Classification and Marking
4	Congestion Management
5	Policing and Shaping
6	Congestion Avoidance
7	Link-Efficiency Tools
8	Call Admission Control
9	QoS Management Tools, QoS Design
10	None
Appendix B	None

Changes to the DQOS Exam Topics

One day, the DQOS exam will change. Most likely, Cisco will announce changes, or Cisco will change the exam topics and maybe even announce a new DQOS 9E0-601 exam with a new exam number. There is even precedent for Cisco changing the entire question database for an exam without telling the public in order to defeat those who would copy questions from the exam and distribute them.

Regardless, the exam will change one day, and you will want to know about the changes if you have not passed the exam yet. First, you should look to www.cisco.com occasionally during the weeks in which you are studying for the DQOS exam, just to make sure nothing new has been announced about the exam. Also, you should look to www.ciscopress.com/1587200589, which is where we will post any new hints about changes to the exam. As you might imagine, www.cisco.com will show any announcements a little before Cisco Press will.

Why Should You Take the QOS Exam?

Most people (myself included) find it odd that Cisco offers two partly overlapping courses on QoS. While the technology covered in each course overlaps, the overall goals of the courses do differ significantly. The DQOS course focuses on the QoS tools specifically needed for converged networks with voice, video, and data, whereas the QOS course provides coverage of almost every router-based IOS QoS feature. So, because the two courses acheive different goals, Cisco has retained both courses.

Because the DQOS course is more focused on the application of QoS for voice, Cisco chose the DQOS exam for the CQS and Channel Partner Telephony certifications. Similarly, when Cisco created the Cisco Certified Internetwork Professional (CCIP) certification, they chose to use the QOS 642-641 exam. Cisco designed CCIP to be like CCNP, but for Service Provider employees—in other words, a wide-ranging general certification for core skills but skills that mattered more to Service Providers. So, the QOS course was a better fit because it covered a wider range of QoS topics, with focus on the tools themselves instead of their application for converged networks.

Currently, the only certification that requires the QOS 642-641 exam is the CCIP certification. Why would you want to be a CCIP? Well, the most obvious reason is to build your resume. Also, Cisco occasionally lets you substitute CCIP instead of CCNP as the prerequisite for some certifications. But there are currently no Channel Partner certifications or CQS certifications that require the CCIP certification or the QOS exam, which simply means that fewer people will be motivated to take the QOS exam as compared with the DQOS exam.

The Cisco CCIP Certification

The CCIP Certification requires that you pass four exams, with the QOS exam being one of those. The four exams are listed in Table I-5.

Table I-5 *CCIP Certification Exams*

Exam Name	Exam Abbreviation and Number
Building Scalable Cisco Internetworks	Building Scalable Cisco Internetworks (BSCI) 642-801
Configuring BGP on Cisco Routers	Border Gateway Protocol (BGP) 642-661
Implementing Cisco QoS	QOS 642-641
Implementing Cisco MPLS	Multiprotocol Label Switching (MPLS) 640-910

The Cisco QOS Exam 642-601

Like the DQOS exam and many other Cisco exams, Cisco lists the exam topics on their web site. And like these other exams, the exam topics have been taken directly from the list of objectives from the corresponding course. You should review the discussion about the DQOS exam objectives in the “Interpreting the DQOS Exam Topics” section of this introduction for some suggestions on how to interpret the exam topics.

The exam topics for the QOS exam are listed here for convenience. You should check Cisco.com web site frequently when you are actively studying for the QOS exam, to see if Cisco has announced any changes. You can also check www.ciscopress.com/qos.

The QOS exam topics, as of the publication of this book, are as follows.

Introduction to IP Quality of Service

- Describe the need for IP QoS.
- Describe the Integrated Services model.
- List the key benefits and drawbacks of the IntServ model.
- Describe the Differentiated Services model.
- List the key benefits of the DiffServ model compared to the IntServ model.
- Describe the interoperability between DSCP-based and IP Precedence-based devices in a network.
- Describe the building blocks of IP QoS mechanisms (classification, marking, metering, policing, shaping, dropping, forwarding, queuing).
- List the IP QoS mechanisms available in the Cisco IOS.
- Describe what QoS features are supported by different IP QoS mechanisms.

Classification and Marking Mechanisms

- Describe policy-based routing and how it can be used to classify and mark IP packets.
- Configure the policy-based routing mechanism on Cisco routers.
- Describe QoS Policy Propagation through BGP and how it can be used to classify and mark IP packets.
- List other mechanisms that also support classification and marking capabilities (committed access rate, class-based marking).

Queuing Mechanisms

- Describe how queuing works on Cisco routers.
- Describe FIFO Queuing (FQ) and its benefits and drawbacks.
- Describe Priority Queuing (PQ) including its benefits and drawbacks.
- Configure Priority Queuing (PQ) on Cisco routers.
- Describe Custom Queuing (CQ) including its benefits and drawbacks.
- Configure Custom Queuing (CQ) on Cisco routers.
- Describe Weighted Fair Queuing (WFQ) including its benefits and drawbacks.
- Configure Weighted Fair Queuing (WFQ) on Cisco routers.

- Describe different distributed Weighted Fair Queuing (dWFQ) mechanisms available on Cisco IOS routers.
- Describe Modified Deficit Round Robin (MDRR) queuing.
- Describe IP RTP Prioritization.
- Configure IP RTP Prioritization on Cisco routers.

Traffic Shaping and Policing Mechanisms

- Describe shaping and policing mechanisms available in Cisco IOS.
- Describe the benefits and drawbacks of traffic shaping and policing mechanisms.
- Describe Generic Traffic Shaping (GTS) on Cisco routers.
- Configure GTS on Cisco routers.
- Describe Frame Relay Traffic Shaping (FRTS) and its differences from GTS.
- Configure Frame Relay Traffic Shaping (FRTS) on Cisco routers.
- Describe the Committed Access Rate (CAR) mechanism including its benefits and drawbacks.
- Configure Committed Access Rate (CAR) on Cisco routers.

Congestion Avoidance Mechanisms

- Describe Random Early Detection (RED).
- Describe the Weighted Random Early Detection (WRED) mechanism.
- Configure Weighted Random Early Detection (WRED) on Cisco routers.
- Describe the flow-based WRED mechanism.

Link-Efficiency Mechanisms

- Describe payload compression and payload compression algorithms available on Cisco routers.
- Describe header compression and header compression algorithms available on Cisco routers.
- Configure Cisco IOS header compression mechanisms.
- Describe Link Fragmentation and Interleaving (LFI).
- Configure Link Fragmentation and Interleaving (LFI) mechanisms on Cisco IOS routers.

Signaling Mechanisms

- Describe Resource Reservation Protocol (RSVP).
- Configure RSVP on Cisco IOS routers.

Modular QoS Command Line Interface

- Describe the Modular QoS CLI (MQC) concept and its structure.
- Describe Modular QoS CLI classification options.
- Configure the Modular QoS CLI to perform classification.
- Describe Network-based Application Recognition (NBAR).
- Describe Modular QoS CLI policy options.
- Configure the Modular QoS CLI to perform service policies.

QoS and IP over ATM

- List the requirements of IP QoS in combination with ATM QoS.
- Describe per-VC queuing.
- Describe and configure per-VC WRED.
- Describe and configure per- VC CB-WRQ.

About the DQOS Exam 9E0-601 Certification Guide

This section provides a brief insight into the contents of the book, the major goals, as well as some of the book features that you will encounter when using this book.

Goals of This Book

Unquestionably, the primary goal for this book is to help you pass the DQOS certification exam. However, the means by which that goal is accomplished follows the Cisco Press Exam Certification Guide philosophy, which makes a statement about helping a reader pass the test through a deeper understanding of the material, as opposed to simply helping the reader memorize the answers to multiple-choice questions.

To accomplish this goal, the book’s main chapters cover all the topics on the DQOS exam, plus an occasional mention of topics outside the scope of the exam just to make a key point. The depth of the conceptual coverage exceeds the depth of coverage in the DQOS course. By doing so, you should be able to pass the exam with greater confidence.

The secondary goal for this book is to help you pass the Cisco QOS exam (642-641). In order to keep the larger DQOS audience focused, the topics covered on the QOS exam, but not on the DQOS exam, are covered in Appendix B, "Topics on the CCIP QOS Exam." As mentioned eariler, DQOS covers about 70 percent of the materials on the QOS exam. Appendix B covers most of the rest of the topics, with a few pointers to good documents on Cisco.com to round out the coverage. For instance, priority queuing gets only passing mention in the DQOS course, with no exam topics mentioning it. Priority queuing gets 20 pages in the QOS course, with a specific mention in the list of exam topics for that exam. So, the priority queuing coverage has been placed into Appendix B.

The third goal is not so obvious. While written to help you pass the exams, we hope that this book will also be useful to anyone who needs to deploy QoS tools using Cisco gear. We hope that if you take the exam, you will keep this book as a desk reference, and for those of you who don't take the exam, we hope you find this book a useful tool for delving into the details and really understanding QoS.

After teaching the DQOS course for the last couple of years, and after hearing students continually ask where they could read more on QoS topics, it became apparent that there were few good options available. This book fills that gap and provides a comprehensive reference for Cisco QoS.

Book Organization

This book contains 10 core chapters with titles that are comparable to the major headings listed in the DQOS exam topics. For DQOS exam candidates, you can simply dive into Chapter 1 and read through Chapter 10.

For those of you taking the QOS exam, the end of this introduction has some tips on what to focus on in Chapters 1 through 10. You will also want to read Appendix B, which covers the topics on the QOS exam but not covered on the DQOS exam.

Chapters 1 and 2 cover most of the core background information needed to understand the different classes of Cisco QoS tools. Chapters 3 through 8 each cover a different major type of QoS tool, covering the concepts, as well as the configuration of the tools. Chapter 9 covers management and design issues that relate to the entire spectrum of QoS tools. Finally, Chapter 10 covers some campus QoS concepts and configuration, which are actually beyond the scope of the current DQOS exam (as of the writing of this book). However, it's a likely topic to be added during the next course and exam revision, and the one topic most DQOS students want to hear more about, so we added the chapter. As always, make sure you check Cisco.com, as well as www.ciscopress.com/1587200589, for the latest news about any future changes to the exam.

Following is a description of each chapter's coverage:

- **Chapter 1, "QoS Overview"**

  QoS affects the characteristics of network traffic. To understand the QoS concepts and configurations discussed in other chapters, you must know what can be manipulated—namely, bandwidth, delay, jitter, and packet loss. Also, different types of traffic have different needs for bandwidth, delay, jitter and loss. Chapter 1 defines QoS terms; explains the concepts relating to bandwidth, delay, jitter, and packet loss; and identifies the traffic characteristcs of data, voice, and video traffic.

- **Chapter 2, "QoS Tools and Architectures"**

  Cisco provides a large number of QoS tools inside the router IOS. Two of the biggest challenges when preparing for either exam is remembering all the of tools and keeping track of which tools provide what features. Chapter 2 begins by listing and describing the classes of tools, and then also listing the tools themselves. The remaining chapters delve into more depth on each particular class of tool.

  QoS tools typically either follow one of two QoS architectural philosophies. The two architectures are called Differentiated Services and Integrated Services. The second part of this chapter explains the two architectures.

- **Chapter 3, "Classification and Marking"**

  Classification and marking defines how a networking device can identify a particular packet and change some bits in the frame or packet header. The changed bits "mark" the packet, so other QoS tools can react to the marked field. This chapter covers the concepts, as well as five different classification and marking tools.

- **Chapter 4, "Congestion Management"**

  Queuing tools on routers manage packets while they are waiting to exit an interface. This chapter discusses the general concepts of queuing in Cisco routers and then covers the concepts and configuration behind a large variety of queuing tools. The Cisco DQOS exam topics refer to queuing as "congestion management."

- **Chapter 5, "Traffic Policing and Shaping"**

  Policing tools discard traffic that exceeds a particular rate. Shaping tools delay traffic so that, over time, the traffic rate does not exceed a particular rate. Both classes of tools use a concept of measuring the rate of sending or receiving bits. This chapter covers the general concepts of policing and shaping in Cisco routers, followed by the detailed concepts and configuration for two policing tools and four shaping tools.

- **Chapter 6, "Congestion Avoidance Through Drop Policies"**

  Interestingly, statistics show that the biggest reason that packets are lost in networks is because a queue fills, leaving no room to hold another packet, forcing the device to discard the packet. Congestion Avoidance tools monitor queue depths, discarding some packets before the queue fills. The early discards cause the computers that sent the dropped packets to slow down the rate of sending packets, abating the congestison. As usual, this chapter covers the concepts and then the configuration behind two congestion avoidance tools.

- **Chapter 7, "Link-Efficiency Tools"**

  Link-efficiency tools deal with how to best use the bandwidth on a link between two routers. Compression, which is one class of link-efficiency tool, reduces the required bandwidth. Fragmentation tools reduce delay for small, delay-sensitive packets by breaking large packets into smaller packets. The smaller delay-sensitive packets can be sent before the fragments of the original larger packet. This chapter covers the base concepts as well as the configuration details.

- **Chapter 8, "Call Admission Control and QoS Signaling"**

  To support voice and video traffic, you need many of the tools covered in the earlier chapters. However, even with those, a network that allows too many voice or video connections can make all the the voice and video traffic degrade to the point of being unusable. Call admission control (CAC) tools allow the network administrator to control the volume of voice and video connections. This chapter covers 10 CAC tools for voice along with configurations, and comparisons of the features of the tools.

- **Chapter 9, "Management Tools and QoS Design"**

  This short chapter covers the QoS management tools provided by Cisco. In addition, this chapter covers a four-step QoS design process that is specifically mentioned in the DQOS exam topics.

- **Chapter 10, "LAN QoS"**

  The DQOS exam, at the time this book was published, covered some very basic concepts about QoS on LAN switches, but no configuration. This chapter covers LAN switch QoS concepts, and configuration. Why? Well, successful voice implementation requires good QoS implementation, even in the campus. Also, when the exam does change one day, it may cover more details of campus QoS. We included this chapter in anticipation of the exam change and because we want this book to serve as a comprehensive reference.

- **Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections"**

  This appendix lists the answers to the questions covered at the beginning and end of each chapter.

- **Appendix B, "Topics on the CCIP QOS Exam"**

  For you QOS exam takers, this appendix fills in the gaps with coverage of the topics on the QOS exam that are not covered in Chapters 1 through 10.

Book Features

The core chapters of this book have several features that help you make the best use of your time:

- **"Do I Know This Already?" Quizes—**Each chapter begins with a quiz that helps you determine the amount of time you need to spend studying that chapter. If you follow the directions at the beginning of the chapter, the "Do I Know This Already?" quiz directs you to study all or particular parts of the chapter.
- **Foundation Topics—**These are the core sections of each chapter. They explain the protocols, concepts, and configuration for the topics in that chapter.
- **Foundation Summary—**Near the end of each chapter, a summary collects the most important tables and figures from the chapter. The "Foundation Summary" section is designed to help you review the key concepts in the chapter if you scored well on the "Do I Know This Already?" quiz. This section is also an excellent tool for last-minute reviews before you take the exam.
- **Q&A—**Each chapter ends with a Q&A section that forces you to exercise your recall of the facts and processes described in the chapter's foundation topics. The questions are generally harder than the actual exam, partly because the questions are in "short answer" format, instead of multiple choice format. These questions are a great way to increase the accuracy of your recollection of the facts and to practice for taking the exam.
- **Examples—**Located inside the Foundation Topics of most chapters, the text includes screen captures from lab scenarios that show how each tool works. The examples include a topology, the configuration, and show command output that matches the examples.
- **CD-ROM-based Practice Exam—**The companion CD-ROM contains multiple-choice questions and a testing engine. As part of your final preparation, you should practice with these questions to help you get used to the exam-taking process, as well as help refine and prove your knowledge of the exam topics.

Study Plan for DQOS 9E0-601 Exam Candidates

The study plan for those of you taking the DQOS exam 9E0-601 follows the book in sequence. However, you do have a few choices about how to proceed through the book and through your study time. Following are some strategies for reading this book:

- Be sure to continually check Cisco.com and www.ciscopress.com/1587200589 for any updates about changes to the DQOS exam. If the exam changes, we will post some study suggestions relating to any changed coverage on the exam.
- Read Chapters 1 through 9 in succession. These chapters cover the topics on the exam.

- Either read or lightly review the concepts in Chapter 10. The DQOS exam, in its current form, does not cover LAN QoS configuration at all and covers just the base classification and marking concepts on the switch. So, your choice of how to approach Chapter 10 depends on whether you only want to pass the exam with minimal time expended, or whether you want to learn more about this important topic.
- When reading each chapter, use the “Do I Know This Already Quiz” to determine how much time you need to spend on that chapter’s topics. These chapter-beginning quizzes are designed to help you plan how to best use your time when studying for the exam. Make sure that you do not “guess” the answers when using these quizzes, because it is important that you give yourself an accurate score for the purposes of determing how to best spend your time.
- Use the test engine on the CD-ROM after you have completed your study. You can use this as part of your final preparation for the exam.
- Use the “Foundation Summary” sections of each chapter for review during your final preparation for the exam as well.
- Throughout your preparation, review the exam topics that are posted at Cisco.com, which are also listed in this introduction for reference.

Study Plan for QOS 642-641 Exam Candidates

The study plan for those of you taking the QOS exam 642-641 also follows the sequence in the book closely but with some minor detours. You should still consider the general advice listed in the section desscribing the study plan for the DQOS exam, but you should also consider the information in Table I-6 for the best use of your study time.

Table I-6 *Deviations from Just Reading the Book Cover-to-Cover When Preparing for the QOS Exam*

Book Chapter	Suggested Deviation as compared with DQOS study plan
1	None; read and study the entire chapter
2	None; read and study the entire chapter
3	Read and study the entire chapter; then go to Appendix B, and read the section about QoS Policy Propagation with BGP (QPPB). QPPB is the one classification and marking tool that is on the QOS exam, but not on the DQOS exam.
4	Read and study the entire chapter; then go to Appendix B, and read the section about Queuing. This section covers several Queuing tools not covered on the DQOS exam

continues

Table I-6 *Deviations from Just Reading the Book Cover-to-Cover When Preparing for the QOS Exam (Continued)*

Book Chapter	Suggested Deviation as compared with DQOS study plan
5	None; read and study the entire chapter
6	None; read and study the entire chapter
7	None; read and study the entire chapter
8	Ignore all topics covering voice CAC, but do read all coverage of RSVP.
9	You do not need to read this chapter; the topics are not covered on the QOS exam
10	You do not need to read this chapter; the topics are not covered on the QOS exam

Now you know what to expect from this book and the exams. It's time to dive in and start the good part!

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Topics

- List five benefits for implementing QoS in the enterprise networks.
- Describe how a converged network behaves without QoS.

QoS Exam Objective

- Describe the need for IP QoS.

CHAPTER 1

QoS Overview

Cisco provides a large number of quality of service (QoS) features inside Cisco IOS Software. When most of us think about QoS, we immediately think of the various queuing mechanisms, such as Weighted Fair Queuing, or Custom Queuing. QoS features include many more categories, however—fragmentation and interleaving features, compression, policing and shaping, selective packet-drop features, and a few others. And inside each of these categories of different QoS tools, there are several competing options—each with varying degrees of similarities both in concept and configuration.

You will have prepared for more than 30 specific Cisco IOS Software QoS tools before you have finished preparing for the Channel Partner DQoS exam or the CCIP QoS exam. Recalling which tools are used for what function, how they compare with similar tools, and how each is configured—these will be your most challenging tasks when preparing for the exams. The chapters in this book cover all the underlying details used by a class of QoS tool, stressing the similarities and differences between tools in the same category.

To remember all the details about QoS tools, you need a firm foundation in the core concepts of QoS. This chapter, as well as Chapter 2, "QoS Tools and Architectures," provides the foundation that you need to organize the concepts and memorize the details in other chapters.

"Do I Know This Already?" Quiz

The purpose of the "Do I Know This Already?" quiz is to help you decide whether you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 12-question quiz, derived from the major sections in the "Foundation Topics" portion of the chapter, helps you determine how to spend your limited study time.

Table 1-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 1-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions	Score
1	QoS: Tuning Bandwidth, Delay, Jitter, and Loss	1 to 6	
2	Traffic Characteristics of Voice, Video, and Data	7 to 12	
All questions		1 to 12	

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **10 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary," and the "Q&A" section.
- **11 or 12 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, proceed to the next chapter.

QoS: Tuning Bandwidth, Delay, Jitter, and Loss Questions

1. List the four traffic characteristics that QoS tools can affect.
2. Describe some of the characteristics of video traffic when no QoS is applied in a network.
3. Define bandwidth. Compare and contrast bandwidth concepts over point-to-point links versus Frame Relay.
4. List the categories of delay that could be experienced by all three types of traffic: data, voice, and video.
5. Define jitter. Give an example that shows a packet without jitter, followed by a packet with jitter.
6. Define packet loss, and describe the primary reason for loss for which QoS tools can help.

Traffic Characteristics of Voice, Video, and Data Questions

7 Describe the contents of an IP packet carrying the payload for a G.729 Voice over IP (VoIP) call.

8 Describe the amount of bandwidth required for G.711 and G.729 VoIP calls, ignoring data-link header/trailer overhead.

9 Define the meaning of the term "packetization delay" in relation to a voice call.

10 Define the term "codec delay" and discuss the two components when using a G.729 codec.

11 Describe a typical video payload flow in terms of packet sizes and packet rates.

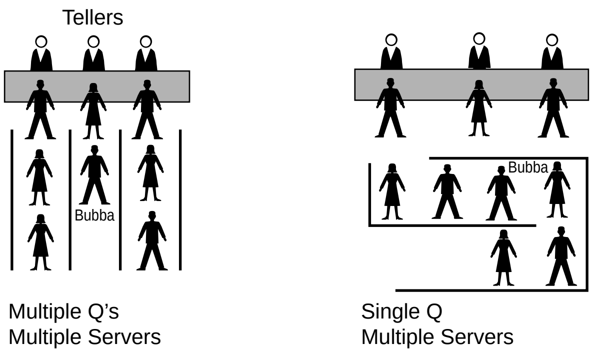
12 Contrast the QoS characteristics needed by interactive data applications to the QoS needs of voice payload flows.

Foundation Topics

When I was a young lad in Barnesville, Georgia, I used to go to the bank with my dad. Each bank teller had his or her own line of people waiting to talk to the teller and transact their business. Invariably, we would always get behind someone who was really slow. (We called that Bubba's law—you always get behind some large, disagreeable guy named "Bubba" in line.) So, someone who came to the bank after we did would get served before we would, because he or she didn't get behind a "Bubba." But, it was the rural South, so no one was in that much of a hurry, and no one really worried about it.

Later we moved to the big city of Snellville, just outside Atlanta. At the bank in Snellville, people were in a bigger hurry. So, there was one line and many tellers. As it turns out, and as queuing theory proves, the average time in the queue is decreased with one queue served by many tellers, rather than one queue for each teller. Therefore, if one slow person (Bubba) was talking to teller 1, when teller 2 became available, my dad and I could go next, rather than the person who showed up at the bank after we did. Figure 1-1 depicts the two competing queuing methods at a typical bank or fast-food chain—multiple queues, multiple servers versus single queue, multiple servers. The single queue/multiple servers method improves average wait time, but also eliminates the possibility of your good luck in choosing a fast line—the one with no Bubbas in it.

Figure 1-1 *Comparing Multiple Server/Multiple Queue to Multiple Server/Single Queue*



The bank in Snellville just chose a different queuing method, and that positively affected everyone, right? Well, the choice of using a single queue did have one negative effect—because there was only one queue, you could never show up, pick one of the many queues, and happen to get in the one with only fast people in it. In this scenario, on average everyone gets better service, but you miss out on the chance to get in and out of the bank really fast. In short, most customers' experience is improved, and some customers' experience is degraded.

In networking, QoS describes a large array of concepts and tools that can be used to affect the packet's access to some service. Most of us think of queuing features when we think of QoS—reordering the output queue so that one packet gets better service than another. But many other QoS features affect the quality—compression, drop policy, shaping, policing, and signaling, to name a few. In the end, whichever mechanism you use, you improve the behavior for one type of packet over another. Just like at the bank, implementing QoS is "managed fairness," and at the same time it is "managed unfairness"—you purposefully choose to favor one packet over another.

All of us can relate to the frustration of waiting in lines (queues) for things in our daily lives. It would be great if there were never any people in line ahead of us at the tollbooths, or waiting to get on a ride at Disneyland (or any other place). For that to be possible, however, there would need to be a lot more tollbooths, Disneyland would need to be 20 times larger, and banks would need to hire a lot more tellers. Even so, adding more capacity would not always solve the problem—the tollbooth would still be crowded at rush hour, Disneyland would still be crowded when schools are not in session, and banks would still be crowded on Friday afternoons when everyone is trying to cash his or her weekly paycheck (at least where I live!). Making Disneyland 20 times larger, so that there are no queues, is financially ridiculous—likewise, the addition of 20 times more bandwidth to an existing link is probably also financially unreasonable. After all, you can afford only so much capacity, or bandwidth in the case of networking.

This chapter begins by taking a close look at the four traffic characteristics that QoS tools can affect:

- Bandwidth
- Delay
- Jitter
- Packet loss

Whereas QoS tools improve these characteristics for some flows, the same tools might degrade service for other flows. Therefore, before you can intelligently decide to reduce one packet's delay by increasing another packet's delay, you should understand what each type of application needs. The second part of this "Foundation Topics" section examines voice, video, and data flows in light of their needs for bandwidth, delay, jitter, and loss.

QoS: Tuning Bandwidth, Delay, Jitter, and Loss

Different types of end-user traffic require different performance characteristics on a network. A file-transfer application might just need throughput, but the delay a single packet experiences might not matter. Interactive applications might need consistent response time. Voice calls need low, consistent delay, and video conferencing needs low, consistent delay as well as high throughput.

Users might legitimately complain about the performance of their applications, and the performance issues may be related to the network. Of course, most end users will believe the network is responsible for performance problems, whether it is or not! Reasonable complaints include the following:

- My application is slow.
- My file takes too long to transfer now.
- The video freezes.
- The phone call has so much delay we keep talking at the same time, not knowing whether the other person has paused.
- I keep losing calls.

In some cases, the root problem can be removed, or at least its impact lessened, by implementing QoS features.

So, how do voice, video, and data traffic behave in networks that do not use QoS? Well, certainly the performance varies. Table 1-2 outlines some of the behaviors in a network without QoS.

Table 1-2 *Traffic Behavior with No QoS*

Type of Traffic	Behavior Without QoS
Voice	Voice is hard to understand.
	Voice breaks up, sounds choppy.
	Delays make interacting difficult; callers do not know when other party has finished talking.
	Calls are disconnected.
Video	Picture displays erratically; jerky movements.
	Audio not in sync with video.
	Movement slows down.
Data	Data arrives after it is no longer useful.
	Customer waiting for customer care agent, who waits for a screen to display.
	Erratic response times frustrate users, who may give up or try later.

QoS attempts to solve network traffic performance issues, although QoS is not a cure-all. To improve network performance, QoS features affect a network by manipulating the following network characteristics:

- Bandwidth
- Delay
- Jitter (delay variation)
- Packet loss

Unfortunately, improving one QoS characteristic might degrade another. *Bandwidth* defines the capacity of the transmission media. Compression tools reduce the amount of bandwidth needed to send all packets, but the compression process adds some delay per packet and also consumes CPU cycles. *Jitter* is the variation in delay between consecutive packets, so it is sometimes called "delay variation." A router can reduce jitter for some traffic, but that usually increases delay and jitter for other traffic flows. QoS features can address jitter problems, particularly the queuing features that have priority queuing for packets that need low jitter. *Packet loss* can occur because of transmission errors, and QoS mechanisms cannot do much about that. However, more packets might be lost due to queues filling up rather than transmission errors—and QoS features can affect which packets are dropped.

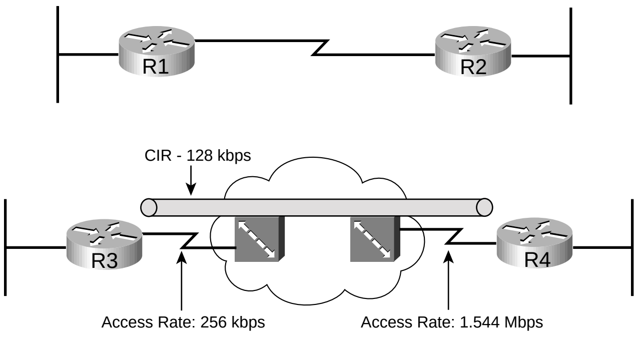
You can think of QoS as "managed fairness" and, conversely, as "managed unfairness." The real key to QoS success requires you to improve a QoS characteristic for a flow that needs that characteristic, while degrading that same characteristic for a flow that does not need that characteristic. For instance, QoS designs should decrease delay for delay-sensitive traffic, while increasing delay for delay-insensitive traffic.

The next four short sections take a closer look at each of these four traffic characteristics.

Bandwidth

The term "bandwidth" refers to the number of bits per second that can reasonably be expected to be successfully delivered across some medium. In some cases, bandwidth equals the physical link speed, or the clock rate, of the interface. In other cases, bandwidth is smaller than the actual speed of the link. Consider, for example, Figure 1-2, which shows two typical networks, one with a point-to-point serial link, and the other using Frame Relay.

Figure 1-2 *Two Similar Networks, One with Point-to-Point, One with Frame Relay*



In the point-to-point network, WAN bandwidth equals the physical link speed, or clock rate, on the physical medium. Suppose, for instance, that the link is a 64-kbps link—you could reasonably expect to send 64 kbps worth of traffic, and expect it to get to the other side of the link. You would never expect to send more than that, because you cannot send the bits any faster than the clock rate of the interface. Bandwidth, in this case, is indeed rather obvious; you get 64 kbps in both directions.

The Frame Relay network provides a contracted amount of bandwidth. In practice, however, many installations expect more than that! The committed information rate (CIR) defines how much bandwidth the provider guarantees will pass through their network between the data terminal equipment (DTE) at each end of a virtual circuit (VC). That guarantee is a business proposition—a Layer 8 issue using the OSI reference model. On some occasions, you might not actually even get CIR worth of bandwidth. However, the Frame Relay provider commits to engineering a network so that they can support at least the CIRs of their collective VCs. In effect, bandwidth per VC equals the CIR of each VC, respectively.

Unfortunately, bandwidth on multiaccess networks is not that simple. Consider the fact that R3 has a 256-kbps access rate, and R4 has a T1 access rate. When R3 sends, it must send the bits at access rate—otherwise, Layer 1 functions would not work at all. Similarly, R4 must send at T1 speed. One of Frame Relay's big selling points throughout its large growth years was that you "get something for nothing"—you pay for CIR of *x*, and you get more than *x* worth of bandwidth. In fact, many data network engineers design networks assuming that you will get an average of one and a half to two times CIR over each VC. If R3 and R4 send too much, and the provider's switches have full queues, the frames are discarded, and the data has to be re-sent. If you pay for 128-kbps CIR between R3 and R4, and over time actually send at 192 kbps, or 256 kbps, and it works, how much bandwidth do you really have? Well, on a multiaccess network, such as Frame Relay or ATM, the actual amount of bandwidth is certainly open to argument.

Frame Relay bandwidth might even be less than CIR in practice. Suppose that R4 is the main site, and there are 15 remote sites identical to R3 (including R3). Can you reasonably expect to send at 128 kbps (CIR) to all 15 sites, at the same time, when R4 has a 1.544-Mbps access rate? No! All 15 sites sending at 128 kbps requires 1.920 Mbps, and R4 can only send and receive at 1.544 Mbps. Would an engineer design a network like this? Yes! The idea being that if data is being sent to all VCs simultaneously, either R4 will queue the data (for packets going right to left), or the Frame Relay switch will queue the data (for packets going left to right). If data is not being sent to all 15 remote sites at the same time, you get (at least) 128 kbps to each site that needs the bandwidth at that time. The negative effect is that a larger percentage of packets are dropped due to full queues; however, for data networks that is typically a reasonable tradeoff. For traffic that is not tolerant to loss, such as voice and video, this type of design may not be reasonable.

Throughout this book, when multiaccess network bandwidth is important, the discussion covers some of the implications of using the more conservative bandwidth of CIR, versus the more liberal measurements that are typically a multiple of CIR.

The **clock rate** Command Versus the **bandwidth** Command

When you are using a Cisco router, two common interface commands relate to bandwidth. First, the **clock rate** command defines the actual Layer 1 bit rate. The command is used when the router is providing clocking, typically when connecting the router using a serial interface to some other nearby device (another router, for instance). The **bandwidth** command tells a variety of Cisco IOS Software functions how much bandwidth is assumed to be available on the interface. For instance, Enhanced Interior Gateway Routing Protocol (EIGRP) chooses metrics for interfaces based on the **bandwidth** command, not based on a **clock rate** command. In short, bandwidth only changes the behavior of other tools on an interface, and it affects the results of some statistics, but it never changes the actual rate of sending bits out an interface.

Some QoS tools refer to interface bandwidth, which is defined with the **bandwidth** command. Engineers should consider bandwidth defaults when enabling QoS features. On serial interfaces on Cisco routers, the default bandwidth setting is T1 speed—regardless of the actual bandwidth. If subinterfaces are used, they inherit the bandwidth setting of the corresponding physical interface. In Figure 1-2, for example, R3 would have a default bandwidth setting of 1544 (the units are in kbps), as opposed to a more accurate 128, 192, or 256 kbps, depending on how conservative or liberal the engineer can afford to be in this network.

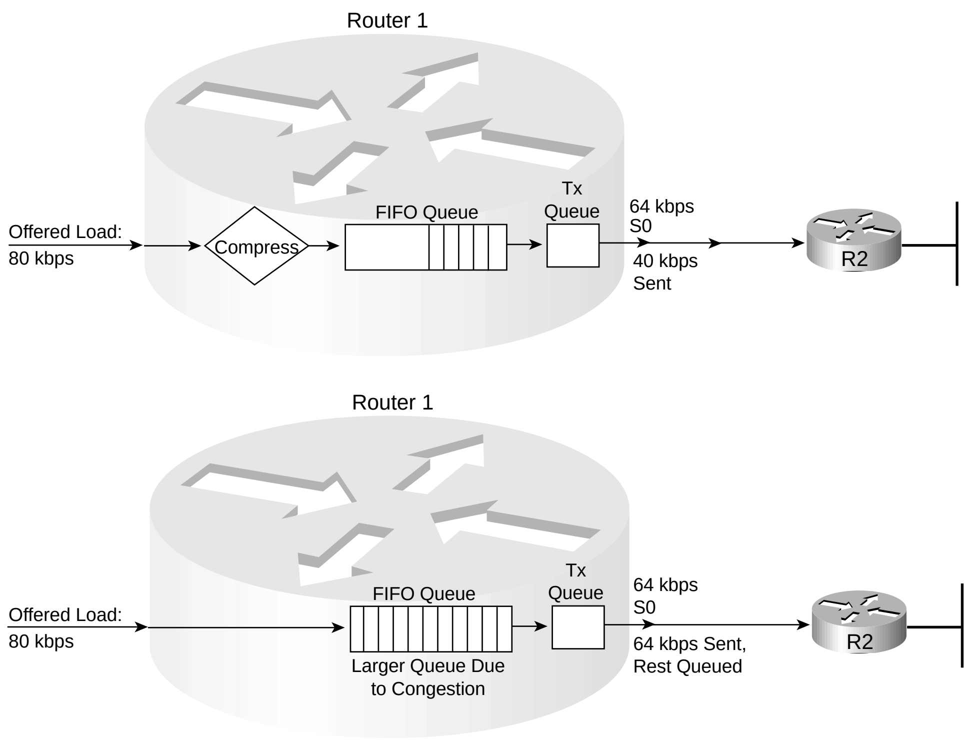
QoS Tools That Affect Bandwidth

Several QoS features can help with bandwidth issues. You'll find more detail about each of these tools in various chapters throughout this book. For now, however, knowing what each class of QoS tool accomplishes will help you sift through some of the details.

The best QoS tool for bandwidth issues is more bandwidth! However, more bandwidth does not solve all problems. In fact, in converged networks (networks with voice, video, and data), adding more bandwidth might be masking delay problems that are best solved through other QoS tools or through better QoS design. To quote Arno Penzias, former head of Bell Labs and a Nobel Prize winner: "Money and sex, storage and bandwidth: Only too much is ever enough." If you can afford it, more bandwidth certainly helps improve traffic quality.

Some link-efficiency QoS tools improve bandwidth by reducing the number of bits required to transmit the data. Figure 1-3 shows a rather simplistic view of the effect of compression, assuming the compression ratio is 2:1. Without compression, with 80 kbps of offered traffic, and only a 64-kbps point-to-point link, a queue will form. The queue will eventually fill, and packets will be dropped off the end of the queue—an action called *tail drop*. With compression, if a ratio of 2:1 is achieved, the 80 kbps will require only 40 kbps in order to be sent across the link—effectively doubling the bandwidth capacity of the link.

Figure 1-3 *With a 2:1 Compression Ratio Versus No Compression*

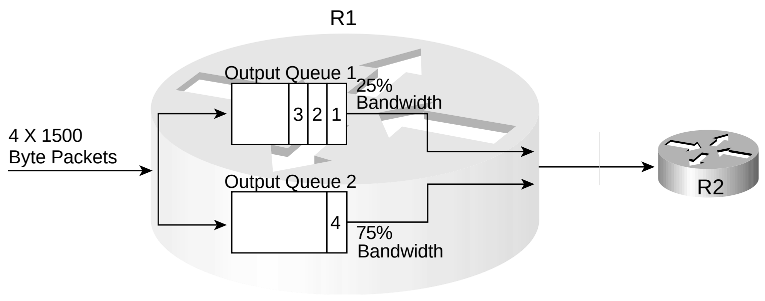


This book covers several options for compression, some of which happen before the queuing process (as shown in Figure 1-3), and some that happen after the queuing process.

The other QoS tool that directly affects bandwidth is call admission control (CAC). CAC tools decide whether the network can accept new voice and video calls. That permission might be based on a large number of factors, but several of those factors involve either a measurement of bandwidth. For example, the design might expect only three concurrent G.729A VoIP calls over a particular path; CAC would be used for each new call, and when three calls already exist, the next call would be rejected. (If CAC did not prevent the fourth call, and a link became oversubscribed as a result, all the quality of all four calls would degrade!) When CAC rejects a call, the call might be rerouted based on the VoIP dial plan, for instance, through the Public Switched Telephone Network (PSTN).

Queuing tools can affect the amount of bandwidth that certain types of traffic receive. Queuing tools create multiple queues, and then packets are taken from the queues based on some queue-servicing algorithm. The queue-servicing algorithm might include a feature that guarantees a minimum amount of bandwidth to a particular queue. Figure 1-4, for example, shows a two-queue system. The first queue gets 25 percent of the bandwidth on the link, and the second queue gets 75 percent of the bandwidth.

Figure 1-4 *Bandwidth Reservation Using Queuing*



With regard to Cisco IOS Software queuing tools that reserve bandwidth, if both queues have packets waiting, the algorithm takes packets such that, over time, each queue gets its configured percentage of the link bandwidth. If only one queue has packets waiting, that queue gets more than its configured amount of bandwidth for that short period.

Although adding more bandwidth always helps, the tools summarized in Table 1-3 do help to improve the efficient utilization of bandwidth in a network.

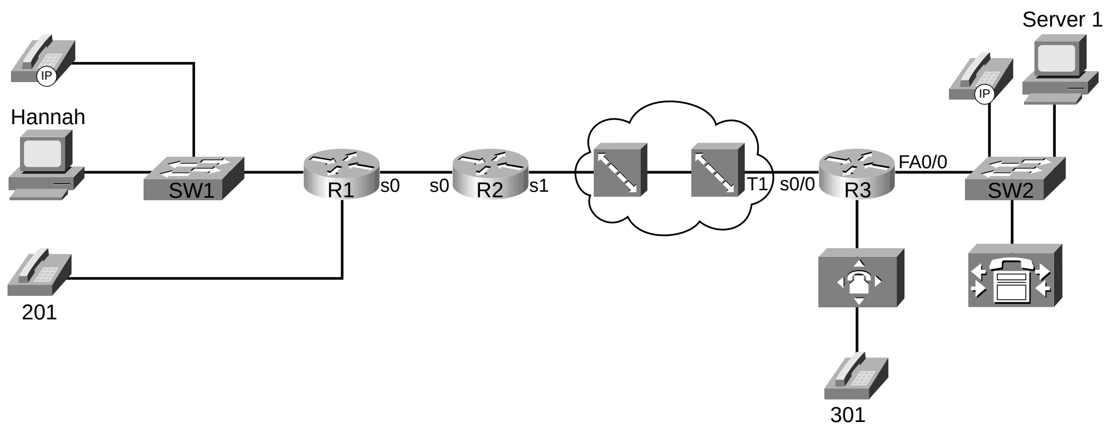
Table 1-3 *QoS Tools That Affect Bandwidth*

Type of QoS Tool	How It Affects Bandwidth
Compression	Compresses either payload or headers, reducing overall number of bits required to transmit the data
CAC	Reduces overall load introduced into the network by rejecting new voice and video calls
Queuing	Can be used to reserve minimum amounts of bandwidth for particular types of packets

Delay

All packets in a network experience some delay between when the packet is first sent and when it arrives at its destination. Most of the concepts behind QoS mechanisms relate in some way to delay. Therefore, a deeper look into delay is useful. Take a look at Figure 1-5; this sample network is used often in this book.

Figure 1-5 *Sample Network for Discussion of Delay*



At what points will delay occur in this network? Well, at all points, in actuality. At some points in the network, the delay is so small that it can just be ignored for practical purposes. In other cases, the delay is significant, but there is nothing you can do about it! For a fuller understanding, consider the following types of delay:

- Serialization delay (fixed)
- Propagation delay (fixed)
- Queuing delay (variable)
- Forwarding/processing delay (variable)
- Shaping delay (variable)
- Network delay (variable)
- Codec delay (fixed)
- Compression delay (variable)

Each of these types of delay is explained over the next several pages. Together, the types of delay make up the components of the end-to-end delay experienced by a packet.

Serialization Delay

Imagine you are standing at a train station. A train comes by but doesn't stop; it just keeps going. Because the train cars are connected serially one to another, a time lag occurs between when the engine car at the front of the train first gets to this station and when the last car passes by. If the train is long, it takes more time until the train fully passes. If the train is moving slowly, it takes longer for all the cars to pass. In networking, serialization delay is similar to the delay between the first and last cars in a train.

Serialization delay defines the time it takes to encode the bits of a packet onto the physical interface. If the link is fast, the bits can be encoded onto the link more quickly; if the link is slow, it takes longer to encode the bits on the link. Likewise, if the packet is short, it does not take as long to put the bits on the link as compared with a long packet.

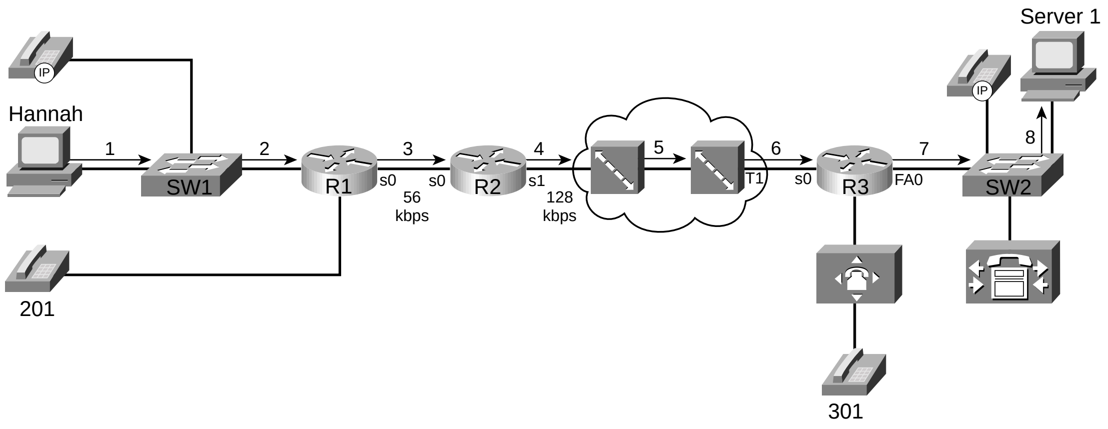
Use the following formula to calculate serialization delay for a packet:

$$\frac{\text{\#bits sent}}{\text{Link speed}}$$

Suppose, for instance, that Hannah send a 125-byte packet to Server1. Hannah sends the packet over the Fast Ethernet to the switch. The 125 bytes equal 1000 bits, so at Fast Ethernet speeds, it takes 1000 bits/100,000,000 bits per second (bps), or .01 ms, to serialize the packet onto the Fast Ethernet. Another .01 ms of serialization delay is experienced when the switch sends the frame to R1. (I ignored the data-link header lengths to keep the math obvious.)

Next, when that same packet leaves R1 over a 56 kbps link to R2, serialization takes 1000 bits/ 56,000 bps, or 17.85 ms. The serialization component over Fast Ethernet is insignificant, whereas serialization becomes a more significant number on lower-speed serial links. Figure 1-6 shows the various locations where the packet from Hannah to Server1 experiences serialization delay.

Figure 1-6 *Serialization Delay*



As Figure 1-6 shows, serialization delay occurs any time a frame is sent. On LAN links, the delay is insignificant for most applications. At steps 3 through 6 in the figure, the serialization delay is 17.85 ms, 7.8 ms, .02 ms, and .65 ms for the 125-byte packet, respectively. Also note that serialization delays do occur inside the Frame Relay cloud. (You can read more about delays inside the cloud in the "Network Delay" section later in this chapter.)

Table 1-4 lists the serialization delay for a couple of frame sizes and link speeds.

Table 1-4 *Example Serialization Delay Values*

Clock Rate of Link	Serialization Delay (125-Byte Frame; Milliseconds)	Serialization Delay (1500-Byte Frame; Milliseconds)
100 Mbps	.01	.12
1.544 Mbps	.65	8
512 kbps	2	24
128 kbps	7.8	93
56 kbps	17.85	214

Propagation Delay

Imagine you are watching a train again, this time from a helicopter high in the air over the tracks. You see the train leaving one station, and then arriving at the second station. Using a stopwatch, you measure the amount of time it takes from the first car leaving the first station until the first car arrives at the second station. Of course, all the other cars take the same amount of time to get there as well. This delay is similar to propagation delay in networking.

Propagation delay defines the time it takes a single bit to get from one end of the link to the other. When an electrical or optical signal is placed onto the cable, the energy does not propagate to the other end of the cable instantaneously—some delay occurs. The speed of energy on electrical and optical interfaces approaches the speed of light, and the network engineer cannot override the laws of physics! The only variable that affects the propagation delay is the length of the link. Use the following formula to calculate propagation delay:

$$\frac{\text{Length of Link (meters)}}{3.0 \times 10^8 \text{ meters/second}}$$

or

$$\frac{\text{Length of Link (meters)}}{2.1 \times 10^8 \text{ meters/second}}$$

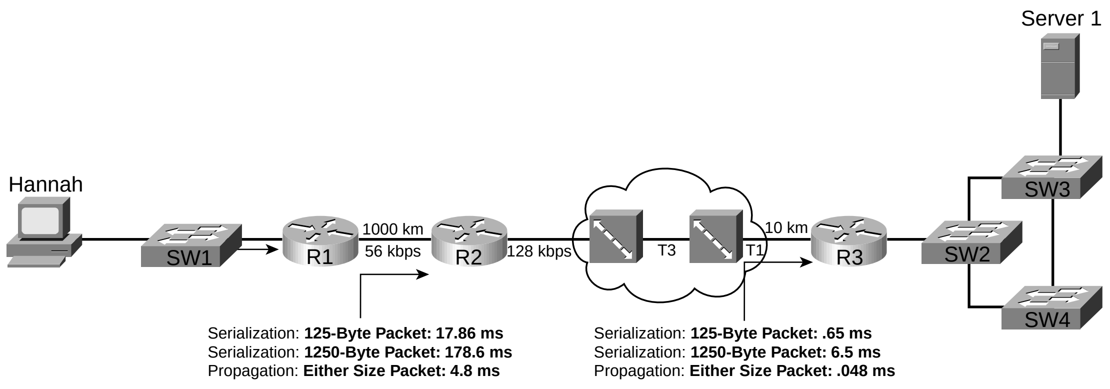
where $3.0 * 10^8$ is the speed of light in a vacuum. Many people use $2.1 * 10^8$ for the speed of light over copper and optical media when a more exact measurement is needed. (Seventy percent of the speed of light is the generally accepted rule for the speed of energy over electrical cabling.)

Propagation delay occurs as the bits traverse the physical link. Suppose, for instance, that the point-to-point link between R1 and R2 is 1000 kilometers (1,000,000 meters) long. The propagation delay would be as follows:

$$\frac{1{,}000{,}000}{2.1 \times 10^8} = 004.8 \text{ ms}$$

Figure 1-7 shows two contrasting examples of serialization and propagation delay.

Figure 1-7 *Serialization and Propagation Delay for Selected Packet and Link Lengths*


Server 1
Hannah
SW1
R1
1000 km
56 kbps
R2
128 kbps
T3
T1
10 km
R3
SW2
SW3
SW4
Serialization: **125-Byte Packet: 17.86 ms**
Serialization: **1250-Byte Packet: 178.6 ms**
Propagation: **Either Size Packet: 4.8 ms**
Serialization: **125-Byte Packet: .65 ms**
Serialization: **1250-Byte Packet: 6.5 ms**
Propagation: **Either Size Packet: .048 ms**

As you can see in Figure 1-7, the length of the link affects propagation delay, whereas the size of the packet and link speed affect serialization delay. The serialization delay is larger for larger packets, but the propagation delay is equal for different-sized packets, on the same link. One common misconception is that the link speed, or clock rate, affects propagation delay—it does not! Table 1-5 lists the various propagation delays and serialization delays for parts of Figure 1-6.

Table 1-5 *Example Serialization and Propagation Delays with Figure 1-6*

Step Number from Figure	Length of Link	Clock Rate of Link	Propagation Delay (Milliseconds)	Serialization delay (125-Byte Packet; Milliseconds)	Serialization Delay (1500-Byte Packet; Milliseconds)
1	50 m	100 Mbps	.002	.01	.12
2	10 m	100 Mbps	.0004	.01	.12
3	1000 km	56 kbps	4.8	17.85	214

continues

Table 1-5 *Example Serialization and Propagation Delays with Figure 1-6 (Continued)*

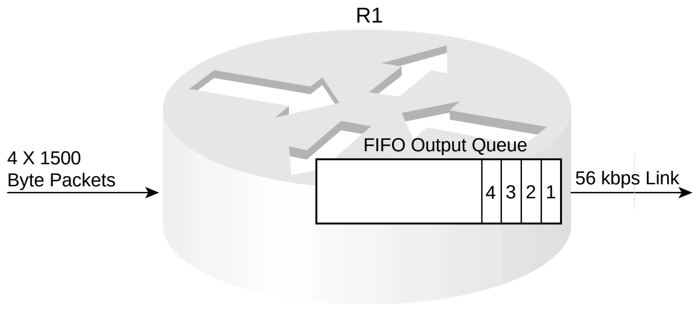
Step Number from Figure	Length of Link	Clock Rate of Link	Propagation Delay (Milliseconds)	Serialization delay (125-Byte Packet; Milliseconds)	Serialization Delay (1500-Byte Packet; Milliseconds)
4	5 km	128 kbps	.024	7.8	94
5	1000 km	44.232 Mbps	4.8	.02	.24
6	10 km	1.544 Mbps	.048	.65	7.8

If the link from Hannah to SW1 is 100 meters, for example, propagation is 100/(2.1 * 10^8), or .48 microseconds. If the T3 between the two Frame Relay switches is 1000 kilometers, the delay is 1,000,000/(2.1 * 10^8), or 4.8 ms. Notice that propagation delay is not affected by clock rate on the link—even on the 56-kbps Frame Relay access link, at 1000 km (a long Frame Relay access link!), the propagation delay would only be 4.8 ms.

Queuing Delay

Packets experience *queuing delay* when they must wait for other packets to be sent. Most people think of queuing delay when they think of QoS, and most people think of queuing strategies and tools when they think of QoS tools—but queuing tools are just one category of QoS tool. Queuing delay consists of the time spent in the queues inside the device—typically just in output queues in a router, because input queuing is typically negligible in a router. However, the queuing time can be relatively large—hundreds of milliseconds, or maybe even more. Consider Figure 1-8, where R1 queues four 1500-byte packets that Hannah sent to Server1.

Figure 1-8 *R1 Queues Four 1500-Byte Packets for Transmission*

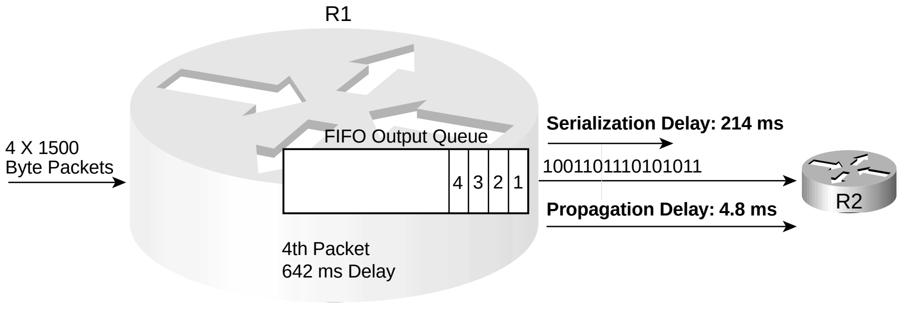


Because it takes 1500 * 8 / 56,000, or 214 ms, to serialize each 1500-byte packet, the other packets need to either be stored in memory or discarded. Therefore, the router uses some memory to hold the packets. The simplest form of queuing is to use a single queue, serviced with first-in, first-out (FIFO) logic—as is shown in the figure. After 856 ms, all four packets would have been sent out the serial link. Assuming that the link was not busy when Hannah sent these four packets, how much queuing delay did each packet experience? Well, the first packet experienced no queuing delay. The second packet waited on the first, or 214 ms. The third packet waited on the first two—or 428 ms. And the fourth packet waited on the first three, for a total of 642 ms.

Queuing provides a useful function, because the second, third, and fourth packets would have had to have been discarded without queuing. However, too much of a good thing is not always good! Imagine that Hannah sends 100 * 1500-byte packets all at once. If the queue in R1 is large enough, R1 could queue all 100 packets. What would the delay be for the one-hundredth packet? Well, 99 * 214 ms per packet, or roughly 21 seconds! If Hannah uses TCP, then TCP has probably timed out, and re-sent the packets—causing more congestion and queuing delay. And what about another user's packet that showed up right after Hannah's 100 packets? Still more delay. So, some queuing helps prevent packet drops, but large queues can cause too much delay.

Figure 1-9 combines all the delay components covered so far into one small diagram. Consider the delay for the fourth of the four 1500-byte packets sent by Hannah. The figure lists the queuing, serialization, and propagation delays.

Figure 1-9 *Delay Components: Three Components, Single Router (R1)*



The overall delay for a packet is the sum of all these delays from end to end. At R1, when all four packets have been received, the fourth packet experiences a total of about 860 ms of delay before it has been fully received at R2. And this example just shows the queuing delay in a single router (R1), and the serialization and propagation delay over a single link—end-to-end delay includes these delays at each router (queuing) and link (serialization and propagation) in the network.

Forwarding Delay

The term “forwarding delay” refers to the time taken to switch the packet inside the router or switch—simply put, the time required to forward the packet. This does not include all the time taken inside the router or switch; a more accurate view is to think of it as the time taken between examining the frame/packet on the input interface, and placing it in the output queue on the output interface. In other words, it does not include a queuing delay. It does include all the processing required between the time that a frame has been received in its entirety until the frame has either been queued on the output interface or has begun transmission on the output interface.

Cisco does not normally quote statistics about forwarding delay numbers for different models of routers with different types of internal processing. However, the higher volume of packets that a router can forward, and the higher volume of packets forwarded using a particular processing method, presumably the lower the forwarding delay.

Most delay components in LAN switches are small enough not to matter. However, switches incur forwarding delay, just like routers—most of the time. Some LAN switches use a “store-and-forward” forwarding logic, when the entire frame must be received before forwarding any part of the frame. However, some switches use cut-through or fragment-free forwarding, which means that the first bits of a frame are forwarded before the final bits are fully received. Technically, if you define forwarding delay as the time between receipt of the entire frame until that frame is queued for transmission, some LAN switches might actually have negative forwarding delay! It just depends on how you decide to define what parts of the overall delay end up being attributed. Forwarding delay is typically a small enough component to ignore in overall delay budget calculations, so this book does not punish you with further discussion about these details!

For more information on internal processing methods such as Cisco Express Forwarding (CEF), you can review the Cisco Press book *Inside Cisco IOS Software Architecture*.

Shaping Delay

Traffic shaping causes additional delays by serving queues more slowly than if traffic shaping were not used. Why should a router slow down sending packets if it does not have to? Well, traffic shaping helps match the overall forwarding rate of traffic when a carrier might discard traffic if the rates exceed the contracted rate. So, which is better?

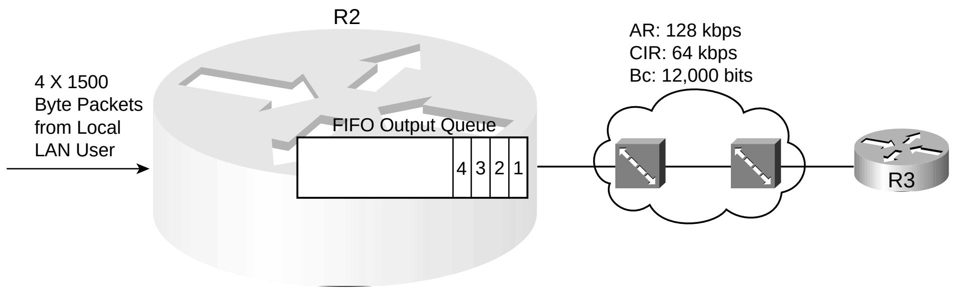
- Sending packets really fast and having them be dropped
- Sending packets more slowly, but not having them be dropped

The right answer is—it depends! If you want to send more slowly, hoping that packets are not dropped, however, traffic shaping is the solution.

Carriers can drop frames and packets inside their network for a variety of reasons. One of the most typical reasons is that most central-site routers use a fast access link, with remote sites using much slower links. If the central site uses a T1, and the remote site uses a 56-kbps link, frames may fill the queue inside the service provider's network, waiting to go across the 56-kbps access link. Many other events can cause the carrier to drop packets; these reasons events explained more fully in Chapter 5, "Traffic Policing and Shaping."

To understand the basic ideas behind shaping in a single router, consider Figure 1-10, where R2 has a 128-kbps access rate and a 64-kbps CIR on its VC to R3.

Figure 1-10 *Traffic Shaping over the Frame Relay Network*



Suppose that the Frame Relay provider agrees to the 64-kbps CIR on the VC from R2 to R3, but the carrier tells you that they aggressively discard frames when you send more than 64 kbps. The access rate is 128 kbps. Therefore, you decide to shape, which means that R2 will want to average sending at 64 kbps, because sending faster than 64 kbps hurts more than it helps. In fact, in this particular instance, if R2 will send packets for this VC only half the time, the rate averages out to 64 kbps. Remember, bits can only be sent at the physical link speed, which is also called the *access rate* in Frame Relay. In effect, the router sends all packets at access rate, but the router purposefully delays sending packets, possibly even leaving the link idle, so that the rate over time averages to be about 64 kbps.

Chapter 5 will clear up the details. The key concept to keep in mind when reading other sections of this book is that traffic shaping introduces additional delay. Like many QoS features, shaping attempts to enhance one particular traffic characteristic (drops), but must sacrifice another traffic characteristic (delay) to do so.

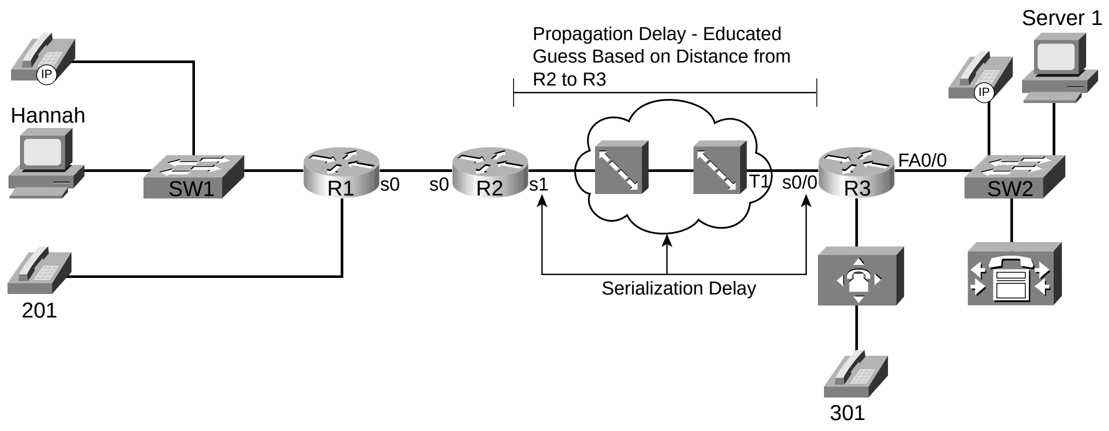
NOTE To QoS test takers: Neither the QoS nor DQoS course books list shaping delay as a delay component. Because it does affect delay, however, it is included here.

Network Delay

Most people draw a big cloud for a Frame Relay or ATM network, because the details are not typically divulged to the customer. However, the same types of delay components seen outside the cloud also exist inside the cloud—and the engineer that owns the routers and switches outside the cloud cannot exercise as much QoS control over the behavior of the devices in the cloud.

So how much delay should a packet experience in the cloud? Well, it will vary. The carrier might commit to a maximum delay value as well. However, with a little insight, you can get a solid understanding of the minimum delay a packet should experience through a Frame Relay cloud. Consider Figure 1-11, focusing on the Frame Relay components.

Figure 1-11 *Frame Relay Network: Propagation and Serialization Delay Components*



The propagation delay and serialization delay can be guessed pretty closely. No matter how many switches exist between R2 and R3, the cumulative propagation delays on all the links between R2 and R3 will be at least as much as the propagation delay on a point-to-point circuit. And with most large providers, because they have many points of presence (PoPs), the Frame Relay VC probably takes the same physical route as a point-to-point circuit would anyway. As for serialization delay, the two slowest links, by far, will be the two access links (in most cases). Therefore, the following account for most of the serialization delay in the cloud:

- The serialization delay to send the packet into the cloud
- The serialization delay at the egress Frame Relay switch, sending the packet to R3

Suppose, for example, that R2 and R3 are 1000 km apart, and a 1500-byte packet is sent. The network delay will at least be the propagation delay plus both serialization delays on the two access links:

Propagation = 1000 km / 2.1 * 10^8 = 4.8 ms

Serialization (ingress R2) = 1500 bytes * 8 / 128,000 bps = 94 ms

Serialization (egress R3) = 1500 bytes * 8 / 1,544,000 = 7.8 ms

For a total of 106.6 ms delay

Of course, the delay will vary—and will depend on the provider, the status of the network's links, and overall network congestion. In some cases, the provider will include delay limits in the contracted service-level agreement (SLA).

Queuing delay inside the cloud creates the most variability in network delay, just as it does outside the cloud. These delays are traffic dependent, and hard to predict.

NOTE To QoS test takers: Neither the QoS nor DQoS course books list network delay as a delay component. Because it does affect delay, however, it is included here.

Delay Summary

Of the types of delay covered so far in this chapter, all except shaping delay occur in every network. Shaping delay occurs only when shaping is enabled.

Two other delay components may or may not be found in a typical network. First, codec delay will be experienced by voice and video traffic. Codec delay is covered in more depth in the section titled "Voice Delay Considerations." Compression requires processing, and the time taken to process a packet to compress or decompress the packet introduces delay. Chapter 7, "Link-Efficiency Tools," covers compression delay.

Table 1-6 summarizes the delay components listed in this section.

Table 1-6 *Components of Delay Not Specific to One Type of Traffic*

Delay Component	Definition	Where It Occurs
Serialization delay (fixed)	Time taken to place all bits of a frame onto the physical medium. Function of frame size and physical link speed.	Outbound on every physical interface; typically negligible on T3 and faster links.
Propagation delay (fixed)	Time taken for a single bit to traverse the physical medium from one end to the other. Based on the speed of light over that medium, and the length of the link.	Every physical link. Typically negligible on LAN links and shorter WAN links.

continues

Table 1-6 *Components of Delay Not Specific to One Type of Traffic (Continued)*

Delay Component	Definition	Where It Occurs
Queuing delay (variable)	Time spent in a queue awaiting the opportunity to be forwarded (output queuing), or awaiting a chance to cross the switch fabric (input queuing).	Possible on every output interface. Input queuing unlikely in routers, more likely in LAN switches.
Forwarding or processing delay (variable)	Time required from receipt of the incoming frame, until the frame/packet has been queued for transmission.	On every piece of switching equipment, including routers, LAN switches, Frame Relay switches, and ATM switches.
Shaping delay (variable)	Shaping (if configured) delays transmission of packets to avoid packet loss in the middle of a Frame Relay or ATM network.	Anywhere that shaping is configured, which is most likely on a router, when sending packets to a Frame Relay or ATM network.
Network delay (variable)	Delays created by the components of the carrier's network when using a service. For instance, the delay of a Frame Relay frame as it traverses the Frame Relay network.	Inside the service provider's network.

QoS Tools That Affect Delay

Several QoS features can help with delay issues. You'll find more detail about each of these tools in various chapters throughout this book. For now, however, knowing what each class of QoS tool accomplishes will help you sift through some of the details.

The best QoS tool for delay issues is . . . more bandwidth—again! More bandwidth helps bandwidth-related problems, and it also helps delay-related problems. Faster bandwidth decreases serialization delay. Because packets exit more quickly, queuing delay decreases. Higher CIR on your VCs reduces shaping delay. In short, faster bandwidth reduces delay!

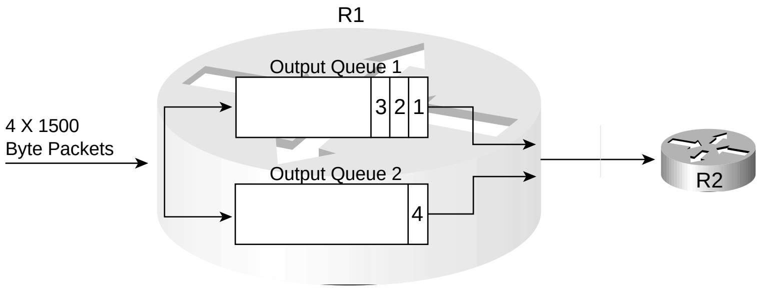
Unfortunately, more bandwidth does not solve all delay problems, even if you could afford more bandwidth! In fact, in converged networks (networks with voice, video, and data), adding more bandwidth might mask delay problems that are best solved through other QoS tools or through better QoS design. The sections that follow address the QoS tools can affect the delay a particular packet receives.

Queuing (Scheduling)

The most popular QoS tool, queuing, involves choosing the packets to be sent based on something other than arrival time. In other words, instead of FIFO queuing with one queue, other queuing mechanisms create multiple queues, place packets into these different queues, and then pick packets from the various queues. As a result, some packets leave the router more quickly, with other packets having to wait longer. Although queuing does not decrease delay for all packets, it can decrease delay for delay-sensitive packets, and increase delay for delay-insensitive packets—and enabling a queuing mechanism on a router does not cost any cash, whereas adding more bandwidth does.

Each queuing method defines some number of different queues, with different methods of scheduling the queues—in other words, different rules for how to choose from which queue the next packet to be sent will be chosen. Figure 1-12 depicts a queuing mechanism with two queues. Suppose Hannah sent four packets, but the fourth packet was sent by a video-conferencing package she was running, whereas the other three packets were for a web application she was using while bored with the video conference.

Figure 1-12 *Sample Queuing Method: Two Queues*



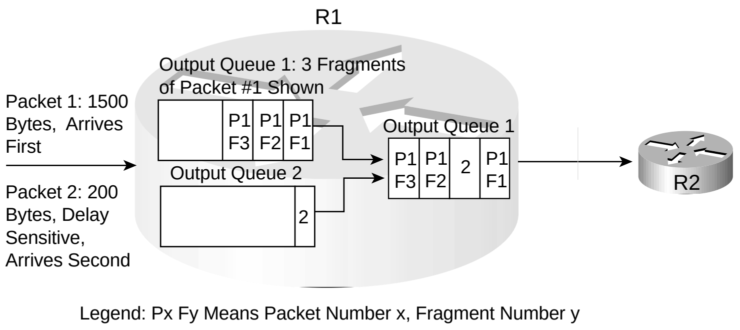
R1 could notice that packet 4 has different characteristics, and place it into a different queue. Packet 4 could exit R1 before some or all of the first three packets.

Link Fragmentation and Interleaving

The time required to serialize a packet on a link is a function of the speed of the link, and the size of the packet. When the router decides to start sending the first bit of a packet, the router continues until the whole packet is sent. Therefore, if a delay-sensitive packet shows up just after a long packet has begun to be sent out an interface, the delay-sensitive packet must wait until the longer packet has been sent.

Suppose, for example, that two packets have arrived at R1. Packet 1 is 1500 bytes, and packet 2 is 200 bytes. The smaller packet is delay sensitive. Because packet 2 arrived just after the first bit of packet 1 was sent, packet 2 must wait 214 ms for packet 1 to be serialized onto the link. With link fragmentation and interleaving (LFI), packet 1 could be broken into three 500-byte fragments, and packet 2 could be interleaved (inserted) and sent on the link after the first of the three fragments of packet 1. Figure 1-13 depicts LFI operation.

Figure 1-13 *Link Fragmentation and Interleaving*


R1
Output Queue 1: 3 Fragments of Packet #1 Shown
Packet 1: 1500 Bytes, Arrives First
P1 F3
P1 F2
P1 F1
Output Queue 1
P1 F3
P1 F2
2
P1 F1
Output Queue 2
Packet 2: 200 Bytes, Delay Sensitive, Arrives Second
2
R2
Legend: Px Fy Means Packet Number x, Fragment Number y

Note that packet 1 was fragmented into three pieces. Because packet 2 arrived after packet 1 had begun to be sent, packet 2 had to wait. With LFI, packet 2 does not have to wait for the entire original packet, but rather it waits for just 1 fragment to be sent.

Compression

Compression takes a packet, or packet header, and compresses the data so that it uses fewer bits. Therefore, a 1500-byte packet, compressed to 750 bytes, takes half as much serialization time as does an uncompressed 1500-byte packet.

Compression reduces serialization delay, because the number of bits used to send a packet is decreased. However, delay may also be increased because of the processing time required to compress and decompress the packets. Chapter 7 covers the pros and cons of each type of compression.

Traffic Shaping

Traffic shaping actually increases delay, in an effort to reduce the chance of packet loss. Shaping is mentioned here just because of its negative impact on delay.

Although adding more bandwidth always helps, the tools summarized in Table 1-7 do help to improve the effects of delay in a network.

Table 1-7 *QoS Tools That Affect Delay*

Type of QoS Tool	How It Affects Delay
Queuing	Enables you to order packets so that delay-sensitive packets leave their queues more quickly than delay-insensitive packets.
Link fragmentation and interleaving	Because routers do not preempt a packet that is currently being transmitted, LFI breaks larger packets into smaller fragments before sending them. Smaller delay-sensitive packets can be sent after a single smaller fragment, instead of having to wait for the larger original packet to be serialized.

Table 1-7 *QoS Tools That Affect Delay (Continued)*

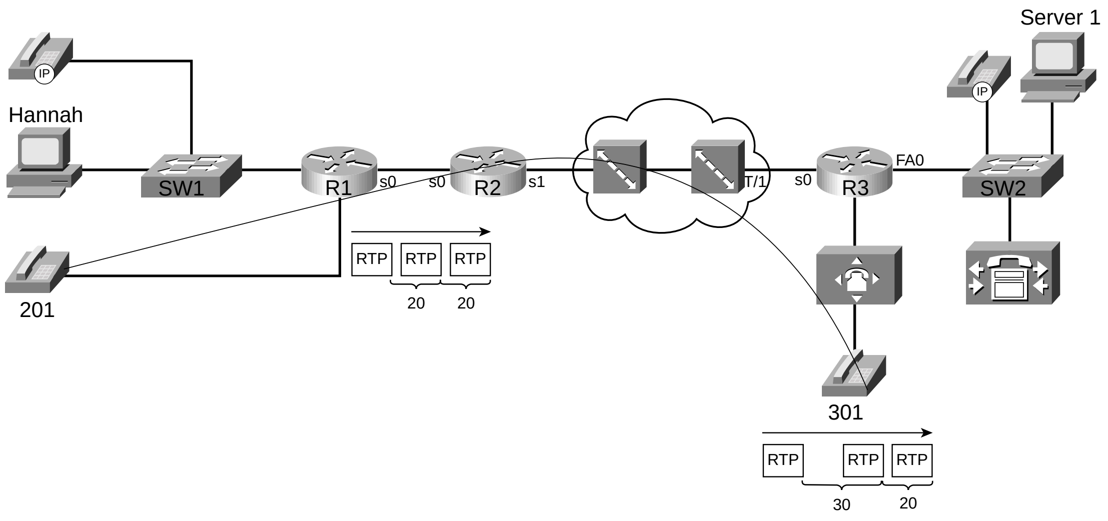
Type of QoS Tool	How It Affects Delay
Compression	Compresses either payload or headers, reducing overall number of bits required to transmit the data. By requiring less bandwidth, queues shrink, which reduces delay. Also serialization delays shrink, because fewer bits are required. Compression also adds some processing delay.
Traffic shaping	Artificially increases delay to reduce drops inside a Frame Relay or ATM network.

Jitter

Consecutive packets that experience different amounts of delay have experienced jitter. In a packet network, with variable delay components, jitter always occurs—the question is whether the jitter impacts the application enough to degrade the service. Typically, data applications expect some jitter, and do not degrade. However, some traffic, such as digitized voice, requires that the packets be transmitted in a consistent, uniform manner (for instance, every 20 ms). The packets should also arrive at the destination with the same spacing between them. (This type of traffic is called *isochronous traffic*.)

Jitter is defined as a variation in the arrival rate (that is, variation in delay through the network) of packets that were transmitted in a uniform manner. Figure 1-14, for example, shows three packets as part of a voice call between phones at extension 301 and 201.

Figure 1-14 *Jitter Example*



The phone sends the packets every 20 ms. Notice that the second packet arrived 20 ms after the first, so no jitter occurred. However, the third packet arrived 30 ms after the second packet, so 10 ms of jitter occurred.

Voice and video degrade quickly when jitter occurs. Data applications tend to be much more tolerant of jitter, although large variations in jitter affect interactive applications.

QoS Tools That Affect Jitter

Several QoS features can help with jitter issues. You'll find more detail about each of these tools in various chapters throughout this book. For now, however, knowing what each class of QoS tool accomplishes will help you sift through some of the details.

The best QoS tool for jitter issues is . . . more bandwidth—again! More bandwidth helps bandwidth-related problems, and it also helps delay-related problems. If it helps to reduce delay, and because jitter is the variation of delay, jitter will be smaller. Faster bandwidth decreases serialization delay, which will decrease jitter. For instance, if delay has been averaging between 100 ms and 200 ms, jitter would typically be up to 100 ms. If delay is reduced to between 50 ms and 100 ms by adding more bandwidth, the typical jitter can be reduced to 50 ms. Because packets exit more quickly, queuing delay decreases. If queuing delays had been between 50 and 100 ms, and now they are between 10 and 20 ms, jitter will shrink as well. In short, faster is better for bandwidth, delay, and jitter issues!

Unfortunately, more bandwidth does not solve all jitter problems, even if you could afford more bandwidth! Several classes of QoS tools improve jitter; as usual, decreasing jitter for one set of packets increases jitter for others.

The same set of tools that affect delay also affect jitter; refer to Table 1-8 for a brief list of these QoS tools.

Table 1-8 *QoS Tools That Affect Jitter*

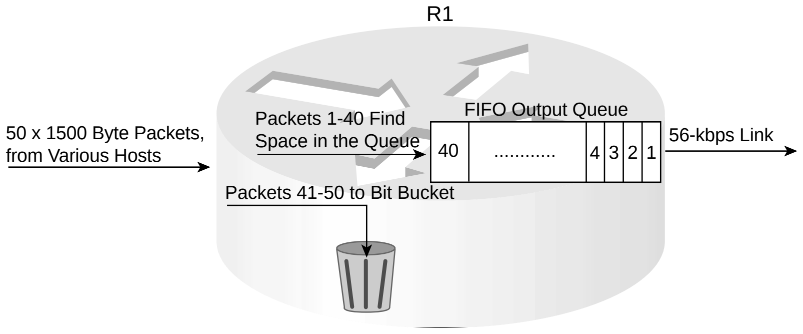
Type of QoS Tool	How It Affects Jitter
Queuing	Enables you to order packets so that delay-sensitive packets leave their queues more quickly than delay-insensitive packets.
Link fragmentation and interleaving	Because routers do not preempt a packet that is currently being transmitted, LFI breaks larger packets into smaller fragments before sending them. Smaller delay-sensitive packets can be sent after a single smaller fragment, instead of having to wait for the larger original packet to be serialized.
Compression	Compresses either payload or headers, reducing overall number of bits required to transmit the data. By requiring less bandwidth, queues shrink, which reduces delay. Also serialization delays shrink, because fewer bits are required. Compression also adds some processing delay.
Traffic shaping	Artificially increases delay to reduce drops inside a Frame Relay or ATM network.

Loss

The last QoS traffic characteristic is *packet loss*, or just *loss*. Routers lose/drop/discard packets for many reasons, most of which QoS tools can do nothing about. For instance, frames that fail the incoming frame check sequence (FCS) are discarded—period. However, QoS tools can be used to minimize the impact of packets lost due to full queues.

In most networks today, the number of packets lost due to bit errors is small, typically less than one in one billion (bit error rate [BER] of 10^{-9} or better). Therefore, the larger concern for packet loss is loss due to full buffers and queues. Consider Figure 1-15, with Hannah sending 50 consecutive 1500-byte packets, and R1 having a queue of size 40.

Figure 1-15 *50 Packets Sent, Only 40 Slots in the Queue*


R1
50 x 1500 Byte Packets, from Various Hosts
Packets 1-40 Find Space in the Queue
FIFO Output Queue
40
....
4
3
2
1
56-kbps Link
Packets 41-50 to Bit Bucket

The term “tail drop” refers to when a router drops a packet when it wants to put the packet at the end or the tail of the queue. As Figure 1-15 shows, when all 40 queue slots are filled, the rest of the 50 packets are dropped. In a real network, a few of the packets might be sent out the serial link before all 50 packets are received, so maybe not all 10 packets are lost, but certainly a large number of packets would be lost.

Some flows tolerate loss better than others do. For instance, the human ear can detect loss of only 10 ms of voice, but the listener can generally understand speech with such small loss. Cisco digital signal processors (DSPs) can predict the contents of lost voice packets, up to 30 ms when using the G.729 codec. By default, each voice packet contains 20 ms of voice; so if two consecutive voice packets are lost, the DSP cannot re-create the voice, and the receiver can actually perceive the silence. Conversely, web traffic tolerates loss well, using TCP to recover the data.

QoS Tools That Affect Loss

Only a few QoS features can help with packet loss issues. You’ll find more detail about each of these tools in various chapters throughout this book. For now, however, knowing what each class of QoS tool accomplishes will help you sift through some of the details.

By now, you should guess that bandwidth will help prevent lost packets. More bandwidth helps—it just does not solve all problems. And frankly, if all you do is add more bandwidth, and you have a converged voice/video/data network, you will still have quality issues.

How does more bandwidth reduce loss? More bandwidth allows packets to be transmitted faster, which reduces the length of queues. With shorter queues, the queues are less likely to fill. Unless queues fill, packets are not tail dropped.

You can use one class of tool to help reduce the impacts of loss. This class is called *Random Early Detection*.

Random Early Detection (RED)

TCP uses a windowing protocol, which restricts the amount of data a TCP sender can send without an acknowledgment. Each TCP window for each different TCP connection grows and shrinks based on many factors. RED works under the assumption that if some of the TCP connections can be made to shrink their windows before output queues fill, the collective number of packets sent into the network will be smaller, and the queue will not fill. During times when the queues are not getting very full, RED does not bother telling the TCP senders to slow down, because there is no need to slow down.

RED just discards some packets before a queue gets full and starts to tail drop. You can almost think of RED tools as managing the end of a queue, while the queuing tool manages the front of the queue! Because most traffic is TCP based and TCP slows down sending packets after a earlier packet is lost, RED reduces the load of packets that are entering the network before the queues fill. RED requires a fairly detailed explanation for a true understanding of what it does, and how it works. However, the general idea can be easily understood, as long as you know that TCP will slow down sending, by reducing its window size, when packets are lost.

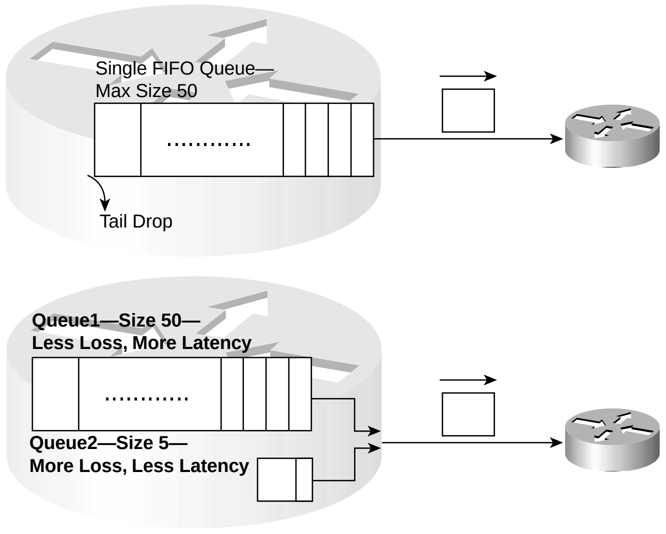
TCP uses a window, which defines how much data can be sent before an acknowledgment is received. The window changes size dynamically, based on several factors, including lost packets. When packets are lost, depending on other conditions, a TCP window shrinks to 50 percent of the previous window size. With most data traffic being TCP in a typical network, when a large amount of tail drop occurs, almost all, if not all TCP connections sending packets across that link have their TCP windows shrunk by 50 percent at least once.

Consider the example in Figure 1-15. As that figure shows, 50 packets were sent, and the queue filled, and 10 of those packets were lost. If those 50 packets were part of 10 different TCP connections, and all 10 connections lost packets in the big tail drop, the next time the hosts send, only 25 total packets would be sent (windows all cut in half).

With RED, before tail drop occurs, RED discards some packets, forcing only a few of the TCP connections to slow down. By allowing only a few of the TCP windows to be reduced, tail drops can be avoided, and most users get better response time. The collective TCP sending rate stabilizes to a level for which tail drops seldom if ever occur. For those TCP connections for which RED dropped packets, response time is temporarily slow. However, that is much better than all users experiencing slow response time!

Queuing accomplishes a lot of tasks—including reducing loss. Because loss occurs when queues fill, and because the queuing methods typically provide the ability to configure the maximum queue size, you can just make the queue longer. With a longer maximum queue size, likelihood of loss decreases. However, queuing delay increases. Consider, for instance, Figure 1-16.

Figure 1-16 *Queuing Affects on Packet Loss*



In the top example, a single FIFO queue is used. For delay-sensitive traffic, waiting behind 49 other packets might not work well. For applications that are loss sensitive, but not as delay sensitive, however, a long queue might work better. One goal might be to put delay-sensitive traffic into a different queue from loss-sensitive traffic, with an extra-long queue length for the loss-sensitive traffic, as shown in the bottom part of the figure. As usual, a tradeoff occurs—in this case, between low loss and low delay.

Table 1-9 summarizes the points concerning the two types of QoS tools that affect loss.

Table 1-9 *QoS Tools That Affect Loss*

Type of QoS Tool	Brief Description
Queuing	Implementing longer queues increases delay, but avoids loss.
RED	Implementing RED drops packets randomly as queues approach the point of being full, slowing some TCP connections. This reduces overall load, shortening the congested queue, while affecting only some user's response times.

Summary: QoS Characteristics: Bandwidth, Delay, Jitter, and Loss

This book covers a wide variety of QoS tools, and every tool either directly or indirectly affects bandwidth, delay, jitter, or loss. Some tools improve a QoS characteristic for one packet, but degrade it for others. For example, queuing tools might let one packet go earlier, reducing delay, while increasing delay for other packets. Some QoS tools directly impact one characteristic, but indirectly affect others. For instance, RED manages loss directly, but it indirectly reduces delay for some flows because RED generally causes queue sizes to decrease.

As this book explains each new feature in detail, you will also find a summary of how the feature manages bandwidth, delay, jitter, and loss.

Traffic Characteristics of Voice, Video, and Data

So why do you need QoS? QoS can affect a network's bandwidth, delay, jitter, and packet loss properties. Applications have different requirements for bandwidth, delay, jitter, and packet loss. With QoS, a network can better provide the right amounts of QoS resources for each application. Table 1-10 lists some major application categories, along with their typical needs for network resources and behavior.

Table 1-10 *Applications and Their QoS Needs*

	Bandwidth	**Delay**	**Jitter**	**Loss**
Interactive	Low	Low	Medium/high	Low
Batch	High	High	High	Low
Voice	Low	Low	Low	Low
Interactive Video	High	Low	Low	Low
1-Way Video	High	Medium/high	Low	Low
"Fragile" (For Instance, SNA*)	Low	Low	Medium/high	None

* SNA = Systems Network Architecture

The table states the QoS requirements of different applications. For instance, interactive data applications typically don't need much bandwidth, although one might argue that a lot of web applications do require a lot of bandwidth. Similarly, voice requires relatively little bandwidth, but requires low delay and low jitter; otherwise, the voice quality suffers significantly. Interactive video, such as video conferencing, has the same characteristics as voice, except it requires much more bandwidth. One-way video, however, can tolerate with long delays, so long as jitter does not exceed the jitter buffer on the receiving side. If you are watching a video over the Internet, for example, and if it takes 20 seconds to start, but it looks and sounds good for the next half hour, you really don't care that the average packet is taking 10 seconds to traverse the network.

The QoS course specifically mentions a category called *fragile*, implying that some applications are totally intolerant of packet loss. I personally would not put SNA in this category—I included it in the book, however, because it is in a table from the QoS course, which of course is the basis for the CCIP QoS Exam questions

The next three sections cover voice, video, and data flows.

Voice Traffic Characteristics

Voice traffic can degrade quickly in networks without QoS tools. This section explains enough about voice traffic flows to enable the typical reader to understand how each of the QoS tools applies to voice.

NOTE This book does not cover voice in depth because the details are not directly related to QoS. For additional information, refer to the following sources:

Deploying Cisco Voice over IP Solutions, Cisco Press, Davidson and Fox

IP Telephony, Hewlett-Packard Professional Books, Douskalis

Voice over IP Fundamentals, Cisco Press, Davidson and Peters

IP Telephone, McGraw Hill, Goralski and Kolon

www.cisco.com/warp/public/788/voip/delay-details.html)

Without QoS, the listener experiences a bad call. The voice becomes choppy or unintelligible. Delays can cause poor interactivity—for instance, the two callers keep starting to talk at the same time, because the delays sound like the other person speaking has finished what he or she had to say. Speech is lost, so that there is a gap in the sound that is heard. Calls might even be disconnected.

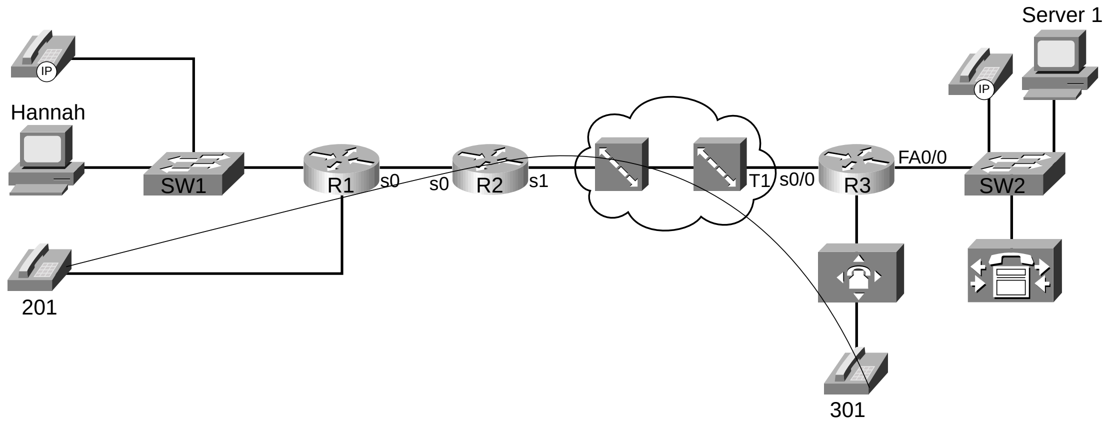
Most QoS issues can be broken into an analysis of the four QoS characteristics: bandwidth, delay, jitter, and loss. The basics of voice over data networks is covered first, followed by QoS details unique to voice in terms of the four QoS characteristics.

Voice Basics

Voice over data includes Voice over IP (VoIP), Voice over Frame Relay (VoFR), and Voice over ATM (VoATM). Each of these three voice over data technologies transports voice, and each is slightly different. Most of the questions you should see on an exam will be related to VoIP, and not VoFR or VoATM, because of the three options, VoIP is the most pervasive. Also calls between Cisco IP Phones use VoIP, not VoFR or VoATM.

Imagine a call between the two analog phones in Figure 1-17, extensions 201 and 301.

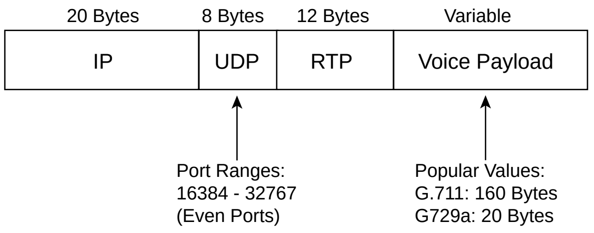
Figure 1-17 *Call Between Analog Phones at Extensions 301 and 201*



Before the voice can be heard at the other end of the call, several things must happen. Either user must pick up the phone and dial the digits. The router connected to the phone interprets the digits and uses signaling to set up the VoIP call. (Because both phones are plugged into FXS analog ports on R1 and R3, the routers use H.323 signaling.) At various points in the signaling process, the caller hears ringing, and the called party hears the phone ringing. The called party picks up the phone, and call setup is complete.

The actual voice call (as opposed to signaling) uses Real-Time Transport Protocol (RTP). Figure 1-18 outlines the format of an IP packet using RTP.

Figure 1-18 *IP Packet for Voice Call: RTP*



In the call between the two analog phones, the router collects the analog voice, digitizes the voice, encodes the voice using a voice codec, and places the encoded voice into the payload field shown in Figure 1-18. For instance, R1 would create an IP packet as shown in Figure 1-18, place the encoded voice bits into the voice payload field, and send the packet. The source IP address would be an IP address on R1, and the destination IP address would be an IP address on R3. When R3 receives the packet, it reverses the process, eventually playing the analog waveform for the voice out to the analog phone.

The IP Phones would experience a similar process in concept, although the details differ. The signaling process includes the use of Skinny Station Control Protocol (SSCP), with flows between each phone and the Cisco CallManager server. After signaling has completed, an RTP flow has been completed between the two phones. CallManager does not participate directly in the actual call, but only in call setup and teardown. (CallManager does maintain a TCP connection to each phone for control function support.) R1 and R3 do not play a role in the creation of the RTP packets on behalf of the IP Phone, because the IP Phones themselves create the packets. As far as R1 and R3 are concerned, the packets sent by the IP Phones are just IP packets.

Finally, the network administrator can choose from various coders/decoders (codecs) for the VoIP calls. Codecs process the incoming analog signal and convert the signal into a digital (binary) signal. The actual binary values used to represent the voice vary based on which codec is used. Each codec has various features, the most significant feature being the amount of bandwidth required to send the voice payload created by the codec. Table 1-11 lists the most popular codecs, and the bandwidth required for each.

Table 1-11 *Popular Voice Codecs and Payload Bandwidth Requirements*

Codec	Bit Rate for Payload* (in kbps)	Size of payload (20-ms Default in Cisco IOS Software)
G.711 Pulse Code Modulation (PCM)	64	160 bytes
G.726 ADPCM	32	80 bytes
G.729	8	20 bytes
G.723.1 ACELP	5.3	20 bytes**

* The payload contains the digitized voice, but does not include headers and trailers used to forward the voice traffic.

** G.723 defaults to a 30-ms payload per packet.

This short section on voice basics (and yes, it is very basic!) can be summarized as follows:

- Various voice signaling protocols establish an RTP stream between the two phones, in response to the caller pressing digits on the phone.
- RTP streams transmit voice between the two phones (or between their VoIP gateways).

Why the relatively simple description of voice? All voice payload flows need the same QoS characteristics, and all voice signaling flows collectively need another set of QoS characteristics. While covering each QoS tool, this book suggests how to apply the tool to "voice"—for two subcategories, namely voice payload (RTP packets) and voice signaling. Table 1-12 contrasts the QoS requirements of voice payload and signaling flows.

Table 1-12 *Comparing Voice Payload to Voice Signaling: QoS Requirements*

	Bandwidth	**Delay**	**Jitter**	**Loss**
Voice Payload	Low	Low	Low	Low
Voice Signaling	Low	Low	Medium	Medium

QoS tools can treat voice payload differently than they treat voice signaling. To do so, each QoS tool first classifies voice packets into one of these two categories. To classify, the QoS tool needs to be able to refer to a field in the packet that signifies that the packet is voice payload, voice signaling, or some other type of packet. Table 1-13 lists the various protocols used for signaling and for voice payload, defining documents, and identifying information.

Table 1-13 *Voice Signaling and Payload Protocols*

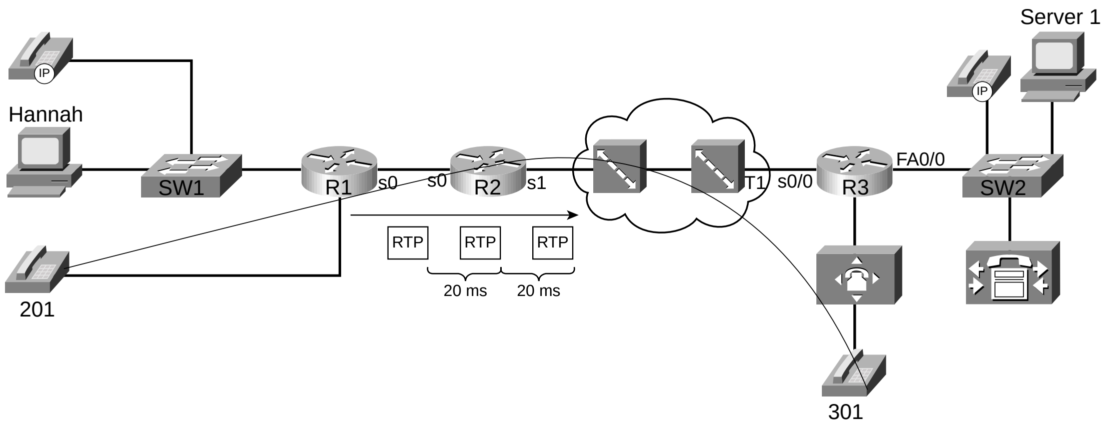
Protocol	**Documented By**	**Useful Classification Fields**
H.323/H.225	ITU	Uses TCP port 1720
H.323/H.245	ITU	TCP ports 11xxx
H.323/H.245	ITU	TCP port 1720 (Fast Connect)
H.323/H.225 RAS	ITU	TCP port 1719
Skinny	Cisco	TCP ports 2000-2002
Simple Gateway Control Protocol (SGCP)		TCP ports 2000-2002
Media Gateway Control Protocol (MGCP)	RFC 2705	UDP port 2427, TCP port 2428
Intra-Cluster Communications Protocols (ICCP)	Cisco	TCP ports 8001–8002
Real-Time Transport Protocol (RTP)	RFC 1889	UDP ports 16384–32767, even ports only
Real-Time Control Protocol (RTCP)	RFC 1889	UDP ports 16385–32767, odd ports only; uses RTP port + 1

The next few sections of this book examine voice more closely in relation to the four QoS characteristics: bandwidth, delay, jitter, and loss.

Voice Bandwidth Considerations

Voice calls create a flow with a fixed data rate, with equally spaced packets. Voice flows can be described as *isochronous*, which, according to Dictionary.com, means “characterized by or occurring at equal intervals of time.” Consider Figure 1-19, where a call has been placed between analog phones at extensions 201 and 301.

Figure 1-19 *Isochronous Packet Flow for Voice Call*



R1 creates the IP/UDP/RTP voice payload packets and sends them, by default, every 20 ms. Because Cisco IOS Software places 20 ms of encoded voice into each packet, a packet must be sent every 20 ms. So, how much bandwidth is really needed for the voice payload call? Well, actual bandwidth depends on several factors:

- Codec
- Packet overhead (IP/UDP/RTP)
- Data-link framing (depends on data links used)
- Compression

Most people quote a G.711 call as taking 64 kbps, and a G.729 call as taking 8 kbps. Those bandwidth numbers consider the payload only—ignoring data-link, IP, UDP, and RTP headers. Consider Table 1-14, with actual bandwidth requirements for various types of voice calls.

Table 1-14 *Bandwidth Requirements with Various Data-Link Types*

L2 Header Type	L2 Header Size	IP/UDP/RTP Header Size	Codec	Payload Bandwidth	Total Bandwidth
Ethernet	14	40 bytes	G.711	64 kbps	85.6
MLPPP/FR	6	40 bytes	G.711	64 kbps	82.4
Ethernet	14	40 bytes	G.729	8 kbps	29.6
MLPPP/FR	6	40 bytes	G.729	8 kbps	26.4

* For DQOS test takers: These numbers are extracted from the DQOS course.

The bandwidth requirements vary dramatically based on the codec and the compression effect if RTP header compression is used. Compressed RTP (cRTP) actually compresses the IP/UDP/RTP headers, with dramatic reduction in bandwidth when used with lower bit-rate codecs. With G.711, because a large percentage of the bandwidth carries the payload, cRTP helps, but the percentage decrease in bandwidth is not as dramatic. In either case, cRTP can increase delay caused while the processor compresses and decompresses the headers.

NOTE Although other codecs are available, this book compares G.711 and G.729 in most examples, noting any conditions where a different specific codec may need different treatment with QoS.

ATM can add a significant amount of data-link overhead to voice packets. Each ATM cell has 5 bytes of overhead; in addition, the last cell holding parts of the voice packet may have a lot of wasted space. For instance, a voice call using G.729 will have a packet size of 60 bytes. ATM adds about 8 bytes of framing overhead, and then segments the 68-byte frame into two cells—one using the full 48 bytes of ATM cell payload, and the other only using 20 bytes of the cell payload area—with 28 bytes "wasted." Therefore, to send one voice "packet" of 60 bytes, two cells, in total 106 bytes, must be sent over ATM. One option to lessen the overhead is to change the payload size to contain 30 ms of voice with a G.729 codec—which interestingly also only takes two ATM cells.

Voice Activity Detection (VAD) also affects the actual bandwidth used for a voice payload call. VAD causes the sender of the voice packets to not send packets when the speaker is silent. Because human speech is typically interactive (I know there are some exceptions to that rule that come to mind right now!), VAD can decrease the actual bandwidth by about 60 percent. The actual amount of bandwidth savings for each call cannot be predicted—simple things such as calling from a noisy environment defeats VAD. Also VAD can be irritating to listeners. After a period of not speaking, the speaker starts to talk. The VAD logic may perform front-end speech clipping, which means that the first few milliseconds of voice are not sent.

Note that the numbers shown in the previous table, Table 1-14, come directly from the DQOS course (and several other sources inside Cisco). Interestingly, the numbers ignore data-link trailer overhead, and ignore ATM and 802.1Q framing. One of the technical editors of this book graciously provided an alternative, more accurate table that you should consider using for planning in production networks. For the exam, trust the numbers in Table 1-14. Table 1-15 lists the updated numbers.

Table 1-15 *Updated Bandwidth Requirements for Various Types of Voice Calls*

Bandwidth Consumption, Including L2 Overhead	802.1Q Ethernet (32 Bytes of L2 Overhead)	PPP (9 Bytes of L2 Overhead)	MLP (13 Bytes of L2 Overhead)	Frame-Relay (8 Bytes of L2 Overhead)	ATM (Variable Bytes of L2 Overhead, Depending on Cell-Padding Requirements)
G.711 at 50 pps*	93 kbps	84 kbps	86 kbps	84 kbps	106 kbps
G.711 at 33 pps	83 kbps	77 kbps	78 kbps	77 kbps	84 kbps
G.729A at 50 pps	37 kbps	28 kbps	30 kbps	28 kbps	43 kbps
G.729A at 33 pps	27 kbps	21 kbps	22 kbps	21 kbps	28 kbps

* pps = packets per second

Voice Delay Considerations

Voice call quality suffers when too much delay occurs. The symptoms include choppy voice, and even dropped calls. Interactivity also becomes difficult—ever had a call on a wireless phone, when you felt like you were talking on a radio? “Hey Fred, let’s go bowling—OVER”—“Okay, Barney, let’s go while Betty and Wilma are out shopping—OVER.” With large delays, it sometimes becomes difficult to know when it is your turn to talk.

Voice traffic experiences delays just like any other packet, and that delay originates from several other sources. For a quick review on delay components covered so far, consider the delay components listed in Table 1-16.

Table 1-16 *Components of Delay Not Specific to One Type of Traffic*

Delay Component	Definition	Where It Occurs
Serialization delay	Time taken to place all bits of a frame onto the physical medium. Function of frame size and physical link speed.	Outbound on every physical interface; typically negligible on T3 and faster links.
Propagation delay	Time taken for a single bit to traverse the physical medium from one end to the other. Based on the speed of light over that medium, and the length of the link.	Every physical link. Typically negligible on LAN links and shorter WAN links.
Queuing delay	Time spent in a queue awaiting the opportunity to be forwarded (output queuing), or awaiting a chance to cross the switch fabric (input queuing).	Possible on every output interface. Input queuing unlikely in routers, more likely in LAN switches.

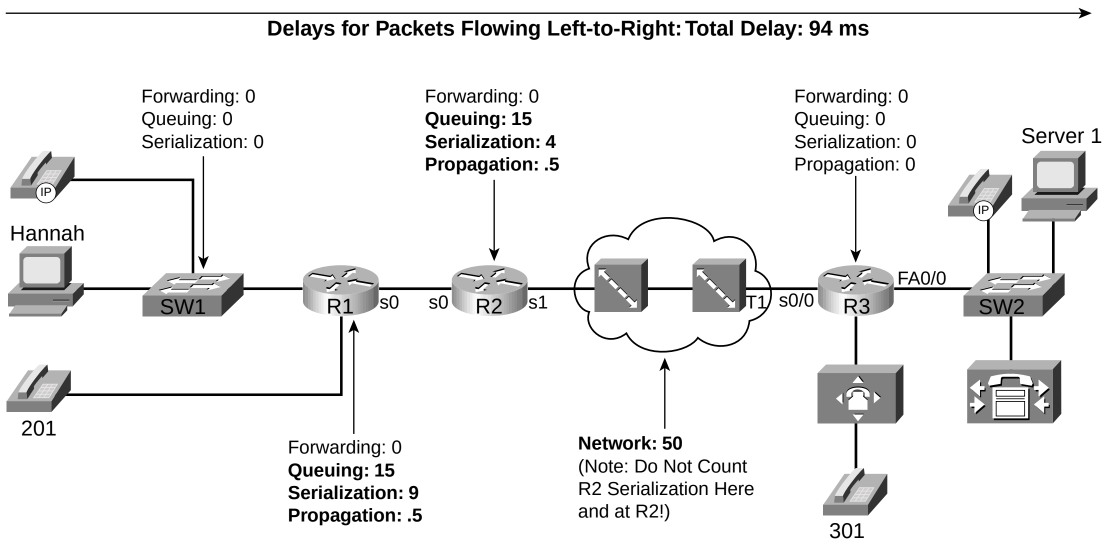
continues

Table 1-16 *Components of Delay Not Specific to One Type of Traffic (Continued)*

Delay Component	Definition	Where It Occurs
Forwarding or processing delay	Time required from receipt of the incoming frame until the frame/packet has been queued for transmission.	On every piece of switching equipment, including routers, LAN switches, Frame Relay switches, and ATM switches.
Shaping delay	Shaping (if configured) delays transmission of packets to avoid packet loss in the middle of a Frame Relay or ATM network.	Anywhere that shaping is configured, which is most likely on a router, when sending packets to a Frame Relay or ATM network.
Network delay	Delays created by the components of the carrier's network when using a service. For instance, the delay of a Frame Relay frame as it traverses the Frame Relay network.	Inside the service provider's network.

Figure 1-20 shows an example of delay concepts, with sample delay values shown. When the delay is negligible, the delay is just listed as zero.

Figure 1-20 *Example Network with Various Delay Components Shown: Left-to-Right Directional Flow*



The figure lists sample delay values. The values were all made up, but with some basis in reality. Forwarding delays are typically measured in microseconds, and become negligible. The propagation delay from R1 to R2 is calculated based on a 100-km link. The serialization delays shown were calculated for a G.729 call's packet, no compression, assuming PPP as the data-link protocol. The queuing delay varies greatly; the example value of 15 ms on R1's 56-kbps link was based on assuming a single 105-byte frame was enqueued ahead of the packet whose delay we are tracking—which is not a lot of queuing delay. The network delay of 50 ms was made up—but that is a very reasonable number. The total delay is only 94 ms—to data network engineers, the delay seems pretty good.

So is this good enough? How little delay does the voice call tolerate? The ITU defines what it claims to be a reasonable one-way delay budget. Cisco has a slightly different opinion. You also may have applications where the user tolerates large delays to save cash. Instead of paying $3 per minute for a quality call to a foreign country, for instance, you might be willing to tolerate poor quality if the call is free. Table 1-17 outlines the suggested delay budgets.

Table 1-17 *One-Way Delay Budget Guidelines*

1-Way Delay (in ms)	Description
0–150	ITU G.114's recommended acceptable range
0–200	Cisco's recommended acceptable range
150–400	ITU G.114's recommended range for degraded service
400+	ITU G.114's range of unacceptable delay in all cases

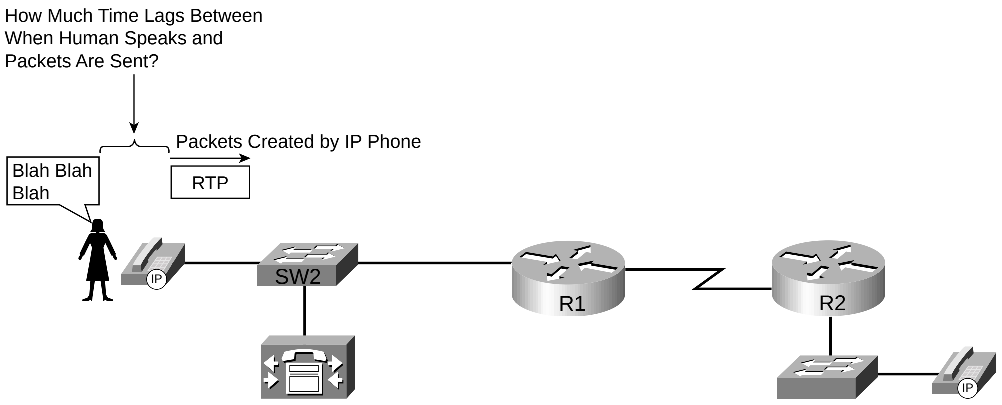
With the example in Figure 1-20, the voice call's delay fits inside the G.114 recommended delay budget. However, voice traffic introduces a few additional delay components, in addition to the delay factors that all data packets experience:

- Codec delay
- Packetization delay
- De-jitter buffer delay (initial playout delay)

Be warned—many books and websites use different terms to refer to the component parts of these three voice-specific types of delay. The terms used in this book are consistent with the Cisco courses, and therefore with the exams.

Codec delay and packetization delay coincide with each other. To get the key concepts of both, consider Figure 1-21, which asks the question, "How much delay happens between when the human speaks, and when the IP packets are sent?"

Figure 1-21 *Codec and Packetization Delays Between the Instant the Speaker Speaks and When the Packet Holding the Speech Is Sent*


How Much Time Lags Between When Human Speaks and Packets Are Sent?
Packets Created by IP Phone
Blah Blah Blah
RTP
IP
SW2
R1
R2
IP

Consider what has to happen at the IP Phones before a packet can be sent. The caller dials digits, and the call is set up. When the call is set up, the IP Phone starts sending RTP packets. When these packets begin, they are sent every 20 ms (default)—in other words, each packet has 20 ms of voice inside the voice payload part of the packet. But how much time passes between when the speaker makes some sound and when the voice packet containing that sound is first sent?

Consider sounds waves for an instant. If you and I sit in the same room and talk, the delay from when you speak and when I hear it is very small, because your voice travels at the speed of sound, which is roughly 1000 km per hour. With packet telephony, the device that converts from sound to analog electrical signals, then to digital electrical signals, and then puts that digital signal (payload) into a packet, needs time to do the work. So there will be some delay between when the speaker speaks and when the IP/UDP/RTP payload packet is sent. In between when the speaker talks and when a packet is sent, the following delays are experienced:

- Packetization delay
- Codec delay

Packetization Delay

The IP Phone or voice gateway must collect 20 ms of voice before it can put 20 ms worth of voice payload into a packet. (The defaults for G.711 and G.729 on IP Phones and Cisco IOS Software gateways are to put 20 ms of voice into an RTP packet; the value can be changed.) Therefore, for the sake of discussion in this book, we consistently consider packet delay always to be 20 ms in examples. That is, the speaker must talk for 20 ms before a packet containing 20 ms of voice can be created.

Codec Delay

Codec delay has two components:

- The time required to process an incoming analog signal and convert it to the correct digital equivalent
- A feature called *look-ahead*

With the first component of codec delay, which is true for all codecs, the IP Phone or gateway must process the incoming analog voice signal and encode the digital equivalent based on the codec in use. For instance, G.729 processes 10 ms of analog voice at a time. That processing does take time—in fact, the actual conversion into G.729 CS-ACELP (Conjugate Structure Algebraic Code Excited Linear Predictive) takes around 5 ms. Some documents from Cisco cite numbers between 2.5 and 10 ms, based on codec load.

The codec algorithm may cause additional delays due to a feature called *look-ahead*. Look-ahead occurs when the codec is predictive—in other words, one method of using fewer bits to encode voice takes advantage of the fact that the human vocal cords cannot instantaneously change from one sound to a very different sound. By examining the voice speech, and knowing that the next few ms of sound cannot be significantly different, the algorithm can use fewer bits to encode the voice—which is one way to improve from 64-kbps G.711 to an 8-kbps G.729 call. However, a predictive algorithm typically requires the codec to process some voice signal that will be encoded, plus the next several milliseconds of voice. With G.729, for example, to process a 10-ms voice sample, the codec must have all of that 10 ms of voice, plus the next 5 ms, in order to process the predictive part of the codec algorithm.

So, the G.729a codec delays the voice as follows, for each 10-ms sample, starting with the time that the 10 ms to be encoded has arrived:

> 5 ms look-ahead + 5 ms processing (average) = 10 ms

Remember, codec delay is variable based on codec load. The white paper, "Understanding Delay in Packet Voice Networks," at Cisco.com provides more information about this topic (www.cisco.com/warp/public/788/voip/delay-details.html).

Considering the Effects of Packetization and Codec Delay

You need to consider packetization delay and codec delay together, because they do overlap. For instance, packetization delay consumes 20 ms while waiting on 20 ms of voice to occur. But what else happens in the first 20 ms? Consider Table 1-18, with a timeline of what happens, beginning with the first uttered sound in the voice call.

Table 1-18 *Typical Packetization and Codec Delay Timeline—G.729*

Timeline	Action	Codec Delay	Packetization Delay
T=0	Begin collecting voice samples for A/D conversion, encoding	Not begun yet (by this book's definition)	Begins
T=10	Collected complete 10-ms sample, which will be encoded with G.729	Codec delay begins	10 ms so far; packetization delay continues
T=15	Collected first 5 ms of second 10-ms sample	5 ms so far; G.729 now has the 5-ms look-ahead that the algorithm needs to encode first 10 ms sample	15 ms so far; packetization delay continues
T=20	Finished collecting second 10-ms sample	10 ms so far; codec delay finished for first 10-ms sample; second 10-ms sample in memory; codec delay for second sample begins	Packetization delay complete at 20 ms, because 20 ms of voice has been collected
T=25	Collected first 5 ms of third 10-ms sample	5-ms delay so far on second 10-ms sample; 15 ms total; G.729 now has the 5-ms look-ahead that the algorithm needs to encode second 10-ms sample	Finished with packetization delay; 20 ms voice has been received
T=30	Finished collecting third 10-ms sample	20 ms total codec delay; RTP and payload ready to be sent	Finished. 20 ms total
Total delays for first packet		20 ms	20 ms

Notice that the packetization and codec delays overlap. Although each takes 20 ms, because there is overlap, the packet actually experiences about 30 ms of total delay instead of a total of 40 ms.

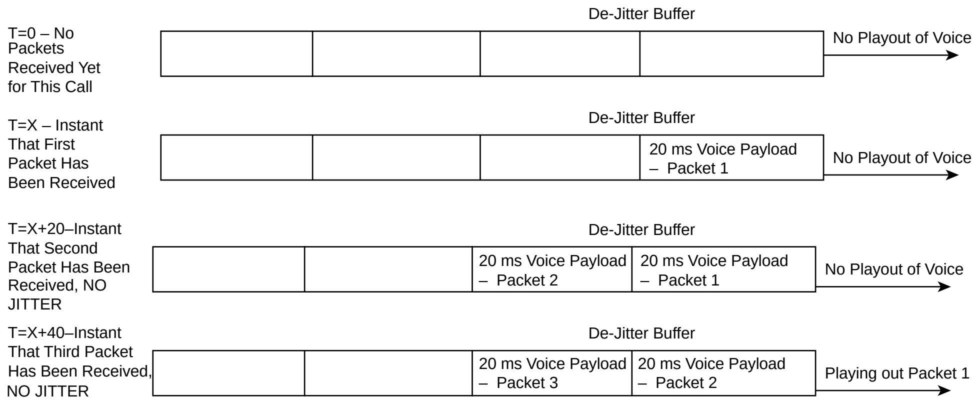
De-Jitter Buffer Delay

De-jitter buffer delay is the third voice delay component. Jitter happens in data networks. You can control it, and minimize it for jitter-sensitive traffic, but you cannot eliminate it. Buy why talk about jitter in the section on delay? Because a key tool in defeating the effects of jitter, the de-jitter buffer (sometimes called the *jitter buffer*) actually increases delay.

The de-jitter buffer collects voice packets and delays playing out the voice to the listener, to have several ms of voice waiting to be played. By doing so, if the next packet experiences jitter and shows up late, the de-jitter buffer's packets can be played out isochronously, so the voice sounds good. This is the same tool used in your CD player in your car—the CD reads ahead several seconds, knowing that your car will hit bumps, knowing that the CD temporarily will not be readable—but having some of the music in solid-state memory lets the player continue to play the music. Similarly, the de-jitter buffers "reads ahead" by collecting some voice before beginning playout, so that delayed packets are less likely to cause a break in the voice.

The de-jitter buffer must be filled before playout can begin. That delay is called the *initial playout delay* and is depicted in Figure 1-22.

Figure 1-22 *De-Jitter Buffer Initial Playout Delay, No Jitter in First Three Packets*


De-Jitter Buffer
T=0 – No Packets Received Yet for This Call
No Playout of Voice
De-Jitter Buffer
T=X – Instant That First Packet Has Been Received
20 ms Voice Payload – Packet 1
No Playout of Voice
De-Jitter Buffer
T=X+20–Instant That Second Packet Has Been Received, NO JITTER
20 ms Voice Payload – Packet 2
20 ms Voice Payload – Packet 1
No Playout of Voice
De-Jitter Buffer
T=X+40–Instant That Third Packet Has Been Received, NO JITTER
20 ms Voice Payload – Packet 3
20 ms Voice Payload – Packet 2
Playing out Packet 1

In this figure, the de-jitter buffer shows the initial playout delay. The time difference between when the initial packet arrives, and when the third packet arrives, in this particular case, is 40 ms. (Cisco IOS gateways default to 40 ms of initial playout delay.) In fact, if the initial playout delay were configured for 40 ms, this delay would be 40 ms, regardless of when the next several packets arrive. Consider, for instance, Figure 1-23, which gives a little insight into the operation of the de-jitter buffer.

In Figure 1-23, the playout begins at the statically set playout delay interval—40 ms in this case—regardless of the arrival time of other packets. A 40-ms de-jitter playout delay allows jitter to occur—because we all know that jitter happens—so that the played-out voice can continue at a constant rate.

Figure 1-23 *De-Jitter Buffer Initial Playout Delay, 10 ms Jitter for Third Packet*

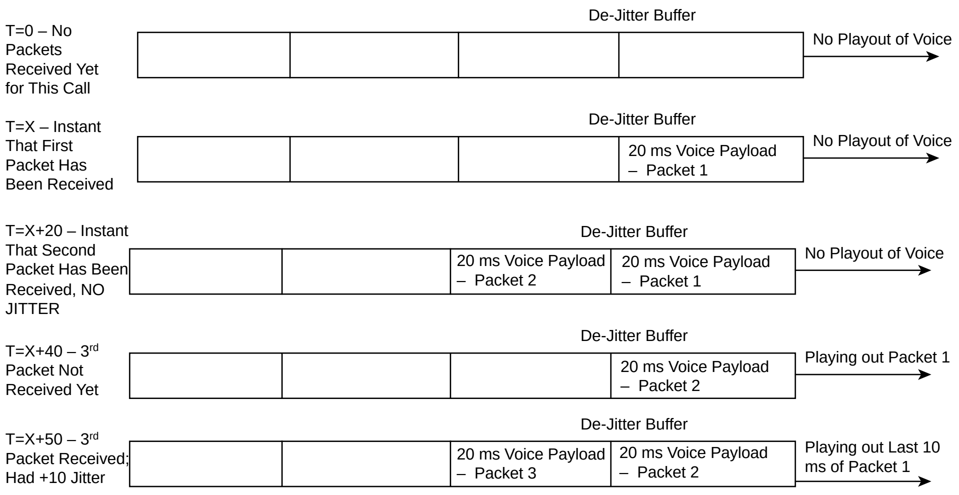
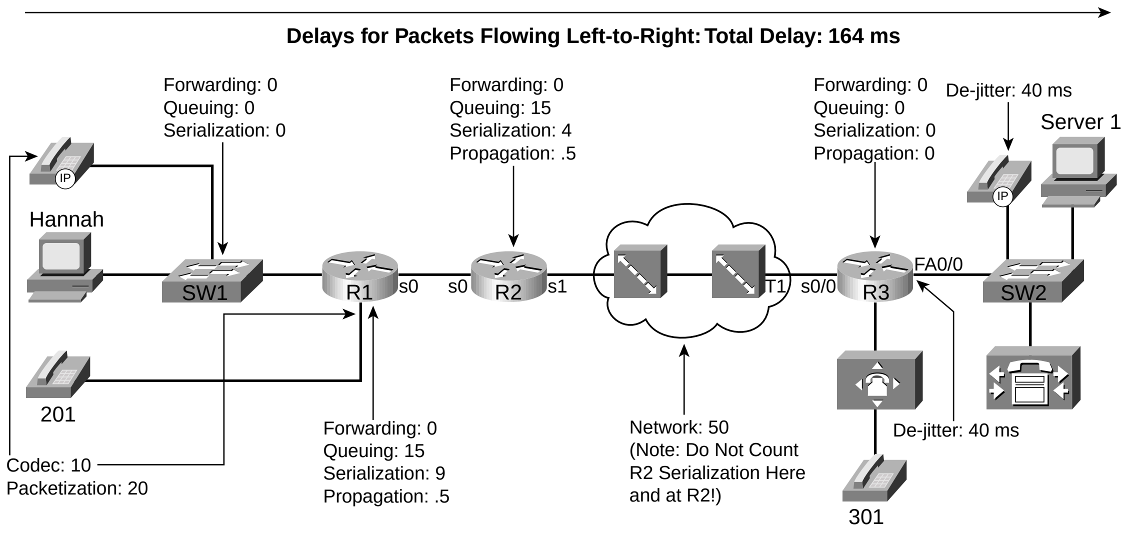

T=0 – No Packets Received Yet for This Call
De-Jitter Buffer
No Playout of Voice
T=X – Instant That First Packet Has Been Received
De-Jitter Buffer
20 ms Voice Payload – Packet 1
No Playout of Voice
T=X+20 – Instant That Second Packet Has Been Received, NO JITTER
De-Jitter Buffer
20 ms Voice Payload – Packet 2
20 ms Voice Payload – Packet 1
No Playout of Voice
T=X+40 – 3rd Packet Not Received Yet
De-Jitter Buffer
20 ms Voice Payload – Packet 2
Playing out Packet 1
T=X+50 – 3rd Packet Received; Had +10 Jitter
De-Jitter Buffer
20 ms Voice Payload – Packet 3
20 ms Voice Payload – Packet 2
Playing out Last 10 ms of Packet 1

Figure 1-24 summarizes all the delay components for a voice call. This figure repeats the same example delay values as did Figure 1-20, but with voice-specific delays added for codec, packetization, and de-jitter delays shown.

Figure 1-24 *Complete End-to-End Voice Delay Example*



The delay has crept beyond the acceptable limits of one-way delay, according to G.114, but slightly under the limit of 200 ms suggested by Cisco. Without the additional voice delays, the 150-ms delay budget seemed attainable. With 30 ms of codec and packetization delay, however, and a (reasonable) default of 40-ms de-jitter delay (actually, de-jitter initial playout delay), 70 ms of that 150/200-ms delay is consumed. So, what can you do to stay within the desired delay budget? You attack the variable components of delay. Table 1-19 lists the different delay components, and whether they are variable.

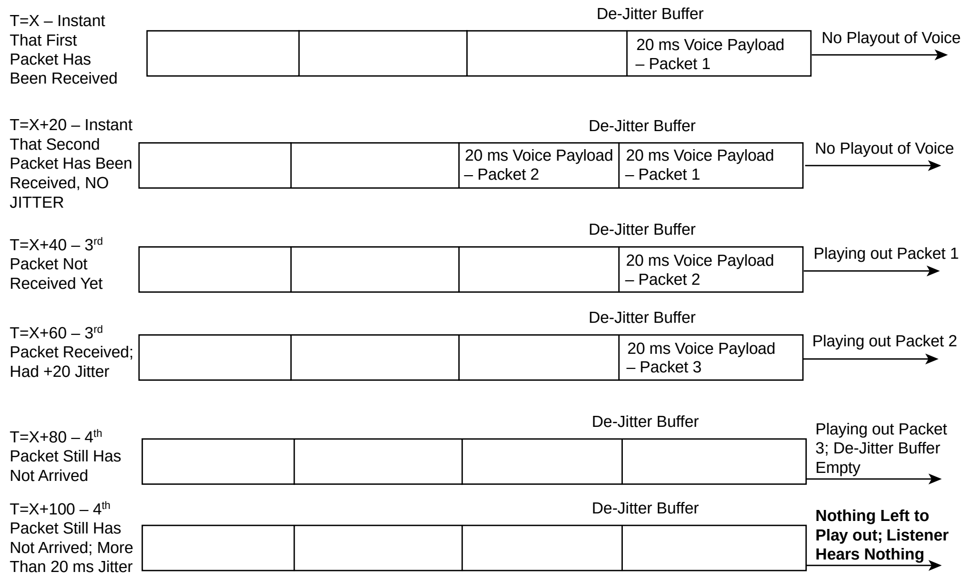
Table 1-19 *Delay Components, Variable and Fixed*

Delay Component	Fixed or Variable	Comments	QoS Tools That Can Help
Codec	Fixed	Varies slightly based on codec and processing load; considered fixed in course books (and probably on exams). Typically around 10 ms.	None.
Packetization	Fixed	Some codecs require a 30-ms payload, but packetization delay does not vary for a single codec. Typically 20 ms, including when using G.711 and G.729.	None.
Propagation	Variable	Varies based on length of circuit. About 5 ms/100 km	Move your facilities to the same town.
Queuing	Variable	This is the most controllable delay component for packet voice	Queuing features, particularly those with a priority-queuing feature.
Serialization	Fixed	It is fixed for voice packets, because all voice packets are of equal length. It is variable based on packet size for all packets.	Fragmentation and compression.
Network	Variable	Least controllable variable component.	Shaping, fragmentation, designs mindful of reducing delay.
De-jitter buffer (initial playout delay)	Variable	This component is variable because it can be configured for a different value. However, that value, once configured, remains fixed for all calls until another value is configured. In other words, the initial playout delay does not dynamically vary.	Configurable playout delay in IOS gateways; not configurable in IP Phones.

Voice Jitter Considerations

The previous section about delay explained most of the technical details of voice flows relating to jitter. If jitter were to cause no degradation in voice call performance, it would not be a problem. However, jitter causes hesitation in the received speech pattern and lost sounds, both when the jitter increases quickly and when it decreases quickly. For instance, consider Figure 1-25, where packets 3 and 4 experience jitter.

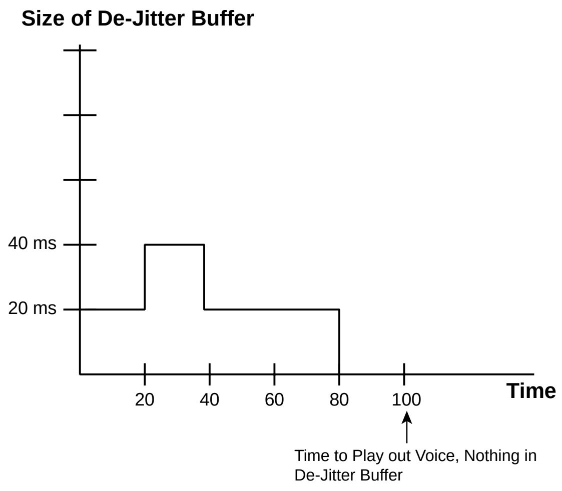
Figure 1-25 *De-Jitter Buffer Underrun Due to Jitter*


T=X – Instant That First Packet Has Been Received
De-Jitter Buffer
20 ms Voice Payload – Packet 1
No Playout of Voice
T=X+20 – Instant That Second Packet Has Been Received, NO JITTER
De-Jitter Buffer
20 ms Voice Payload – Packet 2
20 ms Voice Payload – Packet 1
No Playout of Voice
T=X+40 – 3rd Packet Not Received Yet
De-Jitter Buffer
20 ms Voice Payload – Packet 2
Playing out Packet 1
T=X+60 – 3rd Packet Received; Had +20 Jitter
De-Jitter Buffer
20 ms Voice Payload – Packet 3
Playing out Packet 2
T=X+80 – 4th Packet Still Has Not Arrived
De-Jitter Buffer
Playing out Packet 3; De-Jitter Buffer Empty
T=X+100 – 4th Packet Still Has Not Arrived; More Than 20 ms Jitter
De-Jitter Buffer
Nothing Left to Play out; Listener Hears Nothing

In Figure 1-25, the second packet experiences the same delay as the first packet. How can you tell? The IP Phone or Cisco IOS gateway sends the packets every 20 ms; if they arrive 20 ms apart, the delay for each packet is the same, meaning no jitter. However, packet 3 arrives 40 ms after packet 2, which means packet 3 experienced 20 ms of jitter. Packet 4 does not arrive until 45 seconds later than packet 3; because packet 4 was sent 20 ms after packet 3, packet 4 experienced 25 ms of jitter. As a result, the de-jitter buffer empties, and there is a period of silence. In fact, after packet 4 shows up, the receiver discards the packet, because playing the voice late would be worse than a short period of silence.

Another way to visualize the current size of the de-jitter buffer is to consider the graph in Figure 1-26. The packet arrivals in the figure match Figure 1-25, with the size of the de-jitter buffer shown on the y-axis.

Figure 1-26 *De-Jitter Buffer Underrun Graph*



What caused the jitter? Variable delay components. The two most notorious variable delay components are *queuing delay* and *network delay*. Queuing delay can be reduced and stabilized for voice payload packets by using a queuing method that services voice packets as soon as possible. You also can use LFI to break large data packets into smaller pieces, allowing voice to be interleaved between the smaller packets. And finally, Frame Relay and ATM networks that were purposefully oversubscribed need to be redesigned to reduce delay. Jitter concepts relating to voice, and QoS, can be summarized as follows:

- Jitter happens in packet networks.
- De-jitter buffers on the receiving side compensate for some jitter.
- QoS tools, particularly queuing and fragmentation tools, can reduce jitter to low enough values such that the de-jitter buffer works effectively.
- Frame Relay and ATM networks can be designed to reduce network delay and jitter.

Author’s note: IP Phones display statistics for jitter if you press the information (“i”) button on the phone.

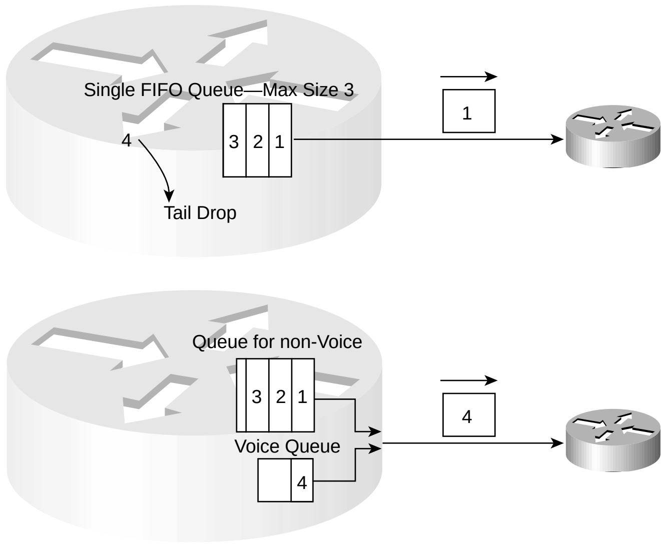
Voice Loss Considerations

Routers discard packets for a variety of reasons. The two biggest reasons for packet loss in routers are

- Bit errors
- Lack of space in a queue

QoS cannot help much with bit errors. However, QoS can help a great deal with loss due to queue space. Figure 1-27 contrasts FIFO queuing (one queue) with a simple queuing method with one queue for voice payload, and another for everything else.

Figure 1-27 *FIFO Queuing Versus Imaginary Two-Queue System (One Queue for Voice, One for Everything Else)*



Suppose, for instance, that four packets arrive almost instantaneously, numbered 1 through 4, with packet 1 being the first to arrive. In the FIFO scheme, the router places the packets into the FIFO output queue, in the same order as arrival. What happens, however, if the output queue only has space for three packets, as shown in the figure? Well, the fourth packet is tail dropped.

Now suppose that the fourth packet is a voice packet, and the two-queue system is used. Each queue has space for three packets. In this case, the router does not drop the voice packet (packet 4). In fact, assuming the router serves the voice queue so that any packets always get to be sent next, this router reduces the delay for the voice packet.

NOTE A queue of size 3 is too small; however, showing a queue of size 40 just seemed to be a little clumsy in the figure!

With the simple example in Figure 1-27, the router does not drop the voice packet. However, the real power of this two-queue system for avoiding lost voice packet shines through with a little closer examination. Suppose that CAC allows only two concurrent G.729a calls to run through this router, and suppose the router does not use cRTP. The bandwidth required would be 26.4 kbps for each call, or a total of 52.8 kbps. Now imagine that the queuing method always sends the voice packets at next opportunity when a voice packet arrives, only waiting for the "currently being sent" packet to finish. Also imagine that the queuing method guarantees at least 60 kbps of this 128-kbps link for the voice queue. With all these features, the voice queue should never get very long (assuming the following parameters):

- The correct choices for maximum queue length.
- Queuing that always takes voice packets at first opportunity.
- Call admission control that prevents too many voice calls.
- LFI that shrinks the time a voice packet must wait for the packet ahead of it to be serialized.
- The voice queue would never fill and voice packets would not be tail dropped on this interface.

Another type of QoS tool, call admission control (CAC), provides a very important component of a strategy to avoid packet loss, and thereby improve voice quality. The best router-based queuing tools for voice include a setting for the maximum amount of bandwidth used by the voice queue. If exceeded, when the interface has other packets waiting, the excess voice packets are discarded. Literally, adding one call too many can make what was a large number of quality voice calls *all* degrade to poor quality. Some form of CAC must be considered to avoid loss for all calls.

Finally, one additional feature helps when a single voice payload packet is lost. G.729 codecs compress the voice payload in part by predicting what the next few milliseconds of voice will look like. G.729 uses this same logic to perform a function called *autofill* when converting from digital to analog at the receiving side of the call. G.729 autofill makes an educated guess at what the next few milliseconds of sound would have been if the next packet in sequence has been lost. Autofill makes this educated guess, filling in up to 30 ms of "lost" voice. Because IP Phone and IOS gateways default to sending 20 ms of voice per packet, with G.729, a single packet can be lost, and the G.729 autofill algorithm will play out a best guess as to what was in the missing voice packet.

Loss considerations for voice can be summarized as follows:

- Routers drop packets because of many reasons; the most controllable reason is tail drop due to full queues.
- Queuing methods that place (isochronous) voice into a different queue than bursty data reduce the chance that voice packets will be tail dropped.
- The QoS tools that help voice already, particularly queuing and LFI, reduce the chance of the voice queue being full, thereby avoiding tail drop.

- Whereas other QoS tools protect voice from other types of packets, CAC protects voice from voice.
- Single voice packet loss, when using G.729, can be compensated for by the autofill algorithm.

Video Traffic Characteristics

Without QoS, video flows typically degrade. The pictures become unclear. Movement is jerky. Movement appears to be in slow motion. Often, the audio becomes unsynchronized with the video. The video can be completely gone, but the audio still works. In short, unless the network has significantly more bandwidth than is needed for all traffic, video quality degrades.

Just like the coverage of voice in this chapter, this section breaks down an analysis of video traffic as it relates to the four QoS characteristics: bandwidth, delay, jitter, and loss. First, the basics of packet video are explained, followed by QoS details unique to video in terms of the four QoS characteristics.

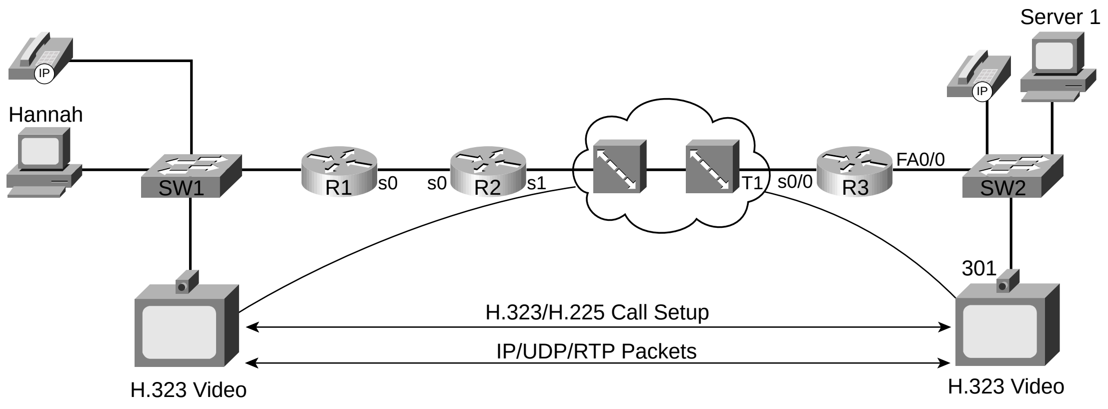
Video Basics

IP packet video can be categorized into two main categories:

- **Interactive video**—Includes H.323-compliant video conferencing systems, such as Cisco’s IP/VC 3500 series of products, and Microsoft’s NetMeeting desktop video-conferencing product. H.323-compliant video-conferencing tools use the familiar RTP protocol for transmission of the voice and audio payload, typically sending the audio in a separate RTP stream than the video.
- **Noninteractive video**—Includes typical e-learning video services and streaming media, and includes products such as Cisco’s IP/TV, Microsoft Windows Media Technologies products, and RealNetworks products. Some noninteractive video uses H.323 standards for video call setup and teardown, and some do not—for instance, RealNetworks most recent servers use Real-Time Streaming Protocol (RTSP) for call setup/teardown, and either the proprietary RealNetworks Data Transport (RDT) or RTP for video payload, depending on the video player used.

Like voice, video codecs convert the analog audio and video to packetized form. Codec delay, packetization delay, and de-jitter initial playout delay are all included in video delay, just like with voice. Familiar voice codecs, including G.711 and G.729, convert the audio stream, which is typically sent as a separate flow from the video signal. The video signals use a large variety of codecs, including ITU H.261, and the popular Moving Pictures Experts Group (MPEG) codecs. Figure 1-28 depicts a typical video conference between two H.323-compliant video-conference systems.

Figure 1-28 *H.323 Video Conference*



Before the video conference can be begin, several things must happen:

- A user must point/click the correct application settings to ask for a conference, typically something as simple as telling the H.323 application that you want a conference with a particular host name.
- The VC units must perform the H.323/H.225 call setup messages.
- Two RTP streams must be established—one for audio, and one for video.

So far, the similarities between voice and video outstrip the differences. The biggest difference is the bandwidth required for video. (Bandwidth requirements are covered in the upcoming section "Video Bandwidth Considerations.") Table 1-20 summarizes the type of QoS characteristics that video requires, as well as voice.

Table 1-20 *Comparing Voice and Video QoS Requirements*

	Bandwidth	**Delay**	**Jitter**	**Loss**
Voice Payload	Low	Low	Low	Low
Video Payload	High	Low	Low	Low
Voice Signaling	Low	Low	Medium	Medium
Video Signaling	Low	Low	Medium	Medium

Just like with voice, most QoS effort goes toward giving the video payload flows the QoS characteristics it needs. However, you might still want to treat video signaling traffic differently than other data traffic, and treat the video payload traffic differently. To classify, the QoS tool needs to be able to refer to a field in the packet that signifies that the packet is video payload, video signaling, or some other type of packet. Table 1-21 lists the various protocols used for signaling and for voice payload, defining documents, and identifying information.

Table 1-21 *Video Signaling and Payload Protocols*

Protocol	Documented By	Useful Classification Fields
H.323/H.225	ITU	Uses TCP port 1720
H.323/H.245	ITU	TCP ports 11*xxx*
H.323/H.225 RAS	ITU	TCP port 1719
RTSP	IETF RFC 2326	TCP or UDP port 554
Real-Time Transport Protocol (RTP)	RFC 1889	UDP ports 16384–32767, even ports only

The next few sections of this book examine video more closely in relation to the four QoS characteristics:

- Bandwidth
- Delay
- Jitter
- Loss

Video Bandwidth Considerations

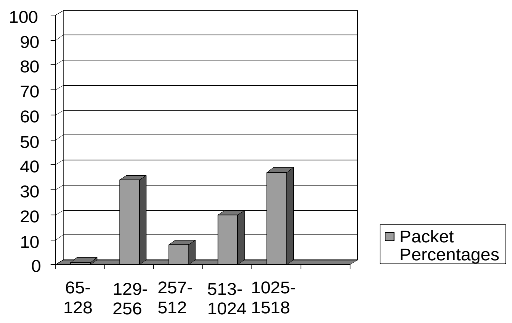
Unlike voice, video uses a variety of packet sizes and packet rates to support a single video stream. Most video codecs take advantage of what can loosely be called “prediction,” by sending an encoded video frame (large packet), followed by a series of vectors describing changes to the previous frame (small packet). Although this type of algorithm greatly reduces the required bandwidth, it does cause video streams to use a dynamic packet rate, with a range of packet sizes. Also the actual average bandwidth required for a section of video depends on the complexity and amount of movement in the video. Table 1-22 lists four popular video codecs and the required bandwidth ranges for each:

Table 1-22 *Video Codecs and Required Bandwidth*

Video Codec	Required Range
MPEG-1	500 to 1500 kbps
MPEG-2	1.5 to 10 Mbps
MPEG-4	28.8 to 400 kbps
H.261	100 to 400 kbps

Different codecs simply provide different tradeoffs for quality and bandwidth, and many different codecs are needed to support applications of all types. For instance, MPEG includes several standards that were created for different types of applications. ITU H.261 provides a video standard for video conferencing, which works well when the callers do not move around too much! If you have ever been on a video conference and watched someone who used their hands a lot to talk, for instance, you might have seen jerky arm movements. All these codecs operate with dynamic bandwidth, and with different-sized packets. Figure 1-29 shows a packet distribution for percentages of packets at various sizes in an H.261 conference call.

Figure 1-29 *Packet Size Distributions in a Video Conference Using H.261*



As mentioned earlier, video flows vary both the size of packets sent and the packet rate. In a high-quality video conference that might average 768 kbps of bandwidth, for example, the packet rates might vary from as low as 35 pps to as many as 120 pps. Some QoS tool configuration options might be affected by not only the bandwidth required (kbps), but also by the packet rate (pps). Remember, queue sizes are measured in packets; so to avoid tail drop, a queue holding video traffic may need a much larger queue size than a queue holding voice. Table 1-23 summarizes some of the key bandwidth differences between voice and video traffic.

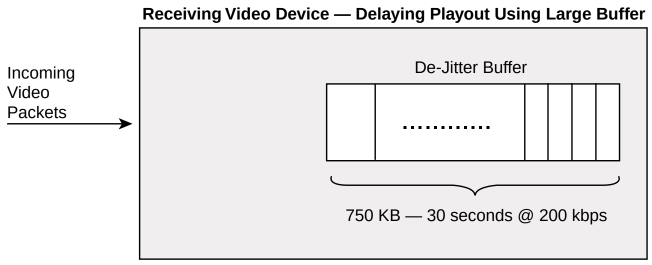
Table 1-23 *Voice and Video Bandwidth Contrasted*

Feature	Voice	Video
Number of flows in each direction	1	2 (1 audio, 1 video)
Packet sizes	Static, based on codec	Variable
Packet rate	Constant (isochronous)	Variable

Video Delay Considerations

Two-way or interactive packet video experiences the same delay components as does a voice call. However, one-way or streaming packet video tolerates a fairly large amount of delay. Consider Figure 1-30, which shows a packet video device receiving a one-way streaming video stream.

Figure 1-30 *Large Playout Buffers for One-Way Video*


Receiving Video Device — Delaying Playout Using Large Buffer
Incoming Video Packets
De-Jitter Buffer
750 KB — 30 seconds @ 200 kbps

The initial playout delay takes 30 seconds—not 30 ms, but 30 seconds—in the example of Figure 1-30. No interaction occurs, and no video flows back to the video server; by far, the most important feature is that, when the video does play, it has high quality. With a 30-second de-jitter buffer, a cumulative jitter of 30 seconds (not milliseconds) can occur without having playout problems due to delay or jitter.

For two-way interactive packet video, delay of course does impact quality. The previous section about voice and voice delay explains most of what you need to know about video delay. Video experiences codec delay, packetization delay, and de-jitter (initial playout) delay. Of particular note, video delay budgets typically run from 0 to 200 ms for high-quality video conferencing, and de-jitter initial playout delays typically range from 20 to 70 ms.

Video Jitter Considerations

Just like for delay, one-way video tolerates jitter much more so than two-way, interactive video. When planning for QoS, two-way video should be treated just like voice in terms of minimizing delay. Furthermore, proper and accurate bandwidth allocation/provisioning is recommended. One-way, streaming video should be given enough bandwidth, but extra delay and jitter can be tolerated.

Jitter concerns for video networks can be summarized as follows:

- Jitter still happens in packet networks.
- De-jitter buffers on the receiving side compensate for some jitter.
- De-jitter buffers for interactive video typically run in the tens of milliseconds, allowing even small amounts of jitter to affect video quality.
- De-jitter buffers for streaming video typically run into the tens of seconds, allowing significant jitter to occur without affecting video quality
- QoS tools, particularly queuing and fragmentation tools, can reduce jitter to low enough values such that the de-jitter buffer for interactive video works effectively.

Video Loss Considerations

Video flows degrade when packets are lost. The picture becomes jerky due to missing frames, the images freeze, and the audio may cut in and out. In short, video does not tolerate loss very well.

Routers drop packets for many reasons; packet loss due to full queues can be addresses with several types of QoS tools. Remember, tail drop describes the situation when a queue fills, another packet needs to be added to the queue, so the router throws away the packet that needs to be added to the tail of the queue. You can use four general QoS strategies to reduce the chance of lost (dropped) video packets:

- Enable queuing and putting video in a different queue than bursty data traffic.
- Configure the video queue to be longer.
- Enable CAC to protect the video queue from having too many concurrent video flows in it—in other words, CAC can protect video from other video streams.
- Use a Random Early Detect (RED) tool on the loss-tolerant flows (typically data, not video!), which causes those flows to slow down, which in turn reduces overall interface congestion.

Comparing Voice and Video: Summary

Table 1-24 summarizes the QoS requirements of video in comparison to voice.

Table 1-24 *Comparing Voice and Video QoS Requirements*

	Bandwidth	**Delay**	**Jitter**	**Loss**
Voice Payload	Low	Low	Low	Low
Video Payload—Interactive (2-Way)	High	Low	Low	Low
Video Payload—Streaming (1-Way)	High	High	High	Low
Video Signaling	Low	Low	Medium	Medium
Voice Signaling	Low	Low	Medium	Medium

Data Traffic Characteristics

This book, as well as the exams about which this book prepares you, assumes you have a fairly high level of knowledge about data traffic before using this book. In fact, the QoS and DQOS courses assume that you have been to the ICND and BSCI courses at least, and hopefully the

BCMSN and CVOICE courses. In fact, when DQOS first came out, the expectation was that students should be CCNPs before attending the course.

QoS has always been important, but QoS has become vitally important with the convergence of data, voice, and video into a single network. As with any convergence of technologies, professionals working in the networking arena come from many different backgrounds. This section about data is intended for those who do not come from a data background.

NOTE If you want to read more on TCP/IP protocols, take a look at the latest Douglas Comer's *Internetworking with TCP/IP* books. Volume 1 is most appropriate for networking engineers. A good alternative is Richard Stevens's *TCP/IP Illustrated*, Volume I.

Just like the coverage of voice and video in this chapter, this section breaks down an analysis of data traffic as it relates to the four QoS characteristics: bandwidth, delay, jitter, and loss. The discussion begins with the basics of data application flows, followed by QoS details as they relate to data in terms of the four QoS characteristics.

IP Data Basics

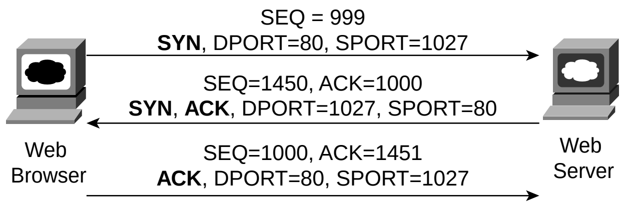
With voice and video, signaling occurred first in order to create the voice or video call. Although it is not called signaling in the data world, something similar does occur—for instance, when you open a web browser, and browse www.cisco.com, several things happen before the first parts of the web page appear. For our purposes for QoS, this book focuses on the actual payload flows—the actual data—rather than the data equivalent of signaling.

Most applications use one of two TCP/IP transport layer protocols: User Datagram Protocol (UDP) or Transmission Control Protocol (TCP). The person writing the application chooses which transport layer protocol to use. Most of the time the application programmer uses standard protocols, which tell them whether to use TCP or UDP. For instance, web servers use TCP, so if writing a web application, TCP is used.

TCP performs error recovery, but UDP does not. To perform error recovery, TCP sends some initialization messages to create a TCP connection, which coincidentally initializes some counters used to perform the error recovery. Figure 1-31 shows an example of connection establishment.

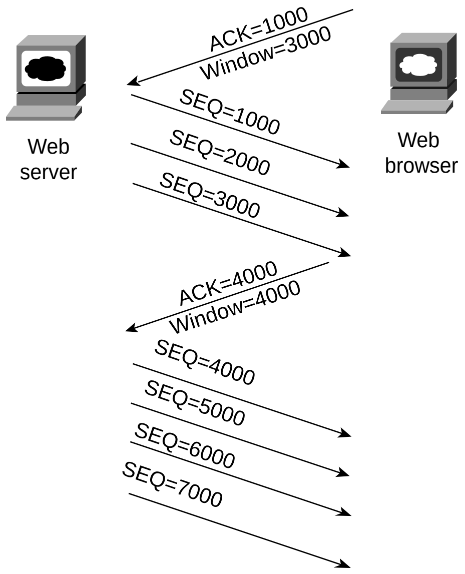
TCP signals connection establishment using 2 bits inside flags field of the TCP header. Called the SYN and ACK flags, these bits have a particularly interesting meaning. SYN means "synchronize the sequence numbers," which is one necessary component in initialization for TCP. The ACK field means "the acknowledgment field is valid in this header."

Figure 1-31 *TCP Connection Establishment*


SEQ = 999
SYN, DPORT=80, SPORT=1027
SEQ=1450, ACK=1000
SYN, **ACK**, DPORT=1027, SPORT=80
SEQ=1000, ACK=1451
ACK, DPORT=80, SPORT=1027
Web Browser
Web Server

When the three-way TCP handshake is complete, the TCP connection is up, and error recovery can be performed. Figure 1-32 shows how the sequence and acknowledgment fields are used, after the connection has been established.

Figure 1-32 *TCP Acknowledgments*



In the figure, the server sends data and labels it with the sequence number. The acknowledgment field in the TCP header sent back to the server by the web client (4,000) implies the next byte to be received; this is called *forward acknowledgment*. In essence, the browser acknowledged the receipt of all three packets, with sequence numbers 1000, 2000, and 3000. (Each packet contains 1000 bytes of data in this example.) The sequence number reflects the number of the first byte in the segment. Keep in mind that on the packet whose sequence number is 3000, with 1000 bytes, that bytes 3000 to 3999 are in the packet—so the browser should expect to get byte 4000 next.

TCP also controls the rate of sending data using windowing. This window field implies the maximum number of unacknowledged bytes allowed outstanding at any instant in time. Figure 1-34 shows windowing with a current window size of 3000, which increases to a window of 4000 by the end of the example. (Remember, each TCP segment has 1000 bytes of data in this example.) The window then "slides" up and down based on network performance, so it is sometimes called a *sliding window*. When the sender sends enough bytes to consume the current window, the sender must wait for an acknowledgment, which controls the flow of data. Effectively, the available window decreases as bytes are sent and increases as acknowledgment for those bytes are received.

The biggest difference between TCP and UDP is that TCP performs error recovery. Therefore, some people refer to TCP as reliable, and UDP as unreliable. And remember, voice and video flows that use RTP also use UDP—so why would voice and video use a protocol that is unreliable? The answer is simple: By the time a voice or video packet was sent, and TCP noticed that the packet was lost, and caused a retransmission, far too much delay would have already occurred. Therefore, resending the voice or video packet would be pointless. For data applications, however, where *all* the data really does need to make it to the other side of the connection, even if it takes additional time, TCP can be very useful. Figure 1-33 outlines the basic error-recovery logic of TCP.

Figure 1-33 *TCP Error Recovery*

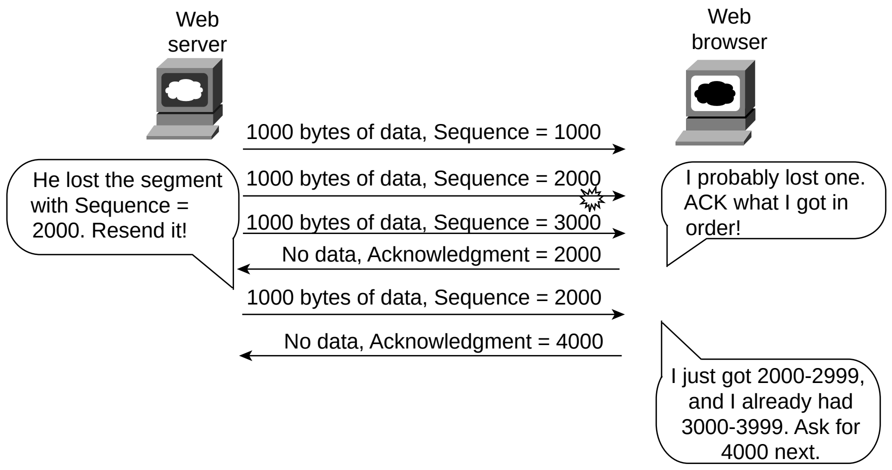
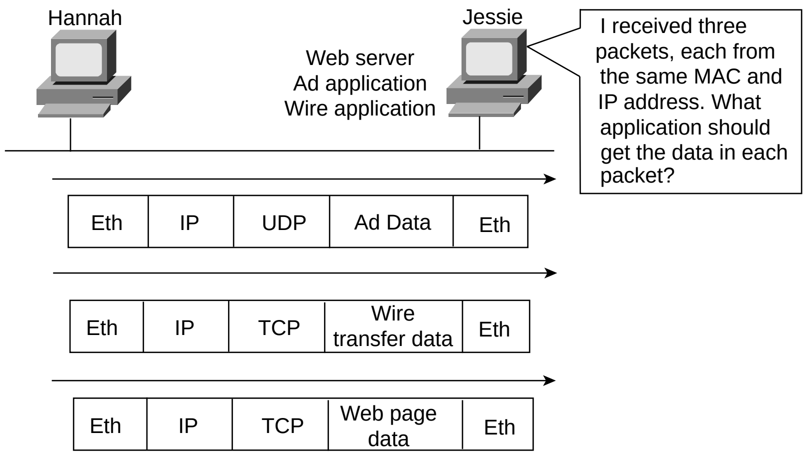

Web server
Web browser
1000 bytes of data, Sequence = 1000
1000 bytes of data, Sequence = 2000
1000 bytes of data, Sequence = 3000
No data, Acknowledgment = 2000
1000 bytes of data, Sequence = 2000
No data, Acknowledgment = 4000
He lost the segment with Sequence = 2000. Resend it!
I probably lost one. ACK what I got in order!
I just got 2000-2999, and I already had 3000-3999. Ask for 4000 next.

Figure 1-33 depicts a flow where the second TCP segment was lost or was in error. The web client's reply has an ACK field equal to 2000, implying that the web client is expecting byte number 2000 next. The TCP function at the web server then could recover lost data by resending the second TCP segment. The TCP protocol allows for resending just that segment and then waiting, hoping that the web client will reply with an acknowledgment that equals 4000.

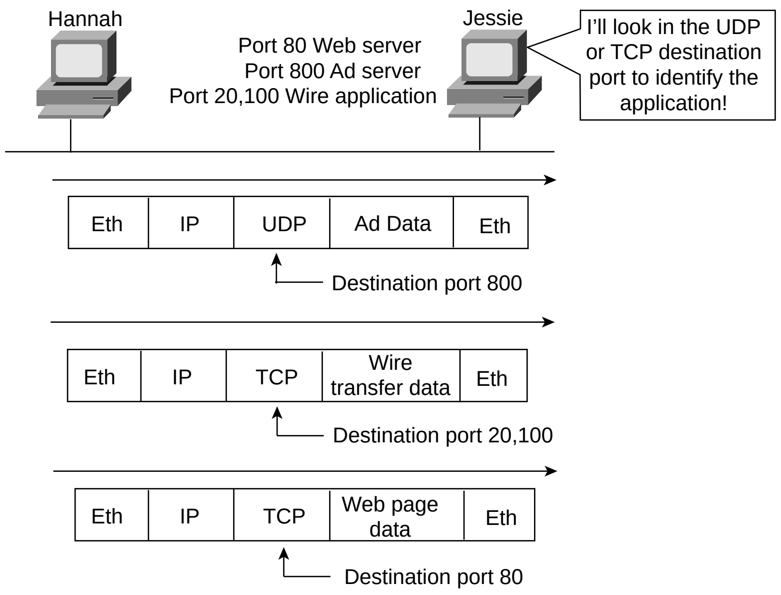
Finally, you should understand one additional feature of TCP and UDP before continuing with your examination of QoS. That feature concerns a part of the TCP and UDP headers called the *source and destination port numbers*. The main purpose for port numbers can be seen with a simple example; for QoS, port numbers can be used to classify a packet, which in turn allows a router or switch to choose a different QoS action. In this case, Hannah is using three applications, and server Jessie is the server for all three applications. This particular company wrote an Advertising application and a wire-transfer application, both in use. In addition, Hannah is using a web-based application, as shown in Figure 1-34.

Figure 1-34 *Hannah Sending Packets to Jessie, with Three Applications*


Hannah
Jessie
Web server
Ad application
Wire application
I received three packets, each from the same MAC and IP address. What application should get the data in each packet?
Eth | IP | UDP | Ad Data | Eth
Eth | IP | TCP | Wire transfer data | Eth
Eth | IP | TCP | Web page data | Eth

After receiving a packet, Jessie needs to know which application to give the data to, but all three packets are from the same Ethernet and IP address. You might think that Jessie could look at whether the packet contains a UDP or a TCP header, but, as you see in the figure, two applications (wire transfer and web) both are using TCP. Well, UDP and TCP designers purposefully included a port number field in the TCP and UDP headers to allow multiplexing. "Multiplexing" is the term generally used to describe the capability to determine which application gets the data for each packet. Each of the applications uses a different port number, so Jessie knows which application to give the data to, as seen in Figure 1-35.

Figure 1-35 *Hannah Sending Packets to Jessie, with Three Applications Using Port Numbers to Multiplex*


Hannah
Jessie
Port 80 Web server
Port 800 Ad server
Port 20,100 Wire application
I'll look in the UDP or TCP destination port to identify the application!
Eth | IP | UDP | Ad Data | Eth
Destination port 800
Eth | IP | TCP | Wire transfer data | Eth
Destination port 20,100
Eth | IP | TCP | Web page data | Eth
Destination port 80

Most well-known applications, such as web, FTP, TFTP, Telnet, SMTP, POP3, and so on, use a well-known port. Using an application would be cumbersome if before you used it you had to call someone to find out what port number it uses. With well-known ports, you can assume that web servers use port 80, for instance. For QoS tools, if you want to classify web traffic, you can just look for packets that use port 80.

Certainly, you could spend a career just learning about, and working with, all the protocols inside the realm of TCP/IP. This brief introduction provides a little background that will help with some of the things you will read about in later sections. Table 1-25 lists the key points to remember about TCP and UDP.

Table 1-25 *TCP and UDP Comparison Chart*

Feature	**TCP**	**UDP**
Error recovery	Yes	No
Uses port number	Yes	Yes
Uses windowing for flow control	Yes	No

The next few sections of this book examine data more closely in relation to the four QoS characteristics: bandwidth, delay, jitter, and loss.

Data Bandwidth Considerations

Unlike voice, which consumes a constant amount of bandwidth, and unlike video, which consumes a range of bandwidth, data bandwidth requirements range wildly. Some applications might literally need less than 1 kbps, whereas others would take literally as much bandwidth as they can get.

The bigger question with data bandwidth revolves around OSI Layer 8—the business layer. For instance, how much bandwidth should web traffic get? Well, in many shops, web traffic consumes 80 percent of the network bandwidth used. However, a better question might be "How much bandwidth does important business web traffic get?" A financial-planning web application ought not to have to compete too hard with my surfing the ESPN.com website to check out the latest scores. A stockbroker might prefer that stock quotes get the lowest latency, for instance, and I would just have to wait a little longer to see the latest sports scores.

The other legitimate question with data application flows concerns whether the application is interactive or not. An interactive application needs less bandwidth typically (not always!) than do the typical noninteractive applications. The traditional comparison between Telnet (very low bandwidth) and FTP (takes as much bandwidth as it can get) implies that interactive implies low bandwidth. Well, many interactive applications today are web based, and if the web pages contain large amounts of graphics, the bandwidth required can also be pretty large.

Bandwidth, delay, jitter, and loss characteristics of data traffic vary based on these factors and others. In fact, one data application's QoS characteristics might be different at two different companies! Because so many factors must be considered, and because it varies from network to network, no one set of requirements really works.

However, do not despair! You should at least consider one perspective on bandwidth, based on business reasons. Simply put, some applications are business critical, and some are not. At a minimum, a QoS strategy should include identifying the critical applications, and giving these application flows the QoS characteristics they need. The rest of the data traffic can take the crumbs that have fallen off the table—or, in QoS terms, be treated as "best-effort" traffic. Table 1-26 summarizes some of the key bandwidth differences between all three types of traffic.

Table 1-26 *Voice, Video, and Data Bandwidth Contrasted*

Feature	**Voice**	**2-Way Video**	**Data**
Number of flows	2 (1 in each direction)	4 (1 audio and 1 video in each direction)	1 bidirectional flow
Packet sizes	Fixed, based on codec	Variable	Varies greatly
Packet rate	Constant (isochronous)	Variable	Varies greatly
Traffic load in opposite directions	Asymmetric	Symmetric	Asymmetric

Data Delay Considerations

Unlike voice and video, the perceived quality of the data application does not degrade quickly when delay increases by mere hundreds of milliseconds. In fact, relative to voice and interactive video, data applications tolerate delay very well. Also unlike voice and video, data applications tend to have round-trip delay requirements.

Two factors affect the delay requirements of a data application, as summarized in Table 1-27.

Table 1-27 *Factors to Consider for Data Delay*

Factor	Mission Critical	Not Mission Critical
Interactive	Should get the lowest delay of all data applications. Most shops strive for 1–2-second application response time—per-packet delay must be shorter.	Applications could benefit from lower delay. Also differentiating between mission critical and not mission critical can be difficult.
Not interactive	While mission-critical, noninteractive applications typically need particular bandwidth requirements met, delay can vary greatly as long as bandwidth is supplied.	Best candidate for getting any left-over bandwidth, with all other voice, video, and data applications getting better QoS treatment.

Because data QoS requirements vary greatly, more communication happens among the staff when deciding how to treat different applications with QoS. With more communication, the chances of miscommunication increase. If you talk to someone who does not just focus on the network, for instance, he might say “we need consistent 1- to 3-second response time on this mission-critical application!” What he means is that all packets, in both directions, that need to flow, in order for the user to see the complete response from the application, must occur in 1 to 3 seconds. Do not be fooled into thinking that a one-way delay for one packet of 1 to 3 seconds is what the user meant when he asked for 1- to 3-second application response time!

Data applications do not suffer from all the same causes of delay as do voice and video traffic. Earlier in the chapter you learned about the core delay components—queuing, serialization, propagation, network, processing, and shaping. Data does not suffer from the additional delay caused by codec, packetization, and de-jitter buffer delays.

Data Jitter Considerations

Just like for delay, data application tolerate jitter much more so than voice and video. Interactive applications are much less tolerant of jitter. I learned networking at a large SNA network inside IBM, and the adage when adjusting routes and QoS (built-in to SNA from the early days, but that’s another story!) was that it was okay to have longer response times, as long as the response times were consistent. Human nature for some reason made consistency more important than net response time to most human interactive users, and that fact remains true today.

Interactive data applications cannot tolerate as much jitter or delay as noninteractive applications can. Voice and video applications can tolerate even less jitter or delay than interactive data applications. Because voice and video cannot tolerate any significant delay and jitter, engineers might choose to use QoS tools to decrease jitter for voice and video—which in turn increases delay and jitter for data applications! If a router sends a voice packet next rather than a data packet, for instance, the voice packet has a better chance of meeting its delay budget—but the data packet experiences more delay. Knowing that data can tolerate delay and jitter more than voice enables the engineer to make this tradeoff.

Jitter concerns for data networks can be summarized as follows:

- Jitter still happens in packet networks.
- Data applications do not have a de-jitter buffer—instead, the user always experiences some jitter, perceived simply as variable response times.
- Interactive applications are less tolerant of jitter, but jitter into the hundreds of milliseconds can be tolerated even for interactive traffic.
- In a converged network, QoS tools that improve (lower) jitter are best applied to voice and video traffic; the penalty is longer delays and more jitter for data applications.

Data Loss Considerations

Unlike voice and video, data does not always suffer when packets are lost. For most applications, the application needs to get all the data; if the lost packets are re-sent, however, no real damage occurs. Some applications do not even care whether a packet is lost. For perspective, consider applications to fall into one of three categories: those that use UDP and the applications perform error recovery, those that use UDP and the applications do not perform error recovery, and applications that use TCP.

UDP Applications That Perform Error Recovery

Some applications need error recovery but choose instead to implement the error recovery with application code. For instance, Network File System (NFS) and Trivial File Transfer Protocol (TFTP) both use UDP, and both perform error recovery inside the application. NFS provides the capability to read from and write to remote file servers—certainly guaranteeing that data is not lost would be important to NFS! Likewise, TFTP transfers files, so ensuring that the file was not missing parts would also be important. So, UDP applications that provide application layer recovery tolerate loss.

UDP Applications That Do Not Perform Error Recovery

Some applications simply do not need error recovery. The most popular of these protocols is Simple Network Management Protocol (SNMP), which allows management software to interrogate managed devices for information. Network management stations retrieve huge amounts of individual data, and often times a missed statistic occasionally does not impact the worker using the management software. SNMP designers purposefully avoided TCP and application layer error recovery to keep SNMP simple, and therefore more scalable, knowing that SNMP would be used in large volumes. Because the application does not care whether a packet is lost, these applications tolerate lost packets.

TCP-Based Applications

Because TCP-based applications expect TCP to recover lost packets, higher loss rates are acceptable. Although the lost packets may be transparent to the user, the added load on the network caused by retransmission of the packets can actually increase the congestion in the network.

Comparing Voice, Video, and Data: Summary

Table 1-28 summarizes the QoS requirements of data, in comparison to voice and video.

Table 1-28 *Comparing Voice, Video, and Data QoS Requirements*

	Bandwidth	Delay	Jitter	Loss
Voice Payload	Low	Low	Low	Low
Video Payload—Interactive (2-Way)	High	Low	Low	Low
Video Payload—Streaming (1-Way)	High	High	High	Low
Video Signaling	Low	Low	Medium	Medium
Voice Signaling	Low	Low	Medium	Medium
Data: Interactive, Mission Critical	Variable, typical medium	Medium	Medium	Medium
Data: Not Interactive, Mission Critical	Variable, typically high	High	High	Medium
Data: Interactive, Not Critical	Variable, typical medium	High	High	Medium
Data: Not Interactive, Not Critical	Variable, typically high	High	High	High

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review will help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures will be a convenient way to review the day before the exam.

Table 1-29 outlines some of the behaviors seen when no QoS is applied in a network.

Table 1-29 *Traffic Behavior with No QoS*

Type of Traffic	Behavior Without QoS
Voice	Voice is hard to understand.
	Voice breaks up, sounds choppy.
	Delays make interacting difficult; callers do not know when other party has finished talking.
	Calls are disconnected.
Video	Picture displays erratically; jerky movements.
	Audio not in sync with video.
	Movement slows down.
Data	Data arrives after it is no longer useful.
	Customer waiting for customer care agent, who waits for a screen to display.
	Erratic response times frustrate users, who may give up or try later.

As shown in Figure 1-36, with compression, if a ratio of 2:1 is achieved, the 80-kbps flow will only require 40 kbps in order to be sent across the link—effectively doubling the bandwidth capacity of the link.

Figure 1-36 *With a 2:1 Compression Ratio Versus No Compression*

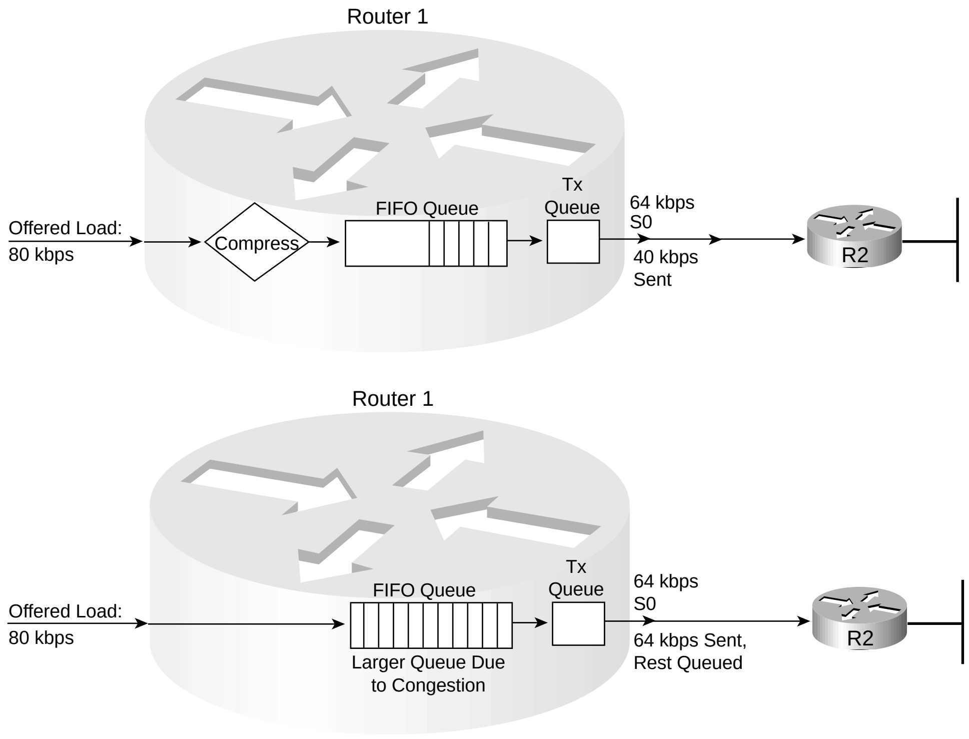
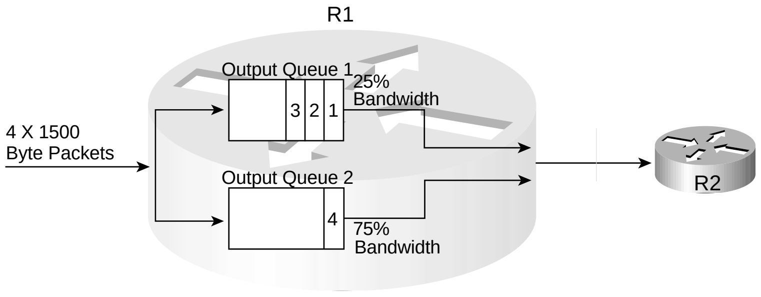


Figure 1-37 shows a two-queue system where the first queue gets 25 percent of the bandwidth on the link, and the second queue gets 75 percent of the bandwidth.

Figure 1-37 *Bandwidth Reservation Using Queuing*



The tools summarized in Table 1-30 help to improve the effects of bandwidth in a network.

Table 1-30 *QoS Tools That Affect Bandwidth*

Type of QoS Tool	How It Affects Bandwidth
Compression	Compresses either payload or headers, reducing overall number of bits required to transmit the data
CAC	Reduces overall load introduced into the network by rejecting new voice and video calls
Queuing	Can be used to reserve minimum amounts of bandwidth for particular types of packets

Figure 1-38 shows two contrasting examples of serialization and propagation delay.

Figure 1-38 *Serialization and Propagation Delay for Selected Packet and Link Lengths*

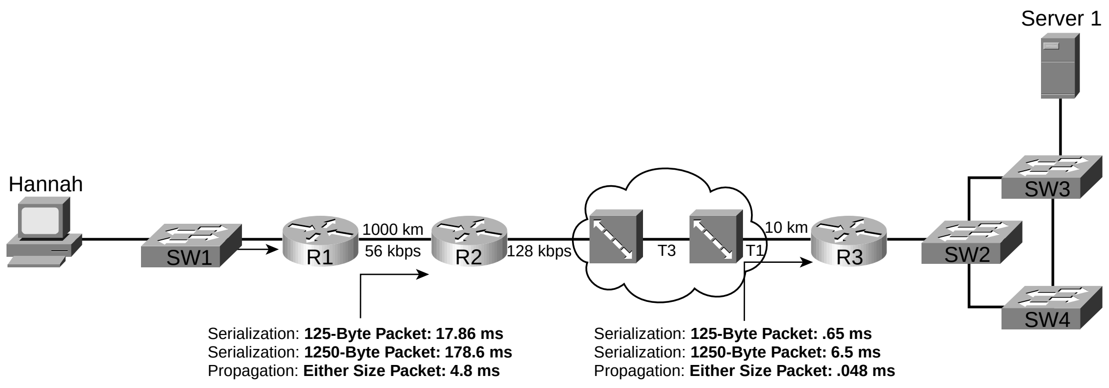

Server 1
Hannah
SW1
R1
1000 km
56 kbps
R2
128 kbps
T3
T1
10 km
R3
SW2
SW3
SW4
Serialization: 125-Byte Packet: 17.86 ms
Serialization: 1250-Byte Packet: 178.6 ms
Propagation: Either Size Packet: 4.8 ms
Serialization: 125-Byte Packet: .65 ms
Serialization: 1250-Byte Packet: 6.5 ms
Propagation: Either Size Packet: .048 ms

Figure 1-39 lists the queuing, serialization, and propagation delays experienced by data, voice, and video traffic.

Figure 1-39 *Delay Components: Three Components, Single Router (R1)*

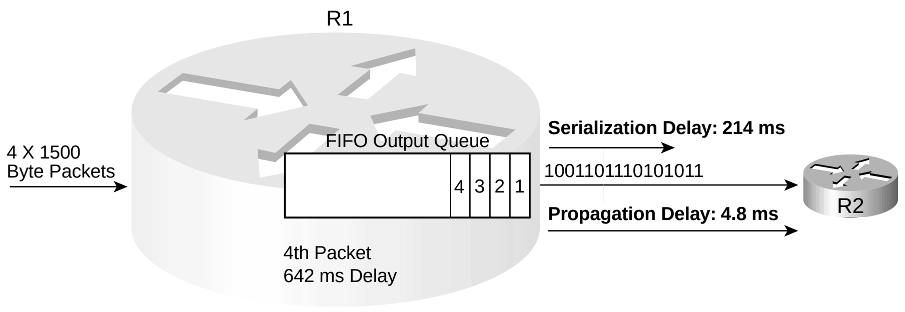
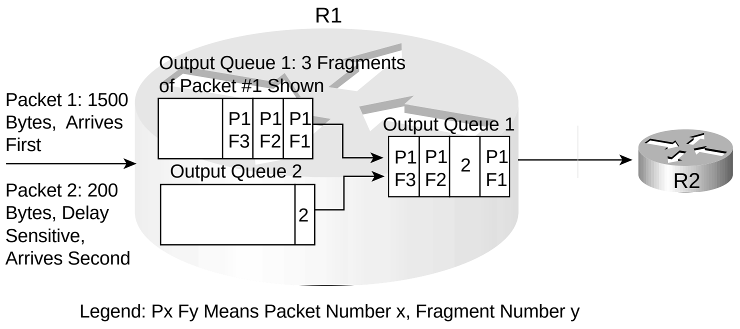


Figure 1-40 depicts LFI operation.

Figure 1-40 *Link Fragmentation and Interleaving*


R1
Output Queue 1: 3 Fragments of Packet #1 Shown
Packet 1: 1500 Bytes, Arrives First
P1 F3
P1 F2
P1 F1
Output Queue 1
Output Queue 2
Packet 2: 200 Bytes, Delay Sensitive, Arrives Second
2
P1 F3
P1 F2
2
P1 F1
R2
Legend: Px Fy Means Packet Number x, Fragment Number y

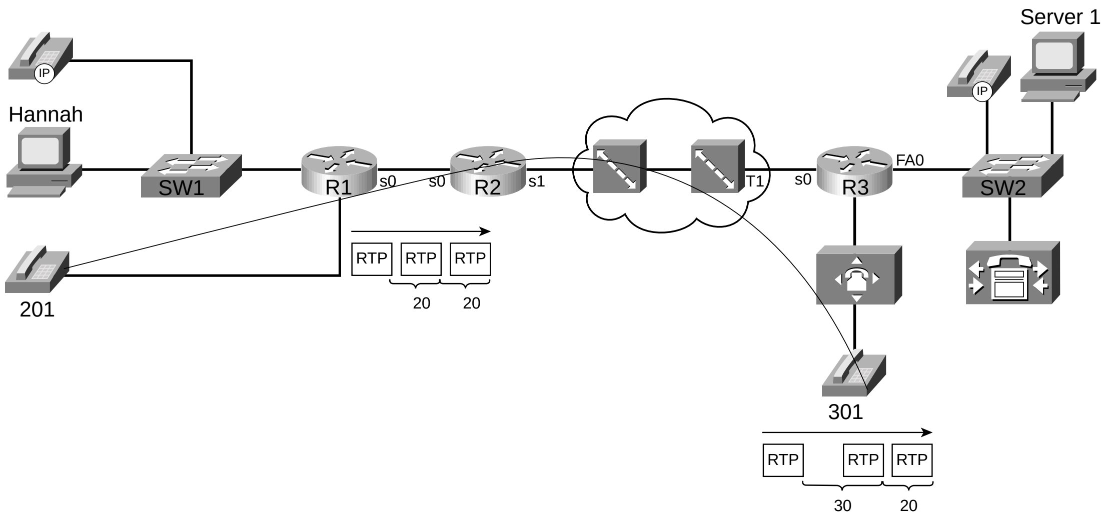
Although adding more bandwidth always helps, the tools summarized in Table 1-31 do help to improve the effects of delay in a network.

Table 1-31 *QoS Tools That Affect Delay*

Type of QoS Tool	How It Affects Delay
Queuing	Enables you to order packets so that delay-sensitive packets leave their queues more quickly than delay-insensitive packets.
Link fragmentation and interleaving	Because routers do not preempt a packet that is currently being transmitted, LFI breaks larger packets into smaller fragments before sending them. Smaller delay-sensitive packets can be sent after a single smaller fragment, instead of having to wait for the larger original packet to be serialized.
Compression	Compresses either payload or headers, reducing overall number of bits required to transmit the data. By requiring less bandwidth, queues shrink, which reduces delay. Also serialization delays shrink, because fewer bits are required. Compression also adds some processing delay.
Traffic shaping	Artificially increases delay to reduce drops inside a Frame Relay or ATM network.

Figure 1-41 shows the jitter experienced by three packets as part of a voice call between phones at extension 301 and 201.

Figure 1-41 *Jitter Example*



The same set of tools that affect delay also affect jitter; Table 1-32 lists some of these QoS tools.

Table 1-32 *QoS Tools That Affect Jitter*

Type of QoS Tool	How It Affects Jitter
Queuing	Enables you to order packets so that delay-sensitive packets leave their queues more quickly than delay-insensitive packets.
Link fragmentation and interleaving	Because routers do not preempt a packet that is currently being transmitted, LFI breaks larger packets into smaller fragments before sending them. Smaller delay-sensitive packets can be sent after a single smaller fragment, instead of having to wait for the larger original packet to be serialized.
Compression	Compresses either payload or headers, reducing overall number of bits required to transmit the data. By requiring less bandwidth, queues shrink, which reduces delay. Also serialization delays shrink, because fewer bits are required. Compression also adds some processing delay.
Traffic shaping	Artificially increases delay to reduce drops inside a Frame Relay or ATM network.

With a longer maximum queue size, likelihood of loss decreases. However, queuing delay increases, as shown in Figure 1-42.

Figure 1-42 *Queuing Effects on Packet Loss*

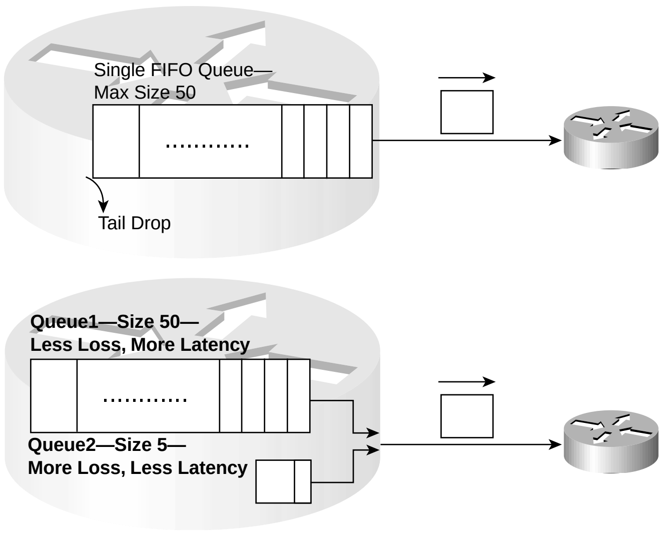


Table 1-33 summarizes the points concerning the two types of QoS tools for affecting loss.

Table 1-33 *QoS Tools That Affect Loss*

Type of QoS Tool	Brief Description
Queuing	Implementing longer queues increases delay, but avoids loss.
RED	Implementing RED drops packets randomly as queues approach the point of being full, slowing some TCP connections. This reduces overall load, shortening the congested queue, while affecting only some users' response times.

Figure 1-43 outlines the format of an IP packet using RTP.

Figure 1-43 *IP Packet for Voice Call—RTP*

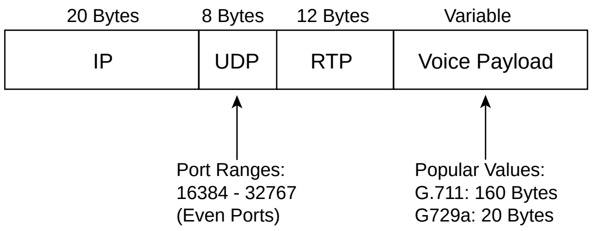


Table 1-34 contrasts the QoS requirements of voice payload and signaling flows.

Table 1-34 *Comparing Voice Payload to Voice Signaling: QoS Requirements*

	Bandwidth	**Delay**	**Jitter**	**Loss**
Voice Payload	Low	Low	Low	Low
Video Signaling	Low	Low	Medium	Medium

Table 1-35 lists the bandwidth requirements for various types of voice calls, as listed in the DQOS course.

Table 1-35 *Bandwidth Requirements with Various Data-Link Types*

L2 Header Type	**L2 Header Size**	**IP/UDP/RTP Header Size**	**Codec**	**Payload Bandwidth**	**Total Bandwidth**
Ethernet	14	40 bytes	G.711	64 kbps	85.6
MLPPP/FR	6	40 bytes	G.711	64 kbps	82.4
Ethernet	14	40 bytes	G.729	8 kbps	29.6
MLPPP/FR	6	40 bytes	G.729	8 kbps	26.4

The delay components that affect all types of traffic are listed in Table 1-36.

Table 1-36 *Components of Delay Not Specific to One Type of Traffic*

Delay Component	**Definition**	**Where It Occurs**
Serialization delay	Time taken to place all bits of a frame onto the physical medium. Function of frame size and physical link speed.	Outbound on every physical interface; typically negligible on T3 and faster links.
Propagation delay	Time taken for a single bit to traverse the physical medium from one end to the other. Based on the speed of light over that medium, and the length of the link.	Every physical link. Typically negligible on LAN links and shorter WAN links.
Queuing delay	Time spent in a queue awaiting the opportunity to be forwarded (output queuing), or awaiting a chance to cross the switch fabric (input queuing).	Possible on every output interface. Input queuing unlikely in routers, more likely in LAN switches.
Forwarding or Processing Delay	Time required from receipt of the incoming frame, until the frame/packet has been enqueued for transmission.	On every piece of switching equipment, including routers, LAN switches, Frame Relay switches, and ATM switches.

continues

Table 1-36 *Components of Delay Not Specific to One Type of Traffic (Continued)*

Delay Component	Definition	Where It Occurs
Shaping delay	Shaping (if configured) delays transmission of packets to avoid packet loss in the middle of a Frame Relay or ATM network.	Anywhere that shaping is configured, which is most likely on a router, when sending packets to a Frame Relay or ATM network.
Network delay	Delays created by the components of the carrier's network when using a service. For instance, the delay of a Frame Relay frame as it traverses the Frame Relay network.	Inside the service provider's network.

Figure 1-44 shows an example of delay concepts, with sample delay values shown. When the delay is negligible, the delay is just listed as zero.

Figure 1-44 *Example Network with Various Delay Components shown: Left-to-Right Direction*

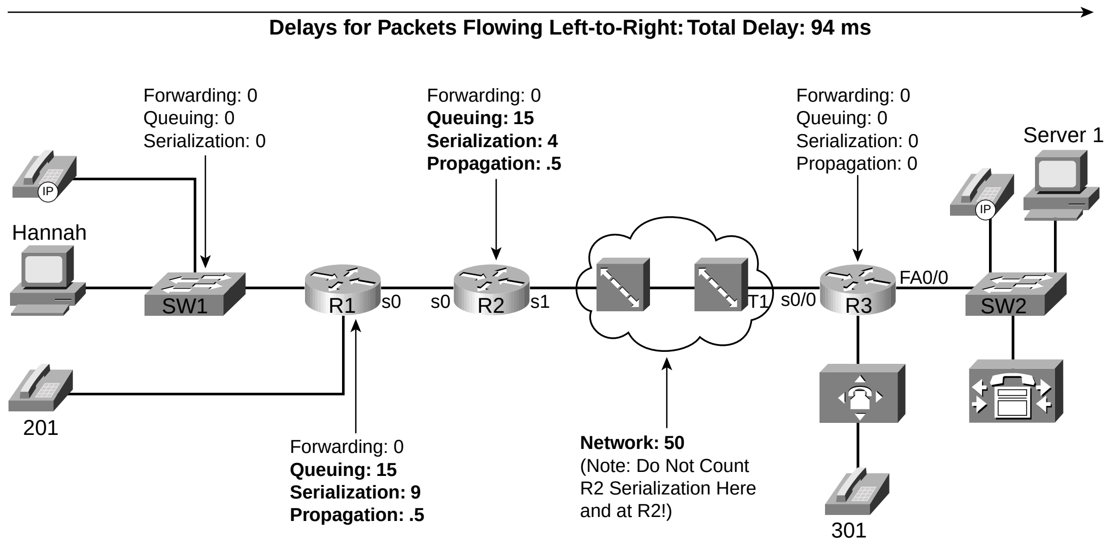


Table 1-37 outlines the suggested delay budgets.

Table 1-37 *One-Way Delay Budget Guidelines*

1-Way Delay (in ms)	Description
0–150	ITU G.114 recommended acceptable range
0–200	Cisco's recommended acceptable range
150–400	ITU G.114's recommended range for degraded service
400+	ITU G.114's range of unacceptable delay in all cases

All the delay components for a voice call are summarized in the example in Figure 1-45.

Figure 1-45 *Complete End-to-End Voice Delay Example*

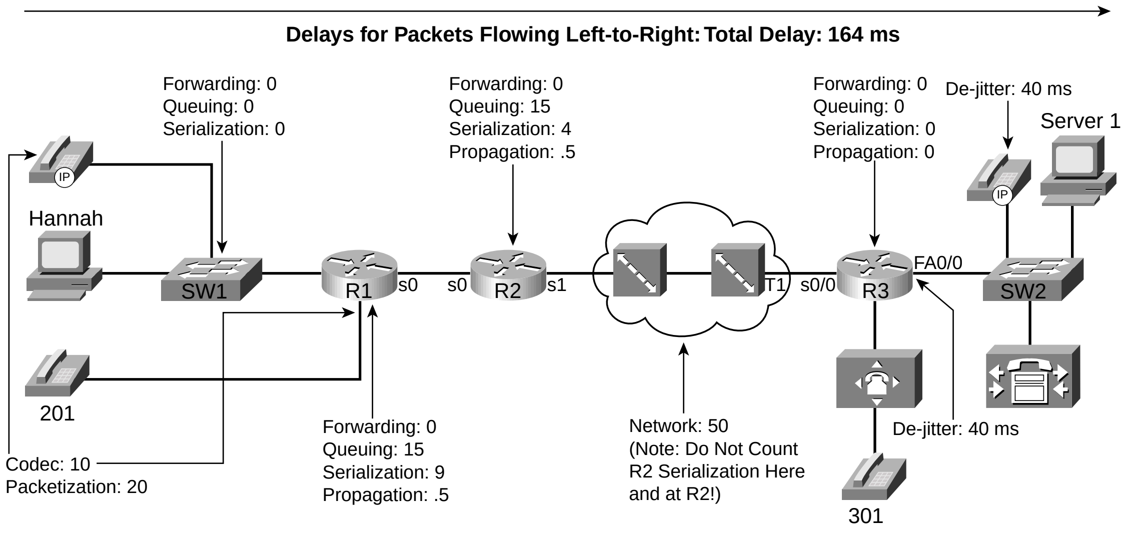


Table 1-38 lists the different delay components and whether they are variable.

Table 1-38 *Delay Components, Variable and Fixed*

Delay Component	Fixed or Variable	Comments	QoS Tools That Can Help
Codec	Fixed	Varies slightly based on codec and processing load; considered fixed in course books (and probably on exams). Typically around 10 ms.	None.

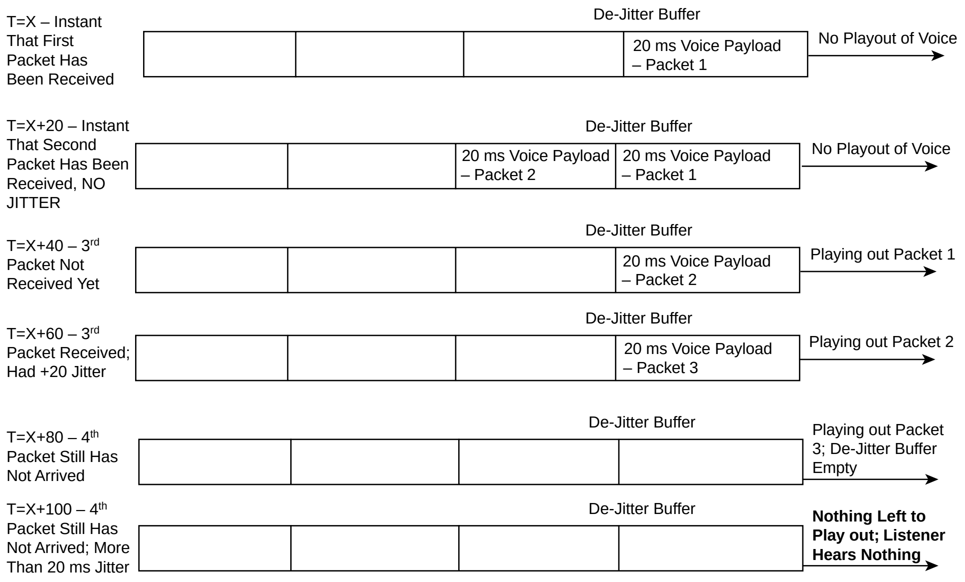
continues

Table 1-38 *Delay Components, Variable and Fixed (Continued)*

Delay Component	Fixed or Variable	Comments	QoS Tools That Can Help
Packetization	Fixed	Some codecs require a 30-ms payload, but packetization delay does not vary for a single codec. Typically 20 ms, including when using G.711 and G.729.	None.
Propagation	Variable	Varies based on length of circuit. About 5 ms/100 km.	Move your facilities to the same town.
Queuing	Variable	This is the most controllable delay component for packet voice.	Queuing features, particularly those with a priority-queuing feature.
Serialization	Fixed	It is fixed for voice packets, because all voice packets are of equal length. It is variable based on packet size for all packets.	Fragmentation and compression.
Network	Variable	Least controllable variable component.	Shaping, fragmentation, designs mindful of reducing delay.
De-jitter buffer (initial playout delay)	Variable	This component is variable because it can be configured for a different value. However, that value, once configured, remains fixed for all calls until another value is configured. In other words, the initial playout delay does not dynamically vary.	Configurable playout delay in IOS gateways; not configurable in IP Phones.

Figure 1-46 shows an example of jitter for packets 3 and 4.

Figure 1-46 *De-Jitter Buffer Underrun Due to Jitter*



De-Jitter Buffer
T=X – Instant That First Packet Has Been Received
20 ms Voice Payload – Packet 1
No Playout of Voice
T=X+20 – Instant That Second Packet Has Been Received, NO JITTER
20 ms Voice Payload – Packet 2
20 ms Voice Payload – Packet 1
No Playout of Voice
T=X+40 – 3rd Packet Not Received Yet
20 ms Voice Payload – Packet 2
Playing out Packet 1
T=X+60 – 3rd Packet Received; Had +20 Jitter
20 ms Voice Payload – Packet 3
Playing out Packet 2
T=X+80 – 4th Packet Still Has Not Arrived
Playing out Packet 3; De-Jitter Buffer Empty
T=X+100 – 4th Packet Still Has Not Arrived; More Than 20 ms Jitter
Nothing Left to Play out; Listener Hears Nothing

Table 1-39 summarizes some of the key bandwidth differences between voice and video traffic.

Table 1-39 *Voice and Video Bandwidth Contrasted*

Feature	Voice	Video
Number of flows in each direction	1	2 (1 audio, 1 video)
Packet sizes	Static, based on codec	Variable
Packet rate	Constant (isochronous)	Variable

Table 1-40 summarizes some of the key bandwidth differences between all three types of traffic.

Table 1-40 *Voice, Video, and Data Bandwidth Contrasted*

Feature	Voice	2-Way Video	Data
Number of flows	2 (1 in each direction)	4 (1 audio and 1 video in each direction)	1 bidirectional flow
Packet sizes	Fixed, based on codec	Variable	Varies greatly
Packet rate	Constant (isochronous)	Variable	Varies greatly

Two factors affect the delay requirements of a data application. Table 1-41 lists these requirements.

Table 1-41 *Factors to Consider for Data Delay*

Factor	Mission Critical	Not Mission Critical
Interactive	Should get the lowest delay of all data applications. Most shops strive for 1–2-second application response time—per-packet delay must be shorter.	Applications could benefit from lower delay. Also differentiating between mission critical and not mission critical can be difficult.
Not interactive	Although mission critical, noninteractive applications typically need particular bandwidth requirements met, delay can vary greatly as long as bandwidth is supplied.	Best candidate for getting any leftover bandwidth, with all other voice, video, and data applications getting better QoS treatment.

Table 1-42 summarizes the QoS requirements of data, in comparison to voice and video.

Table 1-42 *Comparing Voice, Video, and Data QoS Requirements*

	Bandwidth	Delay	Jitter	Loss
Voice Payload	Low	Low	Low	Low
Video Payload—Interactive (2-Way)	High	Low	Low	Low
Video Payload—Streaming (1-Way)	High	High	High	Low
Video Signaling	Low	Low	Medium	Medium
Voice Signaling	Low	Low	Medium	Medium
Data: Interactive, Mission Critical	Variable, typical medium	Medium	Medium	Medium
Data: Not Interactive, Mission Critical	Variable, typically high	High	High	Medium
Data: Interactive, Not Critical	Variable, typical medium	High	High	Medium
Data: Not Interactive, Not Critical	Variable, typically high	High	High	High

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD-ROM included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD-ROM nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD-ROM exam to include, or not include, the topics that are only on the CCIP QoS.

1. List the four traffic characteristics that QoS tools can affect.
2. Describe some of the characteristics of voice traffic when no QoS is applied in a network.
3. Describe some of the characteristics of video traffic when no QoS is applied in a network.
4. Describe some of the characteristics of data traffic when no QoS is applied in a network.
5. Interpret the meaning of the phrase, "QoS is both 'managed fairness,' and at the same time 'managed unfairness'."
6. Define bandwidth. Compare and contrast bandwidth concepts over point-to-point links versus Frame Relay.
7. Compare and contrast bandwidth and clock rate in relation to usage for QoS.
8. List the QoS tool types that affect bandwidth, and give a brief explanation of why each tool can affect bandwidth.
9. Define delay, compare/contrast one-way and round-trip delay, and characterize the types of packets for which one-way delay is important.
10. List the categories of delay that could be experienced by all three types of traffic: data, voice, and video.
11. Define, compare, and contrast serialization and propagation delay.
12. Define network delay.
13. List the QoS tool types that affect delay and give a brief explanation of why each tool can affect delay.
14. Define jitter. Give an example that shows a packet without jitter, followed by a packet with jitter.

15 List the QoS tool types that affect jitter and give a brief explanation of why each tool can affect jitter.

16 Define packet loss and describe the primary reason for loss for which QoS tools can help.

17 List the QoS tool types that affect loss and give a brief explanation of why each tool can affect loss.

18 Describe the contents of an IP packet carrying the payload for a G.729 VoIP call.

19 Describe the amount of bandwidth required for G.711 and G.729 VoIP calls, ignoring data-link header/trailer overhead.

20 List the delay components that voice calls experience, but which data-only flows do not experience.

21 Define the meaning of the term “packetization delay” in relation to a voice call.

22 List the different one-way delay budgets as suggested by Cisco and the ITU.

23 Define the term “codec delay” and discuss the two components when using a G.729 codec.

24 Describe the affects of a single lost packet versus two consecutive lost packets, for a G.729 voice call.

25 Describe a typical video payload flow in terms of packet sizes and packet rates.

26 Discuss the delay requirements of video traffic.

27 List the basic differences between TCP and UDP traffic.

28 Contrast the QoS characteristics needed by interactive data applications, as compared to the QoS needs of voice payload flows.

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Topics

- Correctly describe the QoS framework.
- Describe the differences between integrated services and differentiated services.

QoS Exam Objectives

- Describe the Integrated Services model
- List the key benefits and drawbacks of the IntServ model
- Describe the Differentiated Services model
- List the key benefits of the DiffServ model compared to the IntServ model
- Describe the building blocks of IP QoS mechanisms (classification, marking, metering, policing, shaping, dropping, forwarding, queuing)
- List the IP QoS mechanisms available in the Cisco IOS
- Describe what QoS features are supported by different IP QoS mechanisms
- Describe the interoperability between DSCP-based and IP-precedence based devices in a network

CHAPTER 2

QoS Tools and Architectures

To build a house, you need tools, you need materials, you need labor, and you need architectural plans. To build a network using quality of service (QoS), you need tools and you need an architecture. This chapter lists the various IOS QoS tools and explains the two predominant QoS architectures: integrated services (IntServ) and differentiated services (DiffServ).

Chapter 1, "QoS Overview," covered various types of QoS tools. Now this chapter begins with a review of the types of tools and lists each of the IOS QoS tools in each category. This chapter also introduces the most important acronyms for the QoS exams. All the terms, and the tools behind the terms, get further treatment in later chapters of the book.

As a tool for learning, the second section of this chapter explains something called the Good-Old Common Sense model for QoS. This imaginary QoS model just explains some things about QoS that many people have heard about before. However, taking a few minutes to think about these concepts, and why they make sense, is useful before looking at the two formalized QoS models—namely DiffServ and IntServ.

This chapter then examines the DiffServ architecture in detail. DiffServ attempts to provide Internet-scale QoS, which is a lofty goal indeed! DiffServ uses a class-based approach to differentiate between packets, which scales somewhat better than its predecessor, IntServ. Whether DiffServ succeeds in this goal remains to be seen; however, many of the concepts can be helpful with any QoS implementation.

Finally, the chapter ends with a short discussion on IntServ, which uses the Resource Reservation Protocol (RSVP) to reserve bandwidth for individual flows in the network.

This chapter concludes the introductory materials in this book; the remainder of this book delves into the details of the various QoS tools.

"Do I Know This Already?" Quiz

The purpose of the "Do I Know This Already?" quiz is to help you decide whether you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 12-question quiz, derived from the major sections in the "Foundation Topics" section of this chapter, helps you determine how to spend your limited study time.

Table 2-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 2-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions	Score
1	QoS Tools	1 to 4	
2	Differentiated Services	5 to 8	
3	Integrated Services	9 to 12	
All questions		1 to 12	

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **10 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary," and the "Q&A" sections.
- **11 or 12 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, proceed to the next chapter.

QoS Tools Questions

1. List four queuing tools, including the full names and popular acronyms.
2. List four link-efficiency tools, including the full names and popular acronyms.
3. Which of the following tools can be used for classification and marking? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP
4. Which of the following tools can be used for policing? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

Differentiated Services Questions

5 Define the DiffServ term “DSCP,” including what the acronym stands for.

6 Define the DiffServ term “PHB,” including what the acronym stands for.

7 Compare and contrast the terms “shaper,” “meter,” and “dropper,” according to DiffServ specifications. Suggest typical points in the network where each is used.

8 Compare and contrast the contents of the IP ToS byte before and after the advent of DiffServ.

Integrated Services Questions

9 Imagine an enterprise network, connected to an Internet service provider (ISP), that is connected to a second ISP, which is then connected to another enterprise network. The second ISP does not support IntServ directly. Discuss the two options that allow the other three networks to support IntServ for flows that pass through the nonsupporting ISP.

10 Compare and contrast DiffServ and IntServ in terms of using classes, flows, and scalability.

11 Describe the two options available to a router to perform IntServ admission control.

12 What is the QoS framework, and what does it define?

Foundation Topics

This chapter introduces the long list of QoS tools and two important QoS architectures—differentiated services (DiffServ) and integrated services (IntServ). You can think of QoS implementation as building a house. To properly build a house, you certainly need (and use) a large variety of tools. An experienced builder might be able to use the tools to build a house without an architectural plan. With the architectural plans, however, an experienced house builder can build a better house! Similarly, an experienced network engineer can use QoS tools with good results, without considering the DiffServ and IntServ architectures. By considering these architectures, however, a network engineer can use the tools to implement the QoS policies more consistently and efficiently, and thereby guarantee better QoS.

This chapter begins with an introduction to the various QoS tools in IOS. Then a section titled "The Good-Old Common Sense QoS Model" identifies where you might use a tool in a network, without considering the two formal QoS models. The chapter closes with one section on each of the two formal specifications for QoS architectures and conventions, namely the DiffServ and IntServ architectures.

Introduction to IOS QoS Tools

Ultimately, this book will help you pass one or two QoS exams (the CCIP QoS exam and the Cisco Channel Partner DQOS exam). Both exams cover a wide variety of types of QoS tools. This chapter lists the tools covered on the DQOS exam; any tools only covered on the QOS exam are listed in Appendix B, "Topics on the CCIP QoS Exam."

Of particular note, neither exam covers the implementation details of these tools on LAN switches, so the configurations and features of QoS tools on LAN switches is not listed in this section. Look to Chapter 10, "LAN QoS," for further details about how these QoS tools operate in the LAN. Also make sure to check www.cisco.com, and www.ciscopress.com/1587200589, for the latest information about any changes to the exams.

The coverage here begins with an explanation of classification and marking tools, followed by queuing tools, shaping and policing tools, congestion-avoidance tools, link-efficiency tools, call admission control (CAC), and QoS management tools.

Classification and Marking

Almost every QoS tool uses classification to some degree. To put one packet into a different queue than another packet, the IOS must somehow differentiate between the two packets. To perform header compression on RTP packets, but not on other packets, the IOS must determine which packets have Real Time Protocol (RTP) headers. To shape data traffic going into a Frame Relay network, so that the voice traffic gets enough bandwidth, the IOS must differentiate between Voice over IP (VoIP) and data packets. If an IOS QoS feature needs to treat two packets differently, you must use classification.

Classification involves differentiating one packet from another, typically by examining fields inside the headers. After classification, a QoS tool can treat packets in one class differently than others. To just give all VoIP traffic preference over all other traffic, the queuing tool would need to classify traffic into one of two categories: VoIP or not-VoIP.

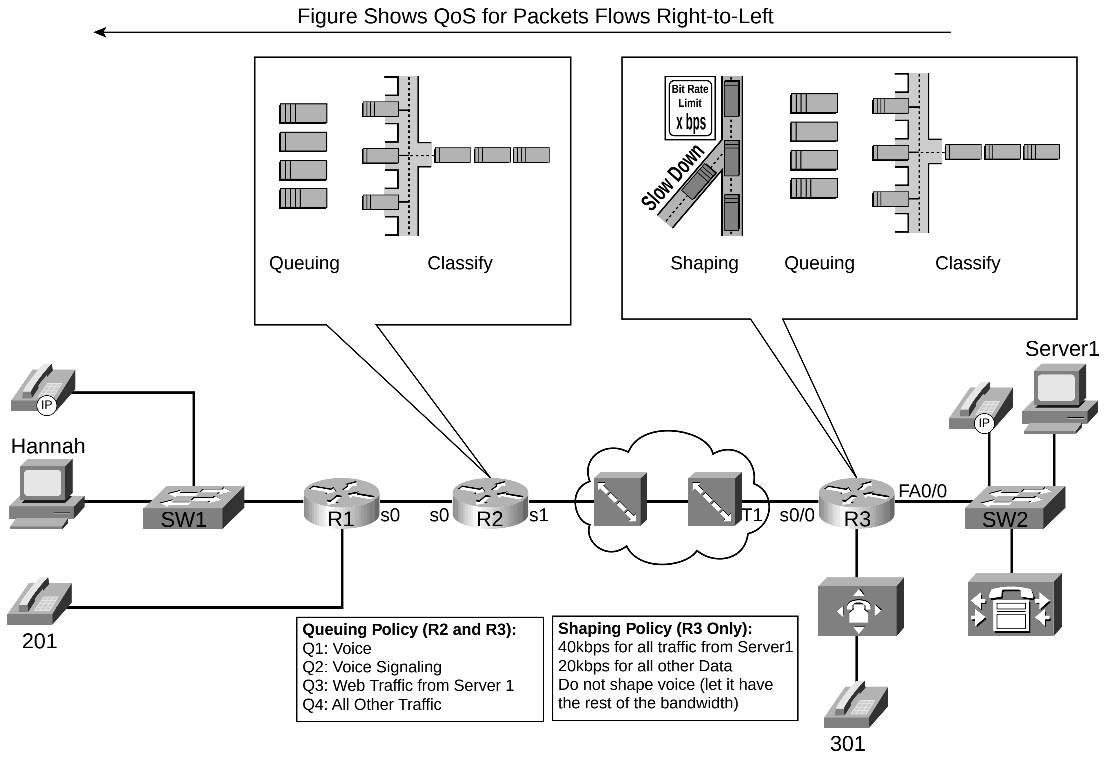
Because most QoS tools need to differentiate between packets, most QoS tools have classification features. In fact, you may already know something about several of the QoS tools described in this book. You may realize that you already know how to perform classification using some of those tools. For instance, many QoS tools enable you to classify using access-control lists (ACLs)—for instance, if ACL 101 "permits" a packet, the packet falls into one queue; if ACL 102 permits a packet, it is placed in a second queue; and so on. In one way of thinking, queuing could instead be called "classification and queuing," because the queuing feature must somehow decide which packets end up in which queue. Similarly, traffic shaping could be called "classification and traffic shaping," policing could be called "classification and policing," and so on. Because most QoS tools classify traffic, however, the names of most QoS tools never evolved to mention the classification function of the tool.

Only one category of QoS tool, called classification and marking, highlights the classification feature in the name of the tool. For other tools, the classification function is just part of the story; with classification and marking tools, classification is the whole point of the tool. To appreciate the need for classification and marking tools, consider Figure 2-1.

The figure shows the QoS policies for traffic flowing right to left. R3 performs queuing and shaping, and R2 performs queuing only. However, for both sets of queues, and for the shaping function, classification must occur. The classification part of the effort seems to be a simple task, but it may cause many comparisons to be made. For instance, each packet exiting R3's S0 and R2's S0 interfaces might be compared for the following:

- From source address 10.1.1.1, TCP source port 80 (Server1 web traffic)
- Using User Datagram Protocol (UDP), port number range 16384 to 32767 (voice payload)—may also want to check IP address ranges to match IP Phones' voice subnets, or voice gateway IP addresses
- Using TCP port 1720 (H.323 voice signaling)
- Using TCP port range 11000 to 11999 (Voice signaling)
- Using TCP port 1719 (Voice signaling)
- Using TCP port 2000 to 2002 (Skinny voice signaling)
- Using UDP port 2427 and 2428 (MGCP voice signaling)

Figure 2-1 *Sample Network, with Queuing and Shaping Tools Enabled*



Queuing Policy (R2 and R3):
Q1: Voice
Q2: Voice Signaling
Q3: Web Traffic from Server 1
Q4: All Other Traffic

Shaping Policy (R3 Only):
40kbps for all traffic from Server1
20kbps for all other Data
Do not shape voice (let it have the rest of the bandwidth)

Classification and marking tools simplify the classification process of the other QoS tools. Even with seemingly simple requirements, the classification functions can require many comparisons to every packet. Rather than have each tool do extensive packet matching, classification and marking tools do the extensive classification once, and mark a field in a packet header. The remaining QoS tools just need to look for this marked field, simplifying the repetitive classification work.

The two most commonly used marking fields in the IP header are the IP Precedence field, and the Differentiated Services Code Point (DSCP) field. You will see the details of these two fields, along with the other fields that can be used for marking, later in this chapter. Consider Figure 2-2, where classification and marking is performed on input of R3.

The queuing and shaping features can now classify more efficiently. Queuing is still performed on R3 and R2, and shaping is still performed on R3. However, the extensive matching logic for each packet done for all incoming traffic can be performed once on R3's FA0/0 interface, or once on one of the LAN switches, such as SW3. Once marked, the other QoS tools can react to the marked value, which each QoS tool can efficiently match in the end-to-end path through the network.

Figure 2-2 *Sample Network, with Simplified Classification as a Result of Classification and Marking*

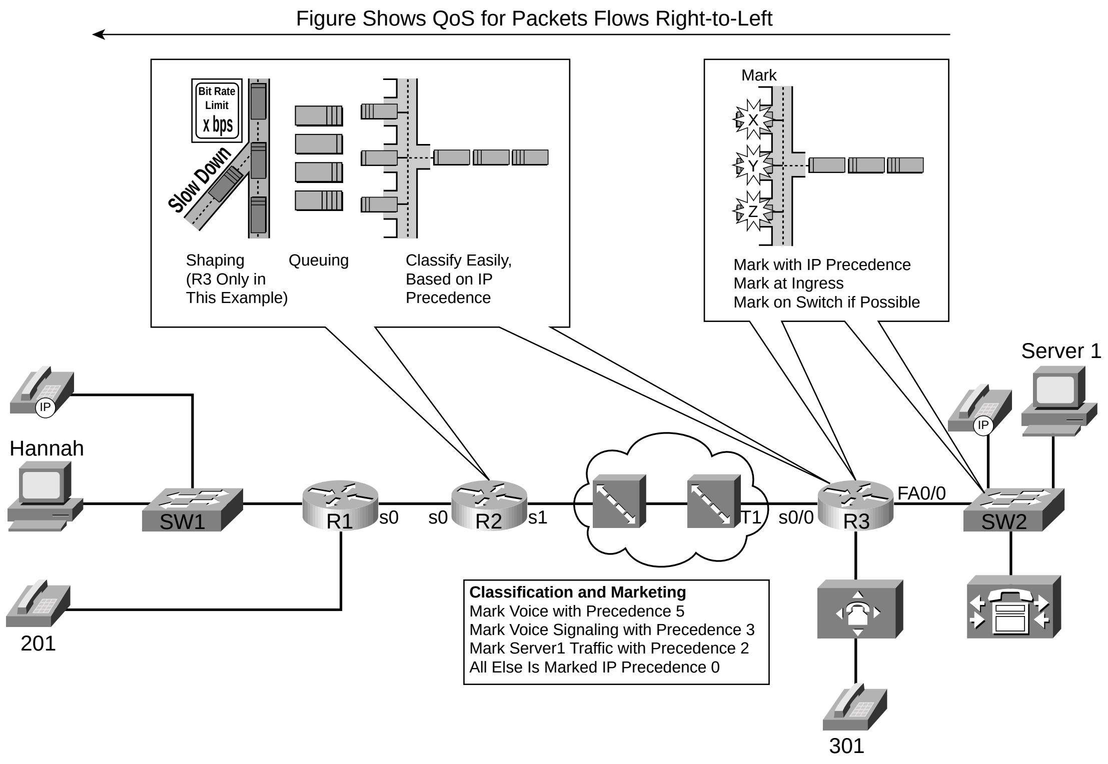


Figure Shows QoS for Packets Flows Right-to-Left
Bit Rate Limit
x bps
Slow Down
Shaping (R3 Only in This Example)
Queuing
Classify Easily, Based on IP Precedence
Mark
X
Y
Z
Mark with IP Precedence
Mark at Ingress
Mark on Switch if Possible
Server 1
IP
Hannah
SW1
R1
s0
s0
R2
s1
T1
s0/0
R3
FA0/0
SW2
201
301
Classification and Marketing
Mark Voice with Precedence 5
Mark Voice Signaling with Precedence 3
Mark Server1 Traffic with Precedence 2
All Else Is Marked IP Precedence 0

Classification and Marking Tools

A variety of classification and marking tools exist. Classification and marking tools first classify by looking at something inside each packet; you can compare these tools by listing the fields the tool can examine. Classification and marking tools mark the frame or packet based on the earlier comparisons; you can compare these tools by listing the fields that can be marked. Some classification and marking tools also perform other functions, as noted in Table 2-2.

Chapter 3 explains the details of each of the tools, all the marked fields, and the configuration of each tool.

Table 2-2 *Comparison of Classification and Marking Tools*

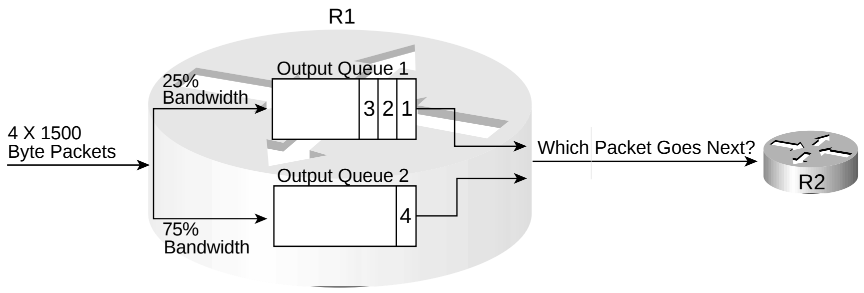
Tool	Other Functions Besides Class and Mark	Fields That Can Be Examined for Classification	Fields That Can Be Marked*
Policy-based routing (PBR)	Routing packets based on something besides destination address	ACLs indirectly through route maps	IP ToS field IP Precedence field QoS Group
Committed access rate (CAR)	Policing	IP ACLs QoS Group IP DSCP	IP Precedence IP DSCP QoS Group MPLS Experimental
Class-based marking (CB marking)	None	IP ACLs Any markable fields Input interface MAC addresses All NBAR-enabled fields	IP precedence DSCP 802.1P CoS ISL Priority ATM CLP Frame Relay DE MPLS Experimental QoS Group
Network based application recognition (NBAR)	Statistical information about traffic mix; recognition of applications that use the dynamic port	Extensive list (see Chapter 3, "Classification and Marking")	None; used in conjunction with CB marking
VoIP dial peers	Call routing for VoIP	None	IP Precedence

* All claims about features/functions that may be affected by IOS versions assume version 12.2, unless otherwise stated.

Queuing

Queuing, also occasionally called "scheduling," provides the ability to reorder packets when congestion occurs. Whereas queuing sometime occurs at the ingress interface, called "input queuing", most queuing methods only implement output queuing. The general idea is simple, but the details can be a little overwhelming. Consider Figure 2-3, with a simple two-queue output queue system.

Figure 2-3 *Simple Output Queuing, Two Queues*



In the figure, four packets arrived in order, at about the same time. The queuing tool's classification feature classified packets 1 through 3 as belonging in Queue 1, and packet 4 as belonging in Queue 2. The figure implies that Queue 2 should receive 75 percent of the bandwidth. But which packet is sent next? In what order do these four packets leave the router? If packet 5 shows up a little later, could it be sent before some of packets 1 through 4? Could the tool support more than two queues? Well, the answers to these questions define the key comparison points between the various queuing tools. You should look for the following when comparing queuing tools:

- Classification capabilities, particularly the packet header fields that can be matched to classify a packet into a particular queue. In some cases, the queuing tool automatically classifies traffic, whereas other tools require you to configure the values to be matched in the packets explicitly.
- The maximum number of queues (sometimes called the maximum number of classes). If you need to distinguish between *x* different types of traffic for queuing, you need at least *x* queues.
- The queue service algorithm. For some queuing tools, Cisco publishes the algorithms used to decide what packet is taken from which queue next; for other tools, Cisco publishes the net effect of the algorithm. In either case, you can still make a good choice as to which tool to use.

Ultimately, you use these queuing features, and other less-obvious features, when choosing the right queuing tool for a particular need in a particular network.

Queuing Tools

QoS queuing tools provide you with a variety of queuing methods. Queuing tools define a number of queues. Cisco publishes the queue service algorithm in some cases; in others, Cisco publishes only the end result (the "what"), but not the algorithm (the "how"). Table 2-3 outlines the key features of IOS queuing methods.

Table 2-3 *Comparison of Queuing Tools*

Tool	Maximum Number of Queues	Classification Capabilities	Queue Service Algorithm/End Result of Algorithm
Priority Queuing (PQ)	4	IP ACL* Input interface Fragments	Strict service; always serves higher-priority queue over lower queue.
Custom Queuing (CQ)	16	IP ACL* Input interface Fragments	Serves a configured number of bytes per queue, per round-robin pass through the queues. Result: Rough percentage of the bandwidth given to each queue under load.
Weighted Fair Queuing (WFQ)	4096	Automatic, based on flows. (Flow identified by source/destination address and port numbers, plus protocol type.)	Each flow uses a different queue. Queues with lower volume and higher IP precedence get more service; high volume, low precedence flows get less service.
Class-Based Weighted Fair Queuing (CBWFQ)	64	IP ACL* NBAR Same as CB marking	Service algorithm not published; results in set percentage bandwidth for each queue under load.
Low Latency Queuing	N/A	Same as CBWFQ	LLQ is a variant of CBWFQ, which makes some queues "priority" queues, always getting served next if a packet is waiting in that queue. It also polices traffic.
IP RTP Priority	N/A	Even UDP ports between 16384 and 32767 (all VoIP payload ports)	An added feature with WFQ or CBWFQ, all VoIP payload is placed in a special "priority" queue, always getting served next if a packet is waiting in that queue.
Modified Deficit Round-Robin (MDRR)	8	IP precedence	Similar to CQ, but each queue gets an exact percentage of bandwidth. Supports LLQ mechanism as well.

* Some queuing tools support different configuration tools that allow matching the same fields that an ACL can match. In these cases, only the IP ACL method of matching is listed in this summary table.

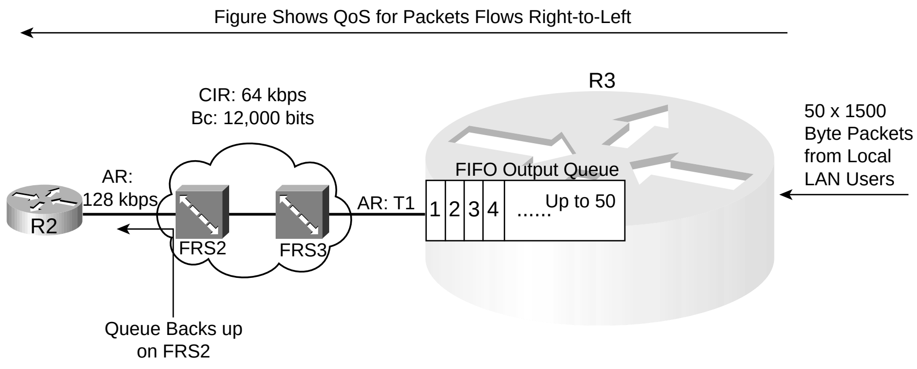
Chapter 4, “Congestion Management,” covers each of the queuing tools in detail.

Shaping and Policing

Because shaping and policing provide two different functions, you may wonder why shaping and policing are covered here at the same time. The simple answer is this: Networks that use policing typically need shaping as well. Also both shaping and policing measure the rates at which traffic is sent and received in a network, so some of the underlying features are similar. Both can be described using similar metaphors of “token buckets.” Finally, from a business perspective, shaping and policing are typically implemented at or near the edge between an enterprise and a service provider. Therefore, when considering whether you need to use one type of tool, you need to be thinking about the other type.

Traffic shaping, or shaping, delays packets by putting packets in a queue, even when real bandwidth is available. It’s like being at the bank. A teller finishes with a customer, and you’re next in line. You have to wait another minute or two, however, while the teller finishes doing some paperwork for the preceding customer. Why would a router ever want to delay packets? Well, the short answer is “because delaying these packets is better than what happens if you don’t delay them.” Figure 2-4 shows just one example where shaping is useful.

Figure 2-4 *Sample Network, Speed Mismatch (T/1 and 128 kbps)*



This example results in a large queue forming at Frame Relay Switch 2 (FRS2) due to the speed mismatch between the access links at R2 and R3. In this example, 50 1500-byte packets arrive over R3’s Ethernet during a 500-ms span, needing to be delivered to R2. If all 50 packets were to arrive one after the other, with no gaps, a queue would form on R3’s S0 interface. Because it takes a little less than 10 ms to send a 1500-byte packet over T/1, however, all 50 packets would drain from the queue within that 500 ms.

However, because the access link between FRS2 and R2 clocks at 128 kbps, it takes almost 100 ms to serialize a 1500-byte packet. So, although some queuing happens at R3, FRS2's egress queue on the link to R2 fills—in this case, it needs to be 45 packets long. (Five packets could be sent over this link during the 500 ms that the rest of the packets are arriving.)

What happens if FRS2's maximum egress queue size is only 20 frames? In such a scenario, around half of the frames are discarded. What is the impact? The quality of voice and video streams degrades. Most data applications resend their data—which may well cause the same phenomena all over again. Both results, of course, are bad.

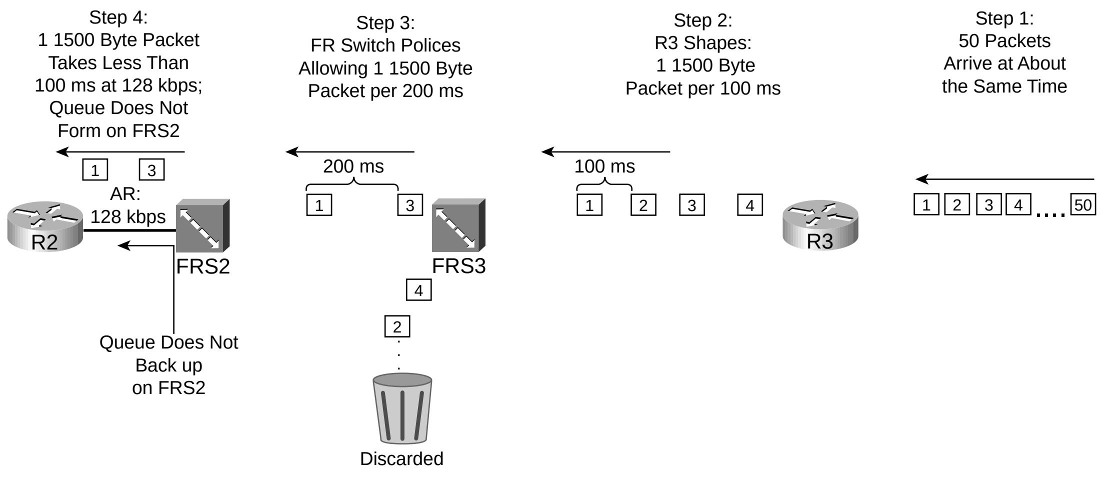
Traffic shaping solves this particular problem. If R3 had just waited and sent one 1500-byte packet every 100 ms, because FRS2 can send one 1500-byte packet in a little less than 100 ms, no queue would have formed on FRS2's egress interface. Even if R3 were to send one 1500-byte packet every 50 ms, FRS2's egress queue would grow, but only a few packets would be lost.

Whenever a speed mismatch occurs, shaping may be able to reduce the chance that packets get dropped. In the previous example, a speed mismatch occurred on the access rates of two Frame Relay-connected routers. In other cases, it may be that many VCs terminate at one router, and the collective VC committed information rate (CIRs) far exceed the access rate (oversubscription). In either case, queues may form, and they may form in a place where the engineer cannot control the queue—inside the Frame Relay cloud.

Shaping may help in one other specific case: when the Frame Relay service provider uses policing. The service provider may need to limit a VC to use just the CIR amount of bandwidth. Most providers, as well as their customers, expect the Frame Relay data terminal equipment (DTE) to send more than the CIR across each VC. However, the provider may decide that in this case, they need to prevent R3 and R2 from sending more than CIR. Why? For many reasons, but one common reason may be that a particular part of their network may have enough capacity to support the CIRs of all VCs for all customers, but not much bandwidth beyond that. To protect customers from each other, the provider may limit each VC to CIR, or some percentage over CIR, and discard the excess traffic.

The QoS tool used to monitor the rate, and discard the excess traffic, is called traffic policing, or just policing. Because the provider is monitoring traffic sent by the customer, traffic policers typically monitor ingress traffic, although they can monitor egress traffic as well. Figure 2-5 shows the same network, but with policing and shaping enabled for traffic entering FRS3 from R3.

Figure 2-5 *Traffic Policing and Shaping*



In the shaping discussion, one solution involved sending only one 1500-byte packet every 100 ms, which prevented an egress queue from forming on FRS2. As seen in this figure, however, the ingress policer on FRS3 monitors incoming traffic on the VC from R3 to R2, allowing only one 1500-byte packet per *200* ms. Policers discard the excess traffic, which in this case, even with shaping enabled on R3, half of the packets will be discarded when the network is busy! The solution, when the provider enables policing, is to configure shaping such that R3 only sends traffic at a rate that the policer function allows. In summary, some of the reasons behind shaping and policing are as follows:

- Packets might be lost in a multiaccess WAN due to access rate speed mismatch, oversubscription of CIRs over an access link, or by policing performed by the provider.
- Traffic shaping queues packets when configured traffic rates are exceeded, delaying those packets, to avoid likely packet loss.
- Traffic policing discards packets when configured traffic rates are exceeded, protecting other flows from being overrun by a particular customer.

Shaping and Policing Tools

QoS shaping and policing tools provide you with a variety of methods. As usual, you may consider many factors when comparing these tools. (Table 2-4 lists a few of these factors.) First, not all shaping and policing tools support every data-link protocol. Second, some tools can be enabled on a subinterface, but not on a per data-link connection identifier (DLCI); therefore, in cases where a network uses multipoint subinterfaces, one tool may give more granularity for shaping/policing. With regard to policers, some categorize packets as either conforming to or exceeding a traffic contract (called a two-headed policer), and some categorize packets as either conforming to, exceeding, or violating a traffic contract (a three-headed policer).

Table 2-4 *Comparison of Shaping and Policing Tools*

Tool	Policer or Shaper	Interfaces Supported	Per Subinterface, and Per VC, Support
Class-based policing (CB policing; sometimes just called policer)	Policer	All that are supported by Cisco Express Forwarding (CEF)	Per subinterface
Committed access rate (CAR)	Policer	All that are supported by CEF	Per subinterface
Class-based shaping	Shaper	All that are supported by CEF	Per subinterface
Generic traffic shaping/ distributed traffic shaping (GTS/DTS)	Shaper	Frame, ATM, SMDS, Ethernet	Per subinterface
Frame Relay traffic shaping (FRTS)	Shaper	Frame	Per DLCI

All functions listed are based on 12.2 mainline IOS code levels.

Chapter 5, “Traffic Policing and Shaping,” covers each of the policing and shaping tools in detail.

Congestion Avoidance

When networks become congested, output queues begin to fill. When new packets need to be added to a full queue, the packet is dropped—a process called *tail drop*. Tail drop happens in most networks every day—but to what effect? Packet loss degrades voice and video flows significantly; for data flows, packet loss causes higher-layer retransmissions for TCP-based applications, which probably increases network congestion.

Two solutions to the tail-drop problem exist. One solution is to lengthen queues, and thereby lessen the likelihood of tail drop. With longer queues, fewer packets are tail dropped, but the average queuing delay is increased. The other solution requires the network to ask the devices sending the packets into the network to slow down before the queues fill—which is exactly what the congestion-avoidance QoS tools do.

Congestion-avoidance tools operate under the assumption that a dropped TCP segment causes the sender of the TCP segment to reduce its congestion window to 50 percent of the previous window. If a router experiences congestion, before its queues fill, it can purposefully discard several TCP segments, making a few of the TCP senders reduce their window sizes. By reducing these TCP windows, these particular senders send less traffic into the network, allowing the congested router's queues time to recover. If the queues continue to grow, more TCP segments are purposefully dropped, to make more TCP senders slow down. If the queues become less congested, the router can stop discarding packets.

Congestion-Avoidance Tools

This book covers three congestion-avoidance tools. One of the tools was never implemented in IOS (Random Early Detection, or RED)—but because the other two features are based on RED concepts, Chapter 6, "Congestion Avoidance Through Drop Policies," covers the basics of RED as well.

All Congestion-avoidance tools consider the queue depth—the number of packets in a queue—when deciding whether to drop a packet. Some tools weigh the likelihood of dropping a packet based on the IP precedence or IP DSCP value. Finally, one congestion-avoidance tool considers the actual flow in addition to the queue depth and weight, and treats some flows differently based on their characteristics. Table 2-5 lists the tools and the various points for comparison.

Table 2-5 *Comparison of Congestion-Avoidance Tools*

Tool	Can Be Enabled in IOS?	Weights Based on IP Precedence or DSCP?	Considers Flow Information When Deciding to Drop Packets?
Random Early Detection (RED)	No	No	No
Weighted Random Early Detection (WRED)	Yes	Yes	No
Flow-Based Random Early Detection (FRED)	Yes	Yes	Yes

All functions listed are based on 12.2 mainline IOS code levels.

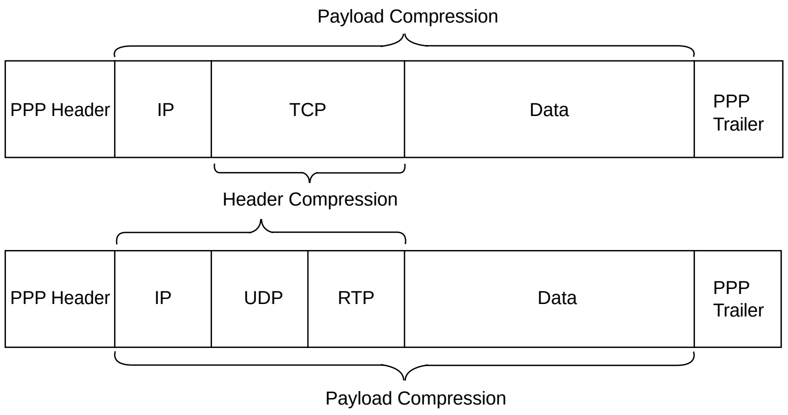
Chapter 6 covers each of the congestion-avoidance tools in detail.

Link Efficiency

The category of link efficiency encompasses two real topics: compression and fragmentation. Rather than treat these topics in two separate chapters, I have included them in one chapter (Chapter 7, "Link-Efficiency Tools") to match the organization of the Cisco QoS courses (and the IOS documentation to some degree).

Compression reduces bandwidth utilization by making packets smaller before transmission. Two general types of compression tools exist in IOS—payload compression and header compression. Payload compression compresses the "packet"—the portion of the data link frame between the frame header and trailer. Header compression compresses just particular headers. Figure 2-6 shows the typical scope of the compressed portions of a frame over a PPP link.

Figure 2-6 *Scope of Compression for Payload and Header Compression Types*



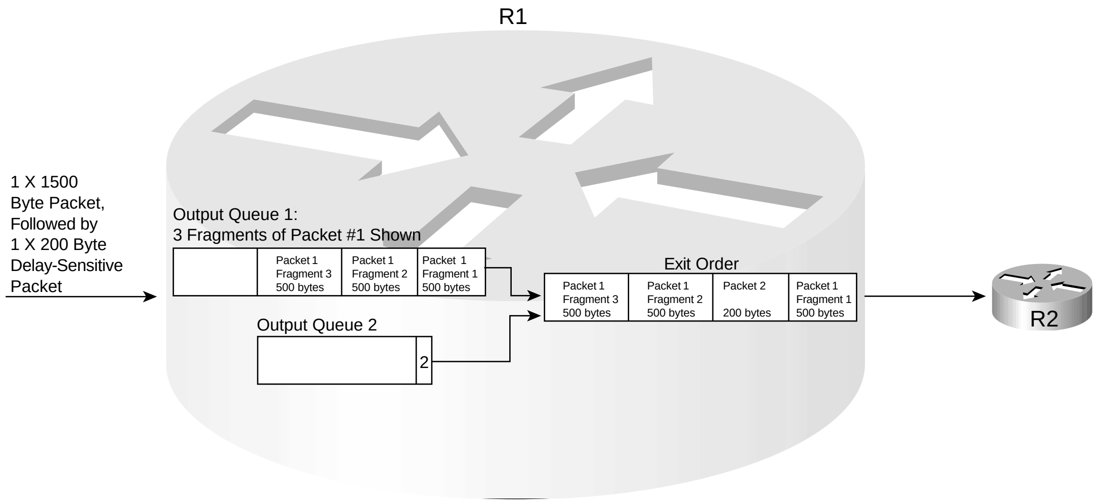
Compression tools differ in how much CPU load they create and which parts of the frame they compress. Based on the CPU load and what is compressed, you can make good decisions about when to use each tool.

Payload compression can be applied to all packets, with some good results. Suppose that the compression algorithm manages to compress *x* bytes of payload into (1/2*x*)—a reasonable 2:1 compression ratio. The router saves a lot of bandwidth with the compression a 1500-byte packet into a 750-byte packet. Given the variation and unpredictable nature of the contents of the packets, compression ratios between 2:1 and 4:1 are reasonable with payload compression.

Header compression takes advantage of the fact that the headers being compressed are predictable. Much larger compression ratios can be achieved, many times with less CPU load than payload compression. However, header compression only operates on headers. For instance, compressed RTP compresses packets with IP/UDP/RTP headers, as shown in Figure 2-6. The 40 bytes of the IP/UDP/RTP headers compress to between 2 and 4 bytes. For a minimum packet size of 60 bytes, typical of G.729 VoIP calls, cRTP reduces the packet from 60 bytes to between 22 to 24 bytes, a significant improvement.

The other major category of link-efficiency tools is link fragmentation and interleaving (LFI), also just called fragmentation. The concept is simple: When a router starts sending a packet, it never just stops sending that packet in order to send a higher-priority packet—it finishes sending the first packet, and then sends the higher-priority packet. On slow links, the time it takes for one large packet to be serialized may cause too much delay, particularly for VoIP and video traffic. LFI tools fragment large packets into smaller packets, and then interleave the high-priority packet between the fragments. For instance, it takes 214 ms to serialize one 1500-byte packet over a 56-kbps link—which blows the VoIP one-way delay budget. (As described in Chapter 7, Cisco recommends that you considered LFI when the link speed is 768 kbps or less.) Figure 2-7 shows the process of fragmentation.

Figure 2-7 *Link Fragmentation and Interleaving*


R1
1 X 1500 Byte Packet, Followed by 1 X 200 Byte Delay-Sensitive Packet
Output Queue 1: 3 Fragments of Packet #1 Shown
Packet 1 Fragment 3 500 bytes
Packet 1 Fragment 2 500 bytes
Packet 1 Fragment 1 500 bytes
Output Queue 2
2
Exit Order
Packet 1 Fragment 3 500 bytes
Packet 1 Fragment 2 500 bytes
Packet 2 200 bytes
Packet 1 Fragment 1 500 bytes
R2

Without LFI, packet 2 has to wait 214 ms on the 1500-byte packet. With LFI fragmenting packet 1 into 3 parts, the serialization delay is reduced to about 72 ms.

Link-Efficiency Tools: Summary

Most link-efficiency tools have a very specific application that becomes obvious when you discover what each tool can and cannot do. Not all compression and LFI tools support every type of data link. Both compression and LFI tools may operate on a subset of the packets that exit an interface. For instance, TCP header compression just compresses IP packets that also have TCP headers. Frame Relay fragmentation only operates on a subset of the packets, based on which of two styles of fragmentation is configured. So depending on what you want to accomplish with link efficiency, you can typically use a single tool. Table 2-6 lists the link-efficiency tools and some of the pertinent comparison points.

Chapter 7 covers each of the link-efficiency tools in detail.

Table 2-6 *Comparison of Link-Efficiency Tools*

Tool	Data Links Supported	Types of Packets to Which Tool Can Be Applied
Payload compression	All; recommended on serial links (T/1, E/1, and slower)	All IP packets
RTP header compression (cRTP)	All; recommended on serial links (T/1, E/1, and slower)	All packets with IP/UDP/RTP headers
TCP header compression	All; recommended on serial links (T/1, E/1, and slower)	All IP packets with TCP headers
Multilink PPP fragmentation and interleaving (MLPPP LFI)	Multilink PPP	All packets larger than a configured length
Frame Relay fragmentation (FRF*)	Frame Relay	All packets larger than a configured length (FRF.12) or all non-VoFR frames (FRF.11c)
Link fragmentation and interleaving for Frame Relay and ATM VCs	Frame Relay and ATM	All IP packets

All functions listed are based on 12.2 mainline IOS code levels.

* The Frame Relay Forum is often referred to as FRF; their document names tend to begin with the letters FRF as well. The QoS feature called Frame Relay fragmentation is also referred to as FRF. In this book, FRF refers to fragmentation, and not the Frame Relay Forum, unless otherwise stated.

Call Admission Control and RSVP

Call admission control (CAC) protects network bandwidth by preventing more concurrent voice and video flows than the network can support. By doing so, it not only protects data traffic, but it also protects the quality of voice and video calls that are already set up. If a network engineer designs a network to support 3 concurrent G.729 calls, for instance, which take roughly 85 kbps, depending on the data links used, but if 10 concurrent calls occur, taking roughly 285 kbps, many bad things happen. Data applications may not get enough bandwidth. Also all the voice calls tend to degrade quickly—not just the “extra” calls.

Cisco defines three general categories of CAC tools: local, measurement based, and resource based. When a device makes a decision about whether to allow a new call to occur, a CAC decision has been made. When a single device makes a CAC decision without asking other devices, or without any perception of network health beyond itself, the CAC tool used is considered to be a local CAC tool—in other words, the decision was made locally. Measurement-based CAC tools measure the network performance and load by sending probes that measure delay and packet loss through the network. This “measurement” of the current health of the network determines whether a new call is allowed. The third category, resource-based CAC, uses information about usage of resources to decide whether a new call occurs. Resources include things such as voice digital signal processors (DSPs), available DS0 channels on trunks, and bandwidth inside the IP network. Table 2-7 lists a general definition of each type of CAC.

Table 2-7 *Definitions of Types of CAC Tools*

Type of CAC Tool	Definition
Local	When making the decision to allow or disallow a new call, a single device considers information about itself, but does not consider information about the current network behavior, and does not ask about resource utilization in other devices in the network.
Measurement based	When making the decision to allow or disallow a new call, a single device considers the results of probes that measure delay and loss in the network.
Resource based	When making the decision to allow or disallow a new call, a single device considers information about resource utilization, as perceived by itself and by other devices in the network. Resources include DSPs, DS0 channels on trunks, and IP bandwidth.

The Resource Reservation Protocol (RSVP) enables devices to reserve bandwidth inside the IP network. RSVP is one of the resource-based VoIP CAC methods, because IP bandwidth is a resource in the network. However, RSVP can be used as a more general tool, not just for VoIP CAC. Chapter 8, "Call Admission Control and QoS Signaling," happens to be a good place to cover RSVP in some depth, because some RSVP discussion is already required for the CAC coverage in that chapter.

CAC Tools

Table 2-8 lists each of the specific CAC tools. The table categorizes each tool based on whether it is a local CAC, resource-based CAC, or measurement-based CAC mechanism. Chapter 8 covers each tool to varying levels of detail.

Table 2-8 *Comparison of CAC Tools*

Tool	CAC Type	CAC Decision Is Based on Whether . . .
Physical DS0 limitation	Local	A DS0 channel is available on a trunk
Max-connections	Local	A configured number of maximum connections on the dial peer used for the call has been exceeded
Voice-bandwidth for Frame Relay	Local	VoFR PVC CIR has been exceeded; VoFR only (not VoIP)
Trunk conditioning	Local	Keepalives sent to other end of network keep working or not; used for "connection trunk" calls only*

continues

Table 2-8 *Comparison of CAC Tools (Continued)*

Tool	CAC Type	CAC Decision Is Based on Whether . . .
Local Voice Busy-Out (LVBO)	Local	One or more local interfaces fail; if they all fail, no IP connectivity would exist, so the trunk is placed in busy-out state
Advanced Busy-Out Monitor (AVBO)	Measurement based	Probe measurements are better than a configured "impairment factor"; if value is higher, the entire trunk is placed in busy-out
PSTN Fallback	Measurement based	Probe measurements are better than a configured "impairment factor"; instead of busying-out the trunk, calls are allowed or rejected on a call-by-call basis
Resource Availability Indicator (RAI)	Resource based	Terminating gateway's calculation of available DSPs and DS0s implies it has adequate number of resources or not
Gatekeeper Zone Bandwidth (GK Zone Bandwidth)	Resource based	Gatekeeper believes that the bandwidth into the zones in question has been oversubscribed or not
Resource Reservation Protocol (RSVP)	Resource based	The RSVP reservation request flow can, both at call setup and ongoing throughout the call, reserve the needed bandwidth on all RSVP-supporting links in the IP network

* Trunk conditioning acts like a measurement-based CAC tool in my opinion; the IOS documentation, and the DQOS course, list this CAC tool as a local CAC tool. Cisco QoS exam questions are not based on my opinion, so it is listed as a local CAC tool in this book!

Management Tools

Cisco provides several management tools, and features of management tools, that assist in managing the QoS policies and configuration in a network. Table 2-9 lists these tools and provides a short description of the functions of each tool.

Table 2-9 *QoS Management Tools*

Tool	Features
QoS Device Manager (QDM)	Uses code that is stored in Flash memory, running inside each router; user can use a web browser to manage QoS configuration and view statistics for an individual router. The product is free.
QoS Policy Manager (QPM)	Application runs on Windows NT or Windows 2000, accessible by a browser. Enables the engineer to manager QoS policies network wide; QPM takes policies and creates QoS configurations, stages, implements, and allows back-out of QoS configurations throughout the network.

Table 2-9 *QoS Management Tools (Continued)*

Tool	Features
Service Assurance Agent (SAA)	The feature formerly known as Response Time Reporter (RTR), SAA is a feature of IOS. This feature can be configured to create, send, respond to, and measure the performance of probe packets. Measurement-based CAC mechanisms create SAA probes on routers; the routers send and receive the probes; and then the routers tell SAA the results.
Internetwork Performance Monitor (IPM)	Formerly a separate product, this Cisco Works feature monitors network performance in real time. It also provides an easy GUI interface to configure SAA probes.
Service Management Solution (SMS)	SMS is a feature of Cisco Works that provides performance statistics similar to IPM, but with the intent to save historical data and to provide reporting about whether configured service level agreements (SLAs) are being met. IPM is used for operating the network and looking at current network statistics; SMS is used for historical reporting, trending, and SLAs.

Summary

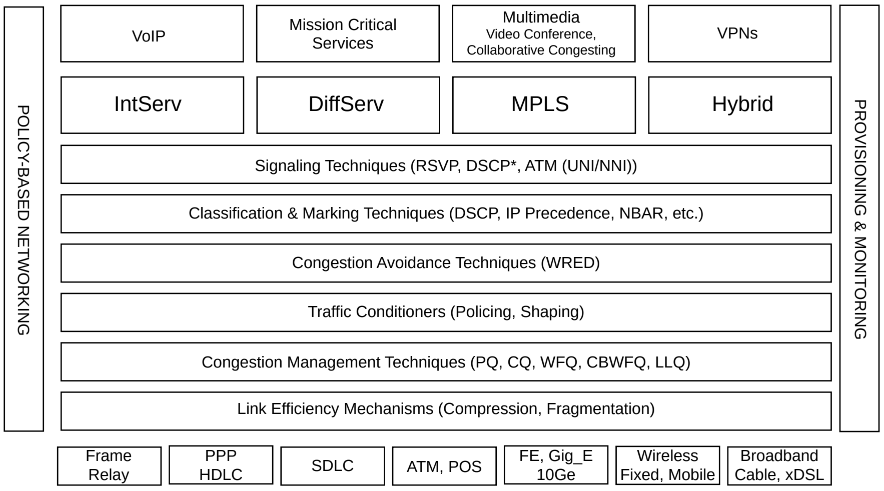
If you do not work with QoS tools every day, and you are reading this book to pass one of Cisco's QoS exams, you may be feeling overwhelmed after reading the first section of this chapter! However, do not be dismayed—the rest of the chapters in this book are devoted to a deeper description of the tools introduced here. You need to memorize a lot of facts for the exam—facts for which you can just refer to this book after you pass the exam—but each chapter purposefully includes the most important facts to memorize in tables, and these tables are included in the "Foundation Summary" section of each chapter.

Cisco has created a "QoS framework" that shows the various components of a network relating to QoS. The exam does cover the QoS framework, and each chapter reminds you about which parts of the framework are covered in that chapter. Figure 2-8 shows the Cisco QoS framework.

The bottom half of the figure shows five key categories of QoS tools, all of which are covered in Chapters 3 through 7 of this book. Chapter 8 covers CAC and QoS signaling; RSVP is mentioned as one of the signaling features in the upper part of the framework. Management tools, as covered in Chapter 9, are shown on the right side of the QoS framework. Not all parts of the QoS framework are covered on the Cisco QoS exams. The framework itself is covered on the exams, so make sure that you are familiar with the terms and the organization of this diagram.

The row of the framework that lists IntServ, DiffServ, MPLS, and Hybrid, refer to different QoS architectures. Two of these architectures are covered on the Cisco QoS exams—IntServ and DiffServ. Most of the rest of this chapter explains these two QoS architectures.

Figure 2-8 *The Cisco QoS Framework*


VoIP
Mission Critical Services
Multimedia Video Conference, Collaborative Congesting
VPNs
IntServ
DiffServ
MPLS
Hybrid
Signaling Techniques (RSVP, DSCP*, ATM (UNI/NNI))
Classification & Marking Techniques (DSCP, IP Precedence, NBAR, etc.)
Congestion Avoidance Techniques (WRED)
Traffic Conditioners (Policing, Shaping)
Congestion Management Techniques (PQ, CQ, WFQ, CBWFQ, LLQ)
Link Efficiency Mechanisms (Compression, Fragmentation)
POLICY-BASED NETWORKING
PROVISIONING & MONITORING
Frame Relay
PPP HDLC
SDLC
ATM, POS
FE, Gig_E 10Ge
Wireless Fixed, Mobile
Broadband Cable, xDSL

The Good-Old Common Sense QoS Model

Most people already have some idea of what would be useful, and not so useful, regarding implementing QoS in a network. If you have been reading this book from the beginning of Chapter 1, you already know about the how QoS tools affect bandwidth, delay, jitter, and loss. You also know some of the traffic characteristics of voice, video, and data flows. You have seen the basics of how each general category of QoS tool works (Chapter 1), and you have at least seen a list of QoS tools in IOS (within this chapter).

With that in mind, you can mostly ignore DiffServ and IntServ and succeed at deploying QoS. Of course, you need to know more about each specific QoS tool, and you need to know how to configure the tools. You need to follow some good QoS design principles as well. However, you do not have to understand a lot about the DiffServ and IntServ models of QoS to succeed with QoS deployments.

However, most readers will find information about DiffServ and IntServ important for at least two reasons. First, the information is on the Cisco QoS exams! The other reasons is that the two formal models for QoS, DiffServ and IntServ, formalize and describe good QoS design models. You can deploy QoS without considering the concepts behind DiffServ and IntServ, but you will be better prepared for deploying QoS if you take the time to understand each model.

For those who are relatively new to QoS, before covering the depth and details in the upcoming chapters, you should consider some basic terms, concepts, and strategies for applying QoS in a network. This book summarizes these common strategies into what I call the “Good-Old Commonsense” (GOCS) QoS model, which summarizes what most people think of as just plain common sense about how to apply QoS. If you understand the GOCS model, you will already understand more than half of the concepts behind DiffServ, without letting the vast amount of terminology get in the way.

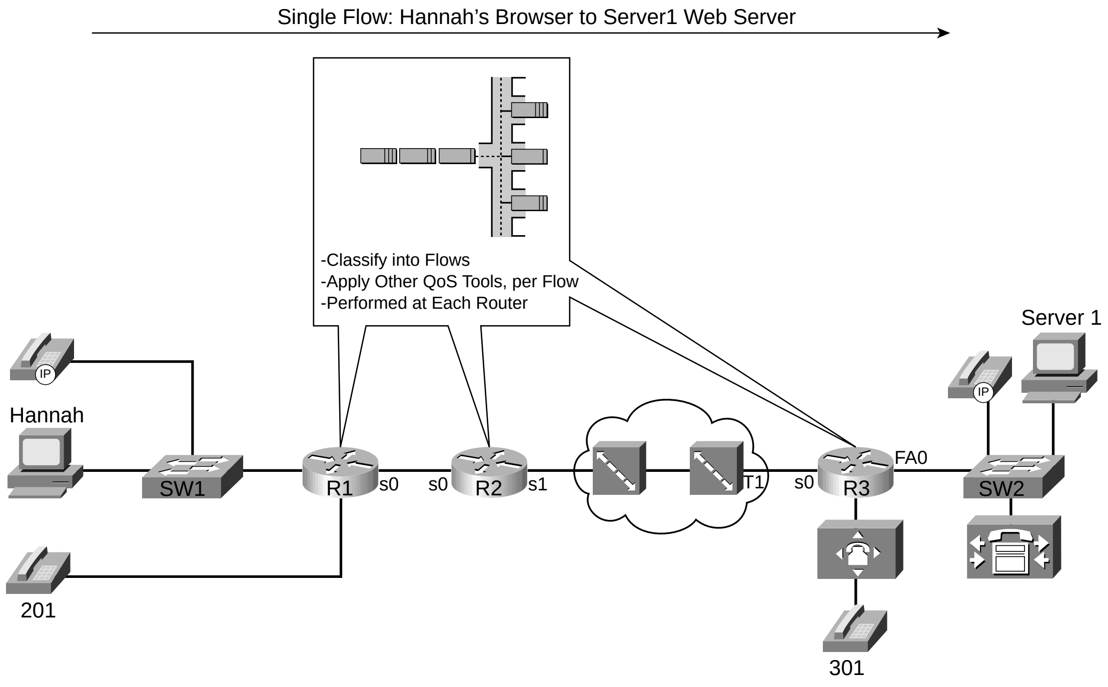
GOCS Flow-Based QoS

A flow consists of all the packets about which the following are true:

- All packets use the same transport layer protocol (for instance, UDP or TCP).
- All packets have the same source IP address.
- All packets have the same source port number.
- All packets have the same destination IP address.
- All packets have the same destination port number.

The slightly different, but just as exact definition, is that a flow consists of the packets between two IP sockets. For instance, a web browser on a PC, connecting to a server, creates a flow. (Actually, in the case of HTTP, it may create multiple TCP connections, which would actually be multiple flows.) Regardless, consider Figure 2-9. In this network, one flow exists from Hannah to Server1's web server.

Figure 2-9 *GOCS Approach to QoS for a Single Flow*

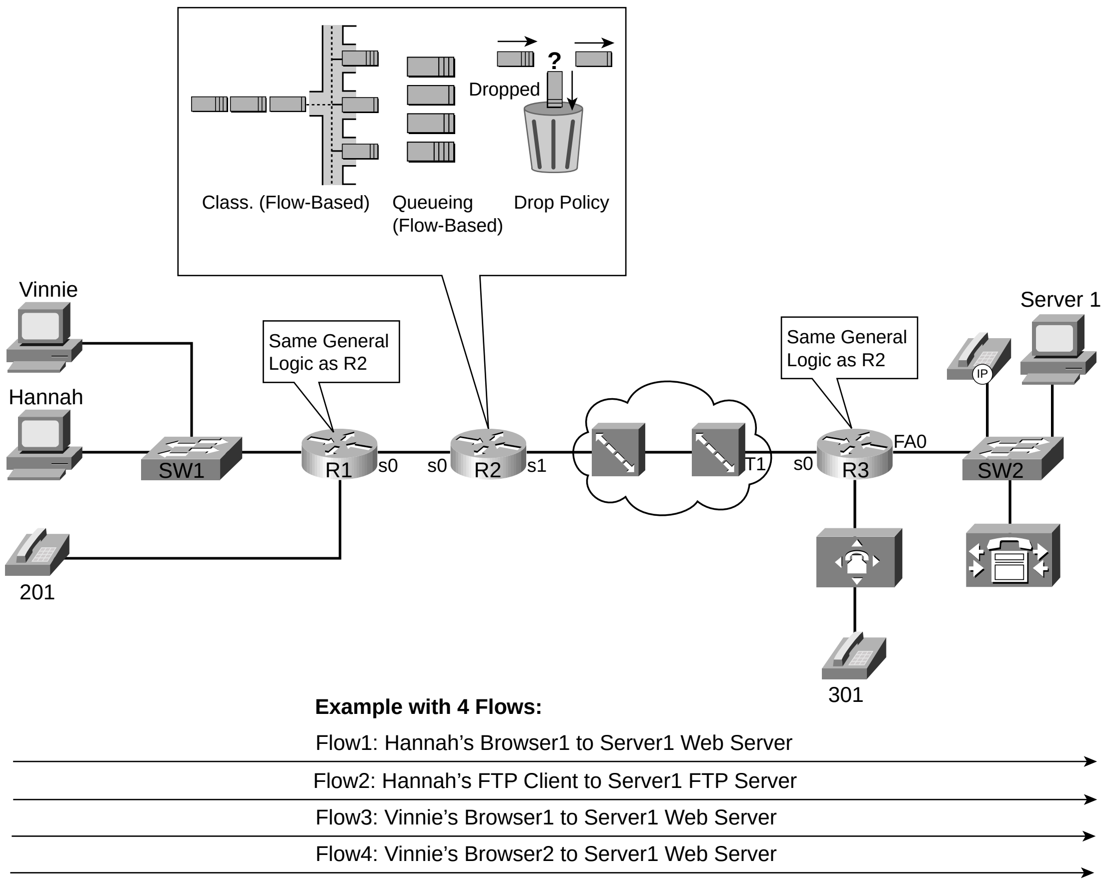

Single Flow: Hannah's Browser to Server1 Web Server
-Classify into Flows
-Apply Other QoS Tools, per Flow
-Performed at Each Router
IP
Hannah
SW1
R1
s0
s0
R2
s1
T1
s0
R3
FA0
SW2
Server 1
IP
201
301

The single flow shown in the figure consists of the packets sent by Hannah to Server1. Note that flows are unidirectional—in effect, two flows, one in each direction, would exist. To reduce the complexity, the samples in this section show flows going left to right only.

Flow-based QoS tools behave like the logic shown in the figure. Some QoS tools recognize flows and treat packets in one flow differently than another flow. So, common sense may imply that the QoS tools must first identify the packets that belong to this single flow, and then take some QoS action—such as queuing, LFI, shaping, and so on—for the packets in that flow. The tools may be applied for packets entering an interface, and other QoS tools may be applied for packets exiting an interface. Some tools may even be applied on the LAN switches. Some QoS tools may be applied in the Frame Relay cloud—but those are not typically under your control.

Real networks will have a widely varying number of flows, but the general ideas behind flow-based QoS tools do not change when more flows exist. Take a look at Figure 2-10, which shows a pretty small network with only four flows at this point.

Figure 2-10 *GOCS Approach to QoS for Multiple Flows*


?
Dropped
Class. (Flow-Based)
Queueing (Flow-Based)
Drop Policy
Vinnie
Hannah
201
SW1
Same General Logic as R2
R1
s0
s0
R2
s1
T1
s0
Same General Logic as R2
R3
FA0
SW2
Server 1
IP
301
Example with 4 Flows:
Flow1: Hannah's Browser1 to Server1 Web Server
Flow2: Hannah's FTP Client to Server1 FTP Server
Flow3: Vinnie's Browser1 to Server1 Web Server
Flow4: Vinnie's Browser2 to Server1 Web Server

This figure shows four flows. Even though Vinnie sends all packets for both flows to Server1, they are in two different flows, because each of the two browser windows would open separate TCP connections, with different source TCP port numbers. (Note: HTTP 1.1 could cause multiple connections to be opened; FTP uses a control and data connection, which would result in two flows.) However, assume that only four flows exist at this point.

How does the GOCS model with flow-based tools change with multiple flows? Well, not much. Each flow must be identified, based on its unique source/destination address/port values. The QoS actions at each router may be different for each flow—for instance, maybe FTP just gets leftover bandwidth, or maybe Hannah gets better treatment for her web flow than does Vinnie. The reasons and rationale behind deciding what traffic gets what QoS treatment will change from network to network, but the basic process works the same:

- Identify each packet, *determining which flow* it belongs to.
- Apply some QoS action to the packets in each *flow*.
- The QoS actions on a single router may be different for each *flow*.
- The QoS actions among all routers may be different for each *flow*.

Flow-based QoS tools provide some advantages and some disadvantages. Because each separate flow is identified, the QoS tools can literally provide different levels of service to every flow. A single queue could be used for each flow, for instance, giving each a different reserved bandwidth. With a large number of flows, however, the overhead associated with keeping state information about each flow can be a lot of work. Imagine a router with 1000, or even 10,000, concurrent flows, which would not be surprising in the Internet—and then imagine the overhead in trying to perform queuing with 1000 or 10,000 queues! So, flow-based QoS tools provide a great deal of granularity, but they do not scale as well as some other QoS tools that do not consider flows (particularly with very large networks or the Internet).

Engineers gain another advantage and disadvantage when configuring flow-based QoS tools. Suppose that your job is to explicitly configure the routers in Figure 2-10 as to which source and destination IP addresses and ports to use to find each flow. And instead of 4 flows, there are 1000 concurrent flows. Over the course of a day, there may be hundreds of thousands of flows. Your job is to find the details that make each flow unique and configure it! Well, that would be rather ridiculous, a lot of work, and mostly impractical. Therefore, flow-based tools typically require no configuration to match and classify packets into a flow. However, some configuration control is lost.

The following list summarizes the key points about flow-based QoS tools:

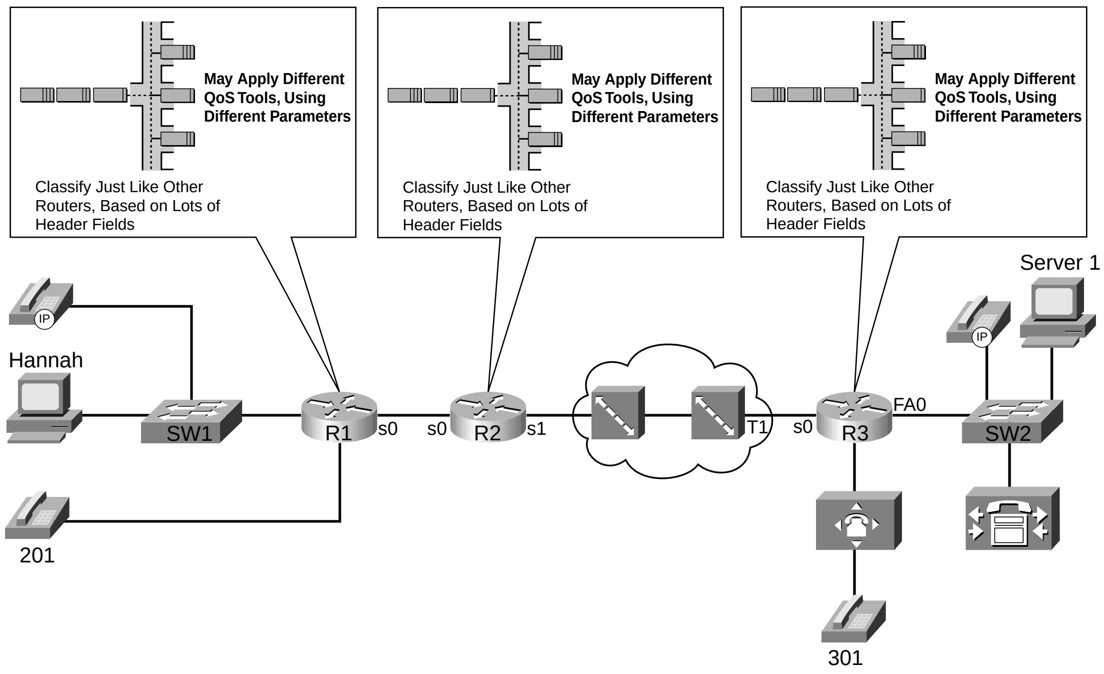
- Flow-based QoS tools automatically recognize flows based on the source and destination IP address and port numbers, and the transport layer protocol.
- Flow-based tools automatically identify flows, because it would be impractical to configure parameters statically to match the large number of dynamically created flows in a network.

- Flow-based tools provide a great amount of granularity, because each flow can be treated differently.
- The granularity may create scaling problems when the number of flows becomes large.

GOCS Class-Based QoS

Most QoS tools do not need to differentiate between each flow. In the Figure 2-10, for instance, flows to web Server1 were identified. Most network engineers would want to treat those collective web flows the exact same way with their QoS tools. Therefore, most QoS tools tend to operate on the idea of a category, or class, of flows and packets. Consider Figure 2-11, for example, which has thousands of flows, all of which are classified into four types of traffic.

Figure 2-11 *GOCS Approach to QoS with Classes*


May Apply Different QoS Tools, Using Different Parameters
Classify Just Like Other Routers, Based on Lots of Header Fields
May Apply Different QoS Tools, Using Different Parameters
Classify Just Like Other Routers, Based on Lots of Header Fields
May Apply Different QoS Tools, Using Different Parameters
Classify Just Like Other Routers, Based on Lots of Header Fields
Server 1
IP
Hannah
SW1
R1
s0
s0
R2
s1
T1
s0
R3
FA0
SW2
IP
201
301

Class-based QoS tools do not have to identify each flow. However, they do need to identify packets based on something in the packet header—such as TCP destination port 80 for web traffic—and consider that traffic to be in one category or class for QoS treatment. Once again, the reasons and rationale behind deciding what traffic gets what QoS treatment changes from network to network, but the basic process works the same, but per class rather than per flow:

- Identify each packet, determining which class it belongs to.
- Apply some QoS action to the packets in each class.
- The QoS actions on a single router may be different for each class.
- The QoS actions among all routers may be different for each class.

Unlike flow-based QoS tools, class-based QoS tools typically require the engineer to specify exactly what must be seen in the packet header to classify a packet. If this network currently has 4 flows to the web server, or 400, or 4000, if the classification criteria just states "all TCP port 80 traffic," no additional configuration is required as the network scales. Both flow-based and class-based tools need to examine every packet to classify the packet into the appropriate flow or class. Because class-based tools typically only need a small number of classifications, however, the tool can reasonably be configured to specify the types of traffic that get added to each class.

Class-based QoS tools can use more complex rules to classify packets than do flow-based tools. For instance, a class-based tool can examine subsets of the IP addresses (matching a subnet, for example), the incoming interface, the URL for web traffic, and anything that an IP ACL can match. For flow-based tools, the router always look at five fields, all in the IP header—Source and Destination Address, Source and Destination Port, and the Protocol Type field (which identifies the transport layer protocol). In short, classification options for class-based tools tend to be much more varied and functional, but they require more configuration work to take advantage of the different options.

Flow-based and class-based QoS tools both have a useful place in a QoS strategy for a network. Most QoS tools tend to be based on general classes, as opposed to looking at each individual flow.

Classification and Marking at the Edge

Class-based tools have advantages and disadvantages as well. The engineer can exercise great control over the packets placed into the classes or categories used in a network. Because a small number of categories are used, in most cases fewer than 10, the configuration scales. For instance, you build a network and choose four categories for packets. As the network grows, you can keep using the same four categories, and even with growing numbers of packets and routers, the number of classes can remain the same.

However, the same explicit classification configuration details can be the cause of two particular types of problems. For instance, suppose that you have 500 sites, with at least one router and several switches at each site. At each site, you want to classify packets into the same four categories. You need to use six different IOS QoS tools—which is not an unreasonable choice, because you might need different types of queuing, shaping in some cases, fragmentation only in some cases, compression in other cases, and so on. Each of the six tool's classification configuration differs. In some cases, the switches can perform classification and marking at Layer 3, but not in other cases. You also have 5 different engineers, plus 10 operational staff, who have to enable secret passwords to the routers and switches. What are the chances that the classification configurations will be correct, on each router, and stay that way? Even if the configurations remain correct and unchanged, do you really want all routers looking at 10 different things in every packet header, taking CPU cycles, to classify each packet? Well, common sense tells us that there may be a better method.

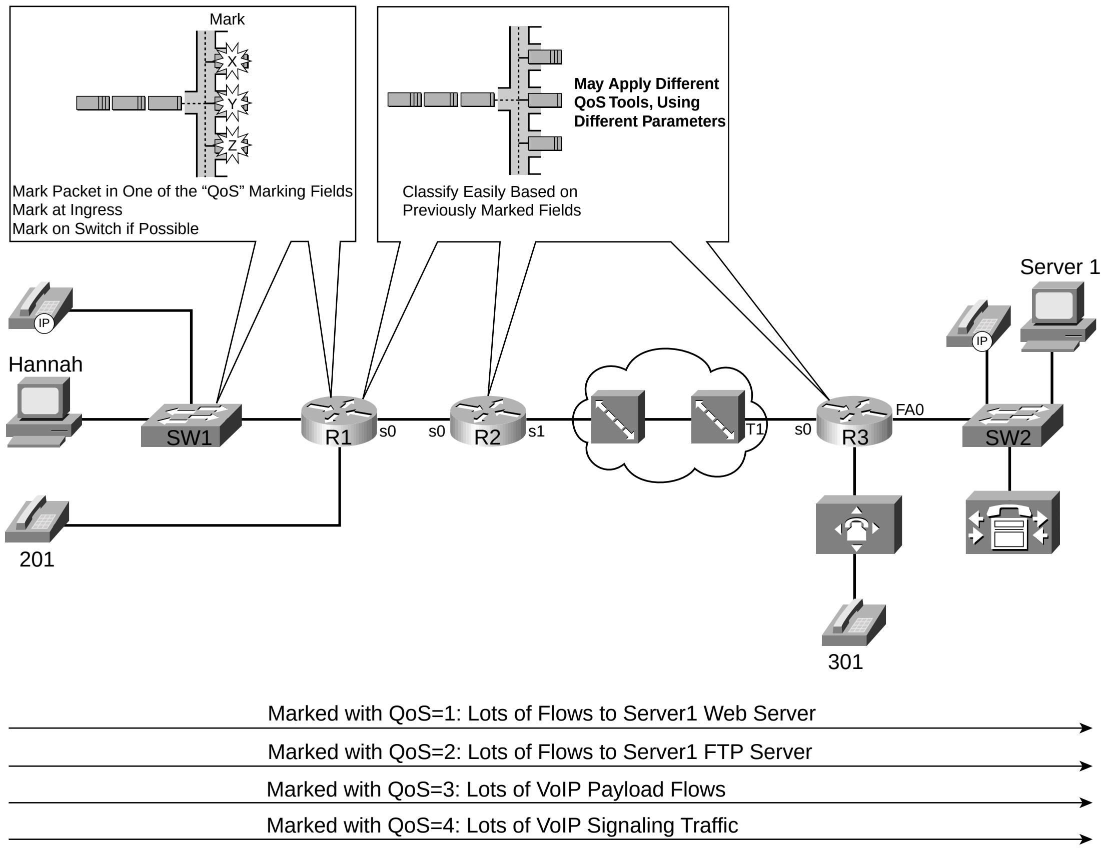
You can use the Cisco QoS Policy Manager (QPM) to overcome the configuration correctness and consistency problem. QPM creates the QoS configurations for you, based on your input about QoS policies using a GUI. QPM loads the configurations, and re-verifies the QoS configurations to discover whether changes have been made. It can also reconfigure a router after someone has inadvertently changed the QoS configuration—automatically. Any large QoS implementation begs for the use of QPM.

Through good QoS design, you can solve the problem of having every router in the network examining a lot of fields in every packet header. This design choice reduces the complexity of the configurations as well. Packets can be classified and marked near the ingress points of traffic into the network; the QoS tools in the center of the network can just look for the marked field in the packet header, as shown in Figure 2-12.

Classifying and marking the packets near the edge of the network simplifies the classification logic and configuration at the rest of the routers in the network. For instance, the figure shows the same classes as Figure 2-11, but with classification and marking performed near the ingress edge of the network. Ideally, packets should be marked even before they reach the first router, typically by a LAN switch. If not, packets entering R1's E0 interface can be classified and marked. R1, R2, and R3 all still need to classify packets, but now the classification details just look for a single well-known field inside the IP header. (Note: The two IP header fields used are the IP Precedence and IP DSCP fields.) This design choice simplifies the QoS configuration and reduces the processing effort for the intermediate routers. (Note: Classification and marking can happen in the switches, IP Phones, and end-user computers—all of which are discussed in detail in Chapter 3.)

Proper planning prevents poor policies when using the "class and mark near the edge" GOCS strategy. For instance, the Figure 2-12 shows four classes, one of which is "all web traffic." Four classes may work well for this network, but other networks may need to treat web traffic to Server1 differently than web traffic to Server2 or Server3. Before beginning to deploy QoS, the network architects and engineers should agree on what types of traffic need to be in a separate class. Then they must classify and mark at the edge and use simplified classification configurations, based on the marked fields in the IP header, in the interior of the network. With flow-based tools, there is no requirement to plan the different traffic classifications, because flow-based tools categorize based on flow.

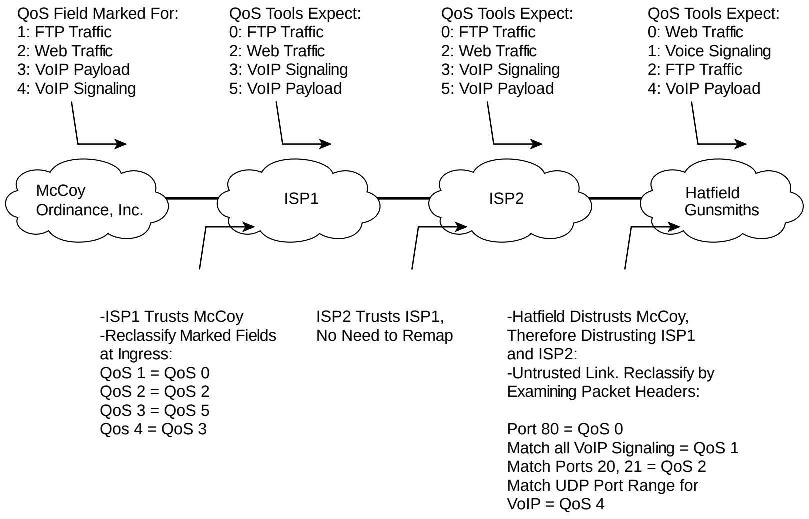
Figure 2-12 *GOCS Design: Mark Packets near the Edge of the Network*


Mark
X
Y
Z
Mark Packet in One of the "QoS" Marking Fields
Mark at Ingress
Mark on Switch if Possible
May Apply Different QoS Tools, Using Different Parameters
Classify Easily Based on Previously Marked Fields
Hannah
SW1
R1
s0
s0
R2
s1
T1
s0
R3
FA0
SW2
Server 1
IP
201
301
Marked with QoS=1: Lots of Flows to Server1 Web Server
Marked with QoS=2: Lots of Flows to Server1 FTP Server
Marked with QoS=3: Lots of VoIP Payload Flows
Marked with QoS=4: Lots of VoIP Signaling Traffic

It is possible to plan QoS classes for an enterprise network. However, planning classes for all networks in the Internet is a bit more challenging! For instance, although everyone may agree that VoIP and video may need different treatment than data over the Internet needs, they probably do not agree about differentiation between different types of data traffic. Also a company paying a premium to a service provider may expect better service—translated, better QoS treatment—so the classification may be based on the source IP addresses, and may be different for different ISPs.

Therefore, although the political and business challenges of Internet-scale QoS may be difficult, cases will still exist where QoS can be implemented over the Internet. Consider Figure 2-13, which shows two companies, each connected to two different ISPs.

Figure 2-13 *QoS Between Different Companies over the Internet*



Two key QoS issues exist in this case. First, the parties must agree to the different classifications of packets. In this example, all four networks agree to the need for four classes. (Agreement will not always occur!) For instance, McCoy Enterprises may want a different class for customer web traffic versus supplier web traffic. Even if all companies want these same general categories, it is difficult to effectively match the correct traffic for all companies connected to the Internet, because every company has different customers and suppliers. Therefore, QoS across the Internet may well end up using general categories—categories such as voice, video, voice/video signaling, important data, and not-so-important data.

Even with general categories agreed upon, not every network chooses to mark the IP packets with the same value to denote the same class. Figure 2-13 shows just such a case, where ISP1 and ISP2 agree to the values to use when marking packets, but McCoy Ordinance and Hatfield Gunsmiths, two long-time competitors, do not agree on what marked values to use.

Three commonsense QoS design choices help overcome common Internet QoS issues:

- If neighboring autonomous systems do not agree about what traffic should be in each class, each autonomous system should reclassify ingress traffic based on more complex matching of packets based on the large variety of packet header fields.
- If neighboring autonomous systems do agree about the classes, but not the marked values, each autonomous system should reclassify ingress traffic based on simple matching of packets based on the previously marked fields in the IP header, as shown in Figure 2-13.

- If an autonomous system does not trust its neighbor regarding QoS, neighboring autonomous systems should also reclassify traffic at ingress, based on detailed matching of packets.

The GOCS QoS model introduces some basic concepts for QoS. The following two sections cover in detail the two formal QoS architectural models, DiffServ and IntServ. Table 2-10 summarizes the key points from the GOCS model.

Table 2-10 *Summary of GOCS Features*

Feature	**Comments**
Flow	A flow is a unidirectional set of packets, sent from one source IP address and port, to one destination IP address and port, using the same transport layer protocol.
Flow-based QoS tools	Flow-based tools automatically recognize flows without explicit classification configuration. Each flow can be treated differently, providing a great amount of granularity for QoS.
Class-based QoS tools	Class-based tools treat packets differently based on the category or class to which they belong. The number of classes chosen in a typical network is small (typically fewer than 10). Network engineers choose which types of packets are placed into each class, so class-based QoS tools require explicit configuration of classification parameters.
QoS tools that are neither flow or class based	Some QoS tools operate on all traffic entering or exiting an interface. Other tools may work on a predetermined type of traffic. For example, compressed RTP only operates on packets with RTP headers. Therefore, some tools do not need to consider flows or explicitly defined classes.
QoS class planning—enterprise	For a single enterprise network, a considered, thorough analysis of the network can yield a relatively small set of useful QoS packet categories. Agreement to these categories should occur before beginning the classification and marking strategy of marking near the edge.
Classification and marking	Packets should be classified and marked near the edge of the network, as packets enter the network. By doing so, classification configuration on the remaining routers in the packets' path is simplified, reducing processing overhead, complexity, the risk of misconfiguration.
QoS class planning—Internet	For the Internet, anything more than a handful of general class conventions is unlikely to be agreed upon by a large number of ISPs and customers. Cisco suggests a short list with five categories: VoIP payload, video payload, voice/video signaling, important data, and not-so-important important data.
Internet reclassification and re-marking	Between different autonomous systems, reclassification and re-marking may need to occur. If the autonomous systems distrust each other, packets must be matched based on the contents of the various fields in their headers. If the autonomous systems trust each other, and agree on what packets belong to each of the traffic classes, all that may be required is to reclassify and re-mark based on the marked field inside the IP header.

The Differentiated Services QoS Model

If you understood the Good-Old Common Sense (GOCS) model for QoS, you already understand at least half of the concepts behind DiffServ. DiffServ goes into a lot more depth, includes a large number of terms, but the core concepts of DiffServ can be summarized as follows:

- Takes advantage of the scaling properties of class-based QoS tools to differentiate between types of packets, with the goal of "scalable service differentiation in the Internet."
- In a single network, packets should be marked at the ingress point into a network, with other devices making QoS choices based on the marked field.
- The marked field will be in the IP header, not a data-link header, because the IP header is retained throughout the network.
- Between networks, packets can be reclassified and re-marked at ingress into another network.
- To facilitate marking, the IP header has be redefined to include a 6-bit Differentiated Services Code Point (DSCP) field, which allows for 64 different classifications.

To some extent, DiffServ formally defines a QoS architecture using common sense, or "best practices," for QoS design today. Along with the formal definitions comes a lot of terminology—terminology that is purposefully not vendor specific. So, after learning the DiffServ terms, you need to relate them to Cisco tools and terms. But DiffServ is more than just recording some good ideas about QoS—DiffServ defines another useful field in the IP header (DSCP), as well as some conventions for usage of the new DSCP field. Finally, DiffServ defines general categories of QoS functions and the purpose of the tools in each category. This book has already covered those same concepts and terms from Cisco's perspective, so in this chapter, you will read about the DiffServ terms for categories or types of QoS tools and how they relate to Cisco's terms.

DiffServ Specifications and Terminology

DiffServ is defined by the RFCs listed in Table 2-11.

Table 2-11 *DiffServ RFCs*

RFC	Title	Comments
2474	Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers	Contains the details of the 6-bit DSCP field in IP header.
2475	An Architecture for Differentiated Service	This is the core DiffServ conceptual document.
2597	Assured Forwarding PHB Group	Defines a set of 12 DSCP values and a convention for their use.

Table 2-11 *DiffServ RFCs (Continued)*

RFC	Title	Comments
2598	An Expedited Forwarding PHB	Defines a single DSCP value as a convention for use as a low-latency class.
3260	New Terminology and Clarifications for DiffServ	Clarifies, but does not supercede, existing DiffServ RFCs.

The RFCs introduce many new terms. Table 2-12 lists the terms and their definitions. This table provides a reference for study for the Cisco QoS exams; the rest of this section relates the terms to some network diagrams.

Table 2-12 *DiffServ Terminology and Their Definitions*

Term	Definition
Behavior aggregate (BA)	A DS behavior aggregate.
BA classifier	A classifier that selects packets based only on the contents of the DS field.
Classifier	An entity that selects packets based on the content of packet headers according to defined rules.
DS behavior aggregate	A collection of packets with the same DS code point crossing a link in a particular direction.
DS boundary node	A DS node that connects one DS domain to a node either in another DS domain or in a domain that is not DS capable.
DS code point	A specific value of the DSCP portion of the DS field, used to select a PHB.
DS compliant	Enabled to support differentiated services functions and behaviors as defined in [DSFIELD], this document, and other differentiated services documents; usually used in reference to a node or device.
DS ingress node	A DS boundary node in its role in handling traffic as it enters a DS domain.
DS field	The IPv4 header ToS octet or the IPv6 traffic class octet when interpreted in conformance with the definition given in [DSFIELD]. The bits of the DSCP field encode the DS code point, whereas the remaining bits are currently unused.
Dropper	A device that performs dropping.
Marker	A device that performs marking.
Meter	A device that performs metering.

continues

Table 2-12 *DiffServ Terminology and Their Definitions (Continued)*

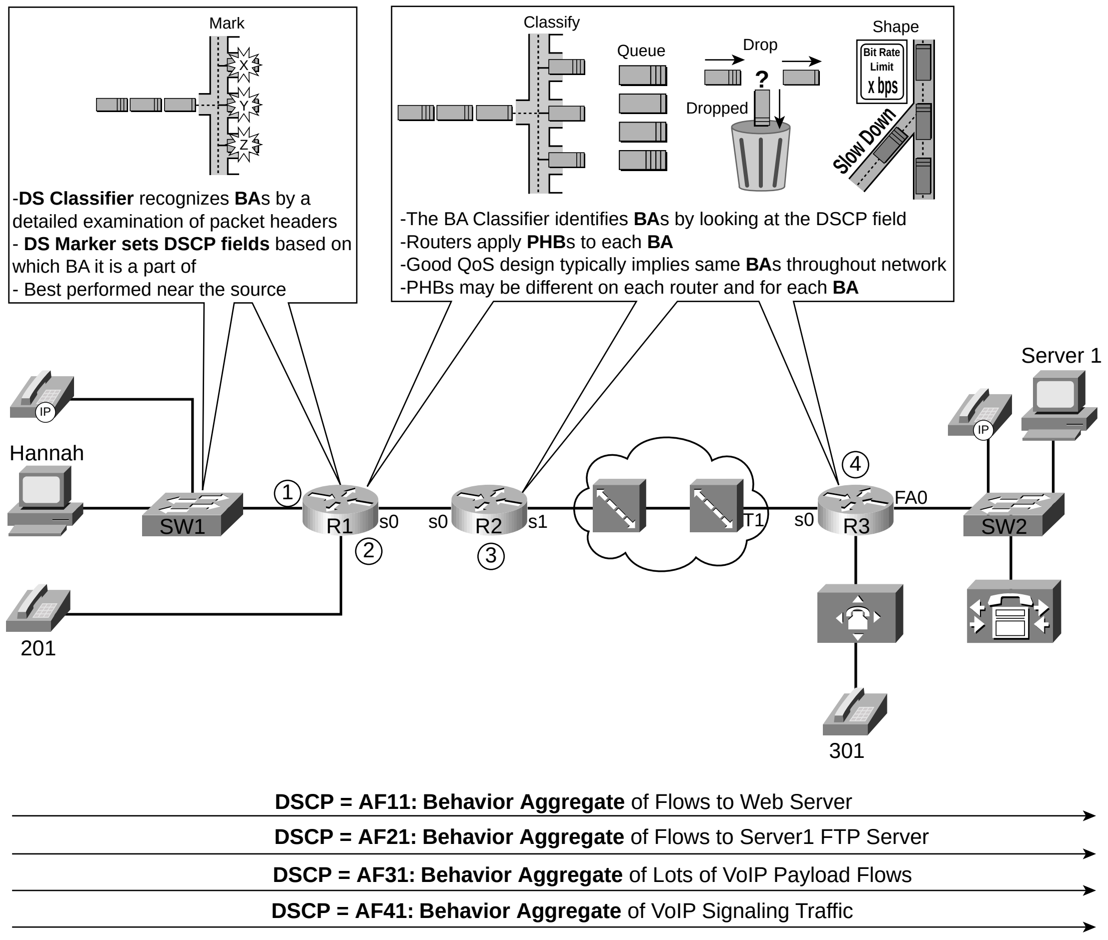
Term	Definition
MF classifier	A multifield (MF) classifier that selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field, protocol ID, source port and destination port.
Per-hop behavior (PHB)	The externally observable forwarding behavior applied at a DS-compliant node to a DS BA.
Policing	The process of discarding packets (by a dropper) within a traffic stream in accordance with the state of a corresponding meter enforcing a traffic profile.
Re-mark	To change the DS code point of a packet, usually performed by a marker in accordance with a TCA.
Shaper	A device that performs shaping.
Traffic conditioner	An entity that performs traffic-conditioning functions and which may contain meters, markers, droppers, and shapers. Traffic conditioners are typically deployed in DS boundary nodes only. A traffic conditioner may re-mark a traffic stream or may discard or shape packets to alter the temporal characteristics of the stream and bring it into compliance with a traffic profile.

Table 2-12 contains material reprinted from RFC 2475.

DiffServ terminology overwhelms most people when first learning the architecture. Not all the DiffServ terms are even listed in the table. In fact, I wouldn't be surprised if you are already wondering which of these terms you really need to know when using QoS and which of these terms you need to know for the Cisco QoS exams. Neither of the exams covered by this book focus on DiffServ as an end to itself. If you glance over the table, and read this section, you should become familiar enough with the terms to do well on those questions on the exams.

The rest of this section explores some examples of usage of DiffServ terminology. The first two terms are "behavior aggregate" and "per-hop behavior." If you read the previous section about the fictitious "GOCS model" for QoS, you already know the concepts behind the terms. Figure 2-14 shows the terms in a figure that is a duplicate of Figure 2-12.

Figure 2-14 *Behavior Aggregates and Per-Hop Behavior*


Mark
X
Y
Z
-**DS Classifier** recognizes **BA**s by a detailed examination of packet headers
- **DS Marker sets DSCP fields** based on which BA it is a part of
- Best performed near the source
Classify
Queue
Drop
?
Dropped
Shape
Bit Rate Limit
x bps
Slow Down
-The BA Classifier identifies **BA**s by looking at the DSCP field
-Routers apply **PHB**s to each **BA**
-Good QoS design typically implies same **BA**s throughout network
-PHBs may be different on each router and for each **BA**
Server 1
IP
Hannah
1
SW1
R1
s0
2
s0
R2
s1
3
T1
s0
4
R3
FA0
SW2
201
301
DSCP = AF11: Behavior Aggregate of Flows to Web Server
DSCP = AF21: Behavior Aggregate of Flows to Server1 FTP Server
DSCP = AF31: Behavior Aggregate of Lots of VoIP Payload Flows
DSCP = AF41: Behavior Aggregate of VoIP Signaling Traffic

Consider the flow of packets from left to right in this network. The following list numbers correspond to the steps in the figure:

1 The packets are classified or categorized by matching fields in the header. For instance, packets with Server1's destination IP address, and destination port 80, would be in the first class. The process of classifying the packets is performed by the DS classifier, MF classifier, or just classifier. The classifier marks the DSCP field inside the IP header; DSCP is a

6-bit field inside the DS field (byte) inside the IP header. Classification and marking are considered to be two different steps—the DiffServ marker actually performs the process of marking the packets. DiffServ defines each class or category of packets as a BA.

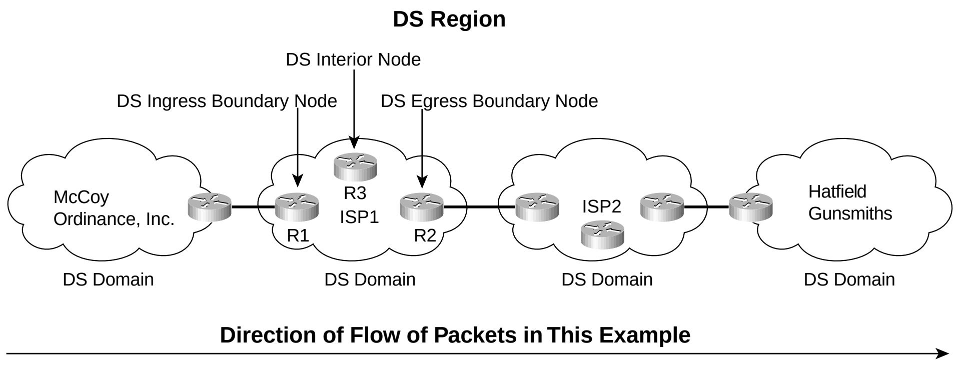
2 Router R1 determines which packets are part of which BA by using a BA classifier. A BA classifier only examines the DSCP field, so technically it differs from an MF classifier, as described in step 1, because the MF classifier can look at many fields besides the DSCP field. When R1 decides to apply a QoS tool to a BA (for example, queuing), the action is called a per-hop behavior. The term PHB makes sense to most people, particularly if you think of it as a per-hop QoS behavior.

3 Router R2 performs the same types of tasks as R1; these tasks are described with the same terms as in step 2. Also note that the PHBs can be, and often are, different on one router to the next. In this case, R2 may want to use a shaping PHB—DiffServ would call the shaping tool a shaper—but because all implemented shaping tools need to calculate the rate at which packets are sent, DiffServ would consider both a meter and shaper to be used.

4 Likewise, no new terminology is required to describe step 4, as compared with the two preceding steps. However, the terms “AF11,” “AF21,” “AF31,” and “AF41” have not yet been defined. DiffServ defines several suggested values to be used in the DSCP field. Most installations do not need all 64 values possible in DSCP. The next section in this chapter covers the details, but in this case, AF11, AF21, AF31, and AF41 represent different DSCP values.

DiffServ models good QoS design specifically to support Internet-scale QoS. Reading through the RFCs, you will notice that DiffServ focuses on issues between different networks. Figure 2-15 shows the same two enterprise networks and the same two ISPs shown in Figure 2-13 in the GOCS section of this chapter. The figure shows examples of several of the DiffServ terms that relate to interconnecting networks.

Figure 2-15 *DiffServ Domains, Regions, and Nodes*



The terms in this figure only apply in cases where multiple organizations' networks are interconnected. The entire figure comprises one DS region, which includes connected networks that are providing differentiated services. Each individual network, typically an autonomous system, is a single DiffServ domain.

The remaining terms in the figure relate to the particular direction of flow of the packets. In this figure, packets flow left to right. Therefore, R1 is a DS ingress boundary node, because it is on the boundary between two domains, and packets in the flows shown first enter the DS domain through R1. Similarly, R2 is a DS egress boundary node. R3 is a DS interior node, because it is not on the boundary of the network. Ingress and egress DS boundary nodes typically perform reclassification and re-marking work.

DiffServ formalizes the same general goals and concepts described in the GOCS section earlier in this chapter, but it goes beyond those basic concepts. The next two sections examine two additional important aspects of DiffServ more closely, namely the DSCP field and the different types of PHBs. As described so far, DiffServ operation can be summarized as follows:

1. Good planning must be performed to define the BAs needed for a network.
2. To mark packets to signify what BA they belong to, DiffServ suggests using MF classifiers, which can look at all fields in the packet header.
3. The classifier should be used near the ingress point of the network to assign unique DSCP values to packets inside each BA.
4. After marking has occurred, interior DS nodes use BA classifiers. BA classifiers only look at the DSCP field. When the BA is identified, that node's PHBs can take action on that packet.
5. The ingress DS boundary node in a neighboring downstream DS domain network may not trust the neighboring upstream DS domain at all, requiring an MF classifier and marker at the DS ingress boundary node to reclassify and re-mark all traffic.
6. If the ingress DS boundary node trusts the neighboring DS domain, but the domains use different DSCP values for the same BA, a BA classifier function can be used to reclassify and re-mark the ingress traffic.

DiffServ Per-Hop Behaviors

Other than the general QoS strategies described in this chapter, DiffServ really provides two additional key features: the DSCP field, and some good suggestions on how to use the DSCP field. In fact, two of the DiffServ RFCs, 2597 and 2598, are devoted to describing a set of DSCP values, and some suggested PHBs that should be associated with each DSCP value.

IP defined a type of service (ToS) byte in RFC 791, which came out in September 1981. The IP protocol creators intended the ToS byte to be used as a field to mark a packet for treatment with QoS tools. Inside the ToS byte, the first 3 bits were defined as a field called IP Precedence, which can be marked for the purposes of implying a particular class of service. The Precedence field values imply that the larger the value, the more important the traffic. In fact, names were given to each value 0 from routine (precedence 0) to critical (precedence 5) and network control (precedence 7). The complete list of values from the ToS byte's original IP Precedence 3-bit field, and the corresponding names, are listed in Table 2-13.

Table 2-13 *IP Precedence Values and Names*

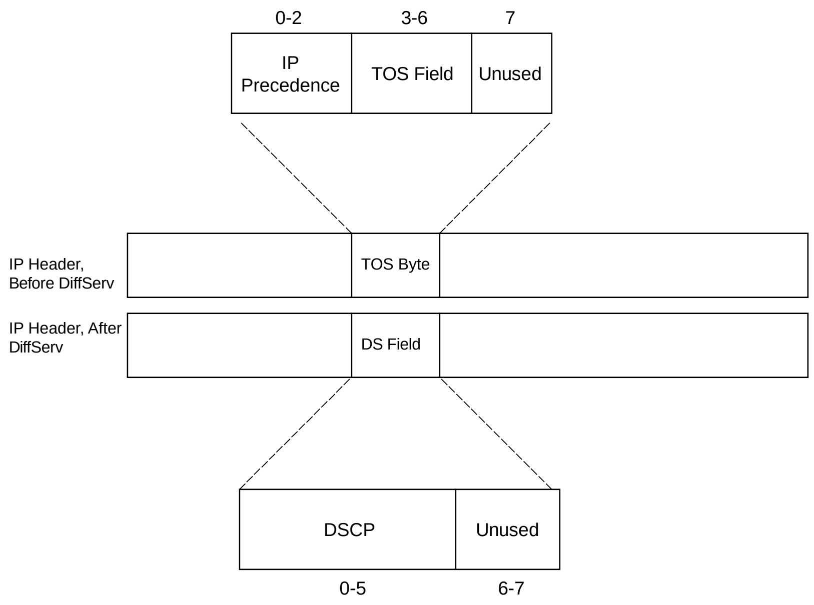
Field and Value (Decimal)	Binary Value	Name
Precedence 0	**000**	Routine
Precedence 1	**001**	Priority
Precedence 2	**010**	Immediate
Precedence 3	**011**	Flash
Precedence 4	**100**	Flash Override
Precedence 5	**101**	Critic/ECP
Precedence 6	**110**	Internetwork Control
Precedence 7	**111**	Network Control

In additon to the Precedence field, the ToS byte included other flag fields that were toggled on or off to imply a particular QoS service—for instance, low or high delay would be signaled by a 1 or a 0 in the delay bit. Bits 4 through 6 (RFC 795) comprised the ToS field inside the ToS byte, with flags for throughput, delay, and reliability. RFC 1349 expanded the ToS field to bits 4 through 7, adding a cost flag. For instance, the original ToS byte creators envisioned the ability to choose a different route, using a more reliable link, for packets with the reliability flag set.

The DS field redefines the ToS byte in the IP header. It removes the definition of the 4 ToS bits (bits 3 through 6). DiffServ creates a replacement for the Precedence field with a new 6-bit field called the Differentiated Services (DS) field. (The last 2 bits of the ToS bytes are used by another specification in RFC 3168.) Figure 2-16 shows the fields inside the ToS byte (per RFC 1349) and the DS field (per RFC 2474).

Changing a protocol that is used in production may result in compatibility issues. If the protocol has available unused fields in the header, and those can be added to the protocol specifications, then all is well. When changing the meaning of an already defined field, however, problems can occur. In this case, DiffServ took advantage of the fact that the ToS field (not ToS byte, but just bits 3 through 6 of the ToS byte) were seldom used. Therefore, DiffServ only had to build compatibility with the Precedence field.

Figure 2-16 *IP ToS Byte and DS Field*

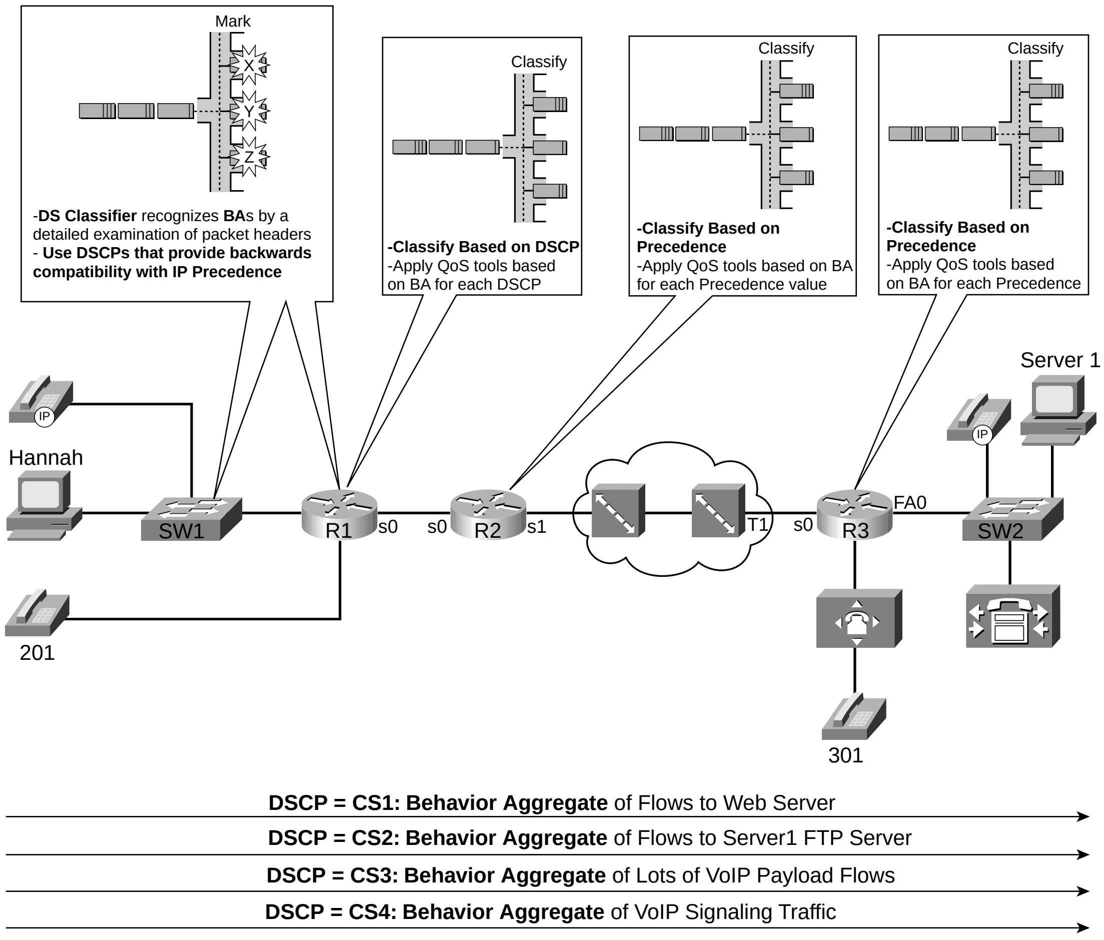


The Class Selector PHB and DSCP Values

RFC 2475, which defines DiffServ, became an RFC in December 1998. Even today, some QoS features in IOS do not support DiffServ! Some QoS features will never support DiffServ, because newer, better tools that can do the same thing may have been introduced. All tools that support Cisco's strategic direction for QoS configuration, using the Modular QoS command-line interface (MQC), support DSCP. However, depending on the tools you need to use, and the IOS revisions you use in your network, you may not be able to use only tools that support DiffServ.

So how does the lack of DiffServ support affect a network based on the DiffServ model? With a well-chosen binary value in the DSCP field, PHBs performed by QoS tools can react to the whole DSCP, or just the first 3 bits, with good effect. Consider Figure 2-17. The DSCP values are marked near the edge. R1 performs PHBs based on the DSCP value, and R2 performs PHBs based on what it thinks is IP precedence, but is really just the first 3 bits of the DSCP.

Figure 2-17 *Supporting IP Precedence in a DiffServ Domain*


Mark
X
Y
Z
-**DS Classifier** recognizes **BA**s by a detailed examination of packet headers
- **Use DSCPs that provide backwards compatibility with IP Precedence**
Classify
-Classify Based on DSCP
-Apply QoS tools based on BA for each DSCP
Classify
-Classify Based on Precedence
-Apply QoS tools based on BA for each Precedence value
Classify
-Classify Based on Precedence
-Apply QoS tools based on BA for each Precedence
Hannah
SW1
R1
s0
s0
R2
s1
T1
s0
R3
FA0
SW2
Server 1
201
301
DSCP = CS1: Behavior Aggregate of Flows to Web Server
DSCP = CS2: Behavior Aggregate of Flows to Server1 FTP Server
DSCP = CS3: Behavior Aggregate of Lots of VoIP Payload Flows
DSCP = CS4: Behavior Aggregate of VoIP Signaling Traffic

The figure lists text telling us that R1 only reacts to DSCP, R2 only reacts to precedence, and R3 has tools that react to both. A QoS tool without DS support may just look at precedence, whereas other QoS tools can look at the DSCP field. The DSCP values marked in this figure were designed to provide backwards-compatability with the IP Precedence field. Table 2-14 lists the DSCP values specifically designed for backwards-compatability. (Note: DiffServ calls DSCP values used for backwards-compatibility with IP Precedence "class selectors.")

Table 2-14 *Default and Class Selector DSCP Values*

Name of DSCP Class Selector Values Used by IOS	Binary Values of DSCP	Equivalent Precedence Value (Decimal)
Default	**000**000	0
CS1	**001**000	1
CS2	**010**000	2
CS3	**011**000	3
CS4	**100**000	4
CS5	**101**000	5
CS6	**110**000	6
CS7	**111**000	7

The names of the code points in Table 2-14 match parameters found on IOS DiffServ-compliant classification commands. Because an “all-zeros” DSCP called “default” was already defined, there was no need to create a CS0 DSCP name.

The class selector PHB and DSCP values defined by DiffServ are listed in Table 2-14. These DSCP values provide backward compatibility with precedence. By examining the first 3 bits in each binary DSCP value in the table, you can see that these 8 DSCP values match the 8 different values that can be encoded in the 3-bit Precedence field. Any router looking instead for the Precedence field will just find the first 3 bits of the DSCP field. And just like with IP precedence, the CS DSCP values all imply that the bigger the binary number, the better the PHB.

Although DiffServ supplies the eight CS DSCP values for backward compatibility with IP precedence, many DSCP values actually provide backward compatibility. For instance, DSCP values decimal 8 through 15 all begin with the binary string 001 in the 6-bit DSCP field, making each of these 8 DSCP values compatible with IP precedence 1 (binary 001). In fact, there are 8 DSCP values that provide backward compatibility with every IP precedence value. Table 2-15 lists the values.

Table 2-15 *Range of DSCP Values Compatible with IP Precedence*

Range of DSCP Values, in Decimal	Binary Value	Compatible with These IP Precedence Values
0–7	**000**xxx	0
8–15	**001**xxx	1
16–23	**010**xxx	2

continues

Table 2-15 *Range of DSCP Values Compatible with IP Precedence (Continued)*

Range of DSCP Values, in Decimal	Binary Value	Compatible with These IP Precedence Values
24–31	**011**xxx	3
32–39	**100**xxx	4
40–47	**101**xxx	5
48–55	**110**xxx	6
56–63	**111**xxx	7

As you will read in the upcoming sections, the DSCP values suggested for use by DiffServ include the consideration of making the values meaningful to devices that do not understand DSCP, but only understand IP precedence.

NOTE It is important to distinguish between what the values of the precedence and DSCP fields *can* mean and what they *should* mean if following suggested QoS design practices. IP precedence value 0 should imply the lowest QoS service possible, with precedence 7 implying the best QoS service. The class selector PHB values follow that same logic. However, most QoS tools can be configured to do just the opposite—for instance, giving precedence 0 traffic the best service, and precedence 7 the worst. Conversely, some other QoS tools are not as flexible and assume a bigger precedence is better. For instance, Weighted Fair Queuing (WFQ) always gives more queuing preference to higher-precedence value flows, all other facts being equal.

NOTE As seen later with the assured forwarding (AF) PHB and DSCP values, the actual binary values for DSCP do not conform to the "bigger-is-better" logic for the actual values.

DiffServ suggests two other sets of PHBs and DSCP values besides the class selector values, namely assured forwarding (AF) and expedited forwarding (EF). Can you just decide to make up random 6-bit values to associate with each BA? Yes. Can you configure most QoS tools to give each BA the PHB that you desire? Sure. If you take the time to learn and follow DiffServ's suggestions, such as CS, AF, and EF, however, then you can take advantage of some good defaults in IOS, increase the odds of compatibility between your DS domain and others, and avoid a lot of extra configuration.

Table 2-16 summarizes some of the key points about choosing to follow DiffServ's suggestions.

Table 2-16 *Comparing Choices: Making Up DSCP Values, or Using Suggested Values from the RFCs*

Using Suggested DSCP Values	Making Up DSCP Values
More likely to be using the same values as neighboring DS domains; still dependent on whether BAs match	Unlikely to be using the same values as neighboring DS domains
Can configure all QoS tools to create the needed PHBs	Can configure most, but not all, QoS tools to create the needed PHBs
Defaults for some QoS tools already set to good values	Must create more configuration to override defaults that Cisco chose based on DSCP suggestions
Can use well-known names for DSCP values, ignoring actual value; IOS stores values as names in configuration file	Must configure DSCP values as decimal numbers

The next two sections cover the DiffServ RFCs' main suggestions for DSCP values to be assigned.

The Assured Forwarding PHB and DSCP Values

RFC 2597 defines something called "the assured forwarding per-hop behaviors." This RFC suggests that one good DiffServ design choice would be to allow for four different classes for queuing purposes. Within each queue, three levels of drop probability could be implied. The RFC title rightfully suggests that the focus is on the QoS behavior—the PHB—at each node. Most engineers also think of the RFC as defining the 12 DSCPs that are used in conjunction with the AF PHBs.

An individual PHB describes what happen in a single hop, most typically a router. In the case of AF, each PHB contains two separate QoS function, typically performed by two different QoS tools. The first function is queuing. Each router classifies the packets into four different classes, and packets from each class are placed in a separate queue. AF does also specify that the queuing method support the ability to reserve a minimum configured bandwidth for each class.

The AF PHB defines congestion avoidance as the second behavior that comprises the AF PHB. Routers drop packets when a queue is full and the router needs to place the packet in the queue; this action is called tail drop. Congestion-avoidance tools discard packets before tail drop is required, hoping that fewer packets are dropped, as described earlier in this chapter. In Cisco routers, this part of the AF PHB is implemented using some form of RED.

The AF PHB does not define that it wants a guarantee that each packet will be delivered, nor does it imply any form of error recovery. The name *assured forwarding* sometimes evokes visions that the bandwidth throughout the network is guaranteed, but it is not. The real objective can be seen with a short read of RFC 2597, which reveals a view of an enterprise and an ISP. The ISP wants to assure the customer that his traffic will get through the network, so long as the customer does not send more data than is contracted. For instance, maybe the enterprise has three classes (BAs) of traffic, for which they contract 300K for the first, 200K for the second, and 100K for the third. The ISP would like to assure that these levels are met. Because many queuing tools effectively guarantee bandwidth over time, the ISP can give these BAs the appropriate amount of bandwidth. Because packet arrival times and rates vary, however, the queues inside the ISP's network will grow and shrink. Congestion will occur; the ISP knows it, and the enterprise customer knows it. So, when temporary congestion occurs in the ISP network, it would be nice to know which type of packets to throw away under limited congestion, and which type to throw away under moderate congestion, and which ones to throw away under heavy congestion. Letting the customer define which traffic is discarded most aggressively also lets the customer achieve some control of how its traffic is treated by the ISP. Therefore, assured forwarding does not mean that an individual packet is assured of making it across the network; it does mean that attempts will be made to assure that queuing tools provide enough bandwidth, and when congestion does occur, less important traffic will be discarded first.

RFC 2597, and the AF PHB concepts, can be summarized as follows:

- Use up to four different queues, one for each BA.
- Use three different congestion thresholds inside each queue to determine when to begin discarding different types of packets.
- To mark these packets, 12 DSCP values are needed; the names of these values as start with "AF" (assured forwarding).

NOTE RFC 2360 clarifies some of the meaning and terminology used with DiffServ. Technically, the AF PHB, according to RFC 3260, is actually four different PHBs—one for each class. Frankly, for purposes of passing the Cisco QoS exams, I do not think that will ever matter; I only mention it here to be fully correct.

Table 2-17 lists the names of the DSCP values, the queuing classes, and the implied drop likelihood.

Table 2-17 *Assured Forwarding DSCP Values and Meaning*

	Low Drop Probability Within Class	Medium Drop Probability Within Class	High Drop Probability Within Class
Class 1	AF11	AF12	AF13
Class 2	AF21	AF22	AF23
Class 3	AF31	AF32	AF33
Class 4	AF41	AF42	AF43

Pay particular attention to the explanation of Table 2-17 inside this paragraph, because the values in the chart can be counterintuitive. Unlike the CS PHB, AF does not follow the "bigger-is-better" logic for the AF DSCPs. First, AF11, AF12, and so on are names for DSCP values, not the binary of decimal equivalent. (Those values are listed momentarily.) Given the names, at least you can think of the first "digit" after the AF to be the queuing classification—for example, all AF4x code points are in the same class for queuing. No specific queuing parameters are implied for any of these classes, so there is no inherent advantage to being in class 4 versus class 1.

Similarly, the second numeric digit in the AF DSCP names imply the drop preference—with 3 meaning highest likelihood of being dropped, and 1 meaning the least likelihood. In other words, inside a single class, an AFx3 DSCP would mean that these packets would be dropped more quickly (more aggressively) than AFx2, which would be dropped more aggressively than AFx1 packets. In the actual DSCP names, a bigger number for the second numeric digit actually implies a less-desirable QoS behavior. (This convention is also true of the actual binary values.)

You can read about DiffServ AF PHBs, configure DiffServ-compliant IOS tools, and never really have to know the underlying binary values and their decimal equivalents. For reference, however, Table 2-18 includes them. As noted earlier, the numeric parts of the AF code points did not follow the same bigger-is-better scheme that IP precedence did in the past. Likewise, the actual underlying binary values do *not* follow a bigger-is-better scheme, either.

Table 2-18 *Assured Forwarding DSCP Values—Names, Binary, and Decimal*

	Low Drop Probability Within Class	**Medium Drop Probability Within Class**	**High Drop Probability Within Class**
	Name/Decimal/Binary	**Name/Decimal/Binary**	**Name/Decimal/Binary**
Class 1	AF11 / 10 / 001010	AF12 / 12 / 001100	AF13 / 14 / 001110
Class 2	AF21 / 18 / 010010	AF22 / 20 / 010100	AF23 / 22 / 010110
Class 3	AF31 / 26 / 011010	AF32 / 28 / 011100	AF33 / 30 / 011110
Class 4	AF41 / 34 / 100010	AF42 / 36 / 100100	AF43 / 38 / 100110

The binary DSCP values imply the queuing class with the first 3 bits (bits 0 through 2), and the drop preference in the next two bits (bits 3 and 4). Queuing tools that operate only on IP precedence can still react to the AF DSCP values, essentially making the AF DSCPs backward compatible with non-DiffServ nodes for queuing, at least. Note that the "bigger-is-not-always-better" attitude continues with the actual binary values—the smaller the value of bits 3 and 4, the lower the probability of being discarded.

NOTE To convert from the AF name to the decimal equivalent, you can use a simple formula. If you think of the AF values as AFxy, the formula is

$8x + 2y$ = decimal value. For example, AF41 gives you a formula of $(8 * 4) + (2 * 1) = 34$.

In summary, DiffServ AF PHB provides the following:

- An overriding goal to provide PHBs that provide enough bandwidth for each class, with the ability to drop less important traffic, hoping to avoid dropping more important traffic, if congestion does occur.
- A convention that provides 4 classes for queuing.
- The convention includes three drop preferences inside each class.
- Twelve code point names and values to use to create the four classes with three drop levels each.

The Expedited Forwarding PHB and DSCP Values

RFC 2598 defines the expedited forwarding per-hop behaviors. This RFC defines a very simple PHB (low latency, with a cap on bandwidth), and a single DSCP (EF) to represent it. Expedited forwarding simply states that a packet with the EF DSCP should minimize delay, jitter, and loss, up to a guaranteed bandwidth level for the class.

Like AF, the EF RFC suggests two QoS actions be performed to achieve the PHB. First, queuing must be used to minimize the time that EF packets spend in a queue. A queuing scheme that sends packets from this queue whenever a packet is waiting reduces delay. Anything that reduces delay reduces jitter. Always servicing the EF queue first greatly reduces the queue length, which in turn greatly reduces the chance of tail drop due to the queue being full. Therefore, EF's goal of reducing delay, jitter, and loss can be achieved with a queuing method such as Priority Queuing, with EF traffic in the most important queue.

The second component of the EF PHB is policing. If the input load of EF packets exceeds a configured rate, the excess packets are discarded. If 100 kbps is reserved for the EF BA, and 200 kbps enters the network, for example, supporting the extra traffic may be unreasonable. Why not just accept the extra traffic? The queuing method used to achieve low delay, low jitter, and low loss, would prevent other types of traffic from getting any bandwidth, because the queuing method always gives preference to EF traffic. Thus, EF protects the other traffic by capping the amount of bandwidth for the class. In other words, EF suggests a great level of service, but just up to the contracted amount of bandwidth.

The expedited forwarding PHB uses a DSCP name of EF, whose binary value is 101110, with a decimal value of 46.

Table 2-19 summarizes many of the key points about the various DiffServ PHBs.

Table 2-19 *Comparison of DiffServ PHBs*

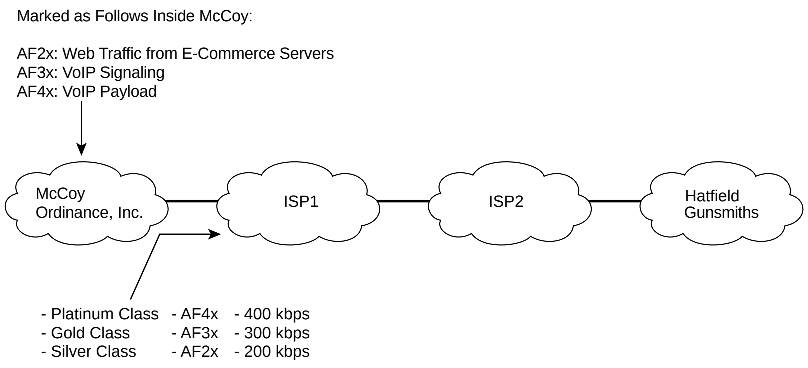
PHB	Key Components	Names of DSCPs
Best effort (BE)	PHB for getting no specific QoS treatment	DSCP BE (default)
Class selector (CS)	Uses 8 DSCPs, all with binary 0s for the last 3 bits. Used for backward compatibility with IP precedence. Uses “bigger-is-better” logic—the bigger the DSCP, the better the QoS treatment.	CS1, CS2, CS3, CS4, CS5, CS6, CS7
Assured forwarding (AF)	PHB consists of 2 components: queuing to provide a minimum bandwidth to each for 4 different queues, and 3 drop thresholds inside each queue. DSCPs do not always follow the “bigger-is-better” logic.	AF11, AF12, AF13, AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43
Expedited forwarding (EF)	PHB also has 2 components: queuing to provide low delay/jitter/loss plus a guaranteed amount of bandwidth, and policing to prevent EF from preventing other types of traffic from getting enough bandwidth.	EF

DiffServ Classifiers and Traffic Conditioners

DiffServ focuses on “scalable service differentiation in the Internet” according to RFC 2475. In times past, the Internet provided a “best-effort” service—networks generated and sent traffic, and the Internet forwarded it—and everyone hoped that market forces helped the Internet grow to accommodate the loads. With the advent of premium offerings from ISPs, new QoS tools and models were needed, with DiffServ being just one component.

Because DiffServ focuses on Internet services, the RFCs focus on QoS tools that are most useful when connecting multiple networks. Consider Figure 2-18, for example, which shows a network with three classes of service defined for the McCoy Ordinance Co. by ISP1.

Figure 2-18 *Three Premium Classes for McCoy Ordinance by ISP1*


Marked as Follows Inside McCoy:
AF2x: Web Traffic from E-Commerce Servers
AF3x: VoIP Signaling
AF4x: VoIP Payload
McCoy Ordinance, Inc.
ISP1
ISP2
Hatfield Gunsmiths
- Platinum Class - AF4x - 400 kbps
- Gold Class - AF3x - 300 kbps
- Silver Class - AF2x - 200 kbps

ISP1 contracted with McCoy to provide three premium service levels. First assume that McCoy conforms to the defined bandwidth levels, and marks the DSCP fields appropriately with AF DSCP values as shown in the figure. Because the QoS issues involve two different networks, however, additional care must be taken with QoS functions at the edge of these networks. The rest of this brief section focuses on these DiffServ boundary node functions.

DiffServ defines two particularly important features that should be implemented on the boundary DS node: classification and conditioning. Classification has been covered in some depth already. Two general categories of classifiers exist—a BA classifier and an MF classifier (multifield classifier). The BA classifier only looks at the DSCP field—in other words, it assumes a DiffServ BA has already been identified using classification and marking. The MF classifier is called "multifield" because it can look at lots of fields in the packet header to classify the traffic. In the example of Figure 2-18, because McCoy already correctly marked the DSCP field, ISP1's DS ingress boundary node just needed to use a BA classifier.

DiffServ defines traffic conditioning as the second important function at the boundary node. Traffic conditioning defines what to do to prevent traffic from exceeding contracts, but if it does, what to do with traffic that exceeds the contract. If McCoy always only sends what the contract defines, great! If McCoy breaks that trust and sends more traffic, and ISP1 does not monitor the traffic and possibly deletes the extra traffic, however, bad things can happen to all of ISP1's customers. It's like making a reservation on a plane. You know some people will change their flights, and some will miss their flights, but the airline will generally not give you a reservation unless there is a seat on the plane. What if someone were to show up at the airport with 100 of their closest friends and tell the gate agent, "Oh, I want my 100 friends to have a seat on the plane as well!"—and they let all 100 people on the plane? Well, the airline would have a lot of other unhappy customers, to be sure. Likewise, although ISP1 may want to accommodate the extra traffic, the ISP might need to protect their network from the congestion created by the extra traffic.

The terms relating to the traffic-conditioning functions of DiffServ, summarized in Table 2-20, should be mostly familiar by now.

Table 2-20 *Traffic Conditioners*

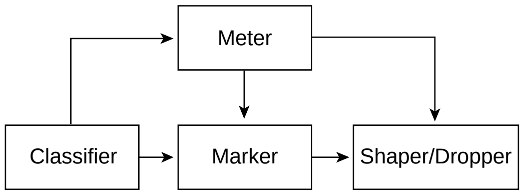
Traffic Conditioner	**Explanation**
Metering	The metering function measures the traffic rate to determine whether the traffic conforms to the stated contract, or exceeds the traffic contract. Metering typically occurs per class.
Dropping (policing)	If the traffic exceeds the contract, one option is to drop some packets, so that the packets that are allowed through meet the contract. Most implementations for this feature are called policing.
Shaping	If the traffic exceeds the contract, one option is to shape the traffic. Shaping just means to buffer or queue the traffic, slowing it down, so that the resulting sending rate is within the contract.

Table 2-20 *Traffic Conditioners (Continued)*

Traffic Conditioner	Explanation
Marking	A third option for traffic that exceeds the contract is to re-mark the DSCP with a different value. For instance, Platinum AF41 (low drop probability) traffic that exceeds the contract might get re-marked to AF43. In such a case, if congestion were to occur, this packet would more likely be dropped than if not re-marked.
Do nothing	Not actually written down in the RFCs; one option is to allow the traffic to keep moving right along.

RFC 2475 contains a block diagram listing the classifier and traffic conditioners. This same figure, or a similar version, is interspersed through many QoS documents and QOS classes. Figure 2-19 does not intend to show a flowchart of what happens to every packet as it passes through a DS boundary node, but rather to show some of the possible paths a packet can take.

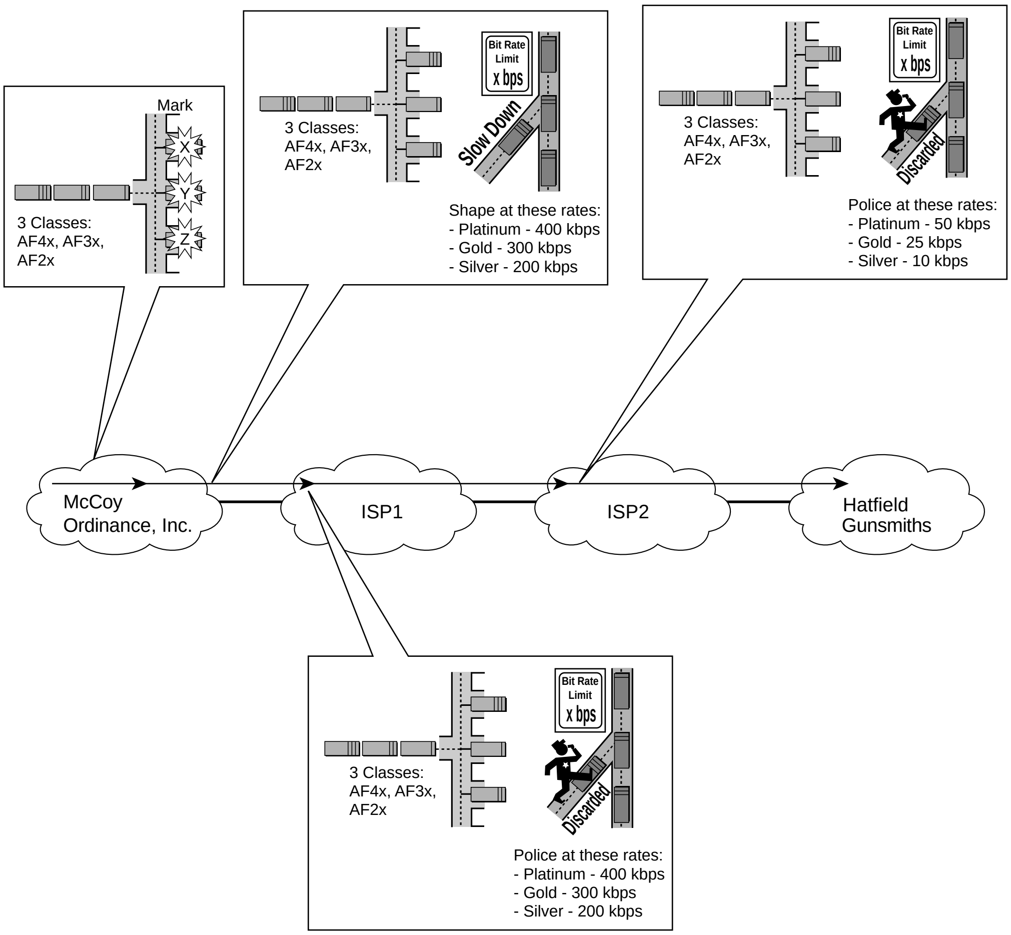
Figure 2-19 *RFC2475—Equivalent Diagram of Classifier and Traffic-Conditioner Functions*



Remember, the figure does not necessarily show a flowchart of how a packet is processed on a boundary node. For instance, the packet can go through a classifier and marker function only, or it can go through the meter and dropper, but not the marker or shaper. So, what is normal? Figure 2-20 shows an example with the more typical usage of the tools shown.

The various classification and traffic-conditioning tools can be used at various points in the network as shown. Egress boundary nodes tend to shape traffic, to prevent sending more traffic than the traffic contract allows; in fact, all Cisco shaping tools shape on egress only. Ingress boundary nodes tend to use the dropper or policer conditioning functions, metering (measuring) ingress traffic, with either a drop or re-mark action as a result of excess traffic. The reason why engineers implement shaping and dropping/policing on egress and ingress, respectively, has to do with who is providing service, and who is receiving a service. For instance, McCoy is sending the traffic, with an expectation that ISP1 will forward the traffic, as long as McCoy conforms to the service contract. By shaping, McCoy makes sure to conform. By policing, ISP1 protects its network and all its customers from McCoy sending too much data into the network.

Figure 2-20 *Examples of When to Use Each Traffic-Conditioning Tool*



DiffServ provides a formalized but sensible approach to QoS over the Internet. By using classes that aggregate the packets of many flows, DiffServ can scale well in the Internet. The next section covers the other specification for an Internet QoS model: IntServ.

The Integrated Services QoS Model

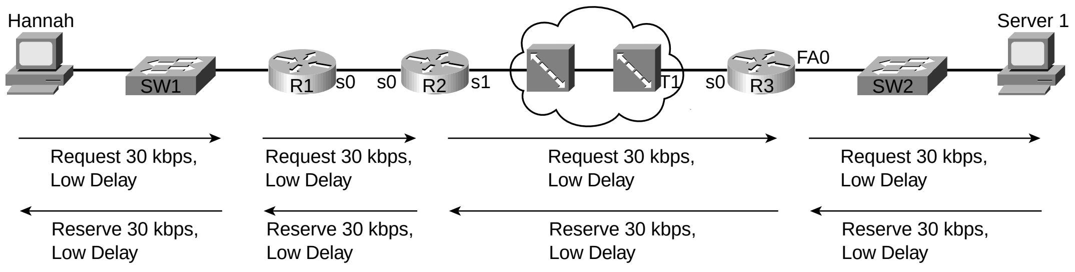
Integrated services (IntServ) defines a different model for QoS than does DiffServ. IntServ defines a signaling process by which an individual flow can request that the network reserve the bandwidth and delay needed for the flow. The original work grew out of the experiences of the IETF in multicasting the audio and video for IETF meetings in the early to mid-1990s.

To provide guarantees per flow, IntServ RFC 1633 describes two components: resource reservation and admission control. Resource reservation signals the network elements about how much bandwidth and delay a particular flow needs. If the signaling completes successfully, the various network components have reserved the needed bandwidth. The collective IntServ nodes (typically routers) reserve the appropriate amount of bandwidth and delay in response to the signaling messages.

IntServ admission control decides when a reservation request should be rejected. If all requests were accepted, eventually too much traffic would perhaps be introduced into the network, and none of the flows would get the requested service.

Figure 2-21 shows the general idea behind the IntServ reservation requests.

Figure 2-21 *Integrated Services Reservation Requests*


Hannah
Server 1
SW1
R1
s0
s0
R2
s1
T1
s0
R3
FA0
SW2
Request 30 kbps, Low Delay
Request 30 kbps, Low Delay
Request 30 kbps, Low Delay
Request 30 kbps, Low Delay
Reserve 30 kbps, Low Delay
Reserve 30 kbps, Low Delay
Reserve 30 kbps, Low Delay
Reserve 30 kbps, Low Delay

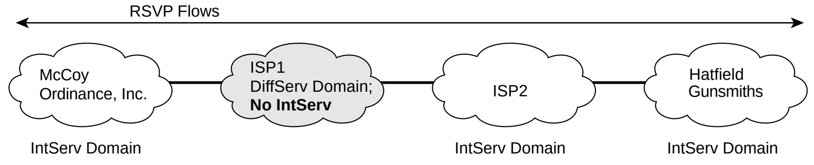
IntServ uses Resource Reservation Protocol (RSVP, RFCs 2205 through 2215) for signaling to reserve the bandwidth. With a full IntServ implementation (more on that later), the originator of the flow (Hannah) begins signaling. At each router along the route, the router asks itself, “Can I support this request?” If the answer is yes, it forwards the request to the next router. Each router holds the bandwidth temporarily, waiting on the confirmation to flow back to the originator (Hannah). When each router sees the reserve RSVP command flow back to the originator, each router completes the reservation.

What does it mean for the router to “reserve” something? In effect, the router reserves the correct queuing preferences for the flow, such that the appropriate amount of bandwidth is allocated to the flow by the queuing tool. RSVP can also request a certain (low) amount of delay, but implementing a guarantee for delay is a little more difficult; IOS, for instance, just reserves the queuing preference. In fact, IntServ RFCs actually define the term “guarantee” as a relatively loose goal, and it is up to the actual implementation to decide how rigorous or general to make the guarantees.

RSVP continues signaling for the entire duration of the flow. If the network changes, or links fail and routing convergence occurs, the network may no longer be able to support the reservation. Therefore, RSVP reserves the bandwidth when the flow initializes and continues to ensure that the flow can receive the necessary amount of bandwidth.

IntServ has some obvious disadvantages, and it has several advantages. IntServ actually predates DiffServ; DiffServ, to some degree, was developed to provide an Internet-scale QoS model, because IntServ scales poorly. IntServ expects the hosts to signal for service guarantees, which brings up two issues—whether the hosts can be trusted by the network and whether the hosts actually support RSVP. Alternatively, routers can be configured to reserve bandwidth on behalf of hosts, but the configuration can quickly become an administrative problem because additional configuration would need to be added for each reserved flow. Also IntServ works best when all intermediate networks support IntServ. Take a look at Figure 2-22, for example. Whereas McCoy, Hatfield, and ISP2 support IntServ, ISP1 does not.

Figure 2-22 *IntServ Through the Internet, with Partial Support*



IntServ, with RSVP signaling, can work in this network. However, for ISP1, one of two things must be done to support IntServ:

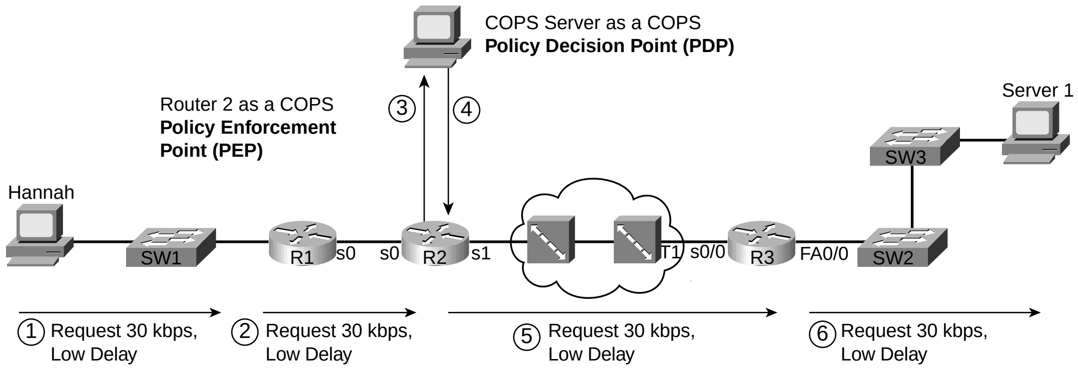
- ISP1 must pass the RSVP messages through; the ISP1 router just treats the traffic as best effort.
- ISP1 passes the RSVP messages; RSVP flows are mapped to DiffServ classes inside the ISP1 DiffServ domain.

Another problem with IntServ in large networks and in the Internet relates to the fact the RSVP reserves bandwidth per flow. With DiffServ, for instance, all web traffic might be marked with DSCP AF21 and placed into a single class—even if there are hundreds, thousands, or tens of thousands of flows. With IntServ, each flow causes a separate reservation. In fact, DiffServ created an “Internet-scale” QoS model in part because the earlier IntServ specification did not scale well for the Internet, even if you could get most or all ISPs to implement IntServ and RSVP.

Are there advantages to IntServ? Of course. It can reserve specific amounts of bandwidth for a particular flow, which can be very useful. However, IntServ may never be pervasively deployed in an enterprise, or throughout the Internet. For instance, reserving bandwidth for flows between key video-conferencing stations may be useful. Allowing voice gateways to request reservations for VoIP calls can also help. The DiffServ model can be used to place all VoIP into a class, and give the class better treatment, but IntServ can guarantee the QoS behavior and reject new calls if the network is currently not ready to accept more traffic.

One final function of IntServ that may be on the QoS exams concerns admission control. One effort to help IntServ scale involves offloading the processing work required for the admission control decision. RSVP allows each router to handle admission control locally, as shown in Figure 2-21. Alternatively, a router can ask a server that supports the Common Open Policy Service (COPS) protocols, and let the server make the choice. Figure 2-23 shows an example.

Figure 2-23 *IntServ and the Common Open Policy Service (COPS)*


COPS Server as a COPS
Policy Decision Point (PDP)
Router 2 as a COPS
Policy Enforcement Point (PEP)
3
4
Server 1
SW3
Hannah
SW1
R1
s0
s0
R2
s1
T1
s0/0
R3
FA0/0
SW2
1 Request 30 kbps, Low Delay
2 Request 30 kbps, Low Delay
5 Request 30 kbps, Low Delay
6 Request 30 kbps, Low Delay

As seen in the figure, R2 requests that the COPS policy decision point (PDP) approve or disapprove of the new reservation. R2 is considered to be a policy enforcement point (PEP), because it enforces the QoS reservation.

The QoS exams cover IntServ-related concepts and features, although DiffServ-related features certainly get much more coverage. The following list summarizes some of the key points about IntServ that you will want to remember for the QoS exams:

- Integrated services defines the need, and suggests mechanisms, to provide bandwidth and delay guarantees to flows that request it. RFC 1633 defines it.
- IntServ contains two components: resource reservation and admission control.
- RSVP, as defined in RFCs 2205 through 2215, provides the IntServ resource reservation function. RFC 2210 specifically discusses RSVP's usage for IntServ.
- With end-to-end RSVP, each intermediate router reserves the needed bandwidth when receiving a reservation request and confirms the request with RSVP reserve messages. If a router in the path does not speak RSVP, it just transparently passes the flow.

- When IntServ has not been implemented end to end, the RSVP messages can be forwarded in the non-IntServ part of the networks. In that case, the non-IntServ networks can either provide best-effort (BE) service, or provide IntServ-DSCP mapping if the intermediate network is a DiffServ domain.
- A router can offload the admission control function to a COPS server.

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures are a convenient way to review the day before the exam.

Table 2-21 lists the IOS classification and marking tools, along with a few key features that differentiate the tools.

Table 2-21 *Comparison of Classification and Marking Tools*

Tool	Other Functions Besides Class and Mark	Fields That Can Be Examined for Classification	Fields That Can Be Marked*
Policy-based routing (PBR)	Routing packets based on something besides destination address	ACLs indirectly through route maps	IP ToS field IP Precedence field QoS Group
Committed access rate (CAR)	Policing	IP ACLs QoS Group IP DSCP	IP Precedence IP DSCP QoS Group MPLS Experimental
Class-based marking (CB marking)	None	IP ACLs Any markable fields Input interface MAC addresses All NBAR-enabled fields	IP precedence DSCP 802.1P CoS ISL Priority ATM CLP Frame Relay DE MPLS Experimental QoS Group
Network based application recognition (NBAR)	Statistical information about traffic mix; recognition of applications that use the dynamic port	Extensive list (see Chapter 3, "Classification and Marking")	None; used in conjunction with CB marking
VoIP dial peers	Call routing for VoIP	None	IP Precedence

* All claims about features/functions that may be affected by IOS versions assume version 12.2, unless otherwise stated.

Table 2-22 outlines the key features of IOS queuing methods.

Table 2-22 *Comparison of Queuing Tools*

Tool	Maximum Number of Queues	Classification Capabilities	Queue Service Algorithm/End Result of Algorithm
Priority Queuing (PQ)	4	IP ACL* Input interface Fragments	Strict service; always serves higher-priority queue over lower queue.
Custom Queuing (CQ)	16	IP ACL* Input interface Fragments	Serves a configured number of bytes per queue, per round-robin pass through the queues. Result: Rough percentage of the bandwidth given to each queue under load.
Weighted Fair Queuing (WFQ)	4096	Automatic, based on flows. (Flow identified by source/destination address and port numbers, plus protocol type.)	Each flow uses a different queue. Queues with lower volume and higher IP precedence get more service; high volume, low precedence flows get less service.
Class-Based Weighted Fair Queuing (CBWFQ)	64	IP ACL* NBAR Same as CB marking	Service algorithm not published; results in set percentage bandwidth for each queue under load.
Low Latency Queuing	N/A	Same as CBWFQ	LLQ is a variant of CBWFQ, which makes some queues “priority” queues, always getting served next if a packet is waiting in that queue. It also polices traffic.
IP RTP Priority	N/A	Even UDP ports between 16384 and 32767 (all VoIP payload ports)	An added feature with WFQ or CBWFQ, all VoIP payload is placed in a special “priority” queue, always getting served next if a packet is waiting in that queue.
Modified Deficit Round-Robin (MDRR)	8	IP precedence	Similar to CQ, but each queue gets an exact percentage of bandwidth. Supports LLQ mechanism as well.

* Some queuing tools support different configuration tools that allow matching the same fields that an ACL can match. In these cases, only the IP ACL method of matching is listed in this summary table.

Figure 2-24 depicts the typical points in a network where policing and shaping are typically deployed.

Figure 2-24 *Traffic Policing and Shaping*

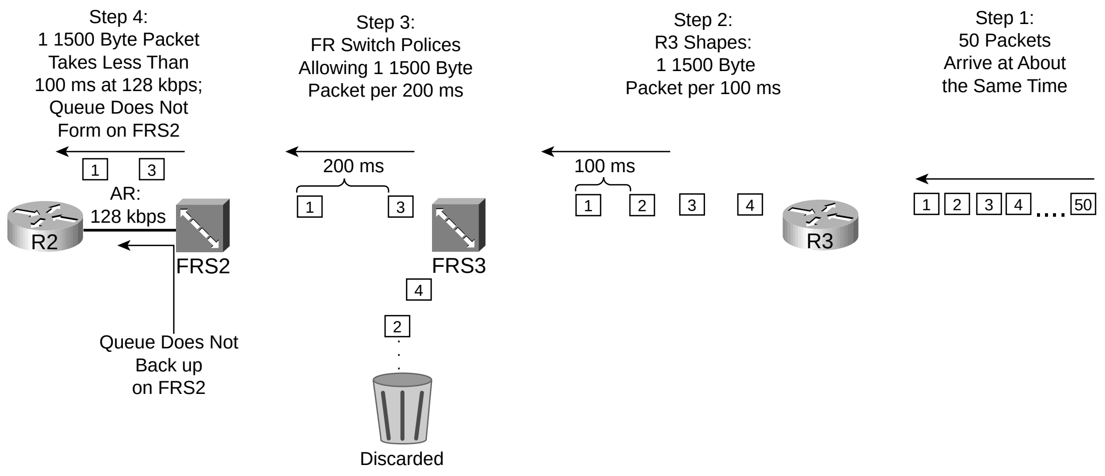

Step 4:
1 1500 Byte Packet Takes Less Than 100 ms at 128 kbps; Queue Does Not Form on FRS2
Step 3:
FR Switch Polices Allowing 1 1500 Byte Packet per 200 ms
Step 2:
R3 Shapes: 1 1500 Byte Packet per 100 ms
Step 1:
50 Packets Arrive at About the Same Time
AR: 128 kbps
R2
FRS2
FRS3
R3
200 ms
100 ms
Queue Does Not Back up on FRS2
Discarded
1
2
3
4
50

Table 2-23 outlines the key features of IOS policing and shaping tools.

Table 2-23 *Comparison of Shaping and Policing Tools*

Tool	Policer or Shaper	Interfaces Supported	Per Subinterface, and Per VC, Support
Class-based policing (CB policing; sometimes just called policer)	Policer	All that are supported by Cisco Express Forwarding (CEF)	Per subinterface
Committed access rate (CAR)	Policer	All that are supported by CEF	Per subinterface
Class-based shaping	Shaper	All that are supported by CEF	Per subinterface
Generic traffic shaping/ distributed traffic shaping (GTS/DTS)	Shaper	Frame, ATM, SMDS, Ethernet	Per subinterface
Frame Relay traffic shaping (FRTS)	Shaper	Frame	Per DLCI

All functions listed are based on 12.2 mainline IOS code levels.

Table 2-24 lists the tools, and the various points for comparison, for congestion-avoidance tools.

Table 2-24 *Comparison of Congestion-Avoidance Tools*

Tool	Can Be Enabled in IOS?	Weights Based on IP Precedence or DSCP?	Considers Flow Information When Deciding to Drop Packets?
Random Early Detection (RED)	No	No	No
Weighted Random Early Detection (WRED)	Yes	Yes	No
Flow-Based Random Early Detection (FRED)	Yes	Yes	Yes

All functions listed are based on 12.2 mainline IOS code levels.

Table 2-25 lists the link-efficiency tools and some of the pertinent comparison points.

Table 2-25 *Comparison of Link-Efficiency Tools*

Tool	Data Links Supported	Types of Packets to Which Tool Can Be Applied
Payload compression	All; recommended on serial links (T/1, E/1, and slower)	All IP packets
RTP header compression (cRTP)	All; recommended on serial links (T/1, E/1, and slower)	All packets with IP/UDP/RTP headers
TCP header compression	All; recommended on serial links (T/1, E/1, and slower)	All IP packets with TCP headers
Multilink PPP fragmentation and interleaving (MLPPP LFI)	Multilink PPP	All packets larger than a configured length
Frame Relay fragmentation (FRF*)	Frame Relay	All packets larger than a configured length (FRF.12) or all non-VoFR frames (FRF.11c)
Link fragmentation and interleaving for Frame Relay and ATM VCs	Frame Relay and ATM	All IP packets

All functions listed are based on 12.2 mainline IOS code levels.

Table 2-26 lists each of the specific CAC tools.

Table 2-26 *Comparison of CAC Tools*

Tool	CAC Type	CAC Decision Is Based on Whether . . .
Physical DS0 limitation	Local	A DS0 channel is available on a trunk
Max-connections	Local	A configured number of maximum connections on the dial peer used for the call has been exceeded
Voice-bandwidth for Frame Relay	Local	VoFR PVC CIR has been exceeded; VoFR only (not VoIP)
Trunk conditioning	Local	Keepalives sent to other end of network keep working or not; used for "connection trunk" calls only*
Local Voice Busy-Out (LVBO)	Local	One or more local interfaces fail; if they all fail, no IP connectivity would exist, so the trunk is placed in busy-out state
Advanced Busy-Out Monitor (AVBO)	Measurement based	Probe measurements are better than a configured "impairment factor"; if value is higher, the entire trunk is placed in busy-out
PSTN Fallback	Measurement based	Probe measurements are better than a configured "impairment factor"; instead of busying-out the trunk, calls are allowed or rejected on a call-by-call basis
Resource Availability Indicator (RAI)	Resource based	Terminating gateway's calculation of available DSPs and DS0s implies it has adequate number of resources or not
Gatekeeper Zone Band-width (GK Zone Bandwidth)	Resource based	Gatekeeper believes that the bandwidth into the zones in question has been oversubscribed or not
Resource Reservation Protocol (RSVP)	Resource based	The RSVP reservation request flow can, both at call setup and ongoing throughout the call, reserve the needed bandwidth on all RSVP-supporting links in the IP network

* Trunk conditioning acts like a measurement-based CAC tool in my opinion; the IOS documentation, and the DQOS course, list this CAC tool as a local CAC tool. Cisco QoS exam questions are not based on my opinion, so it is listed as a local CAC tool in this book!

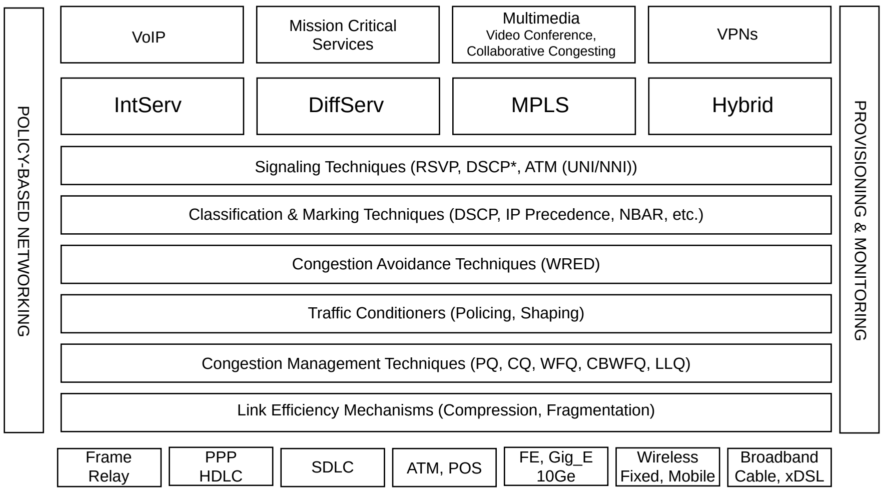
Table 2-27 lists Cisco QoS management tools along with a short description of the functions of each tool.

Table 2-27 *QoS Management Tools*

Tool	Features
QoS Device Manager (QDM)	Uses code that is stored in Flash memory, running inside each router; user can use a web browser to manage QoS configuration and view statistics for an individual router. The product is free.
QoS Policy Manager (QPM)	Application runs on Windows NT or Windows 2000, accessible by a browser. Enables the engineer to manager QoS policies network wide; QPM takes policies and creates QoS configurations, stages, implements, and allows back-out of QoS configurations throughout the network.
Service Assurance Agent (SAA)	The feature formerly known as Response Time Reporter (RTR), SAA is a feature of IOS. This feature can be configured to create, send, respond to, and measure the performance of probe packets. Measurement-based CAC mechanisms create SAA probes on routers; the routers send and receive the probes; and then the routers tell SAA the results.
Internetwork Performance Monitor (IPM)	Formerly a separate product, this Cisco Works feature monitors network performance in real time. It also provides an easy GUI interface to configure SAA probes.
Service Management Solution (SMS)	SMS is a feature of Cisco Works that provides performance statistics similar to IPM, but with the intent to save historical data and to provide reporting about whether configured service level agreements (SLAs) are being met. IPM is used for operating the network and looking at current network statistics; SMS is used for historical reporting, trending, and SLAs.

Cisco has created a QoS framework that shows the various components of a network relating to QoS. Figure 2-25 shows the Cisco QoS framework.

Figure 2-25 *The Cisco QoS Framework*


VoIP
Mission Critical Services
Multimedia Video Conference, Collaborative Congesting
VPNs
IntServ
DiffServ
MPLS
Hybrid
POLICY-BASED NETWORKING
PROVISIONING & MONITORING
Signaling Techniques (RSVP, DSCP*, ATM (UNI/NNI))
Classification & Marking Techniques (DSCP, IP Precedence, NBAR, etc.)
Congestion Avoidance Techniques (WRED)
Traffic Conditioners (Policing, Shaping)
Congestion Management Techniques (PQ, CQ, WFQ, CBWFQ, LLQ)
Link Efficiency Mechanisms (Compression, Fragmentation)
Frame Relay
PPP HDLC
SDLC
ATM, POS
FE, Gig_E 10Ge
Wireless Fixed, Mobile
Broadband Cable, xDSL

A flow consists of all the packets about which the following are true:

- All packets use the same transport layer protocol (for instance, UDP or TCP).
- All packets have the same source IP address.
- All packets have the same source port number.
- All packets have the same destination IP address.
- All packets have the same destination port number.

Packets can be classified and marked near the ingress points of traffic into the network; the QoS tools in the center of the network can just look for the marked field in the packet header, as shown in Figure 2-26.

Figure 2-26 *GOCS Design: Mark Packets near the Edge of the Network*

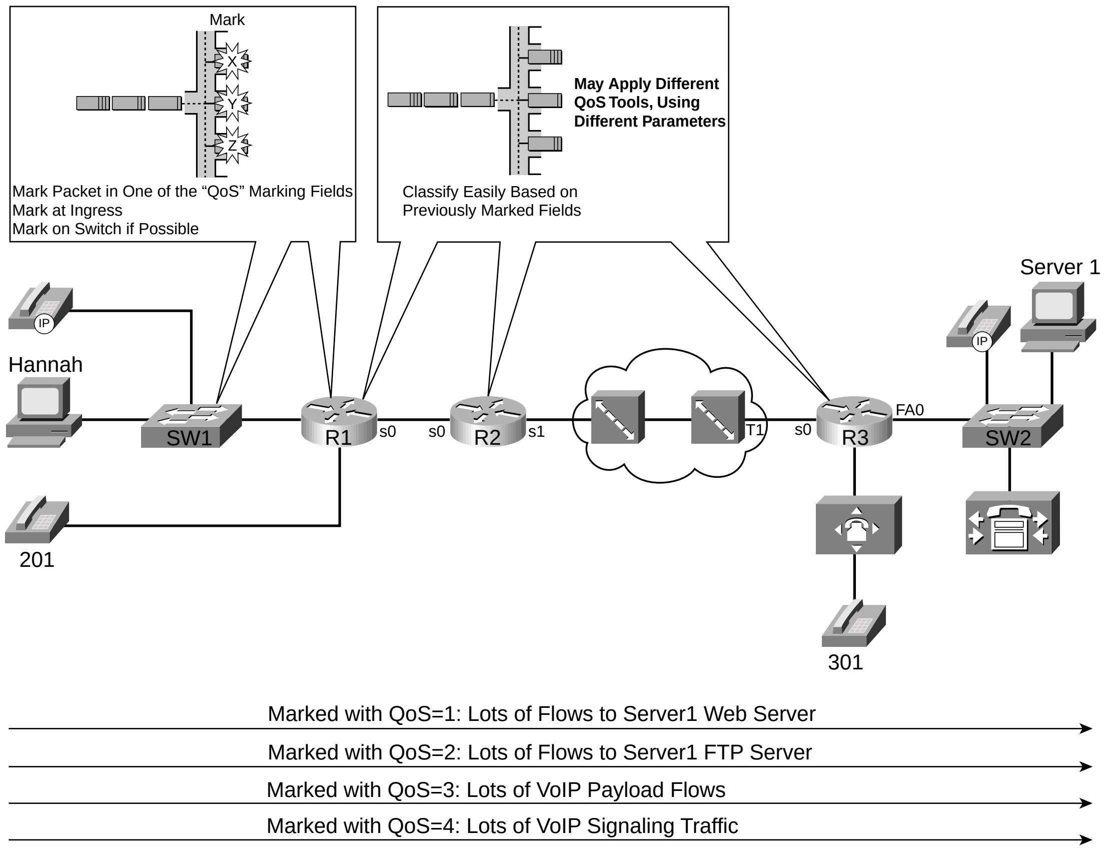

Mark
X
Y
Z
Mark Packet in One of the "QoS" Marking Fields
Mark at Ingress
Mark on Switch if Possible
May Apply Different QoS Tools, Using Different Parameters
Classify Easily Based on Previously Marked Fields
Server 1
IP
Hannah
SW1
R1
s0
s0
R2
s1
T1
s0
R3
FA0
SW2
201
301
Marked with QoS=1: Lots of Flows to Server1 Web Server
Marked with QoS=2: Lots of Flows to Server1 FTP Server
Marked with QoS=3: Lots of VoIP Payload Flows
Marked with QoS=4: Lots of VoIP Signaling Traffic

Table 2-28 summarizes the key points from the GOCS model. Although the exams do not actually cover anything called the GOCS model, the core concepts do provide background information about DiffServ and IntServ.

Table 2-28 *Summary of GOCS Features*

Feature	**Comments**
Flow	A flow is a unidirectional set of packets, sent from one source IP address and port, to one destination IP address and port, using the same transport layer protocol.
Flow-based QoS tools	Flow-based tools automatically recognize flows without explicit classification configuration. Each flow can be treated differently, providing a great amount of granularity for QoS.
Class-based QoS tools	Class-based tools treat packets differently based on the category or class to which they belong. The number of classes chosen in a typical network is small (typically fewer than 10). Network engineers choose which types of packets are placed into each class, so class-based QoS tools require explicit configuration of classification parameters.
QoS tools that are neither flow or class based	Some QoS tools operate on all traffic entering or exiting an interface. Other tools may work on a predetermined type of traffic. For example, compressed RTP only operates on packets with RTP headers. Therefore, some tools do not need to consider flows or explicitly defined classes.
QoS class planning—enterprise	For a single enterprise network, a considered, thorough analysis of the network can yield a relatively small set of useful QoS packet categories. Agreement to these categories should occur before beginning the classification and marking strategy of marking near the edge.
Classification and marking	Packets should be classified and marked near the edge of the network, as packets enter the network. By doing so, classification configuration on the remaining routers in the packets' path is simplified, reducing processing overhead, complexity, the risk of misconfiguration.
QoS class planning—Internet	For the Internet, anything more than a handful of general class conventions is unlikely to be agreed upon by a large number of ISPs and customers. Cisco suggests a short list with five categories: VoIP payload, video payload, voice/video signaling, important data, and not-so-important important data.
Internet reclassification and re-marking	Between different autonomous systems, reclassification and re-marking may need to occur. If the autonomous systems distrust each other, packets must be matched based on the contents of the various fields in their headers. If the autonomous systems trust each other, and agree on what packets belong to each of the traffic classes, all that may be required is to reclassify and re-mark based on the marked field inside the IP header.

Table 2-29 lists the RFCs that define DiffServ.

Table 2-29 *DiffServ RFCs*

RFC	Title	Comments
2474	Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers	Contains the details of the 6-bit DSCP field in IP header.
2475	An Architecture for Differentiated Service	This is the core DiffServ conceptual document.
2597	Assured Forwarding PHB Group	Defines a set of 12 DSCP values and a convention for their use.
2598	An Expedited Forwarding PHB	Defines a single DSCP value as a convention for use as a low-latency class.
3260	New Terminology and Clarifications for DiffServ	Clarifies, but does not supercede, existing DiffServ RFCs.

Table 2-30 lists the terms and their definitions. This table provides a reference for study for the Cisco QoS exams.

Table 2-30 *DiffServ Terminology and Their Definitions*

Term	Definition
Behavior aggregate (BA)	A DS behavior aggregate.
BA classifier	A classifier that selects packets based only on the contents of the DS field.
Classifier	An entity that selects packets based on the content of packet headers according to defined rules.
DS behavior aggregate	A collection of packets with the same DS code point crossing a link in a particular direction.
DS boundary node	A DS node that connects one DS domain to a node either in another DS domain or in a domain that is not DS capable.
DS code point	A specific value of the DSCP portion of the DS field, used to select a PHB.
DS compliant	Enabled to support differentiated services functions and behaviors as defined in [DSFIELD], this document, and other differentiated services documents; usually used in reference to a node or device.
DS ingress node	A DS boundary node in its role in handling traffic as it enters a DS domain.

continues

Table 2-30 *DiffServ Terminology and Their Definitions (Continued)*

Term	Definition
DS field	The IPv4 header ToS octet or the IPv6 traffic class octet when interpreted in conformance with the definition given in [DSFIELD]. The bits of the DSCP field encode the DS code point, whereas the remaining bits are currently unused.
Dropper	A device that performs dropping.
Marker	A device that performs marking.
Meter	A device that performs metering.
MF classifier	A multifield (MF) classifier that selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field, protocol ID, source port and destination port.
Per-hop behavior (PHB)	The externally observable forwarding behavior applied at a DS-compliant node to a DS BA.
Policing	The process of discarding packets (by a dropper) within a traffic stream in accordance with the state of a corresponding meter enforcing a traffic profile.
Re-mark	To change the DS code point of a packet, usually performed by a marker in accordance with a TCA.
Shaper	A device that performs shaping.
Traffic conditioner	An entity that performs traffic-conditioning functions and which may contain meters, markers, droppers, and shapers. Traffic conditioners are typically deployed in DS boundary nodes only. A traffic conditioner may re-mark a traffic stream or may discard or shape packets to alter the temporal characteristics of the stream and bring it into compliance with a traffic profile.

Table 2-12 contains material reprinted from RFC 2475.

Figure 2-27 puts some of the DiffServ terminology in context.

Figure 2-28 shows two enterprise networks and two ISPs, with examples of several of the DiffServ terms relating to interconnecting networks.

Figure 2-27 *Behavior Aggregates and Per-Hop Behavior*

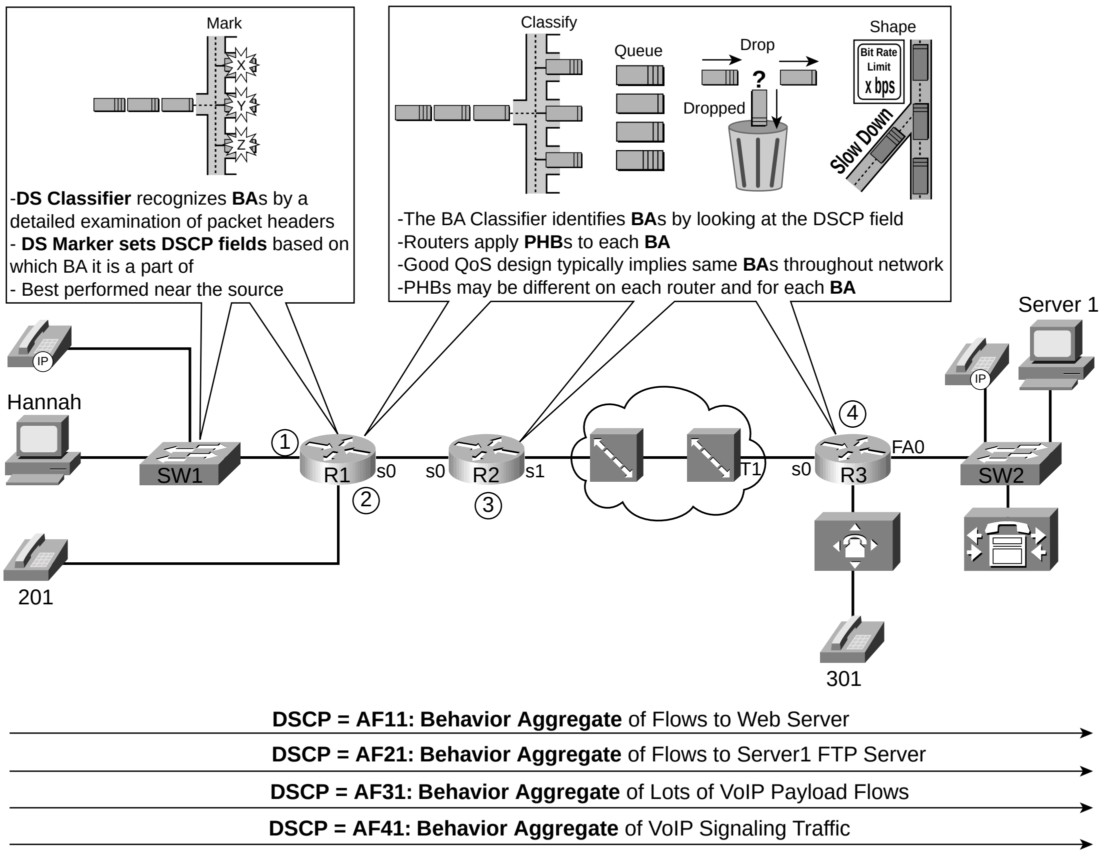

Mark
X
Y
Z
-**DS Classifier** recognizes **BA**s by a detailed examination of packet headers
- **DS Marker sets DSCP fields** based on which BA it is a part of
- Best performed near the source
Classify
Queue
Drop
?
Dropped
Shape
Bit Rate Limit
x bps
Slow Down
-The BA Classifier identifies **BA**s by looking at the DSCP field
-Routers apply **PHB**s to each **BA**
-Good QoS design typically implies same **BA**s throughout network
-PHBs may be different on each router and for each **BA**
Server 1
IP
Hannah
1
SW1
R1
s0
2
s0
R2
s1
3
T1
s0
4
R3
FA0
SW2
IP
201
301
DSCP = AF11: Behavior Aggregate of Flows to Web Server
DSCP = AF21: Behavior Aggregate of Flows to Server1 FTP Server
DSCP = AF31: Behavior Aggregate of Lots of VoIP Payload Flows
DSCP = AF41: Behavior Aggregate of VoIP Signaling Traffic

Figure 2-28 *DiffServ Domains, Regions, and Nodes*

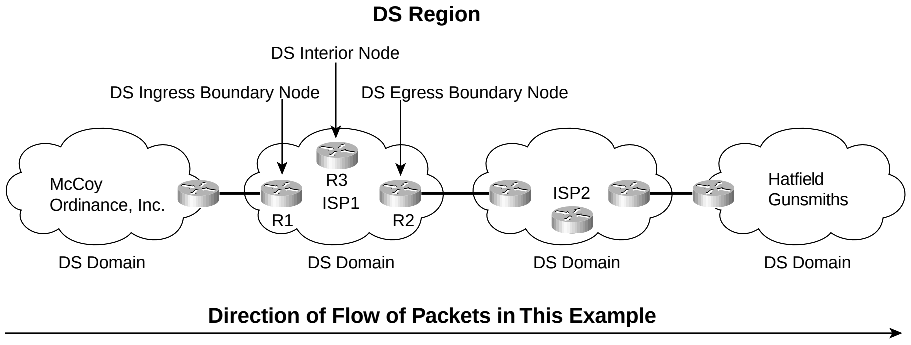


Figure 2-29 shows the fields inside the ToS byte (per RFC 1349) and the DS field (per RFC 2474).

Figure 2-29 *IP ToS Byte and DS Field*

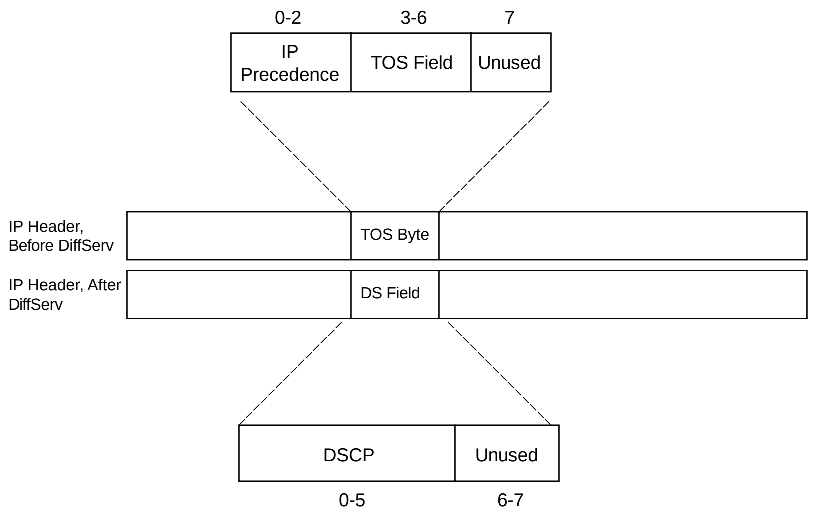


Table 2-31 lists DSCP values useful for QoS tools that only use precedence, and for those that also use DSCP.

Table 2-31 *Default and Class Selector DSCP Values*

Name of DSCP Class Selector Values Used by IOS	Binary Value	Equivalent Precedence Value (Decimal)
Default	**000**000	0
CS1	**001**000	1
CS2	**010**000	2
CS3	**011**000	3
CS4	**100**000	4
CS5	**101**000	5
CS6	**110**000	6
CS7	**111**000	7

The names of the code points in Table 2-14 match parameters found on IOS DiffServ-compliant classification commands. Because an "all-zeros" DSCP called "default" was already defined, there was no need to create a CS0 DSCP name.

Table 2-32 lists the DiffServ AF DSCPs.

Table 2-32 *Assured Forwarding DSCP Values—Names, Binary, and Decimal*

	Low Drop Probability Within Class	**Medium Drop Probability Within Class**	**High Drop Probability Within Class**
	Name/Decimal/Binary	**Name/Decimal/Binary**	**Name/Decimal/Binary**
Class 1	AF11 / 10 / 001010	AF12 / 12 / 001100	AF13 / 14 / 001110
Class 2	AF21 / 18 / 010010	AF22 / 20 / 010100	AF23 / 22 / 010110
Class 3	AF31 / 26 / 011010	AF32 / 28 / 011100	AF33 / 30 / 011110
Class 4	AF41 / 34 / 100010	AF42 / 36 / 100100	AF43 / 38 / 100110

Table 2-33 summarizes many of the key points about the various DiffServ PHBs.

Table 2-33 *Comparison of DiffServ PHBs*

PHB	**Key Components**	**Names of DSCPs**
Best effort (BE)	PHB for getting no specific QoS treatment	DSCP BE (default)
Class selector (CS)	Uses 8 DSCPs, all with binary 0s for the last 3 bits. Used for backward compatibility with IP precedence. Uses "bigger-is-better" logic—the bigger the DSCP, the better the QoS treatment.	CS1, CS2, CS3, CS4, CS5, CS6, CS7
Assured forwarding (AF)	PHB consists of 2 components: queuing to provide a minimum bandwidth to each for 4 different queues, and 3 drop thresholds inside each queue. DSCPs do not always follow the "bigger-is-better" logic.	AF11, AF12, AF13, AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43
Expedited forwarding (EF)	PHB also has 2 components: queuing to provide low delay/jitter/loss and a guaranteed amount of bandwidth, and policing to prevent EF from preventing other types of traffic from getting enough bandwidth.	EF

The terms relating to the traffic-conditioning functions of DiffServ should be mostly familiar by now. Table 2-34 summarizes the terms.

Figure 2-30 does not intend to show a flowchart of what happens to every packet as it passes through a DS boundary node, but rather to show some of the possible paths a packet can take.

Table 2-34 *Traffic Conditioners*

Traffic Conditioner	Explanation
Metering	The metering function measures the traffic rate to determine whether the traffic conforms to the stated contract, or exceeds the traffic contract. Metering typically occurs per class.
Dropping (policing)	If the traffic exceeds the contract, one option is to drop some packets, so that the packets that are allowed through meet the contract. Most implementations for this feature are called policing.
Shaping	If the traffic exceeds the contract, one option is to shape the traffic. Shaping just means to buffer or queue the traffic, slowing it down, so that the resulting sending rate is within the contract.
Marking	A third option for traffic that exceeds the contract is to re-mark the DSCP with a different value. For instance, Platinum AF41 (low drop probability) traffic that exceeds the contract might get re-marked to AF43. In such a case, if congestion were to occur, this packet would more likely be dropped than if not re-marked.
Do nothing	Not actually written down in the RFCs; one option is to allow the traffic to keep moving right along.

Figure 2-30 *RFC 2475—Equivalent Diagram of Classifier and Traffic-Conditioner Functions*

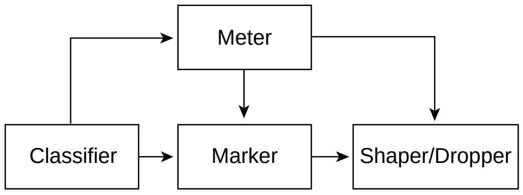
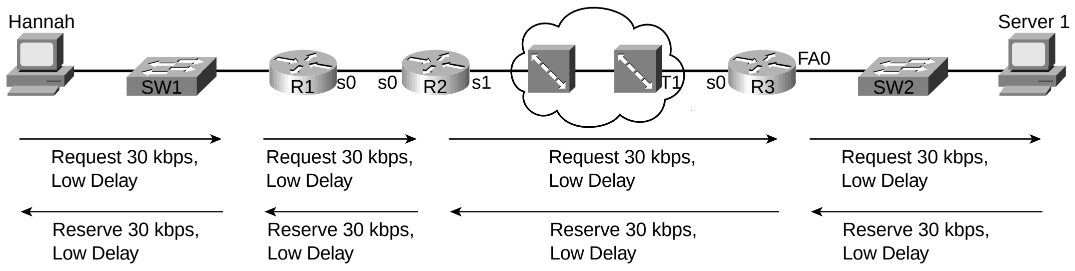


Figure 2-31 shows the general idea behind the IntServ reservation requests.

Figure 2-31 *Integrated Services Reservation Requests*


Hannah
Server 1
SW1
R1
s0
s0
R2
s1
T1
s0
R3
FA0
SW2
Request 30 kbps, Low Delay
Request 30 kbps, Low Delay
Request 30 kbps, Low Delay
Request 30 kbps, Low Delay
Reserve 30 kbps, Low Delay
Reserve 30 kbps, Low Delay
Reserve 30 kbps, Low Delay
Reserve 30 kbps, Low Delay

The following list summarizes some of the key points about IntServ that you will want to remember for the QoS exams:

- Integrated services defines the need, and suggests mechanisms, to provide bandwidth and delay guarantees to flows that request it. RFC 1633 defines it.
- IntServ contains two components: resource reservation and admission control.
- RSVP, as defined in RFCs 2205 through 2215, provides the IntServ resource reservation function. RFC 2210 specifically discusses RSVP’s usage for IntServ.
- With end-to-end RSVP, each intermediate router reserves the needed bandwidth when receiving a reservation request and confirms the request with RSVP reserve messages. If a router in the path does not speak RSVP, it just transparently passes the flow.
- When IntServ has not been implemented end to end, the RSVP messages can be forwarded in the non-IntServ part of the networks. In that case, the non-IntServ networks can either provide best-effort (BE) service, or provide IntServ-DSCP mapping if the intermediate network is a DiffServ domain.
- A router can offload the admission control function to a COPS server.

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD-ROM included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD-ROM nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD-ROM exam to include, or not include, the topics that are only on the CCIP QoS.

1. List four classification and marking tools, including the full names and popular acronyms.
2. List four queuing tools, including the full names and popular acronyms.
3. List four policing and shaping tools, including the full names and popular acronyms.
4. List three congestion-avoidance tools, including the full names and popular acronyms.
5. List four link-efficiency tools, including the full names and popular acronyms.
6. List seven VoIP CAC tools, including the full names and popular acronyms.
7. List four QoS management tools, including the full names and popular acronyms.
8. List the QoS tools that perform some classification function.
9. Which of the following tools can be used for classification and marking? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP
10. Which of the following tools can be used for queuing? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP
11. Which of the following tools can be used for policing? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP
12. Which of the following tools can be used for shaping? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP
13. Which of the following tools can be used for link efficiency? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

14 Which of the following tools can be used for network management? CAR, CB marking, PQ CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

15 Define the DiffServ term "behavior aggregate."

16 Define the DiffServ term "DSCP," including what the acronym stands for.

17 Define the DiffServ term "PHB," including what the acronym stands for.

18 Define the DiffServ term "MF classifier," including what the acronym stands for.

19 Define the DiffServ term "DS ingress node," including what the acronym stands for.

20 Compare and contrast the terms "BA classifier" and "MF classifier," according to DiffServ specifications. Suggest typical points in the network where each is used.

21 Compare and contrast the terms "shaper," "meter," and "dropper," according to DiffServ specifications. Suggest typical points in the network where each is used.

22 Compare and contrast the contents of the IP ToS byte before and after the advent of DiffServ.

23 Describe the QoS behavior at a single DS node when using the AF PHB. Also explain what the acronym "AF PHB" represents and identify the RFC that defines it.

24 Explain (by comparing and contrasting) whether AF and CS PHB DSCPs conform to the concept that "bigger DSCP values are better than smaller values."

25 Describe the QoS behavior at a single DS node when using the EF PHB. Also explain what the acronym "EF PHB" represents and identify the RFC that defines it.

26 Describe the process used by RSVP to reserve bandwidth in a network.

27 Imagine an enterprise network, connected to an Internet service provider (ISP), that is connected to a second ISP, which is then connected to another enterprise network. The second ISP does not support IntServ directly. Discuss the two options that allow the other three networks to support IntServ for flows that pass through the nonsupporting ISP.

28 List and describe the two main features of IntServ.

29 Compare and contrast DiffServ and IntServ in terms of using classes, flows, and scalability.

30 Describe the two options available to a router to perform IntServ admission control.

31 What is the QoS framework, and what does it define?

32 Which five categories of QoS tools are shown in the bottom half of the Cisco QoS framework?

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Topics

- Explain the reason for classification and marking.
- Explain the difference between classification and marking.
- Explain class of service, IP precedence, and DiffServ code points.
- Configure QoS policy using Modular QoS CLI.
- Explain the role of network-based application recognition (NBAR).
- Classify and mark traffic.

QoS Exam Objectives

- Describe policy-based routing and how it can be used to classify and mark IP packets.
- Configure the policy-based routing mechanism on Cisco routers.
- List other mechanisms that also support classification and marking capabilities (committed access rate, class-based marking).
- Describe the Modular QoS CLI (MQC) concept and its structure.
- Describe Modular QoS CLI classification options.
- Configure the Modular QoS CLI to perform classification.
- Describe network-based application recognition (NBAR).
- Describe Modular QoS CLI policy options.
- Configure the Modular QoS CLI to perform service policies.

CHAPTER 3

Classification and Marking

QoS classification tools categorize packets by examining the contents of the frame, cell, and packet headers; whereas marking tools allow the QoS tool to change the packet headers for easier classification. Many QoS tools rely on a classification function to determine to which traffic the tool applies. To place voice and data traffic in separate queues, for example, you must use some form of classification to differentiate the two types of traffic and place the identified traffic in the proper queue. Marking provides a way for QoS tools to change bits in the packet header to indicate the level of service this packet should receive from other QoS tools. For instance, you can use marking tools to change the marking in voice packets to ensure that a classification tool can differentiate a voice packet from a data packet. Without the marking feature, the frame, packet, or cell remains unchanged.

Marking involves placing a value into one of the small number of well-defined frame, packet, or cell header fields specifically designed for QoS marking. By marking a packet, other QoS functions can perform classification based on the marked field inside a header. Marking simplifies the network's QoS design, it simplifies configuration of other QoS tools, and it reduces the overhead required by each of the other QoS tools to classify the packets.

Although classification and marking tools do not directly affect the bandwidth, delay, jitter, or loss experienced by traffic in the network, classification and marking tools are the building blocks for all other QoS tools. With these tools, all traffic on the network is identified for the next QoS tool to act upon.

The concepts that apply to all classification and marking are covered in the first section of this chapter, including the terminology, fields used, and the meaning behind each of the available marked fields. Following that, each of the classification and marking tools is covered, with example configurations, **show**, and **debug** commands.

"Do I Know This Already?" Quiz Questions

The purpose of the "Do I Know This Already?" quiz is to help you decide whether you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 8-question quiz, derived from the major sections in "Foundation Topics" portion of the chapter, helps you determine how to spend your limited study time.

Table 3-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 3-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions
1	Classification and Marking Concepts	1 to 4
2	CAR, PBR, and CB Marking	5 to 8
All questions		1 to 8

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **6 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary," and the "Q&A" sections.
- **7 or 8 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, move to the next chapter.

Classification and Marking Concepts Questions

1. Describe the difference between classification and marking.
2. Describe, in general, how a queuing feature could take advantage of the work performed by a classification and marking feature.
3. Which of the following QoS marking fields are carried inside an 802.1Q header: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?
4. Which of the following QoS marking fields are carried inside an IP header: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

CAR, PBR, and CB Marking Questions

5 Define the meaning of MQC, and spell out what the acronym stands for.

6 What configuration command lists the marking details when configuring CB marking? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

7 What configuration command lists the marking details when configuring CAR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

8 What configuration command lists the classification details when configuring PBR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

Foundation Topics

The contents of the "Foundation Topics" section of this chapter, and most of the rest of the chapters in this book, follow the same overall flow. Each chapter describes a type of category of QoS tool. Each "Foundation Topics" section begins with coverage of the concepts behind these tools. Then, each tool is examined, with coverage of how each tool works like the other tools, and how it works differently than the other tools. So, most of the core concepts are explained in the first part of the chapter; some of the concepts may be explained in the section about a specific tool, however, particularly if the concepts apply only to that tool.

The second part of the chapter covers several classification and marking tools: class-based marking (CB marking), committed access rate (CAR), policy-based routing (PBR), and dial peers. For each tool, the pertinent configuration, **show**, and **debug** commands are also covered.

Classification and Marking Concepts

Most QoS tools classify traffic, which allows for each class of traffic to receive a different level of treatment from other traffic classes. You can use this method to prioritize one type of traffic over another. Classification and marking tools play a large role in this solution. Instead of giving queuing preference, or dropping a packet, or shaping, or policing, or fragmenting, and so on, classification and marking tools change some bits in the packet header. Other QoS tools throughout the network can examine the marked bits to classify packets.

First, this discussion focuses on classification, and then examines marking more closely.

Classification

Almost every QoS tool uses classification to some degree. To put one packet into a different queue than another packet, the IOS must somehow differentiate between the two packets. To perform header compression on Real Time Protocol (RTP) packets, but not on other packets, the IOS must determine which packets have RTP headers. To shape data traffic going into a Frame Relay network, so that the voice traffic gets enough bandwidth, the IOS must differentiate between Voice over IP (VoIP) and data packets. If an IOS QoS feature needs to treat two packets differently, you must use classification.

Because most QoS tools need to differentiate between packets, most QoS tools have classification features. In fact, many of you will already know something about several of the QoS tools described in this book, and you will realize that you already know how to perform classification using some of those tools. For instance, many QoS tools enable you to classify using access-control lists (ACLs). If ACL 101 *permits* a packet, a queuing tool might put the packet into one queue; if ACL 102 permits a packet, it is placed in a second queue; and so on. In one way of thinking, queuing could instead be called *classification and queuing*, because the queuing feature must somehow decide which packets end up in each queue. Similarly, traffic shaping could be called *classification and traffic shaping*, policing could be called *classification and policing*, and so on. Because most QoS tools classify traffic, however, the names of most QoS tools never evolved to mention the classification function of the tool.

Most classification and marking tools, like the other types of QoS tools, generally operate on packets that are entering or exiting an interface. The logic works something like an ACL, but the *action* is marking, as opposed to allowing or denying (dropping) a packet. More generally, classification and marking logic for ingress packets can be described as follows:

- For packets entering an interface, if they match criteria 1, mark a field with a value.
- If the packet was not matched, compare it to criteria 2, and then mark a potentially different field with a potentially different value.
- Keep looking for a match of the packet, until it is matched, or until the classification logic is complete.

NOTE This book uses the following terms describe the data structures used when sending data:

- **Frame**—Bits that include the data link layer header and trailer (for example, Ethernet frame and Frame Relay frame)
- **Cell**—Specifically, an Asynchronous Transfer Mode (ATM) cell
- **Packet**—Bits that include the network layer header, but does not include the data link header (for instance, an IP packet)
- **Segment—**Bits that include the TCP or UDP header, but not the data link or network layer header

---

The key to evaluating the classification features of different classification and marking tools is to examine what can be matched in packet headers. Because many classification and marking tools can refer to IP ACLs to classify packets, Table 3-2 shows the list of items that you can match with an extended IP ACL. Table 3-3 lists the fields that classification and marking tools can match without use of an ACL. Note that some header fields can be matched by an ACL or directly through some other style of configuration—in those cases, it is typically better to match the field directly, rather than with an ACL.

Table 3-2 *IP Extended ACL Matchable Fields—IOS 12.2*

Field	Comments
Source IP address	A range of source IP addresses can be matched by using a wildcard mask.
Destination IP address	A range of source IP addresses can be matched by using a wildcard mask.

continues

Table 3-2 *IP Extended ACL Matchable Fields—IOS 12.2 (Continued)*

Field	Comments
IP Precedence	Format of command uses names for precedence. The following table lists the decimal value for each name. Name IP precedence value **routine** 0 **priority** 1 **immediate** 2 **flash** 3 **flash-override** 4 **Critic** 5 **internet** 6 **network** 7
IP DSCP	Format of the command allows use of differentiated services code point (DSCP) names, as well as decimal values.
IP ToS	Can check to see whether a single Type of Service (ToS) field bit is toggled on; keywords are **normal** (binary **0000**), **max-reliability** (binary **1000**), **max-throughput** (binary **0100**), **min-delay** (binary **0010**), and **min-monetary-cost** (binary **0001**).
TCP ports	Can check source and destination ports; can also check a range of port numbers, whether a port number is larger or smaller than a single value.
TCP Established	Although not typically useful for QoS classification, ACLs can match all TCP segments after the initial segment used for connection establishment.
UDP	Checks the source and destination ports; can also check a range of port numbers, whether a port number is larger or smaller than a single value.

Table 3-2 *IP Extended ACL Matchable Fields—IOS 12.2 (Continued)*

Field	Comments
ICMP	Checks a larger variety of ICMP messages and code types (for example, echo request and echo reply).
IGMP	Checks for Internet Group Management Protocol (IGMP) message types.

Table 3-3 *Fields* Directly *Matchable by Classification and Marking Tools*

Field	Tool	Comments
Source MAC address	CAR, CB marking	Committed access rate (CAR) uses special "access-rate" ACLs; class-based (CB) marking uses the **match** command.
IP Precedence	CAR, CB marking	CAR uses special "access-rate" ACLs specific to CAR; CB marking uses the **match** command; both can match a subset of values.
MPLS Experimental	CAR, CB marking	CAR uses special "access-rate" ACLs specific to CAR; CB marking uses the **match** command; both can match a subset of values.
CoS	CB marking	Checks incoming ISL/802.1P CoS bits. Can match multiple values.
Destination MAC address	CB marking	Checks for destination MAC address. Can match multiple values.
Input Interface	CB marking	Checks for input interface. Can match multiple values.
IP DSCP	CB marking	Can check for multiple values using multiple **match** commands.
RTP's UDP port-number range	CB marking	RTP uses even-numbered UDP ports from 16,384 to 32,767. This option allows matching a subset of these values, even-numbered ports only, because RTP only uses even-numbered ports.
QoS Group	CB marking	The QoS Group field is used to tag packets internal to a single router.
NBAR protocol types	CB marking	Refer to the "Network Based Application Recognition (NBAR)" section in this chapter for more details.
NBAR Citrix applications	CB marking	NBAR can recognize different types of Citrix applications; CB marking can use NBAR to classify based on these application types.

continues

Table 3-3 *Fields* Directly *Matchable by Classification and Marking Tools (Continued)*

Field	Tool	Comments
Host name and URL string	CB marking	NBAR can also match URL strings, including the host name, using regular expressions. CB marking can use NBAR to match these strings for classification.
Outgoing Interface	Policy-based routing (PBR)	Checks the routing table and finds all valid routes for the packet; matches based on the outgoing interface.
Next-Hop	PBR	Similar to the outgoing interface, but it checks the next-hop routers' IP addresses.
Metric	PBR	Checks the routing table entry for this packet, and compares the metric value to match the packet.
Route type	PBR	Checks the routing table, looking at the source of the routing table entry that matches the packet.
Dial Peer	Dial peers	Based on the dial peer and used to connect a VoIP call.

These two tables can be a little intimidating, especially for those of you studying to pass the QoS exams! Rest assured, however, that this book covers approximately 30 QoS tools, and the exams typically have no more than 60 questions on QoS. So, statistically speaking, the exams simply do not have enough questions to ask you about every little item that a tool could use for classification. I just included these tables, and others like them, for reference.

Marking

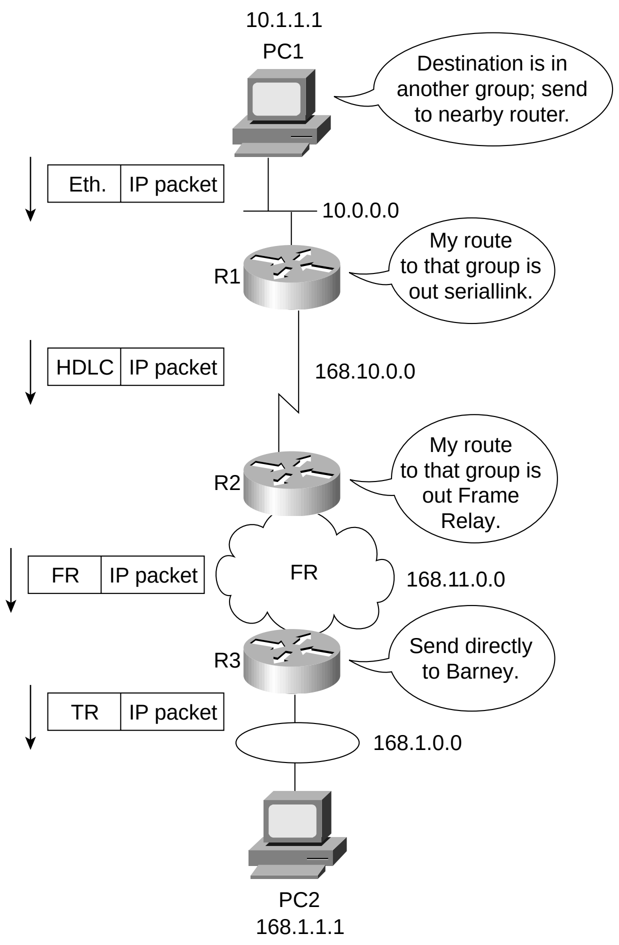
Marking involves setting some bits inside a data link or network layer header, with the goal of letting other devices' QoS tools classify based on the marked values. You can mark a wide variety of fields, and each has a particular purpose. Some fields are more widely used, and some are less widely used. Some marking options make sense for all devices on the LAN, whereas others only when using specific hardware platforms. Marking at the WAN is possible, too.

The following sections list the header fields that you can use for marking, along with explanations of when it is most useful to use that particular field. Recommendations follow these sections as to when to use classification and marking.

IP Header QoS Fields: Precedence and DSCP

The two most popular marking fields for QoS are the IP Precedence and IP DSCP fields that were introduced in Chapter 2, “QoS Tools and Architectures.” QoS tools frequently use these two fields in part because the IP packet header exists from endpoint to endpoint in a network, as shown in Figure 3-1.

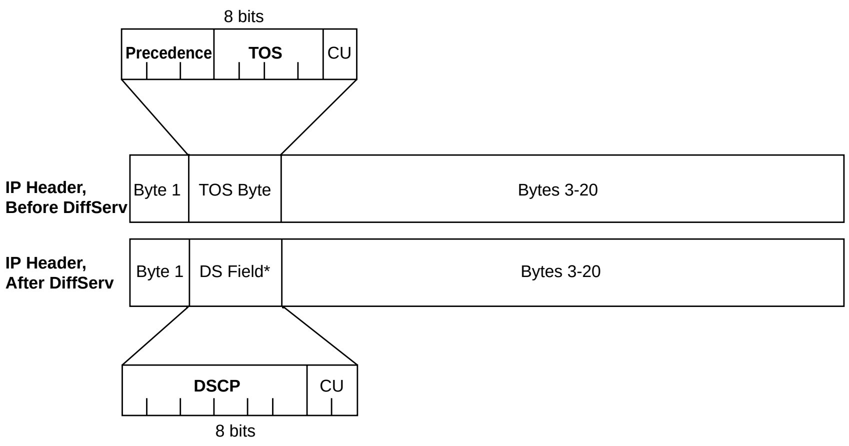
Figure 3-1 *Headers Used During Typical Packet Flow*



As seen in Figure 3-1, the IP packet en route to host PC2 stays intact throughout the network, whereas the data-link headers only exist while a frame is crossing between a host and a router, or between routers.

Figure 3-2 outlines the two marking fields and their positions inside an IP header.

Figure 3-2 *IP Precedence and IP DSCP Fields*



You can mark the Precedence and DSCP fields with any valid binary value of either 3 or 6 bits, respectively. Chapter 2 contains detailed discussion of the recommended values used in these two fields. Briefly, Precedence field values should grow in importance, and in QoS behavior, as the number gets higher. DSCP differs in that several per-hop behavior (PHB) RFCs define suggested DSCP values for which the larger number does not always get a better QoS treatment.

Table 3-4 lists the IP precedence and DSCP values, and their names, for review. Note that not all DSCP values are listed; only the DSCP values suggested by the DiffServ RFCs are listed in the table. QoS tools that are capable of setting DSCP can set any of the actual 64 values.

Table 3-4 *IP Precedence and DSCP—Popular Values and Names*

Field and Value (Decimal)	Binary Value	Name	Defined by This RFC
Precedence 0	**000**	**routine**	791
Precedence 1	**001**	**priority**	791
Precedence 2	**010**	**immediate**	791
Precedence 3	**011**	**flash**	791
Precedence 4	**100**	**flash override**	791
Precedence 5	**101**	**critic**	791
Precedence 6	**110**	**internetwork control**	791
Precedence 7	**111**	**network control**	791
DSCP 0	**000**000	**best effort** or **default**	2475

Table 3-4 *IP Precedence and DSCP—Popular Values and Names (Continued)*

Field and Value (Decimal)	Binary Value	Name	Defined by This RFC
DSCP 8	**001**000	CS1	2475
DSCP 16	**010**000	CS2	2475
DSCP 24	**011**000	CS3	2475
DSCP 32	**100**000	CS4	2475
DSCP 40	**101**000	CS5	2475
DSCP 48	**110**000	CS6	2475
DSCP 56	**111**000	CS7	2475
DSCP 10	**001**010	AF11	2597
DSCP 12	**001**100	AF12	2597
DSCP 14	**001**110	AF13	2597
DSCP 18	**010**010	AF21	2597
DSCP 20	**010**100	AF22	2597
DSCP 22	**010**110	AF23	2597
DSCP 26	**011**010	AF31	2597
DSCP 28	**011**100	AF32	2597
DSCP 30	**011**110	AF33	2597
DSCP 34	**100**010	AF41	2597
DSCP 36	**100**100	AF42	2597
DSCP 38	**100**110	AF43	2597
DSCP 46	**101**110	EF	2598

CS = Class Selector
AF = Assured Forwarding
EF = Expedited Forwarding

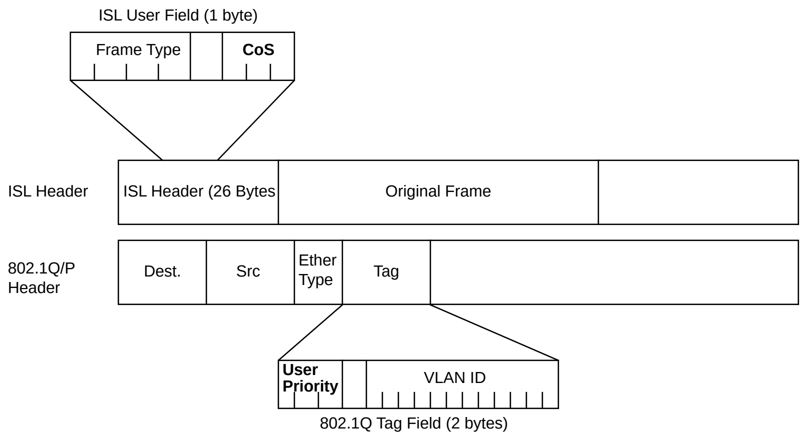
The two IP header QoS marking fields do not provide all the QoS marking fields needed today. One day, all other Layer 3 protocols besides IP may no longer be used. One day, all LAN switches will be capable of looking at IP headers, including IP DSCP and Precedence, and perform QoS based on those fields. Likewise, one day, all WAN services, including Frame Relay and ATM switches, will be able to perform QoS based on these same fields. However, today's reality is that even as more and more devices become capable of marking and reacting to IP precedence and DSCP, it will be a long time before all networking devices are both capable and configured to use these fields for QoS purposes. So, other QoS marking fields are needed.

LAN Class of Service (CoS)

Many LAN switches today can mark and react to a Layer 2 3-bit field called the Class of Service (CoS) located inside an Ethernet header. The CoS field only exists inside Ethernet frames when 802.1Q or Inter-Switch Link (ISL) trunking is used. You can use the field to set 8 different binary values, which can be used by the classification features of other QoS tools, just like IP precedence and DSCP.

Figure 3-3 shows the general location of the CoS field inside ISL and 802.1P headers.

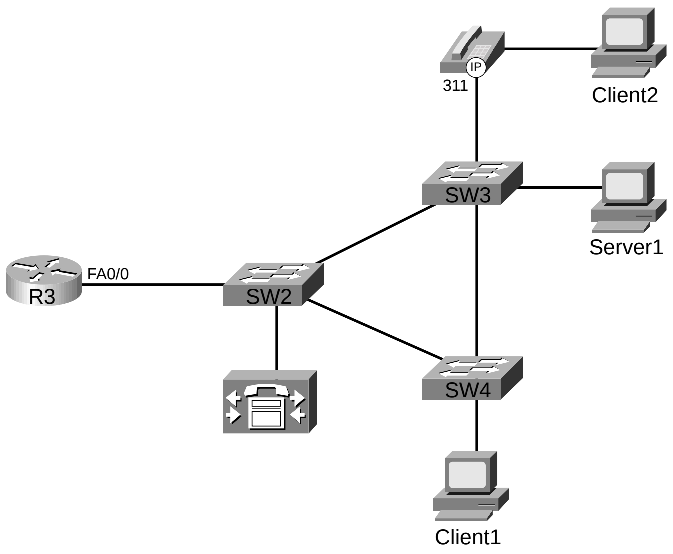
Figure 3-3 *LAN CoS Fields*



The term *CoS* really refers to two different fields—a field inside either the 802.1Q trunking header, or a field inside the ISL header. The IEEE 802.1Q standard uses the 3 most-significant bits of the 2-byte Tag Control field, called the user-priority bits. The Cisco proprietary ISL specification uses the 3 least-significant bits from the 1-byte User field, which is found inside the ISL header's user field byte. In general conversation, and in the QoS courses from Cisco, the term CoS applies to either of these two fields.

When can CoS be marked, and when can it be useful for classification to a QoS tool? First of all, trunking with 802.1Q or ISL must be enabled before the CoS field even exists! Second, as soon as the packet experiences Layer 3 forwarding, either with a router or some Layer 3 switch, the old LAN header gets discarded—which means you lose the CoS field. After a CoS field has been created and populated with the desired markings, routers and switches have several QoS tools that can react to these markings. Consider, for instance, a typical trunking environment, as shown in Figure 3-4, where all LAN switches are only performing Layer 2 switching.

Figure 3-4 *CoS—Trunking Locations in a Typical Network, Layer 2 Switches Only*



To mark the CoS bits, trunking must be used—and in Figure 3-4, trunking could be used on every Ethernet segment. Switches typically use trunking on the Ethernet segments to other switches, routers, and to IP Phones. Typically, switches do not need to use trunking on segments connected to PCs or servers. Because some networking cards have the capability to support 802.1Q or ISL trunking, however, servers and PCs can set the CoS bits.

NOTE Trunking requires a Fast Ethernet interface, or Gigabit, or 10 Gigabit—it is not supported over 10-Mbps Ethernet. This book does not distinguish among the different types of Ethernet upon each mention.

Both routers and switches use QoS tools that can react to the marked CoS bits. Cisco routers can indeed mark CoS bits for frames exiting an Ethernet interface that supports trunking. For instance, R3 could mark CoS 5 on a frame it forwards out its FA 0/0 interface. Other Cisco router QoS tools can react to the marked CoS bits on incoming frames as well. For instance, R3 could mark packets entering its FA0/0 interface with a particular DSCP value based on the incoming CoS value. Later in this chapter, you will see a sample configuration for class-based marking that performs both of these functions.

Cisco switches vary widely regarding their capabilities to set CoS bits and react to previously marked CoS bits. Switches can support marking of CoS, and more often today support marking of IP precedence and DSCP as well. LAN switches that do support QoS features generally perform output queuing, and sometimes input queuing, choosing queues based on CoS values. Congestion avoidance using Weighted Random Early Detection (WRED) is another typical switch QoS feature. In addition, some switches support policing tools, also based on CoS. Although campus QoS is not covered in depth on the QoS exams today, it is an important topic, particularly with converged voice, video, and data networks. Chapter 10, "LAN QoS," covers LAN QoS in additional depth.

Other Marking Fields

You can use single-bit fields in Frame Relay and ATM networks to mark a frame or cell for Layer 2 QoS. Unlike IP precedence, IP DSCP, and 802.1P/ISL CoS, however, these two fields are not intended for general, flexible use. Each of these single-bit fields, when set, imply that the frame or cell is a better candidate to be dropped, as compared with frames or cells that do not have the bit set. In other words, you can mark the bit, but the only expected action by another QoS tool is for the tool to drop the frame or cell.

Frame Relay defines the discard eligibility (DE) bit, and ATM defines the cell loss priority (CLP) bit. The general idea is that when a device, typically a WAN switch, experiences congestion, it needs to discard some frames or cells. If a frame or cell has the DE or CLP bit set, respectively, the switch may choose to discard those frames or cells, and not discard other frames or cells. If the DE or CLP bit is set, there is no requirement that the Frame Relay and ATM switches react to it—just like there is no guarantee that an IP packet with DSCP EF will get special treatment by another router. It's up to the owner of the Frame Relay or ATM switch to decide whether it will consider the DE and CLP bits, and how to react differently.

You can use two other QoS marking fields in specialized cases. The MPLS Experimental bits comprise a 3-bit field that you can use to map IP precedence into an MPLS label. This allows MPLS routers to perform QoS features indirectly based on the original IP Precedence field inside the IP packets encapsulated by MPLS, without the need to spend resources to open the IP packet header and examine the IP Precedence field.

Finally, the QoS Group field, an internal marking that exists only within the router, may be set as a packet passes through the fabric of a Cisco gigabit switch router (GSR) or edge services router (ESR). QoS processing can be performed more quickly inside the switching fabric by using the QoS group. Therefore, you may want to configure GSRs and ESRs to mark the QoS group on ingress so that QoS processing occurs more rapidly.

Summary of Marking Fields

Not all these marked fields receive the same amount of attention on the QoS exams. Refer to the Introduction of this book, and the website suggested there, for the latest information about where to focus your attention. Table 3-5 summarizes the marking fields.

Table 3-5 *Names of Marking Fields*

Field	Location	Length	Comments
IP Precedence	IP header	3 bits	Contained in the first 3 bits of the ToS byte.
IP DSCP	IP header	6 bits	Contained in the first 6 bits of the DS field, which replaces the ToS byte.
DS	IP header	1 byte	Replaces ToS byte per RFC 2475.
ToS	IP header	1 byte	Replaced by DS field per RFC 2475.
ToS	IP header	4 bits	A field inside the ToS byte; superseded by RFC 2475.
CoS	ISL and 802.1Q/P	3 bits	Cisco convention uses “CoS” to describe either trunking headers’ QoS field.
Priority bits	802.1Q/P	3 bits	The name used by IEEE 802.1P for the CoS bits.
Discard Eligible (DE)	Frame Relay header	1 bit	Frame Relay switches may discard DE-marked frames, avoiding discarding frames without DE marked, under congestion.
Cell Loss Priority (CLP)	ATM cell header	1 bit	ATM equivalent of the DE bit
MPLS Experimental values(s)	MPLS header	3 bits	Used to pass QoS marking information across an MPLS network.
QoS Group	Headers internal to IOS	N/A	Uses values between 1–99 inclusive. Used for marking only internal to a single router, specifically only on the GSR/ESR product lines.

The names of the various marking fields can be confusing. Quality of service (QoS) does not refer to any specific marking field, but it is a term that refers to a broad set of tools that effect bandwidth, delay, jitter, and loss. In other words, this whole book is about QoS. Class of service (CoS) refers to both of the two 3-bit fields in Ethernet trunking headers—one in the ISL header, and one in the 802.1Q trunking header. However, CoS also refers to a 2-bit field inside Systems Network Architecture (SNA) Layer 3 headers, which is also used for QoS functions. Type of service (ToS) is my personal favorite—ToS is the 1-byte field in the IP header, which includes a 3-bit Precedence field, and 4 ToS bits. And of course, DiffServ re-defines the ToS Byte as the DS-byte, with the DSCP field in the first 6 bits. Make sure you remember the true meanings of QoS, CoS, ToS, Precedence, and DSCP.

Classification and Marking Design Choices

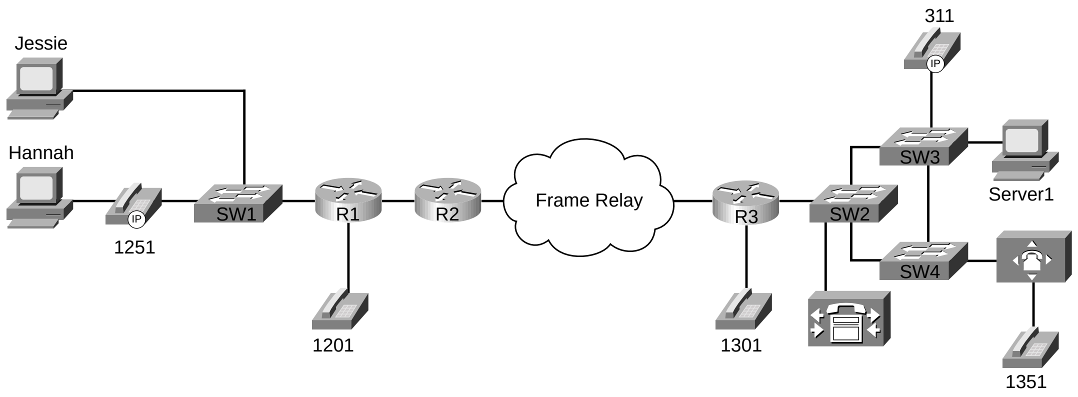
Classification and marking tools provide many options, but sometimes sorting out the best way to use the tools can be difficult. Classification and marking tools can classify based on a large number of frame and packet header fields. They can also mark a number of fields, the most notable being the IP Precedence and DSCP fields. You can use the classification and marking tools on all routers in the network, on many LAN switches, and even on IP Phones and host computers. This brief section discusses some of the classification and marking design choices.

The first step in making good classification and marking design choices is to choose where to mark. The general rule for choosing where to mark is as follows:

> Mark as close to the ingress edge of the network as is possible.

Figure 3-5 diagrams a typical enterprise IP network, which will be used to look more closely at the options for where to mark packets.

Figure 3-5 *Typical Enterprise Network*



Consider packets that flow left to right in Figure 3-5. Hannah and Jessie, both client PCs, can mark IP precedence, IP DSCP, and CoS if their Ethernet card supports ISL or 802.1Q. The IP Phone internally marks its own voice bearer traffic precedence 5, DSCP EF, and CoS 5 by default, its own voice signaling traffic precedence 3, DSCP 31, and CoS 3. The phone can also re-mark the CoS, precedence, and DSCP sent by Hannah's PC. (The phone default action is to re-mark 0 for all three values.) SW1, depending on the type of switch, might be able to re-mark CoS, re-mark precedence or DSCP, or make general (multifield) classification and marking decisions—in other words, it might be able to look at some of the fields listed earlier in Tables 3-2 and 3-3. Finally, R1 can use general multifield classification and marking before sending the packet over the WAN—but over the next link to R2, because the link is a PPP link, the only marking options would be in the IP header.

So marking can be done in many places near the ingress edge of the network—but whom do you trust? Classification and marking should not be performed before the frame/packet reaches a trusted device. This location in the network is called the *trust boundary.* For instance, Jessie formerly marked her packets with DSCP default, but because the user of the PC can change that value, Jessie changed to use DSCP EF to get better service. In most cases, the end-user PCs are beyond the trust boundary. IP Phones can reset CoS, precedence, and DSCP to 0 for Hannah's traffic, and mark the VoIP with CoS 5, precedence 5, and DSCP EF—with the added benefit that the phone user cannot reset those values. The IP Phone trust settings are controlled by the connected Cisco Catalyst switch, enabling the system administrator to trust markings received from the IP Phone while rewriting the values received from the attached PC.

The final consideration when deciding where to mark involves the function of the various devices, and personal preferences. For instance, IP Phones provide three classes—one for voice bearer traffic, one for voice signaling traffic, and one for all packets from the PC. However, a network may need multiple classes for data traffic, so further classification may be required by a switch or router. Some switches provide robust Layer 3 QoS classification and marking functions—in these cases, classification and marking may be performed on the switch; otherwise classification and marking must be performed on the router. Figure 3-6 outlines some of the strategies for classification and marking for three different LAN topologies.

Figure 3-6 shows three typical paths for frames between the end-user device and the first router. The first instance shows a typical installation near the end users—a switch that performs only Layer 2 QoS, and PCs connected to it. *Only Layer 2 QoS* just means that the switch can react to, or possibly set, CoS, but it cannot react to or mark IP precedence or DSCP. In this case, classification and marking is typically performed as packets enter R1's Ethernet interface. In addition, because SW1 can support CoS, but not precedence or DSCP, R1 may want to map incoming CoS values to the Precedence or DSCP fields.

The second part of Figure 3-6 shows a network with a Layer 3 QoS-capable switch. Depending on the type of switch, this switch may not be able to perform Layer 3 switching, but it does have the capability to react to or mark IP precedence or DSCP. In this case, you should classify and mark on the switch. Classification and marking on the Layer 3 switch allows classification and marking closer to the trust boundary of the network, and offers the added benefits of queuing, congestion avoidance, and policing based on the marked values. If only a few sites in the network have Layer 3 QoS-capable switches, you may prefer to perform classification and marking on the router, so all sites' configurations are similar. However, classifying and marking in the router places additional overhead on the router's CPU.

Finally, the third example shows a PC cabled through an IP Phone to a Layer 3 QoS-capable switch. The IP Phone can easily take care of classification and marking into two categories—voice and nonvoice. The switch and router can take advantage of those marked values. If more classes are needed for this network's QoS policy, SW3, or R3, can perform classification and marking. Of course, if the QoS policy for this network only requires the three classes—one for voice bearer traffic, one for voice signaling traffic, one for nonvoice—and all PCs are connected through the switch in the IP Phone, no classification and marking is needed on SW3 or R3!

Figure 3-6 *Three Classification and Marking Placement Strategies*

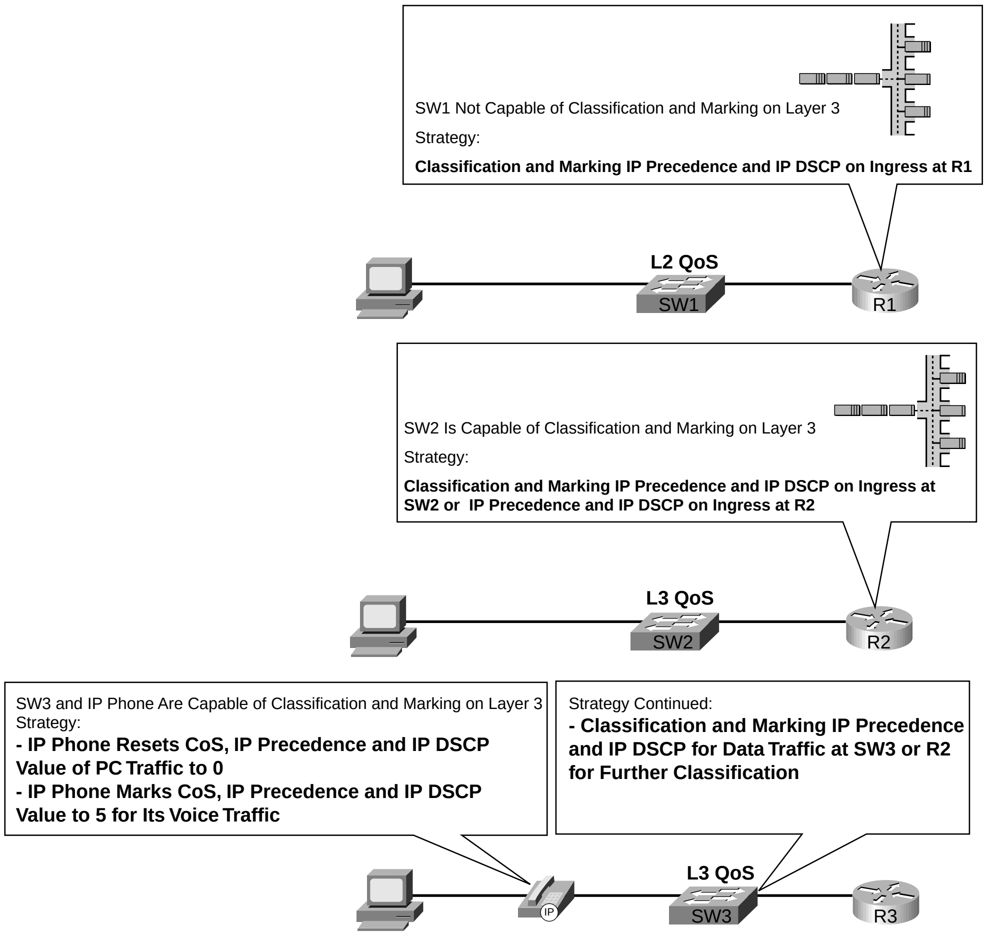
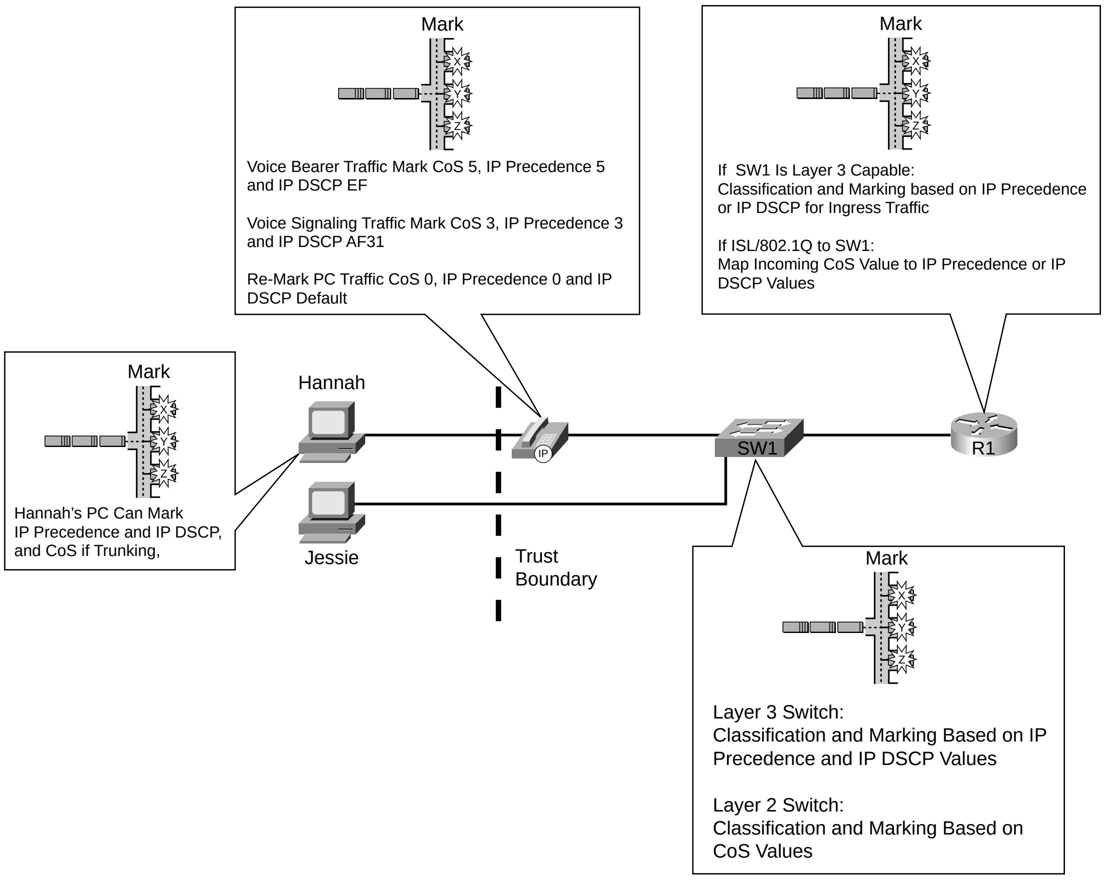

SW1 Not Capable of Classification and Marking on Layer 3
Strategy:
Classification and Marking IP Precedence and IP DSCP on Ingress at R1
L2 QoS
SW1
R1
SW2 Is Capable of Classification and Marking on Layer 3
Strategy:
Classification and Marking IP Precedence and IP DSCP on Ingress at SW2 or IP Precedence and IP DSCP on Ingress at R2
L3 QoS
SW2
R2
SW3 and IP Phone Are Capable of Classification and Marking on Layer 3
Strategy:
- IP Phone Resets CoS, IP Precedence and IP DSCP Value of PC Traffic to 0
- IP Phone Marks CoS, IP Precedence and IP DSCP Value to 5 for Its Voice Traffic
Strategy Continued:
- Classification and Marking IP Precedence and IP DSCP for Data Traffic at SW3 or R2 for Further Classification
L3 QoS
IP
SW3
R3

Figure 3-7 summarizes some of the design options for where to classify and mark, showing the remote site from Figure 3-5.

Figure 3-7 *Classification and Marking Options Applied to a Typical Enterprise Network*

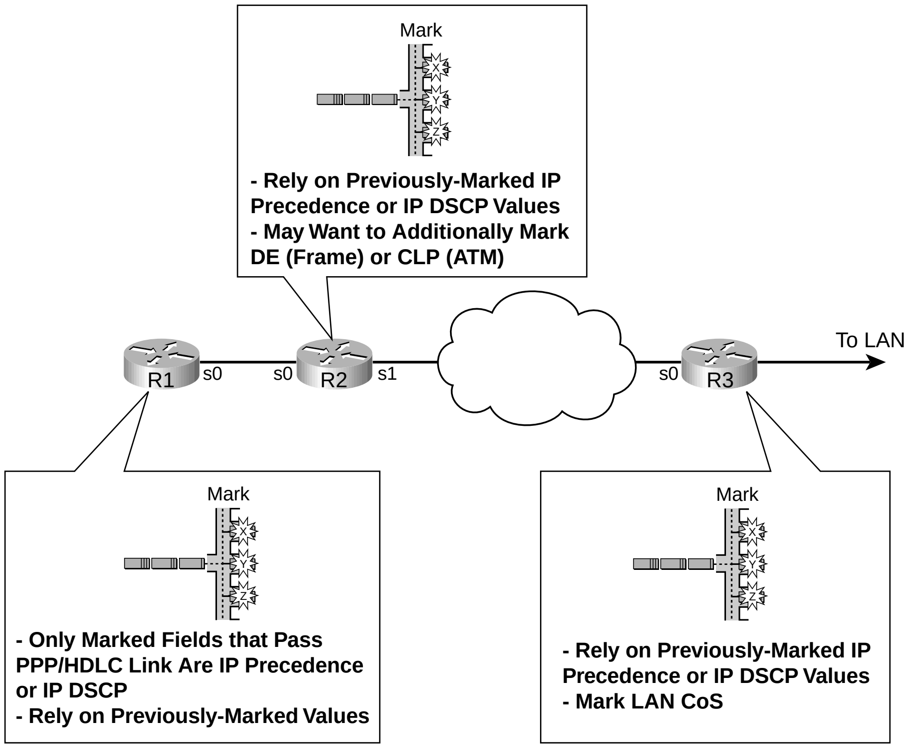

Mark
Voice Bearer Traffic Mark CoS 5, IP Precedence 5 and IP DSCP EF
Voice Signaling Traffic Mark CoS 3, IP Precedence 3 and IP DSCP AF31
Re-Mark PC Traffic CoS 0, IP Precedence 0 and IP DSCP Default
Mark
If SW1 Is Layer 3 Capable: Classification and Marking based on IP Precedence or IP DSCP for Ingress Traffic
If ISL/802.1Q to SW1: Map Incoming CoS Value to IP Precedence or IP DSCP Values
Mark
Hannah's PC Can Mark IP Precedence and IP DSCP, and CoS if Trunking,
Hannah
Jessie
IP
Trust Boundary
SW1
R1
Mark
Layer 3 Switch: Classification and Marking Based on IP Precedence and IP DSCP Values
Layer 2 Switch: Classification and Marking Based on CoS Values
X
Y
Z

The choices of where to perform classification and marking can be summarized as follows:

- Classify and mark as close to the ingress edge as possible.
- Consider the trust boundary in the network, making sure to mark or re-mark traffic after it reaches a trusted device in the network
- Because the two IP QoS marking fields—Precedence and DSCP—are carried end to end, mark one of these fields to maximize the benefits of reducing classification overhead by the other QoS tools enabled in the network.

Typically, when the packet makes it to the first WAN router, the initial marking has occurred. However, there may be other instances where marking should take place. Consider Figure 3-8, which shows several additional options for where marking can occur.

Figure 3-8 *Classification and Marking Options—Typical Enterprise WAN*


Mark
- Rely on Previously-Marked IP Precedence or IP DSCP Values
- May Want to Additionally Mark DE (Frame) or CLP (ATM)
R1
s0
s0
R2
s1
s0
R3
To LAN
Mark
- Only Marked Fields that Pass PPP/HDLC Link Are IP Precedence or IP DSCP
- Rely on Previously-Marked Values
Mark
- Rely on Previously-Marked IP Precedence or IP DSCP Values
- Mark LAN CoS

Most QoS tools can classify based on IP precedence and DSCP. However, the Frame Relay or ATM switches can also react to the DE and CLP bits, respectively. Therefore, you might want to set DE or CLP for the least-important traffic. If the LAN switches connected to R3 react to CoS settings, but not precedence or DSCP, which is typical of switches that only support Layer 2 QoS, you might want to mark the CoS bits on R3 before sending frames onto the Ethernet.

Finally, when you do mark CoS, IP precedence, and IP DSCP, what values should you use? Well, the "bigger is better" attitude is suggested for CoS and precedence, whereas the DiffServ PHB RFCs should be followed for DSCP settings. Cisco also suggests some specific values in cases where your policies allow for voice payload, video payload, voice/video signaling, and two classes of data. Table 3-6 lists these recommended values.

Table 3-6 *Cisco's Recommended Values for Marking*

Type of Traffic	CoS	Precedence	DSCP
Voice payload	5	5	EF
Video payload	4	4	AF41
Voice/Video signaling	3	3	AF31
High-priority or gold data classes*	2	2	AF21 AF22 AF23
Medium-priority or silver data*	1	1	AF11 AF12 AF13
All else	0	0	Default

* Note: The table lists the current recommendations as of early 2003. The DQOS course, and presumably the exam, was created much earlier, when the recommendation for high-priority data was to mark with AF21, with no recommendation for medium-priority data. Keep that in mind when answering exam questions. Also check www.cisco.com and www.ciscopress.com/1587200589 for more information when the exams do change!

In summary, classification and marking tools classify packets based on a large number of different fields inside data link and network layer headers. Based on the classification, the tools then mark a field in a frame or packet header, with the goal that other QoS tools can more easily classify and perform specific QoS actions based on these marked fields. Among all the fields that can be marked, IP Precedence and DSCP, because they are part of the IP header, are the only fields that can be marked and carried from end to end in the network.

Classification and Marking Tools

Three classification and marking tools provide the multifield classifier function of DiffServ—namely, class-Based marking (CB marking), committed access rate (CAR), and policy-based routing (PBR). All three tools enable you to configure matching parameters for a wide variety of fields in a packet header.

Class-Based Marking (CB Marking)

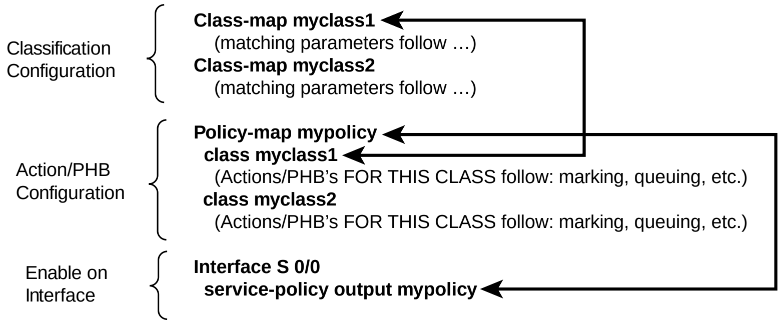
Cisco added CB marking to IOS after all the other classification and marking tools discussed in this book. As of IOS 12.1(5)T and 12.2 mainline, CB marking represents the only classification and marking tool specifically intended for the classification and marking function. NBAR, CAR, PBR, and dial peers have other purposes, whereas CB marking focuses entirely on classification and marking.

You must use a new IOS syntax called the *Modular QoS command-line interface* (MQC) to configure CB marking. After hearing of the term MQC for the first time, many people think that Cisco has created a totally new CLI, different from IOS configuration mode, to configure CB marking. In reality, MQC defines a new set of configuration commands—commands that are typed in using the same IOS CLI, in configuration mode. So, why bother to name these new commands with the term "MQC?" Well, newer IOS QoS tools, as well as future IOS QoS tools, and to some degree older QoS tools, will all use the same MQC commands for QoS configuration in the future. Instead of more than 30 IOS QoS tools, each with different configuration commands, the commands will slowly converge to use the MQC commands. Therefore, MQC is really more of a standard for new and revised configuration commands for QoS features.

All IOS QoS tools that begin with the phrase "class based" use the MQC commands as of IOS 12.2 mainline. These tools include CB marking, CB Weighted Fair Queuing (CBWFQ), CB policing, and CB shaping. Most QoS tools need to perform classification functions; all MQC supporting tools use the same commands for classification. The person configuring the router only needs to learn one set of commands for classification for all four of these tools, which reduces effort and reduces mistakes.

MQC separates the classification, per-hop behavior (PHB), and enabling functions into three separate commands. The **class-map** command defines the matching parameters for classifying packets. Because different tools create different PHBs, the PHB actions (marking, queuing, and so on) are configured under a **policy-map** command. Finally, because these tools operate on packets that either enter or exit an interface, the policy is then enabled on an interface using a **service-policy** command. Figure 3-9 shows the general flow of commands.

Figure 3-9 *MQC Commands and Their Correlation*


Classification Configuration
Class-map myclass1
(matching parameters follow …)
Class-map myclass2
(matching parameters follow …)
Action/PHB Configuration
Policy-map mypolicy
class myclass1
(Actions/PHB's FOR THIS CLASS follow: marking, queuing, etc.)
class myclass2
(Actions/PHB's FOR THIS CLASS follow: marking, queuing, etc.)
Enable on Interface
Interface S 0/0
service-policy output mypolicy

In this example, the network's QoS policy calls for two classes of packets. (The actual types of packets that are placed into each class are not shown, just to keep the focus on the general flow of how the main commands work together.) To classify packets into two classes, two **class-map** commands are used. Each **class-map** would be followed by a **match** subcommand, which defines the actual parameters compared to packet header contents to match packets for classification. For each class, some QoS action needs to be applied—but configuration for these actions is made under the **policy-map** command. Under a single policy map, multiple classes will be referenced—two classes in this example, myclass1 and myclass2. Inside the single policy called mypolicy, under each of the two classes myclass1 and myclass2, you can configure separate QoS actions. For instance, you could apply different marking to packets in class myclass1 and myclass2 at this point. Finally, when the **service-policy** command is applied to an interface, the QoS features are enabled.

MQC provides some good advantages when compared to building each QoS tool with different sets of configuration commands. In many cases, you will use multiple policies in one router, but you need the same classifications. For instance, you might apply slightly different queuing parameters to five different serial links, but because packets have already been marked near the ingress edge of the network, all the classification logic is the same in each case. Therefore, all five policy maps could refer to the same class maps for classification purposes. With multiple tools sharing the same commands, QoS configuration becomes less confusing. Learning how to configure a new MQC QoS tool will be easy as well—for instance, when you know how to configure CB marking, you only need to learn one more command to learn how to configure CBWFQ!

Several specific examples appear over the next several pages. First, Table 3-7 lists the MQC commands used for CB marking. The table shows all the classification options available using the **match** command, and all the marking options available using the **set** command. Table 3-8 lists the **show** commands related to CB marking.

Table 3-7 *Command Reference for CB Marking*

Command	Mode and Function
class-map *class-map-name*	Global config; names a class map, where classification options are configured
Match …	Class-map subcommand; defines specific classification parameters
match access-group {*access-group* | **name** *access-group-name*}	Class-map subcommand; matches an ACL
match source-address mac *address-destination*	Class-map subcommand; matches a source MAC address
match ip precedence *ip-precedence-value* [*ip-precedence-value ip-precedence-value ip-precedence-value*]	Class-map subcommand; Matches an IP precedence value

continues

Table 3-7 *Command Reference for CB Marking (Continued)*

Command	Mode and Function
match mpls experimental *number*	Class-map subcommand; matches an MPLS Experimental value
match cos *cos-value* [*cos-value cos-value cos-value*]	Class-map subcommand; matches a CoS value
match destination-address mac *address*	Class-map subcommand; matches a destination MAC address
match input-interface *interface-name*	Class-map subcommand; matches an input interface
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	Class-map subcommand; matches an IP DSCP value
match ip rtp *starting-port-number port-range*	Class-map subcommand; matches the RTP's UDP port-number range
match qos-group *qos-group-value*	Class-map subcommand; matches a QoS group
match protocol *protocol-name*	Class-map subcommand; matches NBAR protocol types
match protocol citrix app *application-name-string*	Class-map subcommand; matches NBAR Citrix applications
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	Class-map subcommand; matches a host name and URL string
match any	Class-map subcommand; matches all packets
policy-map *policy-map-name*	Global config; names a policy, which is a set of actions to perform
class *class-name*	Policy-map subcommand; identifies which packets on which to perform some action by referring to the classification logic in a class map
set	Class subcommand; for the class, marks (sets) particular QoS fields
set ip precedence *ip-precedence-value*	Class subcommand; set the value for IP precedence
set ip dscp *ip-dscp-value*	Class subcommand; set the value for IP DSCP

Table 3-7 *Command Reference for CB Marking (Continued)*

Command	**Mode and Function**
set cos *cos-value*	Class subcommand; set the value for CoS
set ip qos-group *group-id*	Class subcommand; set the value for the QoS group
set atm-clp	Class subcommand; set the value for the ATM CLP bit
set fr-de	Class subcommand; set the value for the Frame Relay DE bit

Table 3-8 *Exec Command Reference for CB Marking*

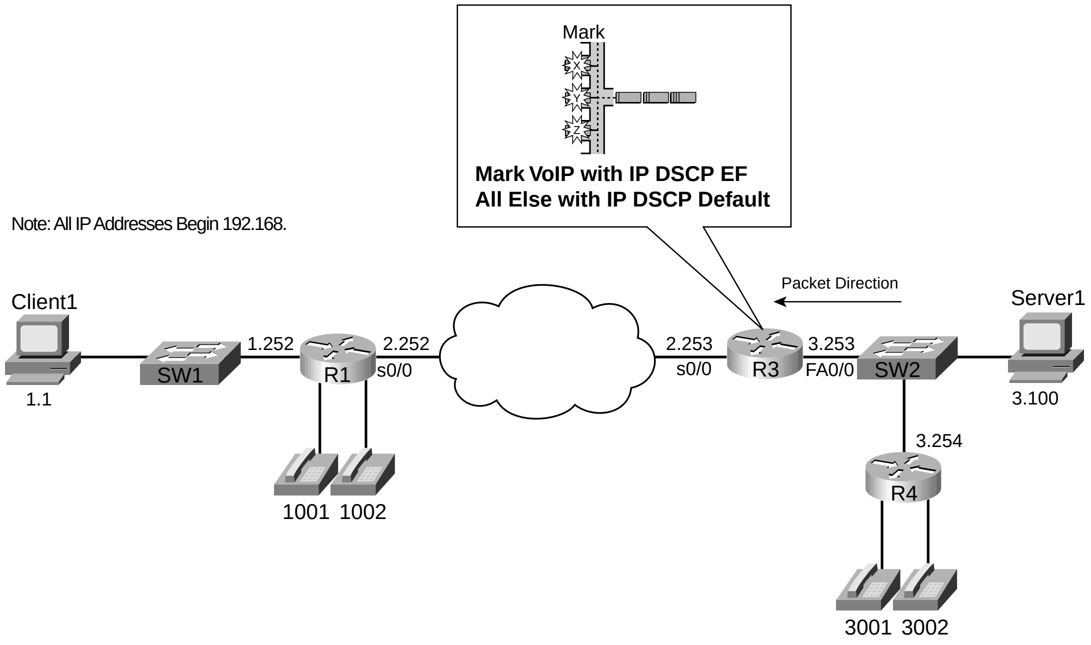
Command	**Function**
show policy-map *policy-map-name*	Lists configuration information about all MQC-based QoS tools
show policy-map *interface-spec* [***input*** | ***output***] [**class** *class-name*]	Lists statistical information about the behavior of all MQC-based QoS tools

QoS configuration should follow the process of planning the QoS policies for the network. After those policies have been defined, and the location of where to perform the marking functions has been determined, however, the CB marking configuration that follows becomes an exercise in deciding how to match or classify the packets, and how to configure the commands correctly. In the first MQC configuration example, for example, the policy has been defined as follows:

- All VoIP traffic should be marked with DSCP EF.
- All other traffic should be marked with DSCP Default.

Figure 3-10 is used for many example configurations in this book. In the first example, marking is performed for packets entering R3's FA0/0 interface. In reality, it also makes sense to mark packets near R1 for packet flows from left to right in the figure. To keep the configurations less cluttered, however, only one direction, right to left, is shown. Example 3-1 lists the configuration for this first example.

Figure 3-10 *CB Marking Sample Configuration 1*



Example 3-1 *CB Marking: Sample 1 Configuration*

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#ip cef
R3(config)#!
R3(config)#class-map voip-rtp
R3(config-cmap)#match ip rtp 16384 16383
R3(config-cmap)#policy-map voip-and-be
R3(config-pmap)# class voip-rtp
R3(config-pmap-c)#  set ip DSCP EF
R3(config-pmap-c)# class class-default
R3(config-pmap-c)#   set ip dscp default
R3(config-pmap-c)#interface e 0/0
R3(config-if)# service-policy input voip-and-be
R3(config-if)#end
R3#
R3#show running-config
```

Example 3-1 *CB Marking: Sample 1 Configuration (Continued)*

```
Building configuration...
!Portions removed to save space…
ip cef
!
class-map match-all voip-rtp
  match ip rtp 16384 16383
!
!
policy-map voip-and-be
  class voip-rtp
   set ip dscp 46
  class class-default
   set ip dscp 0
!
interface Fastethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 service-policy input voip-and-be
```

First, focus on the command prompts in Example 3-1. Note that the **class-map** command moves the CLI into class-map configuration mode, with the prompt R3(config-cmap). The **policy-map** command moves the CLI into policy-map configuration mode, and the **class** command that follows (not **class-map**, but just **class**) moves the CLI into an additional subconfiguration mode that has no specific name.

NOTE I tend to call configuration mode you are in after using the **policy-map** command, and then the **class** command, the *policy-map class* mode when teaching QoS classes.

Next, examine the **match** commands. The solution could have referred to IP ACL 101 with the **match ip access-group 101** command, with ACL 101 matching UDP ports between 16,384 and 32,767, inclusive, to match all VoIP traffic. However, the **match ip rtp** command matches only the even-numbered ports in this same UDP port range. (VoIP payload only uses the even port numbers.) Therefore, the **match ip rtp** command is more efficient for matching VoIP, easier to configure, and only matches the VoIP payload. The other **match** command, **match any**, does exactly that: It matches anything.

Class maps allow multiple **match** commands in a single **class-map** command. You may have noticed the **match-all** parameter on the **class-map** output from **show run**; IOS added the **match-all** parameter, even though it was not typed in. If a **class-map** command has multiple **match** commands, with the default setting of **match-all**, all the **match** commands must match a packet before the packet is considered to be part of the class. The other alternative is to configure the keyword **match-any**, which means that if one or more of the **match** commands in a single class map matches a packet, the packet is part of that class.

Continuing down the configuration, examine the **policy-map set** commands. The first command sets a DSCP of EF for all traffic that matches **class-map voip-rtp**. The other **set** command, which follows the **class class-default** command, sets DSCP of Default for traffic that matches the **class-default class-map**. In other words, the policy map sets DSCP EF for packets that match one class, and DSCP Default, using the keyword **default**, for the other class. IOS includes a class that matches all remaining traffic, called **class-default**, in every **policy-map**. Although the command **class class-default** was not specified in the configuration, note that the **class class-default** command is automatically added to the end of a policy map to match all unspecified traffic. **class-default** was used in the **policy-map** to match all remaining traffic and then mark that traffic as BE.

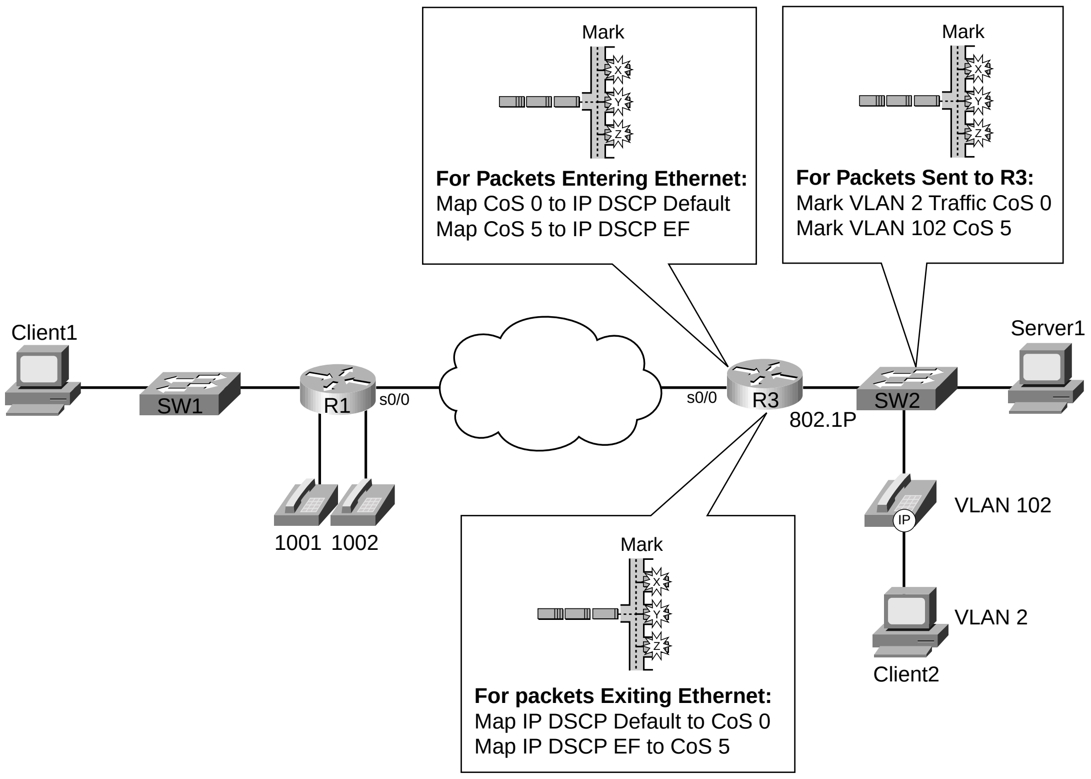
Finally, the **service-policy** command enables CB marking for ingress packets with the **service-policy input voip-and-be** interface subcommand. When enabled, IOS applies the policy map classes in the order they appear in the **policy-map** command. In this example, for instance, the VoIP-RTP class is used to examine the packet first; if a match appears, the packet is marked with DSCP EF. After the packet has been matched and marked, it exits the policy map. If no match occurs, only then is the next class, class-default, used to examine the packet.

Consider two caveats before moving on to more examples. First, examine the output of the **show run** command in Example 3-1, and look for the **set** commands. The MQC enables you to type in the text names of the DSCP values, but IOS records the configuration using the decimal version of the DSCP value! Therefore, you may need a handy reference for the actual DSCP values, as is shown in Table 3-4.

The other caveat only occurs if you already know how to configure Frame Relay traffic shaping (FRTS). FRTS uses a command called **map-class**. Many people who know how to configure **map-class** first look at any MQC-based tool, see the **class-map** command, and don't realize that the commands are indeed two totally different commands. So, ignore all you have learned about FRTS until you learn about MQC configurations using the **class-map** command.

The next example is a CB marking configuration that uses the same network as the one used in Example 3-1. R3 is performing the CB marking function again, but this time R3 expects that SW2 has already set CoS bits to either 0 or 5. The engineers in the meeting to discuss QoS policies for this network decided that SW2 would mark CoS with either 0 or 5, and then R3 would map CoS 0 to DSCP Default, and CoS 5 to DSCP EF. For packets moving left to right, R3 should map DSCP Default back to CoS 0, and DSCP EF back to CoS 5. Figure 3-11 depicts the network and QoS policies, and Example 3-2 lists the configuration.

Figure 3-11 *CB Marking Sample Configuration 2*



Example 3-2 *CB Marking: Sample 2 Configuration*

```
class-map cos0
 match cos 0
!
class-map cos5
 match cos 5
!
class-map BE
 match ip dscp default
!
class-map EF
 match ip dscp EF
!
policy-map map-cos-to-dscp
 class cos0
  set ip DSCP default
```

continues

Example 3-2 *CB Marking: Sample 2 Configuration (Continued)*

```
 class cos5
  set ip DSCP EF
 class class-default
   set ip dscp default
!
policy-map map-dscp-to-cos
 class BE
  set cos 0
 class EF
  set cos 5
 class class-default
   set cos 0
!
interface FastEthernet0/0
!
interface FastEthernet0/0.1
 encapsulation dot1Q 102
 service-policy input map-cos-to-dscp
 service-policy output map-dscp-to-cos
!
interface FastEthernet0/0.2
 encapsulation dot1Q 2 native
```

As you learned earlier in this chapter, to mark and classify CoS values, a VLAN trunking header must exist on the packet. On R3 in this example, subinterface Fast Ethernet 0/0.1 and subinterface Fast Ethernet 0/0.2 have been created and assigned to the voice VLAN 102 and the data VLAN 2, respectively, using 802.1Q trunking. This configuration creates an 802.1Q header for traffic in the voice VLAN 102, without creating a VLAN header for the data VLAN 2 traffic.

The QoS policy required two policy maps in this example. Policy map map-cos-to-dscp matched CoS values for frames entering R3's FA 0/0.1 interface, and marked DSCP values, for packets flowing right to left in the figure. Therefore, the policy map was enabled on input of R3's FA 0/0.1 interface. Policy map map-dscp-to-cos matched DSCP values on packets exiting R3's FA 0/0.1 interface, and marked CoS, for packets flowing left to right in the figure. Therefore, the policy map was enabled on output of R3's FA 0/0.1 interface. Neither policy map could be applied on the WAN interface, because only interfaces configured for 802.1Q accept **service-policy** commands that reference policy maps that either classify or mark based on CoS.

Note that you cannot enable a **policy-map** that refers to CoS on interface FA0/0.2 in this example. That subinterface is in the native VLAN, meaning that no 802.1Q header is used. In a real network, you would probably want to enable a **policy-map** on the subinterface in order to mark traffic, but it must classify based on something beside CoS.

Network-Based Application Recognition (NBAR)

CB marking, and other MQC-based tools, can use NBAR to help classify traffic. By using the **match protocol** class-map subcommand, MQC can match protocols recognized by NBAR. This section describes NBAR, and includes examples of CB marking with NBAR.

NBAR classifies packets that are normally difficult to classify. For instance, some applications use dynamic port numbers, so a statically configured **match** command, looking for a particular UDP or TCP port number, just could not classify the traffic. NBAR can look past the UDP and TCP header, looking at the host name, URL, or MIME type in HTTP requests. NBAR can also look past the TCP and UDP headers to recognize application-specific information. For instance, NBAR allow recognition of different Citrix application types, and allows for searching for a portion of a URL string.

NBAR uses the classification information for two purposes. NBAR, without the help of other IOS features, can classify these difficult-to-classify protocols for the purpose of gathering statistics about the protocols. In fact, NBAR by itself provides classification and statistics, but no marking. NBAR also provides classification help for other QoS tools. Specifically, all MQC tools can refer to NBAR classifications for matching traffic.

The connection between NBAR and CB marking, or any other MQC tool, is through the **match protocol** class-map subcommand. An MQC tool can include the **match protocol** command under a **class-map** command. To do so, NBAR must be enabled on the same interface on which the class map is indirectly enabled through the **service-policy** interface subcommand.

A sample configuration and statistical display may help you make sense of NBAR. Tables 3-9 and 3-10 list the NBAR configuration and exec commands, respectively. Following the tables, Figure 3-12 diagrams the familiar network, where R3 performs CB marking based on NBAR classification of the URL string. Finally, Example 3-3 lists a sample NBAR and CB marking configuration, where CB marking matches a portion of an HTTP URL. The example includes a listing of NBAR statistics gathered on the interface.

Table 3-9 *Configuration Command Reference for NBAR*

Command	Mode and Function
ip nbar protocol-discovery	Interface mode; enables NBAR for traffic entering the interface.
ip nbar port-map *protocol-name* [**tcp** | **udp**] *port-number*	Global; tells NBAR to search for a protocol using a different port number than the well-known port. Also defines ports to be used by custom packet description language modules (PDLMs).
ip nbar pdlm *pdlm-name*	Global; extends the list of protocols recognized by NBAR by adding additional PDLMs.

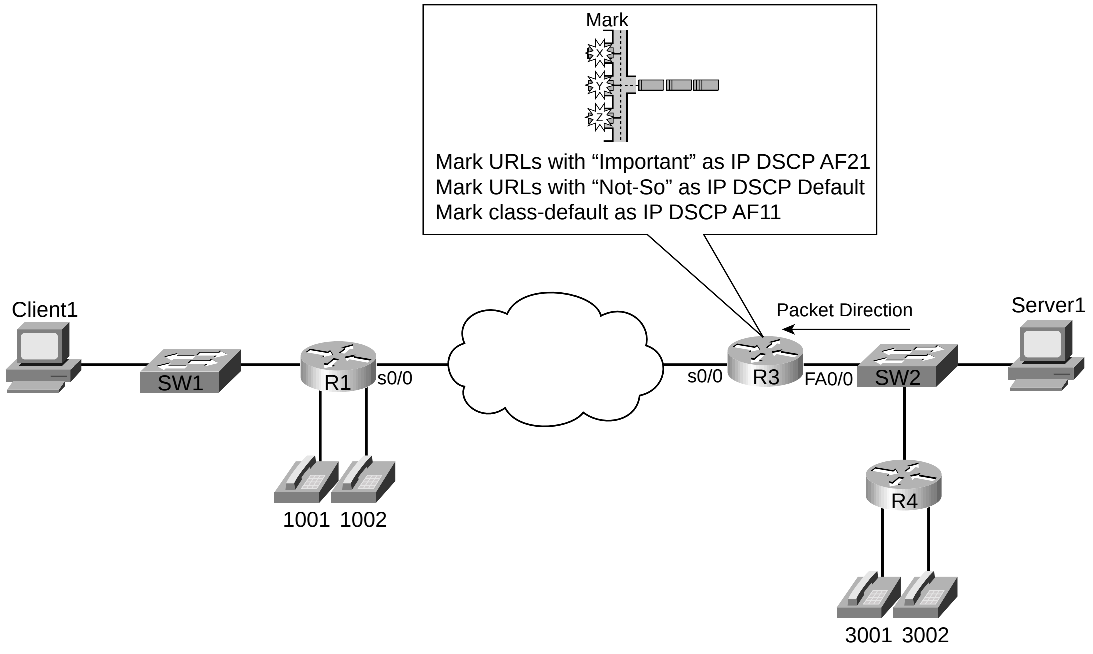
NOTE You can download additional PDLMs from Cisco.com:

www.cisco.com/cgi-bin/tablebuild.pl/pdlm

Table 3-10 *Exec Command Reference for NBAR*

Command	Function
show ip nbar protocol-discovery [**interface** *interface-spec*] [**stats** {**byte-count** | **bit-rate** | **packet-count**}][{**protocol** *protocol-name* | **top-n** *number*}]	Lists information about statistics for the discovered protocols. Statistics can be listed by interface, by protocol, or for just the top *n* protocols by volume.
show ip nbar port-map [*protocol-name*]	Lists the current ports in use by the discovered protocols.

Figure 3-12 *CB Marking Sample Configuration 3*


Mark
X
Y
Z
Mark URLs with "Important" as IP DSCP AF21
Mark URLs with "Not-So" as IP DSCP Default
Mark class-default as IP DSCP AF11
Client1
SW1
R1
s0/0
1001 1002
s0/0
R3
FA0/0
Packet Direction
SW2
Server1
R4
3001 3002

Example 3-3 uses the following criteria for marking packets:

- Any HTTP traffic whose URL contains the string "important" anywhere in the URL is marked with AF21.
- Any HTTP traffic whose URL contains the string "not-so" anywhere in the URL is marked with DSCP default.
- All other traffic is marked with AF11.

Example 3-3 shows the configuration.

Example 3-3 *Sample 3: CB Marking Based on URLs, Using NBAR for Classification*

```
ip cef
!
class-map http-impo
 match protocol http url "*important*"
!
class-map http-not
 match protocol http url "*not-so*"
!
class-map all-else
 match any
!
policy-map http
 class http-impo
  set ip dscp AF21
!
 class http-not
  set ip dscp default
!
 class class-default
  set ip DSCP AF11
!
interface fastethernet 0/0
 ip nbar protocol-discovery
 service-policy input http
!
!

R3# show ip nbar protocol-discovery top-n 5

 FastEthernet0/0
                           Input                    Output
   Protocol                Packet Count             Packet Count
                           Byte Count               Byte Count
                           5 minute bit rate (bps)  5 minute bit rate (bps)
   ----------------------- ------------------------ ------------------------
   eigrp                   76                       0
                           5624                     0
                           0                        0
   bgp                     0                        0
                           0                        0
                           0                        0
   citrix                  0                        0
                           0                        0
                           0                        0
   cuseeme                 0                        0
                           0                        0
                           0                        0
```

continues

Example 3-3 *Sample 3: CB Marking Based on URLs, Using NBAR for Classification (Continued)*

```
   custom-01                0                        0
                            0                        0
                            0                        0
   unknown                  5610                     0
                            5665471                  0
                            135000                   0
   Total                    5851                     0
                            5845277                  0
                            135000                   0

R3#show ip nbar protocol-discovery interface fastethernet 0/0 stats packet-count top-n 5

 FastEthernet0/0
                            Input                    Output
   Protocol                 Packet Count             Packet Count
   ------------------------ ------------------------ ------------------------
   http                     721                      428
   eigrp                    635                      0
   netbios                  199                      0
   icmp                     1                        1
   bgp                      0                        0
   unknown                  46058                    63
   Total                    47614                    492
```

Notice that the class map configuration does not specifically use the term NBAR. Two class maps, http-impo and http-not, use the **match** command, with the **protocol** keyword, which implies that the actual classification uses NBAR. NBAR has been enabled on FA0/0 with the **ip nbar protocol discovery** command—had NBAR not been enabled, the **service-policy** command would have been rejected. Also note that CEF forwarding must be enabled, using the **ip cef** global command, before NBAR will work.

NBAR can match URLs exactly, or with some wildcards. You can use the asterisk (*) to match any characters of any length. In this case, as long as the phrases "important" or "not-so" appear in the URL, the packets are matched by one of the two class maps, respectively. Interestingly, when downloading an object with HTTP, the URL does not flow in every packet. *When classifying based on URL, NBAR matches all packets beginning with the matched URL, and then until another HTTP request for another URL flows inside the same TCP connection.*

The **show ip nbar protocol-discovery** command lists statistics for NBAR-classified packets. However, just using that command in live networks does not help much, because it lists three lines of output per type of protocol that can be discovered by NBAR—not just the protocols NBAR actually discovered. Therefore, the optional parameters on the command are more useful. For instance, both commands shown in the preceding example use the **top-n** parameter to limit the output based on the highest-volume protocols. The **show** command can also limit the statistics for a single interface, or it can limit the statistics to just packet count, or byte count, or bit rate.

Unlike most other IOS features, you can upgrade NBAR without changing to a later IOS version. Cisco uses a feature called packet descriptor language modules (PDLMs) to define new protocols that NBAR should match. When Cisco decides to add one or more new protocols to the list of protocols that NBAR should recognize, it creates and compiles a PDLM. You can then download the PDLM from Cisco, copy it into Flash memory, and add the **ip nbar pdlm** *pdlm-name* command to the configuration, where *pdlm-name* is the name of the PDLM file in Flash memory. NBAR can then classify based on the protocol information from the new PDLM.

CB Marking **show** Commands

CB marking provides only one **show** command that provides statistical information: **show policy-map interface**. The statistics do provide some good insight to the packet volumes being marked by CB marking. The next sample configuration includes a new configuration and several variations of the **show policy-map** command.

The same network is used for the next example as was used in the other CB marking examples, but with different marking criteria. In this case, traffic is generated so that the **show** command output is more meaningful. The following traffic is generated:

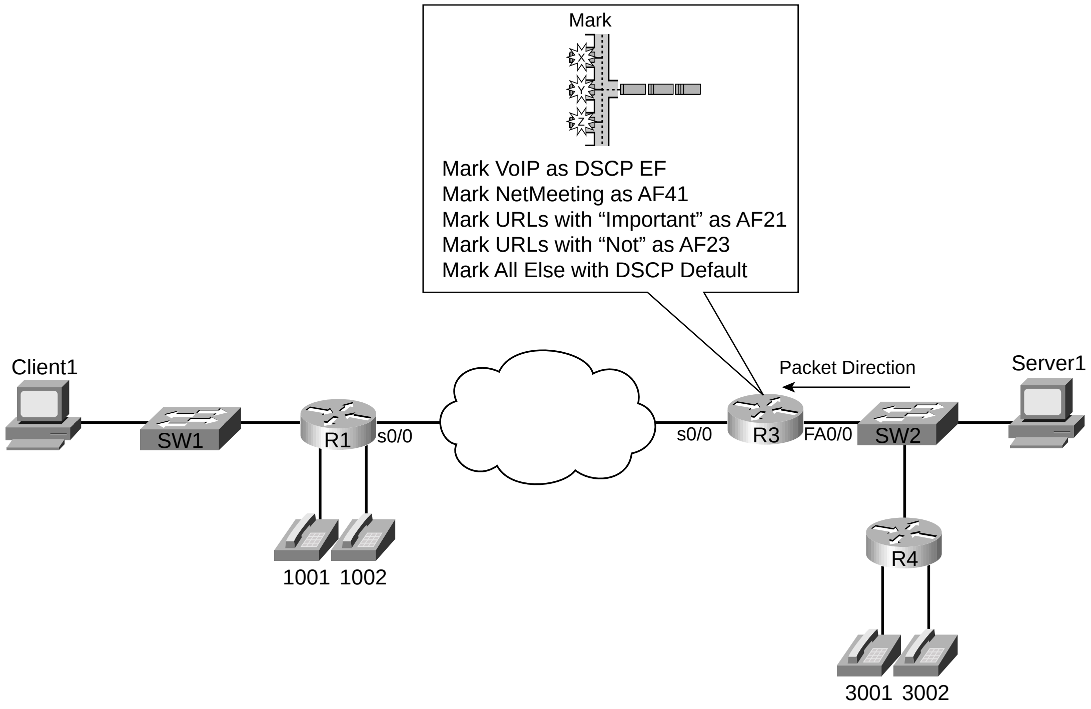
- Two G.711 VoIP calls between R4 and R1 using Foreign Exchange Station (FXS) cards on these two routers. Voice Activation Detection (VAD) is disabled.
- One FTP connection from the client PC to the server, with an FTP get of a 40-MB file called big.zip.
- One Microsoft NetMeeting video/audio conference between the client and server.
- One web page download from the server to the client. The web page has a few small objects. The web page includes two panes, each with a different JPG file: one called important.jpg; the other called not-so.jpg. The JPGs are exact copies of each other, and each JPG is 687 KB. In later examples, the differing performance of the download of these examples is used to demonstrate the behavior of other QoS tools.

Figure 3-13 depicts the same familiar network, and lists the criteria in with the figure for easy reference.

The new criteria for Example 3-4 is as follows:

- VoIP payload is marked with DSCP EF.
- NetMeeting voice and video from Server 1 to Client 1 is marked with DSCP AF41.
- Any HTTP traffic whose URL contains the string “important” anywhere in the URL is marked with AF21.
- Any HTTP traffic whose URL contains the string “not-so” anywhere in the URL is marked with AF23.
- All other traffic is marked with DSCP Default.

Figure 3-13 *Three Classification and Marking Placement Strategies*



Example 3-4 shows the configuration, including the appropriate **show** commands.

Example 3-4 *CB Marking Sample 4, with* **show** *Command output*

```
ip cef
!
interface fastethernet 0/0
 ip nbar protocol-discovery
!
access-list 101 permit udp host 192.168.3.101 gt 16383 192.168.1.0 0.0.0.255 gt 16383
!
class-map voip-rtp
 match ip rtp 16384 16383
!
class-map http-impo
 match protocol http url "*important*"
!
class-map http-not
 match protocol http url "*not-so*"
!
```

Example 3-4 *CB Marking Sample 4, with* **show** *Command output (Continued)*

```
class-map NetMeet
 match access-group 101
!
policy-map laundry-list
!
 class voip-rtp
  set ip dscp EF
!
 class NetMeet
  set ip dscp AF41
!
class http-impo
  set ip dscp AF21
!
 class http-not
  set ip dscp AF23
!
 class class-default
  set ip DSCP default
!
 interface Fastethernet 0/0
 service-policy input laundry-list
end

R3#show policy-map
  Policy Map laundry-list
    Class voip-rtp
      set ip dscp 46
    Class NetMeet
      set ip dscp 34
    Class http-impo
      set ip dscp 18
    Class http-not
      set ip dscp 22
    Class class-default
      set ip dscp 0

R3#show policy-map laundry-list
  Policy Map laundry-list
    Class voip-rtp
      set ip dscp 46
    Class NetMeet
      set ip dscp 34
    Class http-impo
      set ip dscp 18
    Class http-not
      set ip dscp 22
```

continues

Example 3-4 *CB Marking Sample 4, with* **show** *Command output (Continued)*

```
    Class class-default
      set ip dscp 0
R3#show policy-map interface fastethernet 0/0 input
 Fastethernet0/0

  Service-policy input: laundry-list

    Class-map: voip-rtp (match-all)
      35268 packets, 2609832 bytes
      5 minute offered rate 59000 bps, drop rate 0 bps
      Match: ip rtp 16384 16383
      QoS Set
        ip dscp 46
          Packets marked 35268

    Class-map: NetMeet (match-all)
      817 packets, 328768 bytes
      5 minute offered rate 19000 bps, drop rate 0 bps
      Match: access-group 101
      QoS Set
        ip dscp 34
          Packets marked 817

    Class-map: http-impo (match-all)
      2843 packets, 3462611 bytes
      5 minute offered rate 56000 bps, drop rate 0 bps
      Match: protocol http url "*important*"
      QoS Set
        ip dscp 18
          Packets marked 2855

    Class-map: http-not (match-all)
      2828 packets, 3445409 bytes
      5 minute offered rate 56000 bps, drop rate 0 bps
      Match: protocol http url "*not-so*"
      QoS Set
        ip dscp 22
          Packets marked 2842

    Class-map: class-default (match-all)
      33216 packets, 43649458 bytes
      5 minute offered rate 747000 bps, drop rate 0 bps
      Match: any
      QoS Set
        ip dscp 0
          Packets marked 33301
```

Review the configuration before taking a closer look at the **show** commands. The only part of the configuration that was not covered in the first three examples on CB marking is the matching of the Microsoft NetMeeting traffic. NetMeeting uses RTP for the audio and video flows. ACL

101 matches all UDP port numbers over 16,384, for traffic from Server 1 going to the client. This may catch other traffic besides NetMeeting, but it definitely catches all the NetMeeting traffic. Also note that the NetMeet class map uses a combination of capital letters and lowercase letters, as does the **class** command that refers to it. Class map names are case sensitive—you may want to choose to use only uppercase letters for names to avoid confusion.

The **show policy-map laundry-list** command just lists a summary of the configuration. You can gather the same information with a **show running-config** command, but it is summarized nicely with **show policy-map**. The **show policy-map** command lists the same configuration information, but it lists the information for all the configured policy maps in this router.

The **show policy-map** command using the **interface** option provides statistical information about the number of packets and bytes that have matched each class inside the policy maps. Because CB marking is configured, it also notes the number of packets that have been marked. You can select all interfaces, just one interface, either input or output, and even select a single class inside a single policy map for display.

Finally, the **load-interval** interface subcommand can also be useful when looking at any QoS tool's statistics. The **load-interval** command defines the time interval over which IOS measures packet and bit rates on an interface. With a lower load interval, the statistics change more quickly; with a larger load interval, the statistics change more slowly. In a lab when you are just learning to use QoS tools, set the load interval to the minimum of 30 seconds, so you can see the results of new traffic, or changes to the configuration, quickly. (The default setting is 5 minutes.)

CB Marking Summary

Class-based marking provides the most functional general classification and marking tool in IOS, as of IOS 12.2 mainline. Class-based marking provides the largest number of fields for classifying packets, and the largest number of fields that can be marked. It uses MQC for configuration, separating the classification details from the QoS action.

Refer to Table 3-17, in the "Foundation Summary" section of this chapter, for a complete list of classification and marking fields used by CB marking.

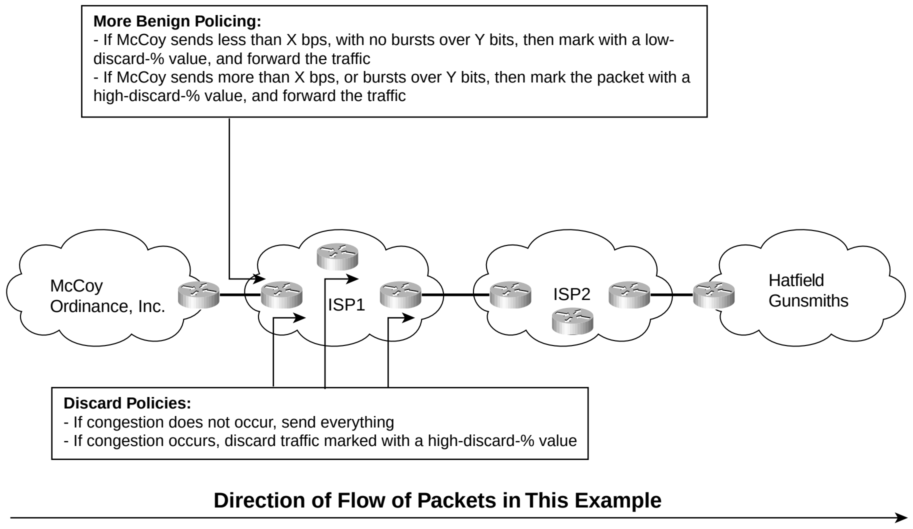
Committed Access Rate (CAR)

CAR provides policing functions and marking. Chapter 5, "Traffic Policing and Shaping," covers the policing details of CAR and CB policing. However, a quick review of policing before getting into CAR's marking features will help you appreciate why CAR includes marking.

Policing, in its most basic form, discards traffic that exceeds a particular traffic contract. The contract has two components: a rate, stated either in bits per second or bytes per second; and a burst size, stated in either bits or bytes. The traffic conforms to the contract if it sends at the rate, or below, and it does not send a burst of traffic greater than the burst size. If the traffic exceeds the traffic rate over time, or exceeds the single burst size limit, the policing function drops the traffic in excess of the rate and the burst size. Therefore, the simplest form of policing has two rigid actions: either to forward packets or to drop them.

CAR's marking function allows for additional policing action besides just forwarding or dropping a packet. Consider a typical case where policing is used, as in Figure 3-14. ISP1 needs to police traffic to protect customers who conform to their contracts from congestion created by customers who do not conform. If the network is not congested, however, it might be nice to go ahead and forward the nonconforming customer traffic. Doing so doesn't really cost the ISP anything, so long as the network is not congested. If the network is congested, however, ISP1 wants to discard the traffic that exceeds the contract before discarding traffic that is within its respective contract.

Figure 3-14 *Policing: Excess Traffic Marked with Higher Discard Value*


More Benign Policing:
- If McCoy sends less than X bps, with no bursts over Y bits, then mark with a low-discard-% value, and forward the traffic
- If McCoy sends more than X bps, or bursts over Y bits, then mark the packet with a high-discard-% value, and forward the traffic
McCoy Ordinance, Inc.
ISP1
ISP2
Hatfield Gunsmiths
Discard Policies:
- If congestion does not occur, send everything
- If congestion occurs, discard traffic marked with a high-discard-% value
Direction of Flow of Packets in This Example

For instance, the conforming traffic can be marked with DSCP AF41, and the nonconforming traffic with DSCP Default. The congestion-avoidance QoS tools in ISP1 can be configured to aggressively discard all DSCP Default traffic at the first signs of congestion. So, when ISP1 experiences congestion, policing indirectly causes the excess traffic to be discarded; in periods of no congestion, ISP1 provides service beyond what the customer has paid for.

You can also use CAR to just mark the traffic. CAR classifies traffic based on a large number of fields in the packet header, including anything that can be matched with an IP ACL. Once matched, CAR can be configured to do one action for conforming traffic, and another for excess traffic. If the two actions (conform and exceed actions) are the same action, in effect, CAR has not policed, but rather has just marked packets in the same way.

CAR configuration includes the classification, marking, and enabling features all in a single configuration command: the **rate-limit** interface subcommand. (CB marking, you may recall, separates classification, marking, and enabling on an interface into three separate commands.) Tables 3-11, 3-12, and 3-13 list the pertinent CAR configuration and exec commands, respectively.

Table 3-11 *Configuration Command Reference for CAR*

Command	Mode and Function
rate-limit {**input** | **output**} [**access-group** [**rate-limit**] *acl-index*] *bps burst-normal burst-max* **conform-action** *conform-action* **exceed-action** *exceed-action*	Interface mode; configures classification, marking, policing, and enabling CAR on the interface
access-list rate-limit *acl-index* {*precedence* | *mac-address* | *exp* **mask** *mask*}	Global mode; creates a CAR ACL, which can match IP precedence, MAC addresses, and MPLS Experimental bits

Table 3-12 *Possible Actions with CAR* **rate-limit** *Command*

rate-limit Conform and Exceed Options	Function
Continue	Evaluates the next **rate-limit** command
Drop	Drops the packet
set-dscp-continue	Sets the differentiated services code point (DSCP) (0–63) and evaluates the next **rate-limit** command
set-dscp-transmit	Sets the DSCP and transmits the packet
set-mpls-exp-continue	Sets the MPLS Experimental bits (0–7) and evaluates the next **rate-limit** command
set-mpls-exp-transmit	Sets the MPLS Experimental bits (0–7) and sends the packet
set-prec-continue	Sets the IP precedence (0–7) and evaluates the next **rate-limit** command
set-prec-transmit	Sets the IP precedence (0–7) and sends the packet
set-qos-continue	Sets the QoS group ID (1–99) and evaluates the next **rate-limit** command
set-qos-transmit	Sets the QoS group ID (1–99) and sends the packet
Transmit	Sends the packet

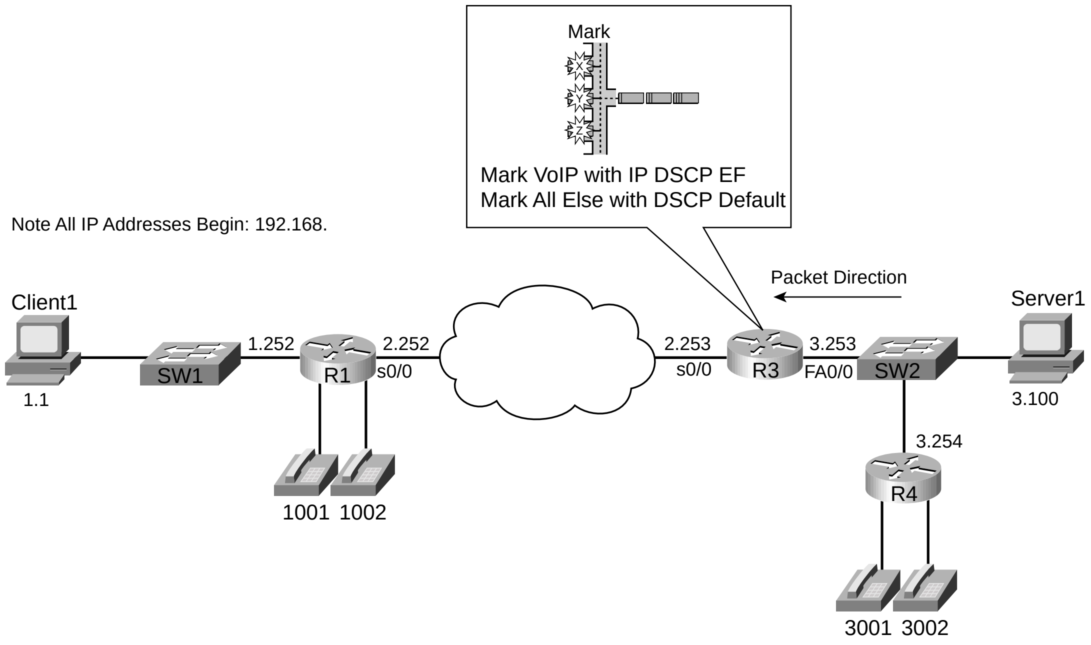
Table 3-13 *Exec Command Reference for CAR*

Command	Function
show interfaces [*interface-type interface-number*] **rate-limit**	Displays CAR statistics on the interface specified, or on all interfaces if the interface is not specified
show access-lists rate-limit [*acl-index*]	Lists information about the configuration of rate-limit ACLs

The first CAR marking example, shown in Example 3-5, uses the following criteria for marking packets. In this example, R3 is marking packets that flow right to left in Figure 3-15. (This example's criteria matches that in Example 3-1 for CB marking.)

- All VoIP payload traffic is marked with DSCP EF.
- All other traffic is marked with DSCP Default.

Figure 3-15 *CAR Marking Sample 1: VoIP Marked with DSCP EF, Everything Else Marked BE*



Example 3-5 *CAR Marking, VoIP as DSCP EF, Everything Else as BE*

```
no ip cef
!
access-list 102 permit udp any range 16384 32768 any range 16384 32768
!
interface fastethernet 0/0
 rate-limit input access-group 102 10000 20000 30000 conform-action
   set-dscp-transmit 46 exceed-action set-dscp-transmit 46
 rate-limit input 10000 20000 30000 conform-action set-dscp-transmit 0
   exceed-action set-dscp-transmit 0
end
```

The configuration does not take nearly as many different commands as the CB marking example, because most of the interesting parameters are contained in the **rate-limit** commands. Cisco Express Forwarding (CEF) is disabled, just to make the point that although you can use CEF with CAR, it is not required. ACL 102 defines some classification parameters that CAR will use to match VoIP packets, looking at UDP ports between 16,384 and 32,767. The ACL logic matches all VoIP payload, but it will also match VoIP Real Time Control Protocol (RTCP) traffic, which uses the odd-numbered UDP ports in the same port range. Finally, two **rate-limit** commands under FA0/0 enable CAR, define policing limits, classification details, and marking details.

The first of the two **rate-limit** commands matches a subset of all traffic using classification, whereas the second **rate-limit** command just matches all traffic. CAR uses the information configured in these two commands sequentially; in other words, if a packet matches the first CAR statement's classification details, the statement is matched, and its actions are followed. If not, CAR compares the next statement, and so on. In this example, the first CAR **rate-limit** command matches VoIP packets by referring to ACL 102, and the second statement, because it does not refer to an ACL, matches all packets.

NOTE CAR can actually match multiple statements on the same interface. Some CAR actions include the keyword **continue**, which means that even after the statement is matched, CAR should keep searching the statements for further matches. This allows CAR to nest statements, to perform features such as "police all traffic at 500 kbps, but police subsets at 250 kbps, 200 kbps, and 150 kbps."

Now examine the first **rate-limit** command, **rate-limit input access-group 102 10000 20000 30000 conform-action set-dscp-transmit 46 exceed-action set-dscp-transmit 46**, in detail. The **input** keyword means that CAR examines traffic entering the interface. The **access-group**

102 command means that packets permitted by ACL 102 are considered to match this **rate-limit** command. The next three values represent the committed rate, the burst size, and the excess size, which make up the traffic contract. The **conform-action** keyword identifies that the next parameter defines the action applied to conforming traffic, and the **exceed-action** keyword identifies that the next parameter defines the action applied to traffic that exceeds the traffic contract. In this example, both the conform and exceed actions are identical: **set-dscp-transmit 46**, which marks the DSCP value to decimal 46, or DSCP EF. (The **rate-limit** command does not allow the use of DSCP names.)

In this example, the actual traffic contract does not matter, because the actions for conforming traffic and excess traffic are the same. The true goal of this example is just to use CAR to mark packets VoIP—not to actually police the traffic. Chapter 5 includes CAR examples with different conform and exceed actions. The three values represent the committed rate (bps), the committed burst size (bytes), and the committed burst plus the excess burst (bytes). The excess burst parameter essentially provides a larger burst during the first measurement interval after a period of inactivity. (Chapter 5 covers the details of these settings.)

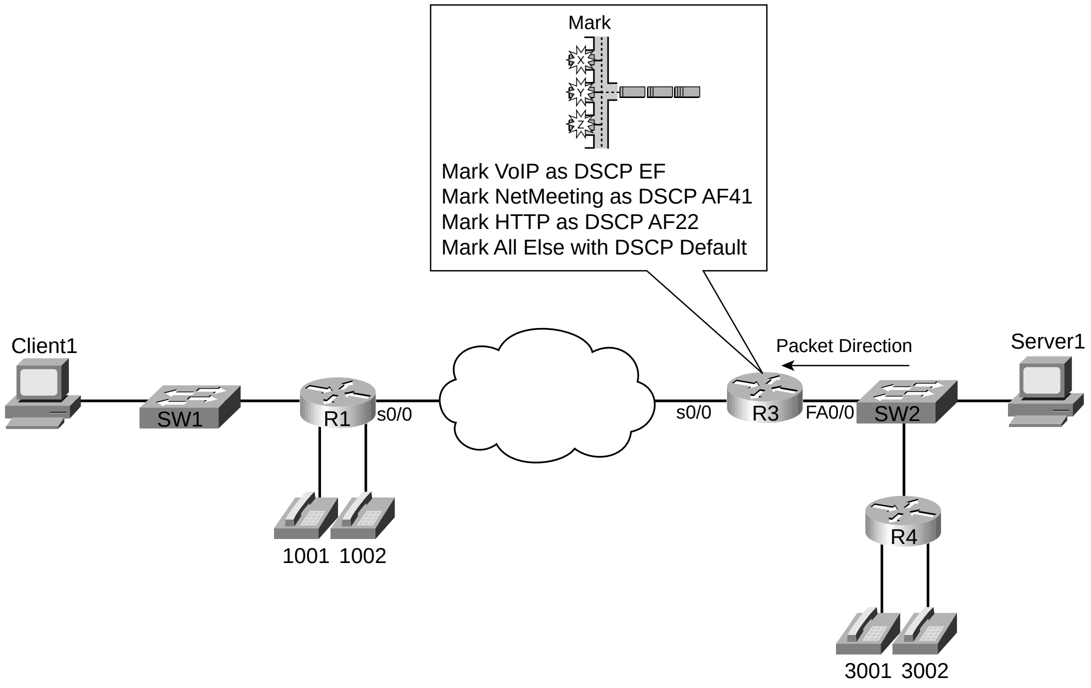
The second **rate-limit** command, **rate-limit input 10000 20000 30000 conform-action set-dscp-transmit 0 exceed-action set-dscp-transmit 0**, matches all remaining traffic. The only way that CAR can classify packets is to refer to an IP ACL, or a CAR rate-limit ACL, from the **rate-limit** command. The second **rate-limit** command does not refer to an ACL with the **access-group** keyword, so by implication, the statement matches all packets. Both actions set the DSCP value to zero. Essentially, this example uses CAR to mark traffic with either DSCP 46 or 0 (decimal), without discarding any packets due to policing.

The second sample CAR configuration, Example 3-6, includes classification options similar to CB marking Example 3-4. Because CAR cannot take advantage of NBAR, CAR cannot look at the URL for HTTP requests, as the CB marking example did. The slightly modified criteria for CAR marking in Example 3-6 is as follows:

- VoIP payload is marked with DSCP EF.
- NetMeeting voice and video from Server1 to Client1 is marked with DSCP AF41.
- Any HTTP traffic is marked with AF22.
- All other traffic is marked with DSCP Default.

Figure 3-16 shows the network in which the configuration is applied, and Example 3-6 shows the configuration.

Figure 3-16 *CAR Marking Sample 2 Network*



Example 3-6 ***CAR Marking Sample 2:*** *VoIP, NetMeeting Audio/Video, HTTP URLs, and Everything Else*

```
no ip cef
!
access-list 110 permit udp any range 16384 32768 any range 16384 32768
!
access-list 111 permit udp host 192.168.1.100 gt 16383 192.168.3.0 0.0.0.255 gt 16383
!
access-list 112 permit tcp any eq www any
access-list 112 permit tcp any any eq www
!
!
interface fastethernet 0/0
rate-limit input access-group 111 8000 20000 30000 conform-action
  set-dscp-transmit 34 exceed-action set-dscp-transmit 34
rate-limit input access-group 110 8000 20000 30000 conform-action
  set-dscp-transmit 46 exceed-action set-dscp-transmit 46
rate-limit input access-group 112 8000 20000 30000 conform-action
  set-dscp-transmit 20 exceed-action set-dscp-transmit 20
```

continues

Example 3-6 ***CAR Marking Sample 2:*** *VoIP, NetMeeting Audio/Video, HTTP URLs, and Everything Else (Continued)*

```
rate-limit input 8000 20000 30000 conform-action set-dscp-transmit 0
  exceed-action set-dscp-transmit 0

end
R3#show interface fastethernet 0/0 rate-limit
Fastethernet0/0 connected to SW2, where Server1 is connected
  Input
    matches: access-group 111
      params:  8000 bps, 20000 limit, 30000 extended limit
      conformed 1346 packets, 341169 bytes; action: set-dscp-transmit 34
      exceeded 2683 packets, 582251 bytes; action: set-dscp-transmit 34
      last packet: 56ms ago, current burst: 29952 bytes
      last cleared 00:07:11 ago, conformed 6000 bps, exceeded 10000 bps
    matches: access-group 110
      params:  8000 bps, 20000 limit, 30000 extended limit
      conformed 6118 packets, 452856 bytes; action: set-dscp-transmit 46
      exceeded 34223 packets, 2552218 bytes; action: set-dscp-transmit 46
      last packet: 12ms ago, current burst: 29989 bytes
      last cleared 00:07:11 ago, conformed 8000 bps, exceeded 47000 bps
    matches: access-group 112
      params:  8000 bps, 20000 limit, 30000 extended limit
      conformed 677 packets, 169168 bytes; action: set-dscp-transmit 20
      exceeded 3631 packets, 5084258 bytes; action: set-dscp-transmit 20
      last packet: 8ms ago, current burst: 29638 bytes
      last cleared 00:07:12 ago, conformed 3000 bps, exceeded 94000 bps
    matches: all traffic
      params:  8000 bps, 20000 limit, 30000 extended limit
      conformed 671 packets, 279572 bytes; action: set-dscp-transmit 0
```

The **show interface Fastethernet 0/0 rate-limit** command lists the pertinent statistical information about CAR's performance. The output has one stanza correlating to each **rate-limit** command on the interface, as highlighted in the example. Under each stanza, the number of packets and bytes that conformed, and the number of packets and bytes that exceeded the traffic contract, are listed. Because this CAR configuration was intended only for marking traffic, the number of packets and bytes in each category does not matter; Chapter 5 takes a closer look at the two values. For comparison purposes, however, consider the bps rates of the combined conformed and exceeded values. For instance, the second **rate-limit** command referenced ACL 110, which matched the two VoIP calls between R1 and R4. These two values total 55 kbps, which is the amount of traffic expected from a pair of G.729a calls over an Ethernet network.

CAR Marking Summary

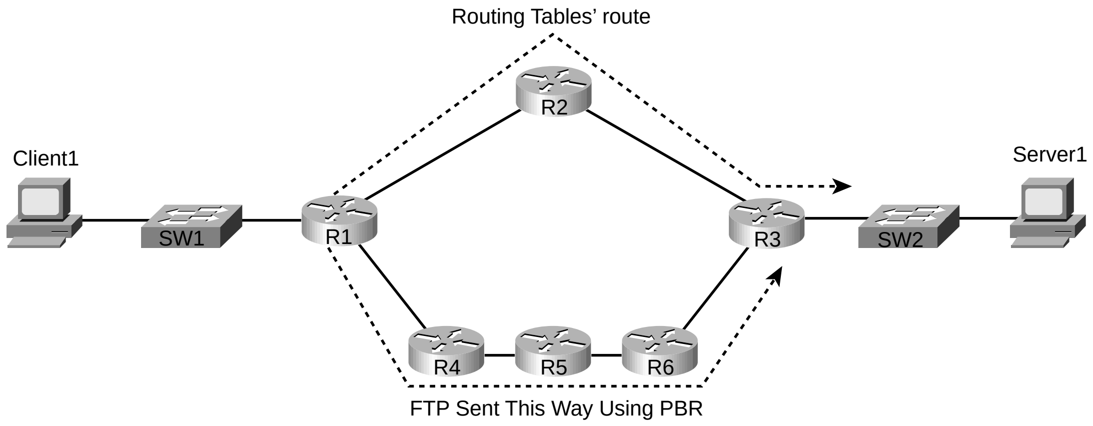
CAR is another tool that examines packet header information to classify and mark packets. CAR provides fewer options for classification and marking than does CB marking, but CAR is considered to be DiffServ compliant because it can classify DSCP using an ACL and mark the DSCP field directly. CAR, along with CB marking and PBR, makes classification decisions based on the contents of packet headers and marks QoS fields based on those classifications. Dial peers provide very different classification options, so fewer direct comparisons can be drawn.

Refer to Table 3-17 for a complete list of classification and marking fields used by CAR.

Policy-Based Routing (PBR)

PBR enables you to route a packet based on other information, in addition to the destination IP address. In most cases, engineers are happy with the choices of routes made by the routing protocol, with routing occurring based on the destination IP address in each packet. For some specialized cases, however, an engineer may want some packets to take a different path. One path through the network may be more secure, for instance, so some packets could be directed through a longer, but more secure, path. Some packets that can tolerate high latency may be routed through a path that uses satellite links, saving bandwidth on the lower-latency terrestrial circuits for delay-sensitive traffic. Regardless of the reasons, PBR can classify packets and choose a different route. Figure 3-17 shows a simple example, where FTP traffic is directed over the longer path in the network.

Figure 3-17 *PBR: FTP Traffic Routed over Longer Path*



PBR supports packet marking and policy routing. As you learned in previous sections, CAR supports marking because CAR's main feature, policing, benefits from having the marking feature available as well. Similarly, PBR includes a marking feature, because in some cases, PBR is used to pick a different route for QoS reasons—for instance, to affect the latency of a packet. So, PBR's core function can benefit from marking a packet, so that the appropriate QoS action can be taken as the packet traverses the network. Just as with CAR, you can use PBR's marking feature without actually using its core feature. In other words, you can use PBR just for classification and marking, without choosing a different route. The examples in this chapter focus only on PBR as a marking tool.

Unlike CB marking and CAR, PBR only processes packets entering an interface; you cannot enable it for packets exiting an interface. The reason PBR only processes incoming packets relates to its core function: policy routing. PBR needs to process packets before a routing decision has been made. Therefore, PBR processes packets entering an interface, preempting the normal routing logic based on destination IP address.

Finally, one other difference between PBR and the other classification and marking tools covered so far (CB marking and CAR) is that PBR can classify based on routing information, instead of totally relying on information in the frame or packet header. PBR can look up the entry in the routing table that matches a packet's destination address, for instance, and then classify based on information about that route. For example, the metric associated with that route, the source of the routing information, or the next-hop interface associated with the route can be checked. In most cases, this routing information does not help you with differentiating between different types of traffic. An FTP server, an IP Phone, a video server, and some web servers may all be in the same subnet, for instance, but the routing information about that subnet could not help PBR distinguish between those different types of traffic. Therefore, typically the most useful classification feature of PBR, when used for marking, is just to refer to an IP ACL.

PBR configuration uses yet another totally different set of configuration commands as compared to CB marking and CAR. PBR does separate the classification, marking, and enabling features into different commands. Tables 3-14 and 3-15 list the pertinent PBR configuration and exec commands, respectively. Following the tables, two example PBR configurations are shown. The two examples use the same criteria as the two CAR samples.

Table 3-14 *Configuration Command Reference for PBR*

Command	Mode and Function
ip local policy route-map *map-tag*	Global; specifies that packets generated by this router should be candidates for policy routing
ip policy route-map *map-tag*	Interface subcommand; refers to a route map, which in turn classifies packets and specifies actions; actions include specifying a different route, and setting IP precedence
route-map *map-tag* **[permit | deny] [***sequence-number***]**	Global command; creates a route map entry
match ip address {*access-list-number* | *access-list-name***} [**... *access-list-number* | ... *access-list-name***]**	Route-map subcommand; used to match IP packets based on parameters that can be matched with an IP ACL
match length *minimum-length maximum-length*	Route-map subcommand; used to match IP packets based on their length

Table 3-14 *Configuration Command Reference for PBR (Continued)*

Command	Mode and Function
set ip precedence *number* | *name*	Route-map subcommand; sets IP precedence value using the decimal number of name
set ip next-hop *ip-address* **[**...*ip-address***]**	Route-map subcommand; defines the IP address(es) of the next-hop router(s) to be used for forwarding packets that match this route map entry
ip route-cache policy	Global command; enables fast switching of PBR-routed packets

Note: Not all PBR-related commands are shown in this table, but commands specifically related to marking are shown.

Table 3-15 *Exec Command Reference for PBR Marking*

Command	Function
show ip policy	Lists configured PBR details, and statistics for numbers of packets matched by each clause.
show route-map	Lists statistical information about packets matched with a route map. PBR uses route maps to classify and mark traffic.

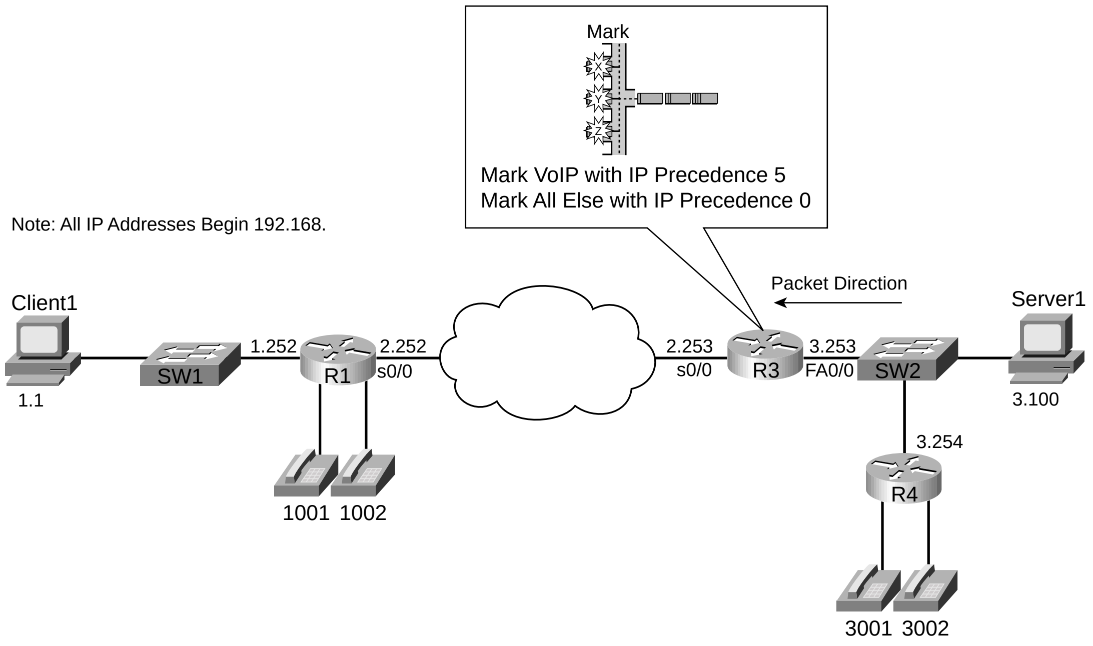
Example 3-7 shows the first PBR marking example, which uses the same criteria as Example 3-1 for CB marking and Example 3-5 for CAR. In this example, R3 is marking packets that flow right to left in Figure 3-18.

- All VoIP payload traffic is marked with IP precedence 5.
- All other traffic is marked with IP precedence 0.

Example 3-7 *PBR Marking, VoIP as DSCP EF, Everything Else as BE*

```
ip route-cache policy
!
ip access-list extended VoIP-ACL
 permit udp any range 16384 32767 any range 16384 32767
!
int fastethernet 0/0
 ip policy route-map voip-routemap
!
route-map voip-routemap permit 10
 match ip address VoIP-ACL
 set ip precedence 5
!
route-map voip-routemap permit 20
set ip precedence 0
```

Figure 3-18 *PBR Marking Sample 1: VoIP Marked with IP Precedence 5, Everything Else Marked IP Precedence 0*



PBR uses **route-map** commands, along with **match** and **set** route-map subcommands, to classify and mark the packets. This configuration uses a route map named voip-routemap, which includes two clauses. The first clause, clause 10, uses a **match** command that refers to VoIP-ACL, which is a named IP ACL. VoIP-ACL matches UDP port numbers between 16,384 and 32,767, which matches all VoIP traffic. If the ACL permits a packet, the route map's first clause acts on the **set** command, which specifies that IP precedence should be set to 5.

The second route map clause, clause 20, matches the rest of the traffic. The route map could have referred to another IP ACL to match all packets; however, by not specifying a match statement in clause 20, all packets will match this clause by default. By not having to refer to another IP ACL to match all packets, less processing overhead is required. The **set** command then specifies to set precedence to zero.

The **ip policy route-map voip-routemap** command enables PBR on interface FA0/0 for incoming packets. Notice that the direction, input or output, is not specified, because PBR can only process incoming packets.

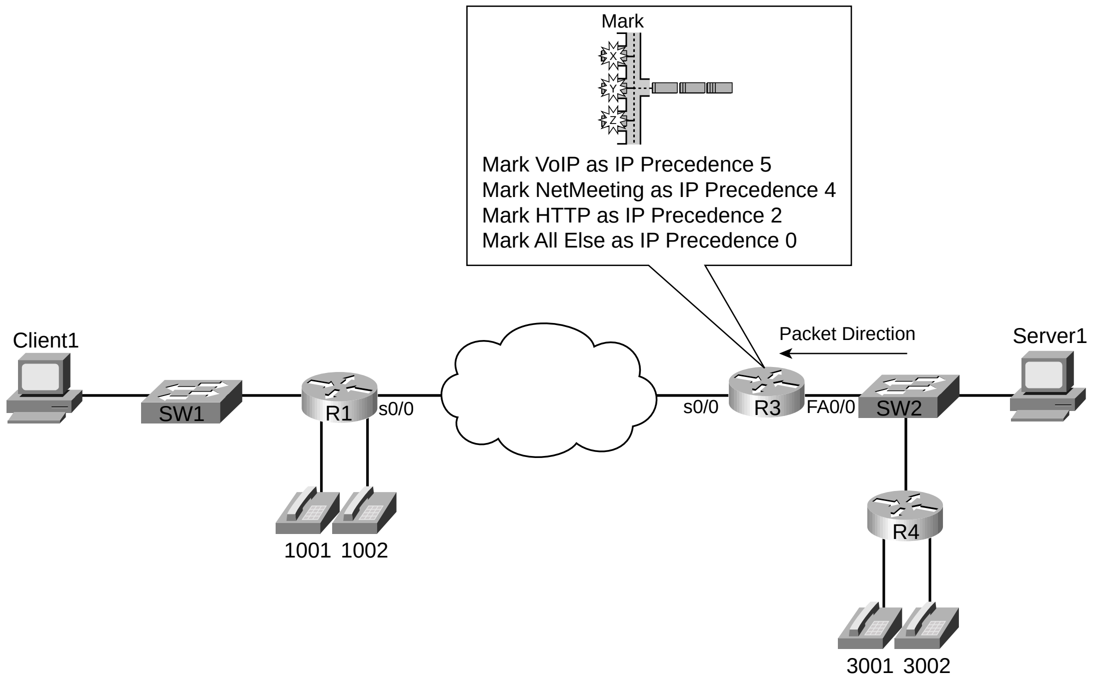
The last PBR-specific command is **ip route-cache policy**. IOS process-switches PBR traffic by default; to use fast switching on PBR traffic, use the **ip route-cache policy** command.

The second PBR configuration (Example 3-8) includes classification options identical to CAR example 2 (see Example 3-6). A major difference between PBR and CAR is that PBR cannot set the DSCP field, so it sets the IP Precedence field instead. The slightly modified criteria, as compared with CAR example 2, for PBR example 2 is as follows:

- VoIP payload is marked with precedence 5.
- NetMeeting voice and video from Server1 to Client1 is marked with precedence 4.
- Any HTTP traffic is marked with precedence 2.
- All other traffic is marked with precedence 0.

Figure 3-19 shows the network in which the configuration is applied, and Example 3-8 shows the configuration.

Figure 3-19 *PBR Marking Sample 2 Network*


Mark
X
Y
Z
Mark VoIP as IP Precedence 5
Mark NetMeeting as IP Precedence 4
Mark HTTP as IP Precedence 2
Mark All Else as IP Precedence 0
Client1
SW1
R1
s0/0
1001 1002
s0/0
R3
FA0/0
Packet Direction
SW2
Server1
R4
3001 3002

Example 3-8 *PBR Marking Sample 2: VoIP, NetMeeting Audio/Video, HTTP URLs, and Everything Else*

```
ip route-cache policy
!
ip access-list extended VoIP-ACL
 permit udp any range 16384 32768 any range 16384 32768
!
ip access-list extended NetMeet-ACL
 permit udp host 192.168.1.100 range 16384 32768 192.168.3.0 0.0.0.255 range 16384 32768
!
!
ip access-list extended http-acl
 permit tcp any eq www any
 permit tcp any any eq www
!
interface fastethernet 0/0
 ip policy route-map voip-routemap
!
route-map voip-routemap permit 10
 match ip-address NetMeet-ACL
 set ip precedence 4
!
route-map voip-routemap permit 20
 match ip-address VoIP-ACL
 set ip precedence 5
!
route-map voip-routemap permit 30
 match ip-address http-acl
 set ip precedence 2
!
route-map voip-routemap permit 40
set ip precedence 0
!
end
R3#sh ip policy
Interface      Route map
Fastethernet0/0    voip-routemap

R3#show route-map
route-map voip-routemap, permit, sequence 10
  Match clauses:
    ip address (access-lists): NetMeet-ACL
  Set clauses:
    ip precedence flash-override
  Policy routing matches: 3 packets, 222 bytes
route-map voip-routemap, permit, sequence 20
  Match clauses:
    ip address (access-lists): VoIP-ACL
  Set clauses:
    ip precedence critical
  Policy routing matches: 14501 packets, 1080266 bytes
```

Example 3-8 *PBR Marking Sample 2: VoIP, NetMeeting Audio/Video, HTTP URLs, and Everything Else (Continued)*

```
route-map voip-routemap, permit, sequence 30
  Match clauses:
    ip address (access-lists): http-acl
  Set clauses:
    ip precedence immediate
  Policy routing matches: 834 packets, 1007171 bytes
route-map voip-routemap, permit, sequence 40
  Match clauses:
  Set clauses:
    ip precedence routine
  Policy routing matches: 8132 packets, 11263313 bytes
```

The output of the **show ip policy** command lists only sparse information. The **show route-map** command enables you to view statistical information about what PBR has performed. This command lists statistics for any activities performed by a route map, including when one is used for PBR. Notice that the four sets of classification criteria seen in the configuration are listed in the highlighted portions of the **show route-map** output, as are packet and byte counters.

PBR Marking Summary

PBR provides another classification and marking tool that examines packet header information to classify and mark packets. PBR is unique compared to the other tools in that it can classify based on information about the route that would be used for forwarding a packet. However, PBR has fewer options for matching header fields for classification as compared with the other tools.

PBR can mark IP precedence, QoS group, as well as the ToS bits. Refer to Table 3-17, in the summary for this chapter, for a complete list of classification and marking fields used by PBR.

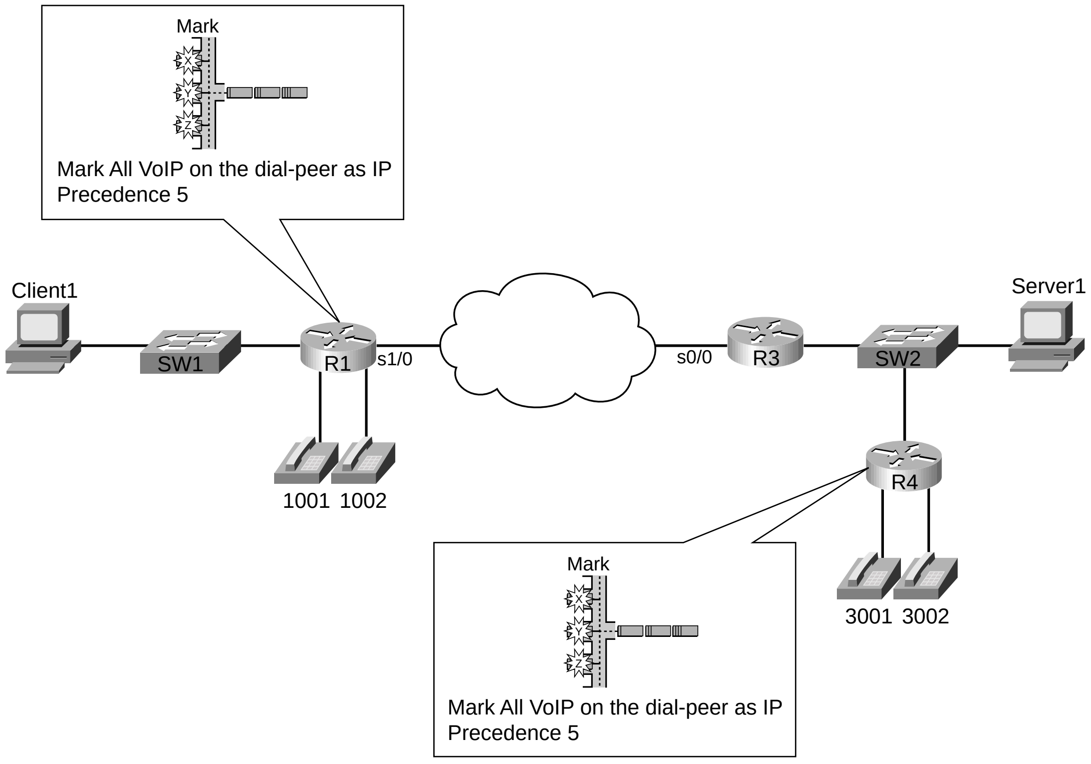
PBR provides a strong option for classification and marking in two cases. For applications when marking based on routing information is useful, PBR can look at details about the route used for each packet, and make marking choices. The other application for PBR marking is when policy routing is already needed, and marking needs to be done at the same time. For more general cases of classification and marking, CB marking or CAR is recommended.

VoIP Dial Peer

IOS voice gateways provide many services to connect the packetized, VoIP network to non-packetized, traditional voice services, including analog and digital trunks. IOS gateways perform many tasks, but one of the most important tasks is to convert from packetized voice to nonpacketized voice, and vice versa. In other words, voice traffic entering a router on an analog or digital trunk is not carried inside an IP packet, but the IOS gateway converts the incoming voice to a digital signal (analog trunks only) and adds the appropriate IP, UDP, and RTP headers around the digital voice (both analog and digital trunks). Conversely, when a VoIP packet arrives, and the voice needs to be sent out a trunk, the IOS gateway removes the packet headers, converts the voice to analog (analog trunks only), and sends the traffic out the trunk.

Although this book does not attempt to explain voice configuration and concepts to much depth, some appreciation for IOS gateway configuration is required for some of the functions covered in this book. In particular, Chapter 8, "Call Admission Control and QoS Signaling," which covers Voice call admission control (CAC), requires a little deeper examination of voice. To understand classification and marking using dial peers, however, only a cursory knowledge of voice configuration is required. Consider Figure 3-20, for instance, which shows two analog IOS voice gateways, R1 and R4, along with Examples 3-9 and 3-10, which show the pertinent configuration on R1 and R4.

Figure 3-20 *Network with Two Analog Voice Gateways*


Mark
X
Y
Z
Mark All VoIP on the dial-peer as IP Precedence 5
Client1
SW1
R1
s1/0
s0/0
R3
SW2
Server1
1001 1002
R4
3001 3002
Mark
X
Y
Z
Mark All VoIP on the dial-peer as IP Precedence 5

Example 3-9 *R1 Voice Gateway Configuration*

```
hostname R1
!
int fastethernet 0/0
ip address 192.168.1.251 255.255.255.0
!
dial-peer voice 3001 voip
 destination-pattern 3001
 session target ipv4:192.168.3.254
!
dial-peer voice 3002 voip
 destination-pattern 3002
 session target ipv4:192.168.3.254
!
dial-peer voice 1001 pots
 destination-pattern 1001
 port 3/0
!
dial-peer voice 1002 pots
 destination-pattern 1002
 port 3/1
```

Example 3-10 *R4 Voice Gateway Configuration*

```
hostname R4
!
int fastethernet 0/0
ip address 192.168.3.254 255.255.255.0
!
dial-peer voice 1001 voip
 destination-pattern 1001
 session target ipv4:192.168.1.251
!
dial-peer voice 1002 voip
 destination-pattern 1002
 session target ipv4:192.168.1.251
!
dial-peer voice 3001 pots
 destination-pattern 3001
 port 3/0
!
dial-peer voice 3002 pots
 destination-pattern 3002
 port 3/1
```

The highlighted portions of the examples focus on the configuration for the physical voice ports on R1, and the VoIP configuration on R4. Both R1 and R4 use **dial-peer** commands to define their local analog voice trunks and to define peers to which VoIP calls can be made. In Example 3-9, for instance, the highlighted portion of the configuration shows R1's configuration of the two local analog lines. The two highlighted **dial-peer** statements use the keyword **pots**, which stands for plain-old telephone service. The **pots** keyword implies that the ports associated with this dial peer are traditional analog or digital telephony ports. The physical analog ports are correlated to each dial peer with the **port** command; in each of these configurations, a two-port FXS card sits inside slot 3 of a 1760-V router. Finally, on R1, the phone number, or dial pattern, associated with each of the analog ports is configured. With just the highlighted configuration in R1, voice calls could be placed between the two extensions (x1001 and x1002).

To place calls to extensions 1001 and 1002 from R4, the **dial-peer** commands highlighted in Example 3-10 are required. These two **dial-peer** commands use a **voip** keyword, which means this dial peer configures information about an entity to which VoIP calls can be placed. The phone number, or dial pattern, is defined with the **destination-pattern** command again—notice that extensions 1001 and 1002 are again configured. Finally, because these two dial peers configure details about a VoIP call, a local physical port is not referenced. Instead, the **session-target ipv4:192.168.1.251** command implies that when these phone numbers are called, to establish a VoIP call, using the IP version 4 IP address shown.

Similarly, R4 defines the local phone numbers and ports for the locally connected phones, and R1 defines VoIP dial peers referring to R4's phones, so that calls can be initiated from R1.

Dial-peer classification and marking, when you know how to configure the basic dial-peer parameters, is easy. POTS dial peers refer to analog or digital trunks, over which no IP packet is in use—so there is nothing to mark. On VoIP dial peers, the dial peer refers to the IP address of another gateway to which a call is placed. So, by placing the **ip precedence 5** dial-peer subcommand under each voip **dial-peer**, the packets generated for calls matching each dial peer will be marked with IP precedence 5. Example 3-11 lists the R4 configuration, with these changes made; the equivalent changes would be made to R1 as well.

Example 3-11 *R4 Voice Gateway Configuration*

```
hostname R4
!
interface fastethernet 0/0
ip address 192.168.3.254 255.255.255.0
!
dial-peer voice 1001 voip
 destination-pattern 1001
 session target ipv4:192.168.1.251
 ip precedence 5
 no vad
!
dial-peer voice 1002 voip
 destination-pattern 1002
 session target ipv4:192.168.1.251
```

Example 3-11 *R4 Voice Gateway Configuration (Continued)*

```
 ip precedence 5
 no vad
!
dial-peer voice 3001 pots
 destination-pattern 3001
 port 3/0
!
dial-peer voice 3002 pots
 destination-pattern 3002
 port 3/1
```

In the example, the highlighted text shows the **ip precedence 5** commands under each **voip dial-peer**. Packets created for VoIP calls for the configured dial patterns of 1001 and 1002 will be marked with IP precedence 5. The identical commands would be added to R1's configuration on the VoIP dial peers to achieve the same effect.

Beginning in IOS Releases 12.2(2)XB and 12.2(2)T the **ip precedence** command has been replaced with the **ip qos dscp** command. This allows the dial peer to set the IP precedence or the DSCP value for VoIP payload and signaling traffic. Also keep in mind that the current DQOS exam, at the time this book was published, was based on IOS 12.1(5)T—so this command would not be on the current exam. Check the URLs listed in the Introduction for any possible changes.

The command uses the following syntax:

```
ip qos dscp [number | set-af | set-cs | default | ef][media | signaling]
```

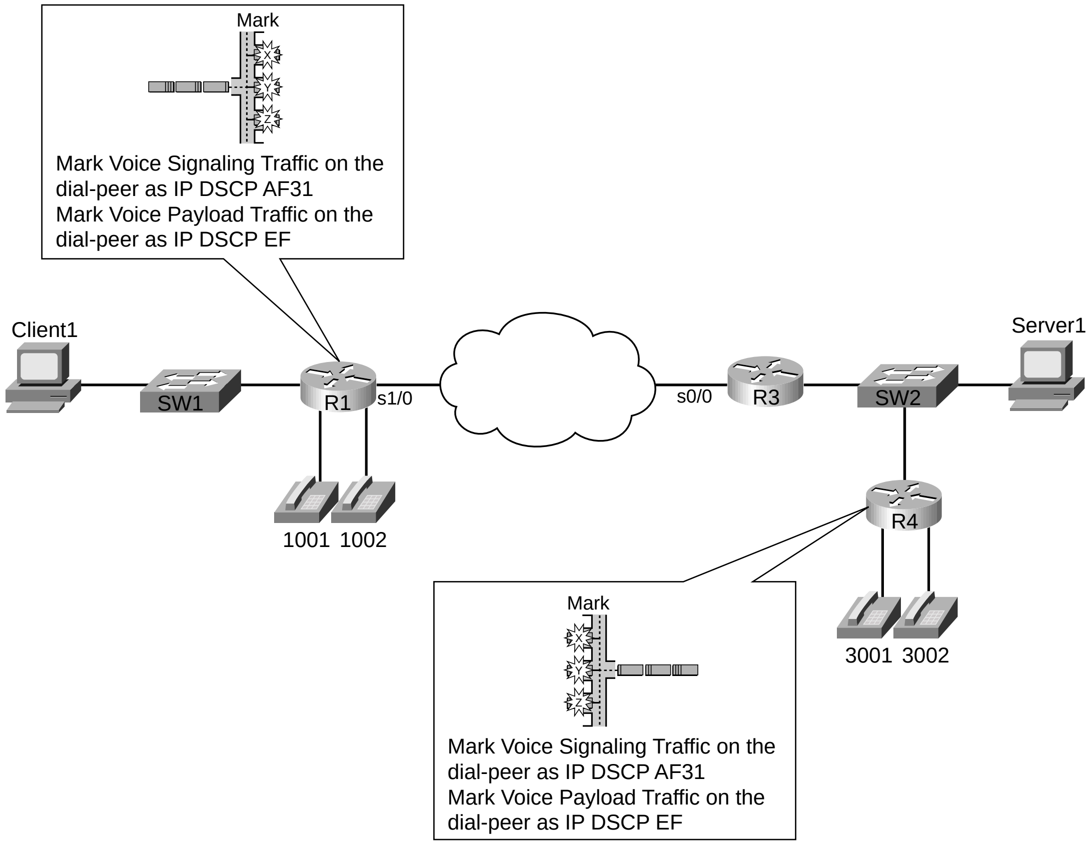
Table 3-16 outlines the meaning of the parameters of the command.

Table 3-16 *IP QoS DSCP Command Options*

IP QoS DSCP Options	Function
number	DSCP value. Valid entries are from 0 to 63.
set-af	Sets DSCP to assured forwarding bit pattern. Acceptable values are as follows: AF11, AF12, AF13, AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43
set-cs	Sets DSCP to class selector code point. Acceptable values are as follows: CS1, CS2, CS3, CS4, CS5, CS6, CS7
default	Sets DSCP to default bit pattern 000000.
ef	Sets DSCP to expedited forwarding bit pattern 101110.
media	Applies the specified DSCP value to the media payload packets.
signaling	Applies the specified DSCP value to the signaling packets.

The **ip qos dscp** command enables you to have much more granular control of how a VoIP packet is marked than the **ip precedence** command, while providing a method to preserve backward compatibility. Examples 3-12 and 3-13 show how R1 and R4 can be configured to use the **ip qos dscp** command to mark voice payload traffic with a DSCP value of EF and voice signaling traffic with a DSCP value of AF31. Figure 3-21 shows the now-familiar network, with the new criteria listed.

Figure 3-21 *Mark Voice Payload Traffic*


Mark
Mark Voice Signaling Traffic on the dial-peer as IP DSCP AF31
Mark Voice Payload Traffic on the dial-peer as IP DSCP EF
Client1
SW1
R1
s1/0
1001
1002
s0/0
R3
SW2
Server1
R4
3001
3002
Mark
Mark Voice Signaling Traffic on the dial-peer as IP DSCP AF31
Mark Voice Payload Traffic on the dial-peer as IP DSCP EF

Example 3-12 *R1 IP QoS DSCP Dial-Peer Configuration*

```
hostname R1
!
int fastethernet 0/0
ip address 192.168.1.251 255.255.255.0
```

Example 3-12 *R1 IP QoS DSCP Dial-Peer Configuration (Continued)*

```
!
dial-peer voice 3001 voip
 destination-pattern 3001
 ip qos dscp ef media
 ip qos dscp af31 signaling
 session target ipv4:192.168.3.254
!
dial-peer voice 3002 voip
 destination-pattern 3002
 ip qos dscp ef media
 ip qos dscp af31 signaling
 session target ipv4:192.168.3.254
!
dial-peer voice 1001 pots
 destination-pattern 1001
 port 3/0
!
dial-peer voice 1002 pots
 destination-pattern 1002
 port 3/1
```

Example 3-13 *R4 IP QoS DSCP Dial-Peer Configuration*

```
hostname R4
!
int fastethernet 0/0
ip address 192.168.3.254 255.255.255.0
!
dial-peer voice 1001 voip
 destination-pattern 1001
 ip qos dscp ef media
 ip qos dscp af31 signaling
 session target ipv4:192.168.1.251
!
dial-peer voice 1002 voip
 destination-pattern 1002
 ip qos dscp ef media
 ip qos dscp af31 signaling
 session target ipv4:192.168.1.251
!
dial-peer voice 3001 pots
 destination-pattern 3001
 port 3/0
!
dial-peer voice 3002 pots
 destination-pattern 3002
 port 3/1
```

In this example, the highlighted text shows the **ip qos dscp** commands used to mark voice signaling with DSCP AF31 and voice payload with DSCP EF. For networks that cannot yet support DSCP markings, you can use the **set-cs** option to mark the voice traffic with IP precedence, providing backward-compatible support.

VoIP Dial-Peer Summary

For voice traffic passing through an IOS gateway, marking the traffic using dial peers provides an easy-to-configure, low-overhead way to mark the packets. Prior to IOS Releases 12.2(2)XB and 12.2(2)T the **ip precedence** command was used to mark all VoIP traffic with an IP precedence value. After these IOS releases, you can use the **ip qos dscp** command to separate and individually mark the voice signaling and voice payload traffic. These markings can be DCSP values, or IP precedence values if backward compatibility is needed. Refer to Tables 3-14 and 3-15 for **ip qos dscp** command options.

Summary of Classification and Marking QoS Features

Classification and marking tools can be easily compared based on three general categories. First, some classification and marking tools are specialized, and some are more general. CB marking, CAR, and PBR all perform the same basic function of matching packets based on fields inside the packet header, and marking based on those fields, so these three tools are the more generalized classification and marking tools. Dial peers perform specialized classification and marking functions. This tool is different from the other three because it does not classify packets based on a variety of fields in a packet header, instead classifying all traffic to the particular dial peer.

The other two points for comparison of classification and marking tools concern what each tool can match for classification, and what each can mark for the marking feature. Tables 3-17 and 3-18 summarize the fields that you can use for classification and marking in the tools, respectively.

Table 3-17 *Classification Fields Used by Classification and Marking Tools*

Classification Field*	CB Marking	CAR	PBR	Dial Peers
Extended IP ACLs	X	X	X	
DSCP	X			
Precedence	X	X		
QoS Group	X			
CoS	X			
NBAR	X			
VoIP Payload (even-numbered RTP UDP ports)	X			

Table 3-17 *Classification Fields Used by Classification and Marking Tools (Continued)*

Classification Field*	CB Marking	CAR	PBR	Dial Peers
MPLS Experimental bits	X	X		
Input Interface	X			
Source MAC address	X	X		
Destination MAC address	X			
BGP ASN			X	
BGP Community			X	
Outgoing Interface			X	
Next-hop IP Address			X	
Routing Protocol Metric			X	
Source of Routing Information			X	
Packet Length			X	
VoIP Dial Peers				X

* Some fields can be matched via ACL as well as matched directly. For instance, DSCP can be matched with an ACL. Because CB marking is the only tool that can directly configure a match for DSCP, only CB marking's box is checked in the table. Refer to Table 3-2, earlier in this chapter, for a list of fields matchable with IP extended ACLs.

Table 3-18 *Marking Fields Used by Classification and Marking Tools*

Marking Field	CB Marking	CAR	PBR	Dial Peers
DSCP	X	X		X
Precedence	X	X	X	X
QoS Group	X	X	X	
CoS	X			
MPLS Experimental bits	X	X		
Frame Relay DE	X			
ATM CLP	X			
IP ToS bits			X	

The choice of when to use each tool can be confusing. Dial peers have specific applications, so the number of instances where they are useful directs you when to consider each tool. Dial peers are used only in IOS voice gateways, which is a convenient place to set IP precedence or DSCP.

The three general classification and marking tools create a more difficult choice, because each can be enabled on almost any interface. CB marking may be ruled out based on the IOS revision being used, because CB marking did not appear in a T-train IOS until 12.1(5)T, and in mainline IOS until 12.2.

NOTE Refer to www.cisco.com/go/fn to use the Cisco feature navigator to find IOS revision levels required for different features.

When at a level of code that support CB marking, the general recommendation is to use CB marking, unless one of the next two statements is true. If you need to use PBR to perform policy routing, and you also need to mark, go ahead and use PBR for marking. Similarly, if you need to perform policing and marking at the same point in the network, use CAR for both.

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures are a convenient way to review the day before the exam.

Table 3-19 shows the list of items that can be matched with an extended IP ACL. Table 3-20 lists the fields that can be matched by classification and marking tools without use of an ACL. Note that some header fields can be matched by an ACL or directly through some other style of configuration—in those cases, it is typically better to match the field directly, rather than with an ACL.

Table 3-19 *IP Extended ACL Matchable Fields—IOS 12.2*

Field	**Comments**
Source IP address	A range of source IP addresses can be matched by using a wildcard mask.
Destination IP address	A range of source IP addresses can be matched by using a wildcard mask.
IP Precedence	Format of command uses names for precedence. The following table lists the decimal value for each name. Name IP precedence value **routine** 0 **priority** 1 **immediate** 2 **flash** 3 **flash-override** 4

continues

Table 3-19 *IP Extended ACL Matchable Fields—IOS 12.2 (Continued)*

Field	Comments
IP Precedence *(Continued)*	**Critic** 5 **internet** 6 **network** 7
IP DSCP	Format of the command allows use of differentiated services code point (DSCP) names, as well as decimal values.
IP ToS	Can check to see whether a single Type of Service (ToS) field bit is toggled on; keywords are **normal** (binary **0000**), **max-reliability** (binary **1000**), **max-throughput** (binary **0100**), **min-delay** (binary **0010**), and **min-monetary-cost** (binary **0001**).
TCP ports	Can check source and destination ports; can also check a range of port numbers, whether a port number is larger or smaller than a single value.
TCP Established	Although not typically useful for QoS classification, ACLs can match all TCP segments after the initial segment used for connection establishment.
UDP	Checks the source and destination ports; can also check a range of port numbers, whether a port number is larger or smaller than a single value.
ICMP	Checks a larger variety of ICMP messages and code types (for example, echo request and echo reply).
IGMP	Checks for Internet Group Management Protocol (IGMP) message types.

Table 3-20 *Fields* Directly *Matchable by Classification and Marking tools*

Field	Tool	Comments
Source MAC address	CAR, CB marking	Committed access rate (CAR) uses special "access-rate" ACLs; class-based (CB) marking uses the **match** command.
IP Precedence	CAR, CB marking	CAR uses special "access-rate" ACLs specific to CAR; CB marking uses the **match** command; both can match a subset of values.
MPLS Experimental	CAR, CB marking	CAR uses special "access-rate" ACLs specific to CAR; CB marking uses the **match** command; both can match a subset of values.

Table 3-20 *Fields* Directly *Matchable by Classification and Marking tools (Continued)*

Field	Tool	Comments
CoS	CB marking	Checks incoming ISL/802.1P CoS bits. Can match multiple values.
Destination MAC address	CB marking	Checks for destination MAC address. Can match multiple values.
Input Interface	CB marking	Checks for input interface. Can match multiple values.
IP DSCP	CB marking	Can check for multiple values using multiple **match** commands.
RTP's UDP port-number range	CB marking	RTP uses even-numbered UDP ports from 16,384 to 32,767. This option allows matching a subset of these values, even-numbered ports only, because RTP only uses even-numbered ports.
QoS Group	CB marking	The QoS Group field is used to tag packets internal to a single router.
NBAR protocol types	CB marking	Refer to the "Network Based Application Recognition (NBAR)" section in this chapter for more details.
NBAR Citrix applications	CB marking	NBAR can recognize different types of Citrix applications; CB marking can use NBAR to classify based on these application types.
Host name and URL string	CB marking	NBAR can also match URL strings, including the host name, using regular expressions. CB marking can use NBAR to match these strings for classification.
Outgoing Interface	Policy-based routing (PBR)	Checks the routing table and finds all valid routes for the packet; matches based on the outgoing interface.
Next-Hop	PBR	Similar to the outgoing interface, but it checks the next-hop routers' IP addresses.
Metric	PBR	Checks the routing table entry for this packet, and compares the metric value to match the packet.
Route type	PBR	Checks the routing table, looking at the source of the routing table entry that matches the packet.
Dial Peer	Dial peers	Based on the dial peer and used to connect a VoIP call.

Figure 3-22 outlines the two IP marking fields and their positions inside an IP header: The suggested values for these fields, and their names, are listed in Table 3-21.

Figure 3-22 *IP Precedence and IP DSCP Fields*

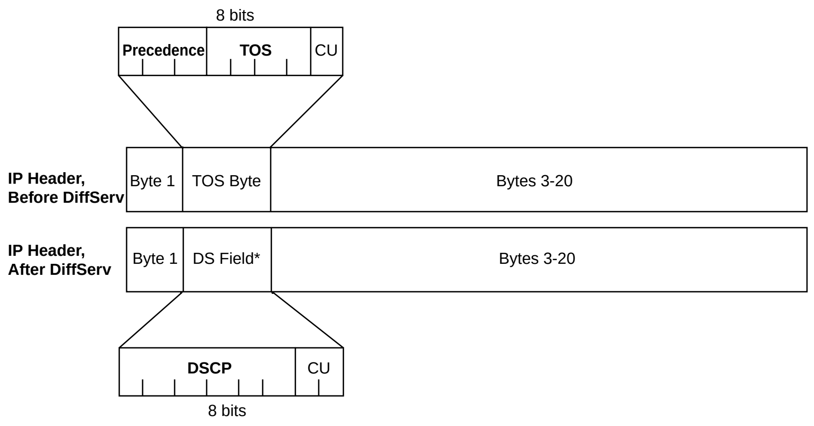


Table 3-21 *IP Precedence and DSCP—Popular Values and Names*

Field and Value (Decimal)	Binary Value	Name	Defined by This RFC
Precedence 0	**000**	**routine**	791
Precedence 1	**001**	**priority**	791
Precedence 2	**010**	**immediate**	791
Precedence 3	**011**	**flash**	791
Precedence 4	**100**	**flash override**	791
Precedence 5	**101**	**critic**	791
Precedence 6	**110**	**internetwork control**	791
Precedence 7	**111**	**network control**	791
DSCP 0	**000**000	**best effort** or **default**	2475
DSCP 8	**001**000	CS1	2475
DSCP 16	**010**000	CS2	2475
DSCP 24	**011**000	CS3	2475
DSCP 32	**100**000	CS4	2475
DSCP 40	**101**000	CS5	2475
DSCP 48	**110**000	CS6	2475
DSCP 56	**111**000	CS7	2475

Table 3-21 *IP Precedence and DSCP—Popular Values and Names (Continued)*

Field and Value (Decimal)	Binary Value	Name	Defined by This RFC
DSCP 10	**001**010	AF11	2597
DSCP 12	**001**100	AF12	2597
DSCP 14	**001**110	AF13	2597
DSCP 18	**010**010	AF21	2597
DSCP 20	**010**100	AF22	2597
DSCP 22	**010**110	AF23	2597
DSCP 26	**011**010	AF31	2597
DSCP 28	**011**100	AF32	2597
DSCP 30	**011**110	AF33	2597
DSCP 34	**100**010	AF41	2597
DSCP 36	**100**100	AF42	2597
DSCP 38	**100**110	AF43	2597
DSCP 46	**101**110	EF	2598

CS = Class Selector
AF = Assured Forwarding
EF = Expedited Forwarding

Figure 3-23 shows the general location of the CoS field inside ISL and 802.1P headers.

Figure 3-23 *LAN Class Of Service Fields*

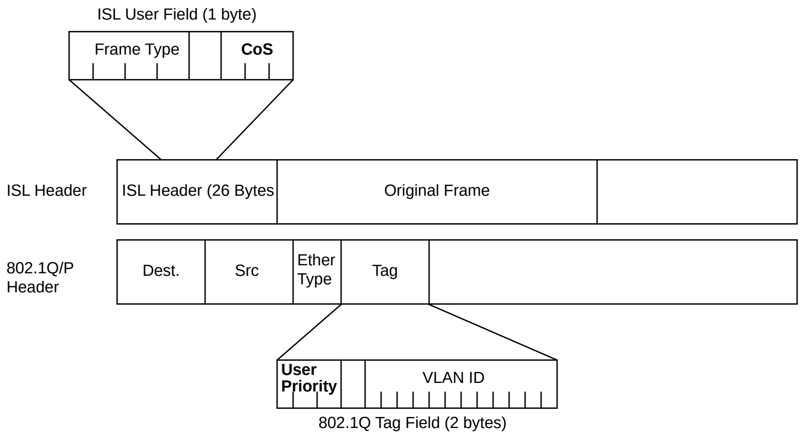


Table 3-22 summarizes the marking fields.

Table 3-22 *Names of Marking Fields*

Field	Location	Length	Comments
IP Precedence	IP header	3 bits	Contained in the first 3 bits of the ToS byte.
IP DSCP	IP header	6 bits	Contained in the first 6 bits of the DS field, which replaces the ToS byte.
DS	IP header	1 byte	Replaces ToS byte per RFC 2475.
ToS	IP header	1 byte	Replaced by DS field per RFC 2475.
ToS	IP header	4 bits	A field inside the ToS byte; superseded by RFC 2475.
CoS	ISL and 802.1Q/P	3 bits	Cisco convention uses “CoS” to describe either trunking headers’ QoS field.
Priority bits	802.1Q/P	3 bits	The name used by IEEE 802.1P for the CoS bits.
Discard Eligible (DE)	Frame Relay header	1 bit	Frame Relay switches may discard DE-marked frames, avoiding discarding frames without DE marked, under congestion.
Cell Loss Priority (CLP)	ATM cell header	1 bit	ATM equivalent of the DE bit
MPLS Experimental values(s)	MPLS header	3 bits	Used to pass QoS marking information across an MPLS network.
QoS Group	Headers internal to IOS	N/A	Uses values between 1–99 inclusive. Used for marking only internal to a single router, specifically only on the GSR/ESR product lines.

Table 3-23 lists the MQC commands used for CB marking. The table shows all the classification options available using the **match** command, and all the marking options available using the **set** command. Table 3-24 lists the **show** commands related to CB marking.

Table 3-23 *Command Reference for Class-Based Marking*

Command	Mode and Function
class-map *class-map-name*	Global config; names a class map, where classification options are configured
Match …	Class-map subcommand; defines specific classification parameters

Table 3-23 *Command Reference for Class-Based Marking (Continued)*

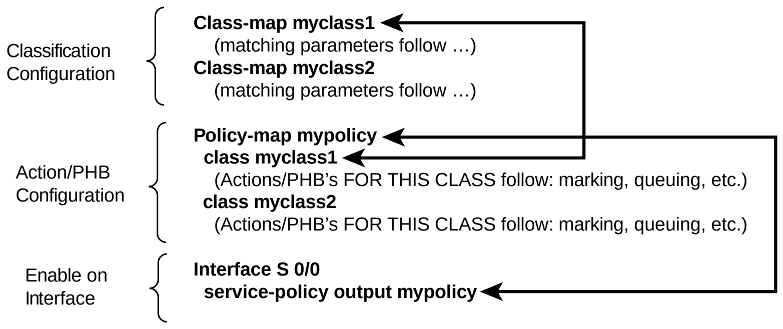
Command	Mode and Function
match access-group {*access-group* | **name** *access-group-name*}	ACL
match source-address mac *address-destination*	Source MAC address
match ip precedence *ip-precedence-value* [*ip-precedence-value ip-precedence-value ip-precedence-value*]	IP precedence
match mpls experimental *number*	MPLS Experimental
match cos *cos-value* [*cos-value cos-value cos-value*]	CoS
match destination-address mac *address*	Destination MAC address
match input-interface *interface-name*	Input interface
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	IP DSCP
match ip rtp *starting-port-number port-range*	RTP's UDP port-number range
match qos-group *qos-group-value*	QoS group
match protocol *protocol-name*	NBAR protocol types
match protocol citrix [**app** *application-name-string*]	NBAR Citrix applications
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	Host name and URL string
match any	All packets
policy-map *policy-map-name*	Global config; names a policy, which is a set of actions to perform.
class *class-name*	Policy-map subcommand; identifies which packets on which to perform some action by referring to the classification logic in a class map
set	For the class, marks (sets) particular QoS fields
set ip precedence *ip-precedence-value*	IP precedence
set ip dscp *ip-dscp-value*	IP DSCP
set cos *cos-value*	CoS
set ip qos-group *group-id*	QoS group
set atm-clp	ATM CLP bit
Set fr-de	Frame Relay DE bit

Table 3-24 *Exec Command Reference for Class-Based Marking*

Command	Function
show policy-map *policy-map-name*	Lists configuration information about all MQC-based QoS tools
show policy-map *interface-spec* [***input*** | ***output***] [**class** *class-name*]	Lists statistical information about the behavior of all MQC-based QoS tools

Figure 3-24 shows the general flow of MQC commands.

Figure 3-24 *MQC Commands and Their Correlation*


Classification Configuration
Class-map myclass1
(matching parameters follow …)
Class-map myclass2
(matching parameters follow …)
Action/PHB Configuration
Policy-map mypolicy
class myclass1
(Actions/PHB's FOR THIS CLASS follow: marking, queuing, etc.)
class myclass2
(Actions/PHB's FOR THIS CLASS follow: marking, queuing, etc.)
Enable on Interface
Interface S 0/0
service-policy output mypolicy

Tables 3-25 and 3-26 list the NBAR configuration and exec commands, respectively.

Table 3-25 *Configuration Command Reference for NBAR*

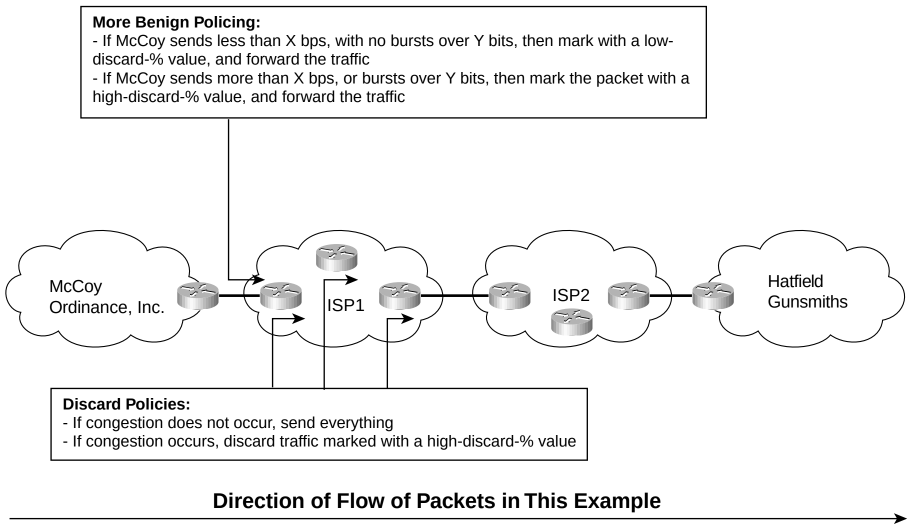
Command	Mode and Function
ip nbar protocol-discovery	Interface mode; enables NBAR for traffic entering the interface.
ip nbar port-map *protocol-name* [**tcp** | **udp**] *port-number*	Global; tells NBAR to search for a protocol using a different port number than the well-known port. Also defines ports to be used by custom packet description language modules (PDLM).
ip nbar pdlm *pdlm-name*	Global; extends the list of protocols recognized by NBAR by adding additional PDLMs.

You can use CAR for policing, but instead of discarding packets, CAR can instead mark nonconforming packets with a value that increases the packets' chances of being discarded when congestion occurs, as seen in Figure 3-25.

Table 3-26 *Exec Command Reference for NBAR*

Command	Function
show ip nbar protocol-discovery [**interface** *interface-spec*] [**stats** {**byte-count** | **bit-rate** | **packet-count**}][{**protocol** *protocol-name* | **top-n** *number*}]	Lists information about statistics for the discovered protocols. Statistics can be listed by interface, by protocol, or for just the top *n* protocols by volume.
show ip nbar port-map [*protocol-name*]	Lists the current ports in use by the discovered protocols.

Figure 3-25 *Policing: Excess Traffic Marked with Lower Value*


More Benign Policing:
- If McCoy sends less than X bps, with no bursts over Y bits, then mark with a low-discard-% value, and forward the traffic
- If McCoy sends more than X bps, or bursts over Y bits, then mark the packet with a high-discard-% value, and forward the traffic
McCoy Ordinance, Inc.
ISP1
ISP2
Hatfield Gunsmiths
Discard Policies:
- If congestion does not occur, send everything
- If congestion occurs, discard traffic marked with a high-discard-% value
Direction of Flow of Packets in This Example

Tables 3-27, 3-28, and 3-29 list the pertinent CAR configuration and exec commands, respectively.

Table 3-27 *Configuration Command Reference for CAR*

Command	Mode and Function
rate-limit {**input** | **output**} [**access-group** [**rate-limit**] *acl-index*] *bps burst-normal burst-max* **conform-action** *conform-action* **exceed-action** *exceed-action*	Interface mode; configures classification, marking, policing, and enabling CAR on the interface
access-list rate-limit *acl-index* {*precedence* | *mac-address* | *exp* **mask** *mask*}	Global mode; creates a CAR ACL, which can match IP precedence, MAC addresses, and MPLS Experimental bits

Table 3-28 *Possible Actions with CAR* **rate-limit** *Command*

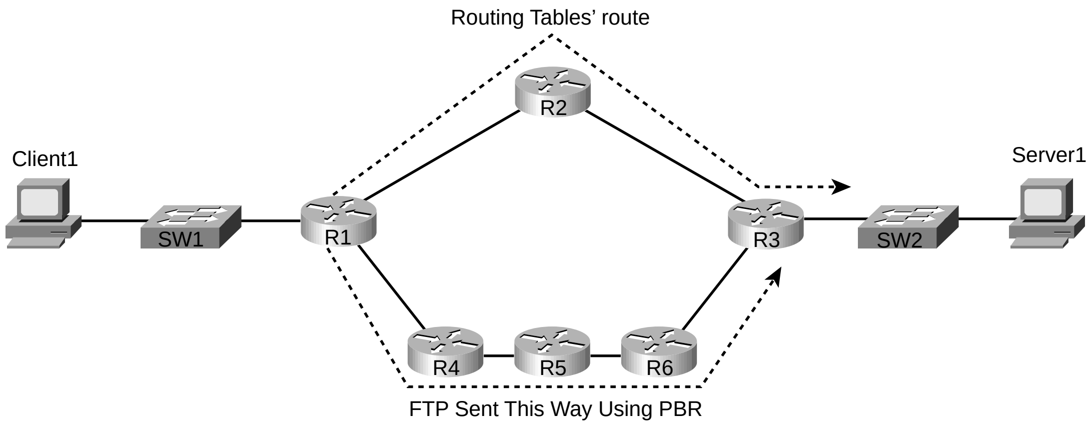
rate-limit Conform and Exceed Options	Function
Continue	Evaluates the next **rate-limit** command
Drop	Drops the packet
set-dscp-continue	Sets the differentiated services code point (DSCP) (0–63) and evaluates the next **rate-limit** command
set-dscp-transmit	Sets the DSCP and transmits the packet
set-mpls-exp-continue	Sets the MPLS Experimental bits (0–7) and evaluates the next **rate-limit** command
set-mpls-exp-transmit	Sets the MPLS Experimental bits (0–7) and sends the packet
set-prec-continue	Sets the IP precedence (0–7) and evaluates the next **rate-limit** command
set-prec-transmit	Sets the IP precedence (0–7) and sends the packet
set-qos-continue	Sets the QoS group ID (1–99) and evaluates the next **rate-limit** command
set-qos-transmit	Sets the QoS group ID (1–99) and sends the packet
Transmit	Sends the packet

Table 3-29 *Exec Command Reference for CAR*

Command	Function
show interfaces [*interface-type interface-number*] **rate-limit**	Displays CAR statistics on the interface specified, or on all interfaces if the interface is not specified
show access-lists rate-limit [*acl-index*]	Lists information about the configuration of rate-limit ACLs

Policy-based routing (PBR) enables you to route a packet based on some other information besides the destination IP address. Figure 3-26 shows a simple example, where FTP traffic is directed over the longer path in the network.

Figure 3-26 *PBR: FTP Traffic Routed over Longer Path*



Tables 3-30 and 3-31 list the pertinent PBR configuration and exec commands, respectively.

Table 3-30 *Configuration Command Reference for PBR*

Command	Mode and Function
ip local policy route-map *map-tag*	Global; specifies that packets generated by this router should be candidates for policy routing
ip policy route-map *map-tag*	Interface subcommand; refers to a route map, which in turn classifies packets and specifies actions; actions include specifying a different route, and setting IP precedence
route-map *map-tag* **[permit | deny] [***sequence-number***]**	Global command; creates a route map entry
match ip address {*access-list-number* | *access-list-name***} [...** *access-list-number* | ... *access-list-name***]**	Route-map subcommand; used to match IP packets based on parameters that can be matched with an IP ACL
match length *minimum-length maximum-length*	Route-map subcommand; used to match IP packets based on their length
set ip precedence *number* | *name*	Route-map subcommand; sets IP precedence value using the decimal number of name

continues

Table 3-30 *Configuration Command Reference for PBR (Continued)*

Command	**Mode and Function**
set ip next-hop *ip-address* **[**...*ip-address***]**	Route-map subcommand; defines the IP address(es) of the next-hop router(s) to be used for forwarding packets that match this route map entry
ip route-cache policy	Global command; enables fast switching of PBR-routed packets

Note: Not all PBR-related commands are shown in this table, but commands specifically related to marking are shown.

Table 3-31 *Exec Command Reference for PBR Marking*

Command	**Function**
show ip policy	Lists configured PBR details, and statistics for numbers of packets matched by each clause.
show route-map	Lists statistical information about packets matched with a route map. PBR uses route maps to classify and mark traffic.

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD-ROM included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD-ROM nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD-ROM exam to include, or not include, the topics that are only on the CCIP QoS.

1. Describe the difference between classification and marking.
2. Describe, in general, how a queuing feature could take advantage of the work performed by a classification and marking feature.
3. Characterize what must be true before the CoS field may be useful for marking packets.
4. Most other QoS tools, besides classification and marking tools, also have a classification feature. Describe the advantage of classification, in terms of overall QoS design and policies, and explain why classification and marking is useful, in spite of the fact that other tools also classify the traffic.
5. Which of the following classification and marking tools can classify based on the contents of an HTTP URL: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?
6. Describe the differences between IP extended ACLs as compared with NBAR for matching TCP and UDP port numbers.
7. Which of the following classification and marking tools can classify based on the outgoing interface of the route used for a packet: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?
8. Which of the following classification and marking tools can classify based on the destination TCP port number of a packet, without using an IP ACL: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?

9 Which of the following classification and marking tools can classify based on the DSCP, without using an IP ACL: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?

10 Which of the following classification and marking tools can classify based on either the source or destination MAC address: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?

11 Which of the following classification and marking tools can classify based on the even numbered UDP ports used for RTP traffic, with or without using an IP ACL: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial-peers?

12 Which of the following QoS marking fields are carried inside an 802.1Q header: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

13 Which of the following QoS marking fields are carried inside an IP header: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

14 Which of the following QoS marking fields are never marked inside a frame that exits a router: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

15 Describe the goal of marking near the edge of a network in light of the meaning of the term “trust boundary.”

16 Define the meaning of MQC, and spell out what the acronym abbreviates.

17 What configuration command lists the classification details when configuring CB marking? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

18 What configuration command lists the marking details when configuring CB marking? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

19 What configuration command enables CB marking? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

20 What configuration command lists the classification details when configuring CAR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

21 What configuration command lists the marking details when configuring CAR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

22 What configuration command enables CAR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

23 What configuration command lists the classification details when configuring PBR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

24 What configuration command lists the marking details when configuring PBR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

25 What configuration command enables PBR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

26 Describe the process dial peers use to classify and mark traffic.

27 What configuration command(s) lists the marking details when configuring dial peers? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

28 What QoS values can a dial peer mark?

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Topics

Congestion Management

- Identify and differentiate between the different IOS queuing techniques.
- Correctly apply each queuing technique to the appropriate application.
- Describe the difference between IP RTP Priority and Low Latency Queuing (LLQ).
- Configure WFQ, CBWFQ, and LLQ.

QoS Exam Topics

- Describe how queuing works on Cisco routers
- Describe FIFO Queuing (FQ) and its benefits and drawbacks
- Describe Priority Queuing (PQ) including its benefits and drawbacks
- Describe Custom Queuing (CQ) including its benefits and drawbacks
- Describe Weighted Fair Queuing (WFQ) including its benefits and drawbacks
- Configure Weighted Fair Queuing (WFQ) on Cisco routers
- Describe IP RTP Prioritization
- Configure IP RTP Prioritization on Cisco routers
- Configure the Modular QoS CLI to perform service policies

CHAPTER 4

Congestion Management

Most people understand the basic concepts of queuing, because most of us experience queuing every day. We wait in line to pay for groceries, we wait for a bank teller, we wait to get into a ride at an amusement park, and so on. So, most of the queuing concepts inside this chapter are intuitive.

Cisco uses the term "congestion management" to refer to queuing systems in their products. Queuing concepts in real life are quite similar to queuing in routers and switches. This chapter begins with a more formalized introduction to queuing, to clarify what most people already intuitively know. This chapter covers such issues as how many queues exist, how routers choose to place packets into each queue, what to do when the queue is full, and so on.

Following the discussion of concepts, the text explains the concepts and configurations related to the queuing tools covered on the DQOS exam. In Appendix B, "Topics on the CCIP QoS Exam," you can find coverage of some other queuing tools that happen to be covered on the QoS exam. Each tool can be compared with the other tools based on the same details covered in the concepts section of the chapter—how to classify a packet into a queue, what to do if the queue is full, and so on. The discussion about queuing tools also examines the details particular to each tool, the configuration, and **show** commands related to each tool.

When Cisco does make the next revision of these two exams, I am guessing that a few queuing tools will be dropped from the QoS exam. However, that's just a guess, so make sure to check www.cisco.com for changes to the exams, and www.ciscopress.com/1587200589 for the latest advice from the authors on a good study plan in case the exams have changed.

"Do I Know This Already?" Quiz

The purpose of the "Do I Know This Already?" quiz is to help you decide if you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 16-question quiz, derived from the major sections in "Foundation Topics" portion of the chapter, helps you determine how to spend your limited study time.

Table 4-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 4-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Foundation Topics Section Covering These Questions	Questions
Queuing Concepts	1 to 4
WFQ and IP RTP Priority	5 to 8
CBWFQ and LLQ	9 to 12
Comparing Queuing Options	13 to 16

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **14 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary," and the "Q&A" sections.
- **15 or 16 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, move to the next chapter.

Queuing Concepts Questions

1 Describe the benefits of having a single FIFO output queue.

2 Explain the purpose of a TX Ring and TX Queue in a Cisco router.

3 Assume a queuing tool has been enabled on interface S0/0. Describe the circumstances under which the queuing tool would actually be used.

4 Explain the circumstances under which it would be useful to enable a queuing tool on a subinterface.

WFQ and IP RTP Priority Questions

5 Characterize the effect the WFQ scheduler has on different types of flows.

6 Describe the WFQ scheduler process. Include at least the concept behind any formulas, if not the specific formula.

7 You previously disabled WFQ on interface S0/0. List the minimum number of commands required to enable WFQ on S0/0.

8 What commands list statistical information about the performance of WFQ?

CBWFQ and LLQ Questions

9 Describe the CBWFQ scheduler process, both inside a single queue and among all queues.

10 Describe how LLQ allows for low latency while still giving good service to other queues.

11 Compare and contrast IP RTP Priority and LLQ. In particular, mention what other queuing tools can be used concurrently with each, how each classifies packets, and which is recommended by Cisco.

12 Compare and contrast the CBWFQ command that configures the guaranteed bandwidth for a class with the command that enables LLQ for a class.

Comparing Queuing Options Questions

13 Which of the following queuing tools allows for WRED inside a single queue? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

14 Which of the following queuing tools can always service a particular queue first, even when other queues have packets waiting? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

15 Which of the following queuing tools allows for a percentage bandwidth to be assigned to each queue? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

16 Which queuing tools could be configured to provide the lowest possible latency for voice traffic? Of these, which does Cisco recommend as the best option for voice queuing today?

Foundation Topics

Queuing has an impact on all four QoS characteristics directly—bandwidth, delay, jitter, and packet loss. Many people, upon hearing the term "QoS," immediately think of queuing, but QoS includes many more concepts and features than just queuing. Queuing is certainly the most often deployed and most important QoS tool.

This chapter begins by explaining the core concepts about queuing. Following that, each queuing tool is covered, including additional concepts specific to that tool, configuration, monitoring, and troubleshooting.

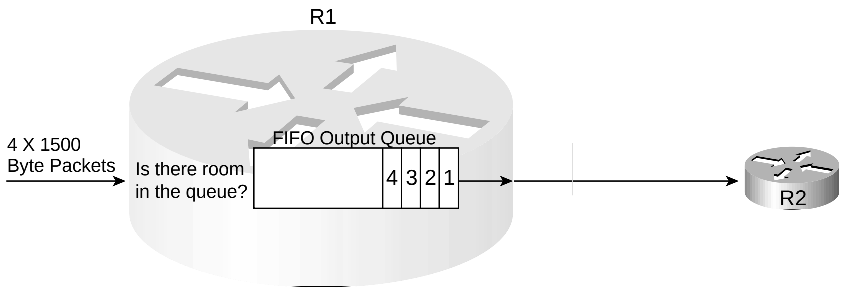
Queuing Concepts

Most people already understand many of the concepts behind queuing. First, this section explains the basics and defines a few terms. Afterward, some of the IOS-specific details are covered.

IOS stores packets in memory while processing the packet. When a router has completed all the required work except actually sending the packet, if the outgoing interface is currently busy, the router just keeps the packet in memory waiting on the interface to become available. To manage the set of packets sitting around in memory waiting to exit an interface, IOS creates a queue. A queue just organizes the packets waiting to exit an interface; the queue itself is nothing more than a series of pointers to the memory buffers that hold the individual packets that are waiting to exit the interface.

The most basic queuing scheme uses a single queue, with first-in, first-out (FIFO) scheduling. What does that mean? Well, when the IOS decides to take the next packet from the queue, of those packets still in the queue, it takes the one that arrived earlier than all the other packets in the queue. Figure 4-1 shows a router, with an interface using a single FIFO queue.

Figure 4-1 *Single FIFO Queue*



Although a single FIFO queue seems to provide no QoS features at all, it actually does affect drop, delay, and jitter. Because there is only one queue, the router need not classify traffic to place it into different queues. Because there is only one queue, the router need not worry about how to decide from which queue it should take the next packet—there is only one choice. And because this single queue uses FIFO logic, the router need not reorder the packets inside the queue.

However, the size of the output queue affects delay, jitter, and loss. Because the queue has a finite size, it may fill. If it fills, and another packet needs to be added to the queue, tail drop would cause the packet to be dropped. One solution to the drops would be to lengthen the queue, which decreases the likelihood of tail drop. With a longer queue, however, the average delay increases, because packets may be enqueued behind a larger number of other packets. In most cases when the average delay increases, the average jitter increases as well. The following list summarizes the key concepts regarding queue length:

- With a longer queue length, the chance of tail drop decreases as compared with a shorter queue, but the average delay increases, with the average jitter typically increasing as well.
- With a shorter queue length, the chance of tail drop increases as compared with a longer queue, but the average delay decreases, with the average jitter typically decreasing as well.
- If the congestion is sustained such that the offered load of bytes trying to exit an interface exceeds the interface speed for long periods, drops will be just as likely whether the queue is short or long.

To appreciate most queuing concepts, you need to consider a queuing system with at least two queues. Consider Figure 4-2, which illustrates two FIFO output queues.

Figure 4-2 *Dual FIFO Output Queues*

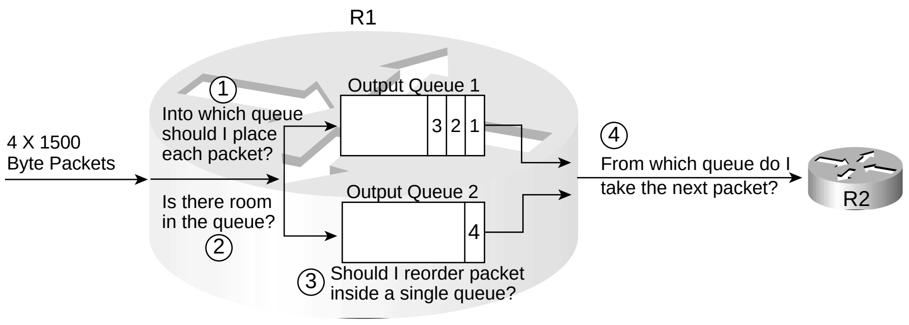

R1
1
Into which queue should I place each packet?
4 X 1500 Byte Packets
Is there room in the queue?
2
Output Queue 1
3 2 1
Output Queue 2
4
3
Should I reorder packet inside a single queue?
4
From which queue do I take the next packet?
R2

Figure 4-2 illustrates the questions that are answered by the queuing tool. Step 1, the classification step, works like classification and marking tools, except the resulting action is to place a packet in a queue, as opposed to marking a packet. So, at step 1, the packet header is examined, and depending on the matched values, the packet is placed into the appropriate queue. Before placing the packet in the queue, the router must make sure that space is available, as shown in step 2 in Figure 4-2. If no space is available, the packet is dropped. Inside each queue, the packets can be reordered (step 3): in this example, however, each queue uses FIFO logic inside each queue, so the packets would never be reordered inside the queue. Finally, the queuing system must choose to take the next packet for transmission from either Queue 1 or Queue 2 (step 4). The scheduler makes the decision at step 4.

Although the classification portion of each queuing tool is relatively obvious, consider two related points when thinking about classification by a queuing tool. First, with a QoS strategy that causes classification and marking to occur near the ingress edge of the network, the queuing tool may enqueue packets that have already been marked. So, the queuing tool can classify based on these marked fields, which was the whole point in marking them in the first place! Second, for each category of traffic for which you want to provide different queuing treatment, you need a different queue. For instance, you may want to classify traffic into six classes for queuing, so each class can get different treatment in a different queue. If the queuing tool only supports four different queues, you may need to consider a different queuing tool to support your QoS policy.

Inside each queue, the queuing methods use FIFO Queuing. The interesting logic for queuing occurs after the packets have been enqueued, when the router decides from which queue to take the next packet. *Queue scheduling* describes the process of the device, in this case a router, choosing which queue to service next. This process is also called a *service algorithm*, or a queue service algorithm. The scheduler may reserve amounts of bandwidth, or a percentage of link bandwidth, for a particular queue. The scheduler may always service one queue first, which means the packets in that queue will experience very low delay and jitter.

For the exams, you need to know what each queuing tool's scheduler accomplishes; for some tools, however, you also need to know the internals of how the scheduler actually works.

NOTE Cisco leads the industry in making details about their products public (being the first large networking vendor to publish bug reports, for instance). However, Cisco must also protect their intellectual assets. So, for some of the newer queuing tools, Cisco has not yet published every detail about how the scheduling algorithm works. For some of the older queuing tools, the details are published. Frankly, the details of how the scheduling code works inside IOS might be interesting, but it is not really necessary for a deep understanding of what a queuing tool does. For the QoS exams, you need to know what each queuing tool's scheduler accomplishes; for some tools, however, you also need to know the internals of how the scheduler actually works. When necessary, this book gives you plenty of details about the internal scheduling algorithms to help prepare you for the exams.

A final comment about the core concepts of queuing: The size of each packet does not affect the length of the queue, or how many packets it can hold. A queue of length 10 holds ten 1500-byte packets as easily as it holds ten 64-byte packets. Queues actually do not hold the packets themselves, but instead just hold pointers to the packets, whose contents are held in buffers.

Table 4-2 summarizes the key concepts of queuing. This table is used to compare the various queuing tools in the "Queuing Tools" section of this chapter.

Table 4-2 *Key Concepts When Comparing Queuing Tools*

Feature	Definition	QoS Characteristic Affected
Classification	The capability to examine packets to determine into which queue the packet should be placed. Many options are available.	None
Drop policy	When the queue has been determined, the drop policy defines the rules by which the router chooses to drop the packet. Tail drop, modified tail drop, WRED (Weighted Random Early Detect), and FRED (Flow-Based Random Early Detect) are the main options.	Loss
Scheduling inside a single queue	Inside a single queue, packets can be reordered. In most cases, however, FIFO logic is used for packets inside each queue.	Bandwidth, delay, jitter, and loss
Scheduling between different queues	The logic that defines how queuing chooses the next packet to take from the output queues and place it in the TX Queue (Transmit Queue).	Bandwidth, delay, jitter, and loss
Maximum number of queues	The maximum number of different queues the queuing tools support, which in turn implies the maximum number of traffic classifications that can be treated differently by the queuing method.	None
Maximum queue length	The maximum number of packets in a single queue.	Loss, delay

Output Queues, TX Rings, and TX Queues

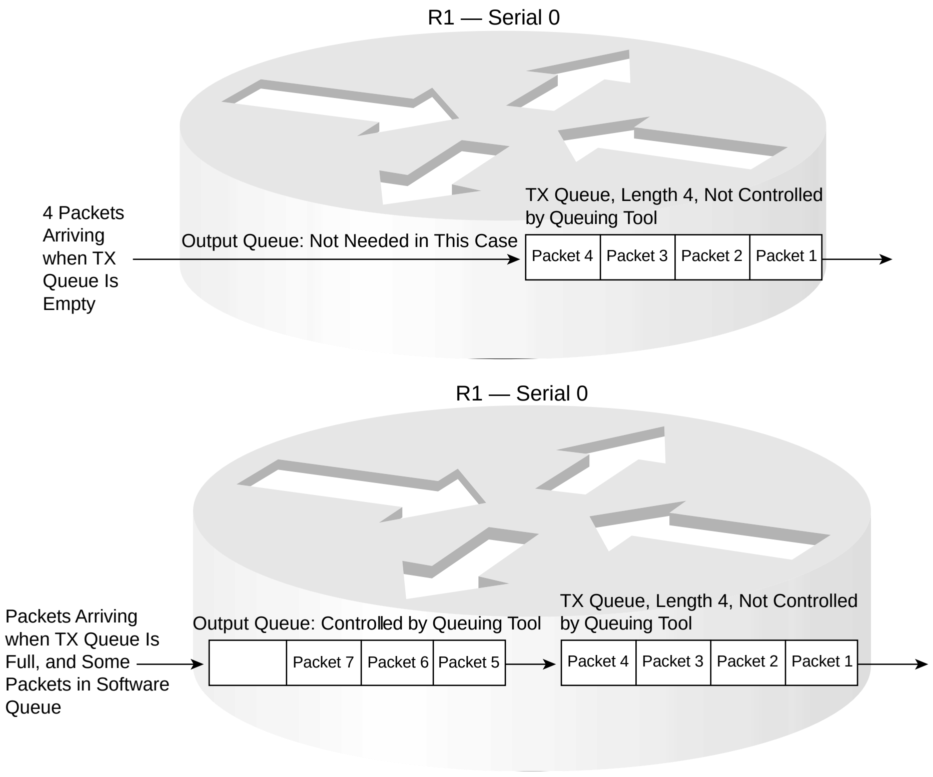
NOTE This section covers the concepts behind TX Rings and TX Queues. Interestingly, the DQOS exam 9E0-601 does not mention these topics at all, but the QoS exam does. For a true understanding of queuing inside IOS, however, you must understand these concepts. Therefore, for you DQOS exam takers, you might choose to just focus on the concepts about TX Queues and TX Rings, and not worry about memorization. As always, check the websites listed in the Introduction for any news about changes to the exams.

Packets do not move directly from an interface's output queue to being transmitted out of an interface. Instead, the router moves the packet from the interface output queue to another small FIFO queue on each interface. Cisco calls this separate, final queue either the *Transmit Queue* (TX Queue) or *Transmit Ring* (TX Ring), depending on the model of the router. Regardless of which name is used, for the purposes of this book, the TX Queue and TX Ring concepts can be considered to be equivalent. I will use the two terms synonymously.

The TX Ring's objective is to drive the link utilization to 100 percent when packets are waiting to exit an interface. The TX Queue holds outgoing packets so that the interface does not have to rely on the general-purpose processor in the router in order to start sending the next packet. The TX Queue can be accessed directly by the application-specific integrated circuits (ASICs) associated with an interface, so even if the general processor is busy, the interface can begin sending the next packet without waiting for the router CPU. Because the most constrained resource in a router is typically the bandwidth on the attached interfaces, particularly on WAN interfaces, the router hopes to always be able to send the next packet immediately when the interface finishes sending the last packet. The TX Queue provides a key component to reach that goal.

However, the existence of the TX Queue does impact queuing to some extent. Figure 4-3 depicts the TX Queue, along with a single FIFO Queuing output queue.

Figure 4-3 *Single FIFO Output Queue, with a Single TX Queue*


R1 — Serial 0
4 Packets Arriving when TX Queue Is Empty
Output Queue: Not Needed in This Case
TX Queue, Length 4, Not Controlled by Queuing Tool
Packet 4
Packet 3
Packet 2
Packet 1
R1 — Serial 0
Packets Arriving when TX Queue Is Full, and Some Packets in Software Queue
Output Queue: Controlled by Queuing Tool
Packet 7
Packet 6
Packet 5
TX Queue, Length 4, Not Controlled by Queuing Tool
Packet 4
Packet 3
Packet 2
Packet 1

Two different examples are outlined in Figure 4-3. In the top part of the figure, the scenario begins with no packets in the output queue, and no packets in the TX Queue. Then, four packets arrive. With a TX Queue with room for four packets, all four packets are placed into the TX Queue.

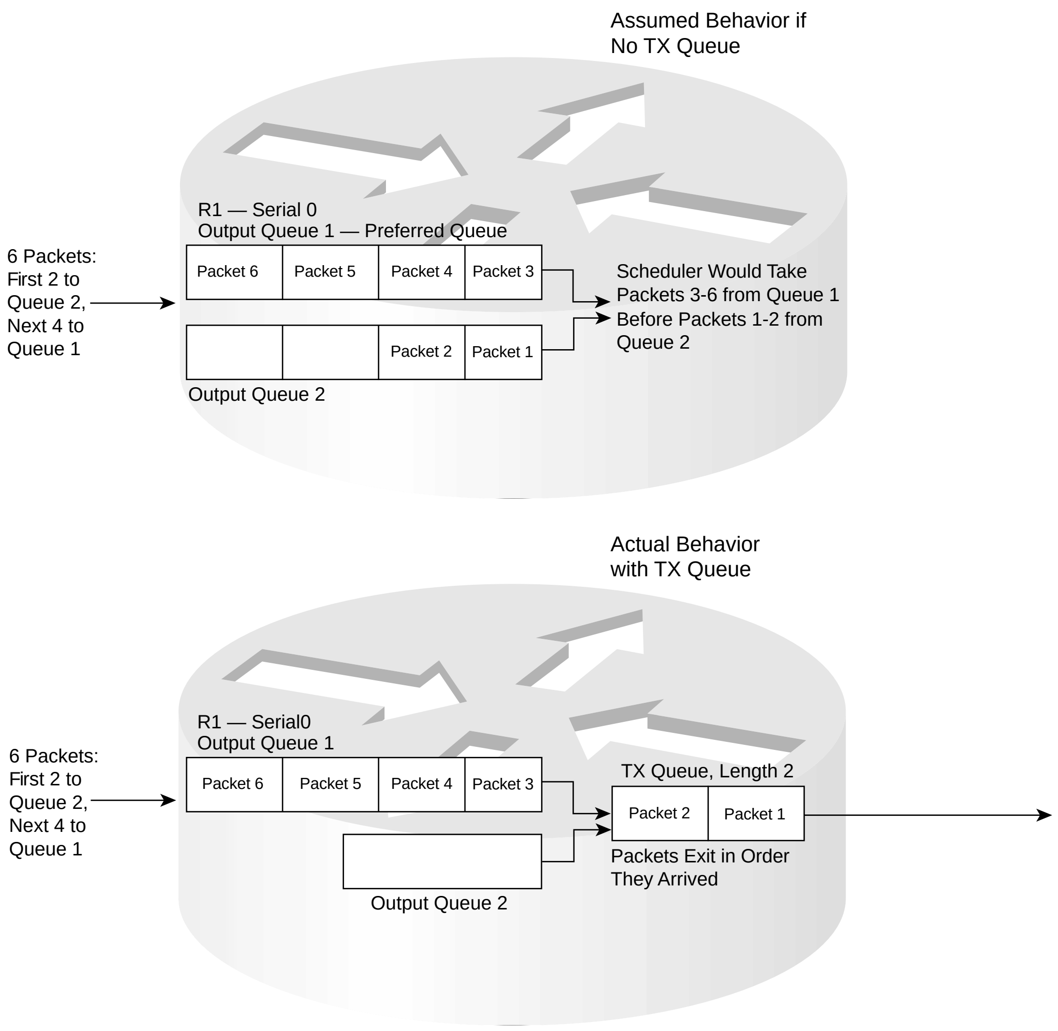
In the second example, any new packets that arrive will be placed into the software queue. Assuming that the software queue and the TX Queue were empty before the seven packets shown in the figure arrived, the first four packets would have been placed in the TX queue, and the last three in the software queue. So, any new packets will be placed at the end of the software queue.

All the queuing tools in IOS create and manage interface output queues, not interface TX Queues or TX Rings. Each interface uses one TX Queue (or TX Ring), and the TX Queue is a FIFO queue, unaffected by the queuing configuration on the interface. In Figure 4-3, the packets are sent in the same order that they would have been sent if the TX Queue did not exist. In some cases, however, the TX Queue impacts the results of the queuing scheduler. For instance, consider Figure 4-4, where queuing is configured with two queues. In this scenario, seven packets arrive, numbered in the order in which they arrive. The output queuing configuration specifies that the first two packets (1 and 2) should be placed into Queue 2, and the next four packets (numbered 3 through 6) should be placed into Queue 1.

Many people assume that the router behaves as shown in the top part of Figure 4-4, with the queuing scheduler determining the order in which packets exit the interface. In reality, IOS behaves as shown in the bottom half of Figure 4-4. In the top half of the figure, if all six packets were to arrive instantaneously, all six packets would be placed into the appropriate output queue. If this particular queuing tool's scheduler always serviced packets from Queue 1, and only serviced Queue 2 if Queue 1 was empty, the packets will leave in a different order than they arrived. In fact, packets 3 through 6 would exit in order, and then packets 1 and 2 would be sent. Ultimately, the order would just depend on the logic of the scheduling part of the queuing tool.

In this particular example, however, the packets would actually exit the interface in the same order that the packets were received because of the existence of the TX Queue. As mentioned earlier, when the router identifies the output interface for a packet, it checks the TX Queue for that interface. If space is available, the packet is placed in the TX Queue, and no output queuing is performed for that packet. In the example, because the scenario assumes that no other packets were waiting to exit R1's S0/0 interface before these six packets arrive, the first two packets are placed in the TX Queue. When packet 3 arrives, S0/0's TX Queue is full, so packets 3 through 6 are placed into an interface output queue, based on the queuing configuration for R1's S0/0 interface. The queuing classification logic places packets 3 through 6 into Queue 1. The router drains the packets in order from the TX Queue, and moves packets 3, 4, 5, and 6, in order, from Queue 1 into the TX Queue. The actual order that the packets exit S0 is the same order as they arrived.

Figure 4-4 *Output Queues, with Scheduler Always Servicing Queue 1 Rather Than Queue 2 When Packets Are in Queue 1*


Assumed Behavior if No TX Queue
R1 — Serial 0
Output Queue 1 — Preferred Queue
6 Packets: First 2 to Queue 2, Next 4 to Queue 1
Packet 6
Packet 5
Packet 4
Packet 3
Scheduler Would Take Packets 3-6 from Queue 1 Before Packets 1-2 from Queue 2
Packet 2
Packet 1
Output Queue 2
Actual Behavior with TX Queue
R1 — Serial0
Output Queue 1
6 Packets: First 2 to Queue 2, Next 4 to Queue 1
Packet 6
Packet 5
Packet 4
Packet 3
TX Queue, Length 2
Packet 2
Packet 1
Packets Exit in Order They Arrived
Output Queue 2

IOS automatically attempts to minimize the impact of the TX Queue to the IOS queuing tools. The IOS maintains the original goal of always having a packet in the TX Queue, available for the interface to immediately send when the interface completes sending the previous packet. When any form of output queuing tool is enabled on an interface, IOS reduces the size of the TX Queue to a very small value (typically 2). The smaller the value, the less impact the TX Queue has on the effects of the queuing method.

In some cases, you may want to change the setting for the size of the TX Queue or TX Ring. You may also want to know the current setting. Example 4-1 lists several commands that enable you to examine the size of the TX Queue, and change the size.

Example 4-1 *TX Queue Length: Finding the Length, and Changing the Length*

```
R3#show controllers serial 0/0
Interface Serial0/0
Hardware is PowerQUICC MPC860
DCE V.35, clock rate 1300000
idb at 0x8108F318, driver data structure at 0x81096D8C
SCC Registers:
General [GSMR]=0x2:0x00000030, Protocol-specific [PSMR]=0x8
Events [SCCE]=0x0000, Mask [SCCM]=0x001F, Status [SCCS]=0x06
Transmit on Demand [TODR]=0x0, Data Sync [DSR]=0x7E7E
Interrupt Registers:
Config [CICR]=0x00367F80, Pending [CIPR]=0x00008000
Mask   [CIMR]=0x40204000, In-srv  [CISR]=0x00000000
Command register [CR]=0x600
Port A [PADIR]=0x1100, [PAPAR]=0xFFFF
       [PAODR]=0x0000, [PADAT]=0xEFFF
Port B [PBDIR]=0x09C0F, [PBPAR]=0x0800E
       [PBODR]=0x0000E, [PBDAT]=0x3E77D
Port C [PCDIR]=0x00C, [PCPAR]=0x000
       [PCSO]=0xC20,  [PCDAT]=0xFC0, [PCINT]=0x00F
Receive Ring
        rmd(68012830): status 9000 length 1F address 3D3FC84
        rmd(68012838): status 9000 length 42 address 3D41D04
        rmd(68012840): status 9000 length F address 3D43D84
        rmd(68012848): status 9000 length 42 address 3D43084
        rmd(68012850): status 9000 length 42 address 3D3E904
        rmd(68012858): status 9000 length 157 address 3D43704
Transmit Ring
        tmd(680128B0): status 5C00 length 40 address 3C01114
        tmd(680128B8): status 5C00 length D address 3C00FD4
        tmd(680128C0): status 5C00 length 40 address 3C00FD4
        tmd(680128C8): status 5C00 length D address 3C00E94
        tmd(680128D0): status 5C00 length 11A address 3D6E394
        tmd(680128D8): status 5C00 length 40 address 3C019D4
        tmd(680128E0): status 5C00 length 40 address 3C01ED4
        tmd(680128E8): status 5C00 length D address 3D58BD4
        tmd(680128F0): status 5C00 length 40 address 3D58954
        tmd(680128F8): status 5C00 length 40 address 3D59214
        tmd(68012900): status 5C00 length D address 3D59494
        tmd(68012908): status 5C00 length 40 address 3D59AD4
        tmd(68012910): status 5C00 length 40 address 3C00214
        tmd(68012918): status 5C00 length D address 3C01C54
        tmd(68012920): status 5C00 length 40 address 3C005D4
        tmd(68012928): status 7C00 length 40 address 3C00714

tx_limited=0(16)
```

continues

Example 4-1 *TX Queue Length: Finding the Length, and Changing the Length (Continued)*

```
SCC GENERAL PARAMETER RAM (at 0x68013C00)
Rx BD Base [RBASE]=0x2830, Fn Code [RFCR]=0x18
Tx BD Base [TBASE]=0x28B0, Fn Code [TFCR]=0x18
Max Rx Buff Len [MRBLR]=1548
Rx State [RSTATE]=0x18008440, BD Ptr [RBPTR]=0x2840
Tx State [TSTATE]=0x18000548, BD Ptr [TBPTR]=0x28C8

SCC HDLC PARAMETER RAM (at 0x68013C38)
CRC Preset [C_PRES]=0xFFFF, Mask [C_MASK]=0xF0B8
Errors: CRC [CRCEC]=0, Aborts [ABTSC]=0, Discards [DISFC]=0
Nonmatch Addr Cntr [NMARC]=0
Retry Count [RETRC]=0
Max Frame Length [MFLR]=1608
Rx Int Threshold [RFTHR]=0, Frame Cnt [RFCNT]=65454
User-defined Address 0000/0000/0000/0000
User-defined Address Mask 0x0000

buffer size 1524

PowerQUICC SCC specific errors:
0 input aborts on receiving flag sequence
0 throttles, 0 enables
0 overruns
0 transmitter underruns
0 transmitter CTS losts
0 aborted short frames
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int s 0/0
R3(config-if)#priority-group 1
R3(config-if)#^Z
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
R3#show controllers serial 0/0
01:03:09: %SYS-5-CONFIG_I: Configured from console by console
Interface Serial0/0
!!!!! Lines omitted to save space
tx_limited=1(2)
!!!!! Lines omitted to save space
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int s 0/0
R3(config-if)#no priority-group 1
R3(config-if)#tx-ring-limit 1
R3(config-if)#^Z
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
R3#show controllers serial 0/0
Interface Serial0/0
! Lines omitted to save space
```

continues

Example 4-1 *TX Queue Length: Finding the Length, and Changing the Length (Continued)*

```
tx_limited=0(1)

! Lines omitted to save space
```

The **show controllers serial 0/0** command lists the size of the TX Queue or TX Ring. In the shaded output in Example 4-1, the phrase "tx_limited=0(16)" means that the TX Ring holds 16 packets. The zero means that the queue size is not currently limited due to a queuing tool being enabled on the interface. For the first instance of **show controllers**, no queuing method is enabled on the interface, so a zero signifies that the size of the TX Ring has not been limited automatically. After enabling Priority Queuing with the **priority-group** interface subcommand, the next **show controllers** command lists "tx_limited=1(2)." The new length of the TX Ring is 2, and 1 shows that the length is automatically limited as a result of queuing being configured. Next, Priority Queuing is disabled with the **no priority-group** interface subcommand, but the length of the TX Ring is explicitly defined with the **tx-ring-limit 1** interface subcommand. On the final **show controllers** command, the "tx_limited=0(1)" output implies that the size is not limited, because no queuing is enabled, but that the length of the TX Ring is 1.

The following list summarizes the key points about TX Rings and TX Queues in relation to their effect on queuing:

- The TX Queue/TX Ring always performs FIFO scheduling, and cannot be changed.
- The TX Queue/TX Ring uses a single queue, per interface.
- IOS shortens the interface TX Queue/TX Ring automatically when an output queuing method is configured.
- The TX Ring/TX queue length can be configured to a different value.

Queuing on Interfaces Versus Subinterfaces and Virtual Circuits (VCs)

IOS queuing tools create and manage output queues associated with an interface, and then the packets drain into the TX Ring/Queue associated with the interface. IOS also supports queuing on subinterfaces and individual VCs when traffic shaping is also enabled. Shaping queues, created by the traffic-shaping feature, drain into the interface output queues, which then drain into the TX Ring/Queue. Like the interface output queues, the shaping queues can be managed with IOS queuing tools.

The interaction between shaping queues associated with a subinterface or VC, and queues associated with a physical interface, is not obvious at first glance. So, before moving into the details of the various queuing tools, consider what happens on subinterfaces, VCs, and physical interfaces so that you can make good choices about how to enable queuing in a router.

Figure 4-5 provides a reasonable backdrop from which to explain the interaction between queues. R1 has many permanent virtual circuits (PVCs) exiting its S0/0 physical interface. The figure shows queues associated with two of the PVCs, a single queue for the physical interface, and the TX Ring for the interface.

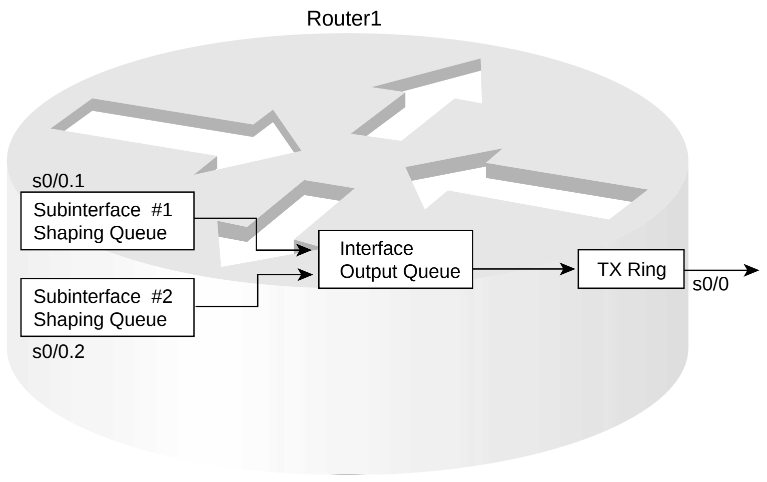


Figure 4-5 *Subinterface Shaping Queues, Interface Queues, and TX Ring*

In this particular example, each subinterface uses a single FIFO shaping queue; the physical interface uses a FIFO output queue. At first glance, it seems simple enough: A packet arrives, and the forwarding decision dictates that the packet should exit subinterface S0/0.1. It is placed into the subinterface 0/0.1 shaping queue, and then into the physical interface output queue, and then into the TX Ring. Then, it exits the interface.

In some cases, the packet moves from the shaping queues directly to the TX Queue. You may recall that packets are not even placed in the output queue if the TX Ring is not full! If no congestion occurs on the interface, the TX Ring does not fill. If no congestion occurs in the TX Ring, the interface output queue does not fill, and the queuing tool enabled on the interface has no effect on the packets exiting the interface.

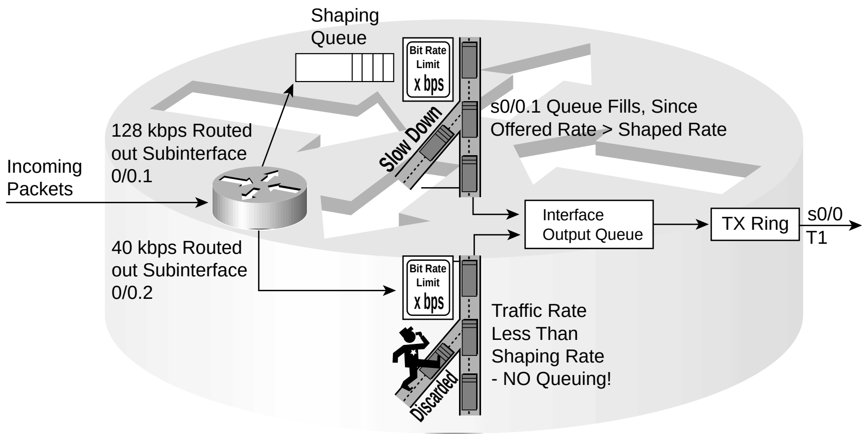
In some cases, IOS does not place the packets into a shaping queue as they arrive, but instead the packets are placed into the interface queue or TX Queue. When the shaping features knows that a newly arrived packet does not exceed the shaping rate, there is no need to delay the packet. In that case, a queuing tool used for managing the shaping queue would also have no effect.

Traffic shaping can cause subinterface shaping queues to fill, even when there is no congestion on the physical interface. Traffic shaping, enabled on a subinterface or VC, slows down the flow of traffic leaving the subinterface or VC. In effect, traffic shaping on the subinterface creates congestion between the shaping queues and the physical interface queues. On a physical interface, packets can only leave the interface at the physical clock rate used by the interface; similarly, packets can only leave a shaping queue at the traffic-shaping rate.

For example, the VC associated with subinterface S0/0.1 uses a 64 kbps committed information rate (CIR), and S0/0 uses a T/1 circuit. Without traffic shaping, more than 64 kbps of traffic could be sent for that PVC, and the only constraining factor would be the access rate (T/1). The Frame Relay network might discard some of the traffic, because the router may send more (up to 1.5 Mbps) on the VC, exceeding the traffic contract (64-kbps CIR). So, traffic shaping could be enabled on the subinterface or VC, restricting the overall rate for this PVC to 64 kbps, to avoid frame loss inside the Frame Relay network. If the offered load of traffic on the subinterface exceeds 64 kbps for some period, traffic shaping delays sending the excess traffic by placing the packets into the shaping queue associated with the subinterface, and draining the traffic from the shaping queue at the shaped rate.

Figure 4-6 shows an updated version of Figure 4-5; this one's PVC is currently exceeding the shaping rate, and the other PVC is not exceeding the shaping rate.

Figure 4-6 *Shaping Active on One VC, and Not Active on the Other*


Shaping Queue
Bit Rate Limit x bps
Slow Down
s0/0.1 Queue Fills, Since Offered Rate > Shaped Rate
128 kbps Routed out Subinterface 0/0.1
Incoming Packets
Interface Output Queue
TX Ring
s0/0 T1
40 kbps Routed out Subinterface 0/0.2
Bit Rate Limit x bps
Traffic Rate Less Than Shaping Rate - NO Queuing!
Discarded

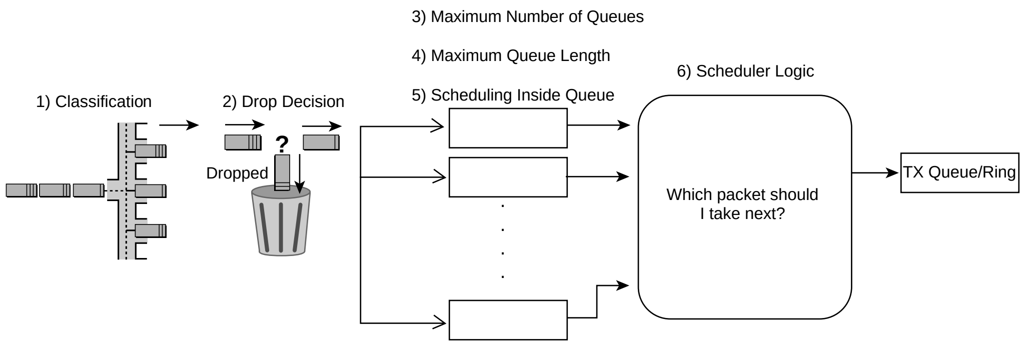
In Figure 4-6, packets arrive and are routed out of each of the two subinterfaces. Traffic for subinterface 0/0.1 exceeds the shaping rate, and packets for subinterface 0/0.2 do not. Therefore, a queue begins to form on the shaping queue for subinterface 0/0.1, because traffic shaping delays packets by queuing the packets. On subinterface 0/0.2, packets will not be enqueued into the shaping queue, because the shaping rate has not been exceeded.

You can configure queuing tools to manipulate the output queue on a physical interface, as well as the shaping queue created by shaping. The concepts in this chapter apply to using queuing on both the main interface, and on any shaping queues. However, this chapter only covers the configuration of queuing to manipulate the interface output queues. Chapter 5, "Traffic Policing and Shaping," which covers traffic shaping, explains how to configure queuing for use on shaping queues. When reading the next chapter, keep these queuing concepts in mind and watch for the details of how to enable your favorite queuing tools for shaping queues.

Summary of Queuing Concepts

For the remainder of this chapter, queuing tools are compared based on the six general points listed in this section. Figure 4-7 outlines these points in the same sequence that each point is listed in the upcoming sections on each queuing tool.

Figure 4-7 *Six Comparison Points for IOS Queuing Tools*



Scheduling gets the most attention when network engineers choose which queuing tool to use for a particular application. However, the other components of queuing are important as well. If the classification part of a queuing tool cannot classify the traffic as defined in the QoS policy for the network, for instance, either the policy must be changed or another tool must be used. For instance, PQ and CQ cannot take direct advantage of network-based application recognition (NBAR), but CBWFQ and LLQ can. In addition, some queuing tools allow a drop policy for each queue, which becomes particularly important when voice and video compete with data traffic in a converged network.

Queuing Tools

Cisco IOS provides a large variety of different queuing tools. Some of the queuing tools have been available for quite some time, whereas others have only become available in recent releases. Some of the tools may not be popular to implement because one queuing tool may provide a better solution than another queuing tool. However, all the queuing tools mentioned in this chapter are covered on one of the current Cisco QoS exams; therefore, we cover them in this book.

The following sections of this book include coverage of First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

FIFO Queuing

NOTE This section covers FIFO Queuing, which like TX Queues, is not currently covered on the DQOS 9E0-601 exam. This section does cover some useful information for all people interested in QoS. Therefore, for you DQOS exam takers, you might choose to just focus on the concepts about FIFO, and not worry about memorization or configuration. And as always, check the websites listed in the Introduction for any news about changes to the exams.

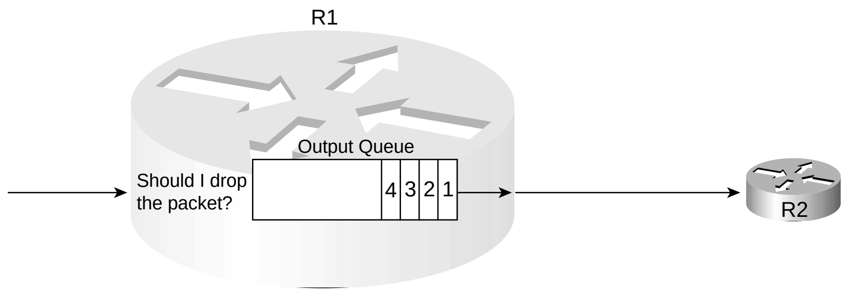
The first reason that a router or switch needs output queues is to hold a packet while waiting for the interface to become available for sending the packet. Whereas the other queuing tools in this chapter also perform other functions, like reordering packets, FIFO Queuing just provides a means to hold packets while they are waiting to exit an interface.

FIFO Queuing does not need the two most interesting features of the other queuing tools, namely classification and scheduling. FIFO Queuing uses a single queue for the interface. Because there is only one queue, there is no need for classification to decide the queue into which the packet should be placed. Also there is no need for scheduling logic to pick which queue from which to take the next packet. The only really interesting part of FIFO Queuing is the queue length, which is configurable, and how the queue length affects delay and loss.

FIFO Queuing uses tail drop to decide when to drop or enqueue packets. If you configure a longer FIFO queue, more packets can be in the queue, which means that the queue will be less likely to fill. If the queue is less likely to fill, fewer packets will be dropped. However, with a longer queue, packets may experience more delay and jitter. With a shorter queue, less delay occurs, but the single FIFO queue fills more quickly, which in turn causes more tail drops of new packets. These facts are true for any queuing method, including FIFO.

Figure 4-8 outlines simple FIFO Queuing. R1 uses FIFO Queuing on the interface connected to R2. The only decision required when configuring FIFO Queuing is whether to change the length of the queue.

Figure 4-8 *Simple FIFO Queuing*



Remember to consider two steps when configuring FIFO Queuing. First, configuring FIFO Queuing actually requires you to turn off all other types of queuing, as opposed to just configuring FIFO Queuing. Cisco IOS uses WFQ as the default queuing method on serial interfaces running at E1 speeds and slower. However, IOS does not supply a command to enable FIFO Queuing; to enable FIFO Queuing, you must first disable WFQ by using the **no fair-queue** interface subcommand. If other queuing tools have been explicitly configured, you should also disable these. Just by removing all other queuing configuration from an interface, you have enabled FIFO!

The second FIFO configuration step that you might consider is to override the default queue length. To do so, use the **hold-queue x out** interface subcommand to reset the length of the queue.

Example 4-2 shows a sample FIFO Queuing configuration.

Example 4-2 *FIFO Queuing Configuration*

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int s 0/0
R3(config-if)#no fair-queue
R3(config-if)#^Z
R3#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent 80, LMI stat recvd 73, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent 0, LMI upd sent 0
  LMI DLCI 1023 LMI type is CISCO frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 171/2, interface broadcasts 155
  Last input 00:00:02, output 00:00:03, output hang never
  Last clearing of "show interface" counters 00:13:48
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue :0/40 (size/max)
  30 second input rate 0 bits/sec, 0 packets/sec
  30 second output rate 0 bits/sec, 0 packets/sec
     235 packets input, 14654 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     2 input errors, 0 CRC, 2 frame, 0 overrun, 0 ignored, 0 abort
     264 packets output, 15881 bytes, 0 underruns
     0 output errors, 0 collisions, 6 interface resets
     0 output buffer failures, 0 output buffers swapped out
     10 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

continues

Example 4-2 *FIFO Queuing Configuration (Continued)*

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int s 0/0
R3(config-if)#hold-queue 50 out
R3(config-if)#^Z
!
R3#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
! Lines omitted for brevity
  Queueing strategy: fifo
  Output queue :0/50 (size/max)
! Line omitted for brevity
```

Example 4-2 shows FIFO Queuing being configured by removing the default WFQ configuration with the **no fair-queue** command. The **show interface** command lists the fact that FIFO Queuing is used, and the output queue has 40 entries maximum. After configuring the output queue to hold 50 packets with the **hold-queue 50 out** command, the **show interface** output still lists FIFO Queuing, but now with a maximum queue size of 50.

FIFO Queuing is pretty basic, but it does provide a useful function: It provides the basic queuing function of holding packets until the interface is no longer busy.

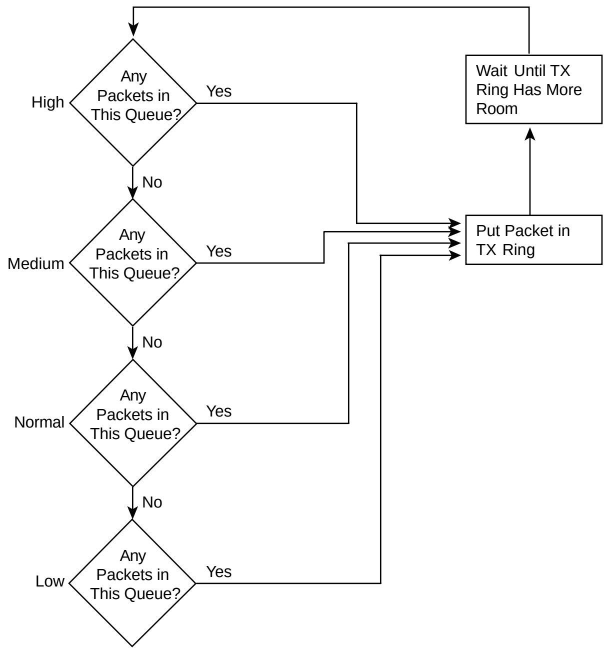
Priority Queuing

Priority Queuing's most distinctive feature is its scheduler. PQ schedules traffic such that the higher-priority queues always get serviced, with the side effect of starving the lower-priority queues. With a maximum of four queues, called High, Medium, Normal, and Low, the complete logic of the scheduler can be easily represented, as is shown in Figure 4-9.

As seen in Figure 4-9, if the High queue always has a packet waiting, the scheduler will always take the packets in the High queue. If the High queue does not have a packet waiting, but the Medium queue does, one packet is taken from the Medium queue—and then the process always starts over at the High queue. The Low queue only gets serviced if the High, Medium, and Normal queues do not have any packets waiting.

The PQ scheduler has some obvious benefits and drawbacks. Packets in the High queue can claim 100 percent of the link bandwidth, with minimal delay, and minimal jitter. The lower queues suffer, however. In fact, when congested, packets in the lower queues take significantly longer to be serviced than under lighter loads. In fact, when the link is congested, user applications may stop working if their packets are placed into lower-priority queues.

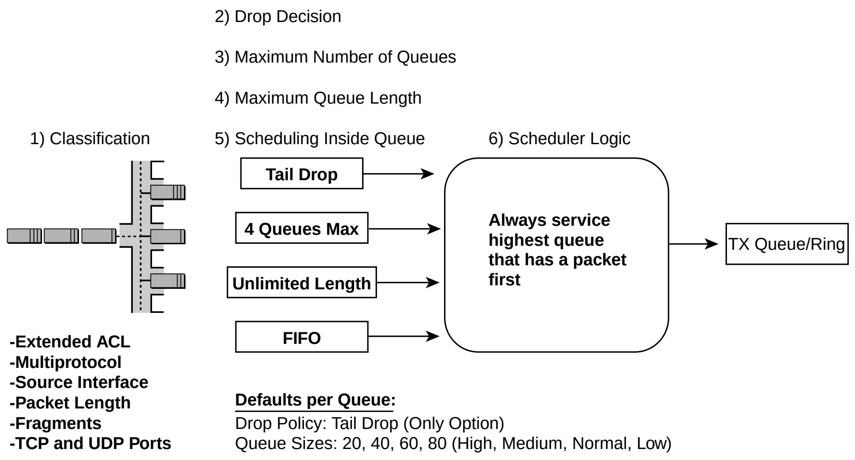
Figure 4-9 *PQ Scheduling Logic*


High
Any Packets in This Queue?
Yes
No
Medium
Any Packets in This Queue?
Yes
No
Normal
Any Packets in This Queue?
Yes
No
Low
Any Packets in This Queue?
Yes
Put Packet in TX Ring
Wait Until TX Ring Has More Room

Most of the rest of the details about PQ can be easily understood. PQ classifies packets based on the content of the packet headers. It uses a maximum of four queues, as mentioned earlier. The only drop policy is tail drop—in other words, after classifying the packet, if the appropriate queue is full, the packet is dropped. The length of each queue, which of course affects packet loss and delay, can be changed—in fact, PQ can set the queue length to a value of zero, which means the queue length is infinite. (Infinite really means that when the router runs out of memory, the packet cannot be queued; however, you have worse problems than queuing a packet if the router is out of memory!) Figure 4-10 summarizes these key features of PQ.

The figure represents the internals of a router, after the routing decision has identified the output interface for the packet. The following list describes each component of the queuing process, with the numbers in the list matching the numbers in the figure.

Figure 4-10 *PQ Features*



1. PQ can classify packets using access-control lists (ACLs) for most Layer 3 protocols, matching anything allowed by any of the types of ACLs. PQ can also directly match, without using an ACL, the incoming interface, packet length, and TCP and UDP port numbers.
2. Tail drop is the only available drop policy.
3. Four queues maximum
4. Maximum queue length can be set to zero, which means the queue has theoretically infinite length. Defaults are 20, 40, 60, and 80 packets for High, Medium, Normal, and Low queues, respectively.
5. Inside a queue, PQ uses FIFO logic.
6. When scheduling among the queues, PQ always services the highest-priority queue.

Independent of the process leading up to placing the packet into the queue, the scheduler continually reacts when the TX Ring empties a packet out the interface, implying that there is now more room in the TX Ring/TX Queue for another packet. When the TX Ring frees space, the PQ scheduler then performs the logic described in Figure 4-9, taking a packet from the highest-priority queue that has a packet waiting. The PQ scheduler moves the packet to the TX Ring, for later transmission on the interface.

As with other queuing tools, none of this work happens if the interface is not congested. When the interface is not congested (in other words, the TX Ring is not full), new packets are placed into the TX Ring directly. When the TX Ring fills, PQ performs queuing. When all the PQ queues drain, as well as the TX Ring drains, congestion has abated. Newly arriving packets would then be placed directly into the TX Ring, until it fills again, which in turn restarts the queuing process with PQ.

PQ works great for QoS policies that need to treat one type of traffic with the absolute best service possible. It has been around since IOS 10.0. However, PQ's service for the lower queues degrades quickly, making PQ impractical for most applications today. For instance, even running one FTP connection, one web browser, one NetMeeting call, and two VoIP calls when creating the output for this section of the book, the TCP connections for the FTP and HTTP traffic frequently timed out.

Table 4-3 summarizes some of the key functions and features of PQ. For those of you pursuing the QoS exam, look at Appendix B for details of how to configure PQ.

Table 4-3 *PQ Functions and Features*

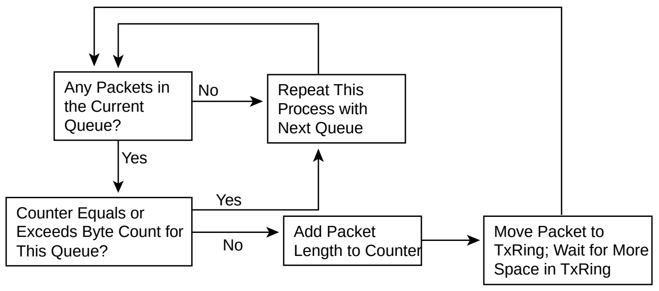
PQ Feature	Explanation
Classification	Classifies based on matching an ACL for all Layer 3 protocols, incoming interface, packet size, whether the packet is a fragment, and TCP and UDP port numbers.
Drop policy	Tail drop.
Maximum number of queues	4.
Maximum queue length	Infinite; really means that packets will not be tail dropped, but will be queued.
Scheduling inside a single queue	FIFO.
Scheduling among all queues	Always service higher-priority queues first; result is great service for the High queue, with potential for 100% of link bandwidth. Service degrades quickly for lower-priority queues.

Custom Queuing

Custom Queuing (CQ) followed PQ as the next IOS queuing tool added to IOS. CQ addresses the biggest drawback of PQ by providing a queuing tool that does service all queues, even during times of congestion. It has 16 queues available, implying 16 classification categories, which is plenty for most applications. The negative part of CQ, as compared to PQ, is that CQ's scheduler does not have an option to always service one queue first—like PQ's High queue—so CQ does not provide great service for delay- and jitter-sensitive traffic.

As with most queuing tools, the most interesting part of the tool is the scheduler. The CQ scheduler gives an approximate percentage of overall link bandwidth to each queue. CQ approximates the bandwidth percentages, as opposed to meeting an exact percentage, due to the simple operation of the CQ scheduler. Figure 4-11 depicts the CQ scheduler logic.

Figure 4-11 *CQ Scheduling Logic for Current Queue*


Any Packets in the Current Queue?
No
Repeat This Process with Next Queue
Yes
Yes
Counter Equals or Exceeds Byte Count for This Queue?
No
Add Packet Length to Counter
Move Packet to TxRing; Wait for More Space in TxRing

The CQ scheduler performs round-robin service on each queue, beginning with Queue 1. CQ takes packets from the queue, until the total byte count specified for the queue has been met or exceeded. After the queue has been serviced for that many bytes, or the queue does not have any more packets, CQ moves on to the next queue, and repeats the process.

CQ does not configure the exact link bandwidth percentage, but rather it configures the number of bytes taken from each queue during each round-robin pass through the queues. Suppose, for example, that an engineer configures CQ to use five queues. The engineer assigns a byte count of 10000 bytes for each queue. With this configuration, the engineer has reserved 20 percent of the link bandwidth for each queue. (If each queue sends 10000 bytes, a total of 50000 bytes are sent per cycle, so each queue sends 10000/50000 of the bytes out of the interface, or 20 percent.) If instead the engineer has assigned byte counts of 5000 bytes for the first 2 queues, 10000 for the next 2 queues, and 20000 for the fifth queue, the total bytes sent in each pass through the queues would again total 50000 bytes. Therefore, Queues 1 and 2 would get 5000/50000, or 10 percent of the link bandwidth. Queues 3 and 4 would get 10000/50000, or 20 percent of the bandwidth, and Queue 5 would get 20000/50000, or 40 percent. The following formula calculates the implied bandwidth percentage for Queue *x*:

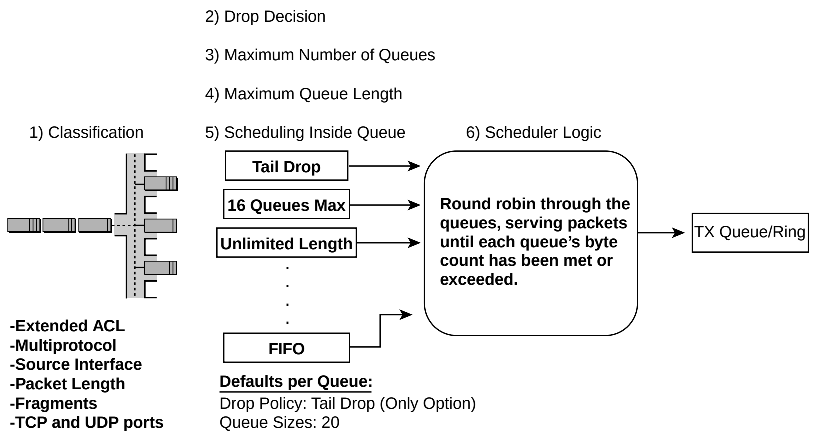
$$\frac{\text{Byte count for Queue } x}{\text{Sum of byte counts for all queues}}$$

The CQ scheduler essentially guarantees the minimum bandwidth for each queue, while allowing queues to have more bandwidth under the right conditions. Imagine that 5 queues have been configured with the byte counts of 5000, 5000, 10000, 10000, and 20000 for queues 1 through 5, respectively. If all 5 queues have plenty of packets to send, the percentage bandwidth given to each queue is 10 percent, 10 percent, 20 percent, 20 percent, and 40 percent, as described earlier. However, suppose that Queue 4 has no traffic over some short period of time. For that period, when the CQ scheduler tries to service Queue 4, it notices that no packets are waiting. The CQ scheduler moves immediately to the next queue. Over this short period of time, only Queues 1 through 3 and Queue 5 have packets waiting. In this case, the queues would receive 12.5 percent, 12.5 percent, 25 percent, 0 percent, and 50 percent of link bandwidth, respectively. (The math to get these percentages is number-of-bytes-per-cycle/40,000 because around 40,000 bytes should be taken from the four active queues per cycle.) Note also that queues that have not been configured are automatically skipped.

Unlike PQ, CQ does not name the queues, but it numbers the queues 1 through 16. No single queue has a better treatment by the scheduler than another, other than the number of bytes serviced for each queue. So, in the example in the last several paragraphs, Queue 5, with 20000 bytes serviced on each turn, might be considered to be the “best” queue with this configuration. Do not be fooled by that assumption! If the traffic classified into Queue 5 comprises 80 percent of the offered traffic, the traffic in Queue 5 may get the worst treatment among all 5 queues. And of course, the traffic patterns will change over short periods of time, and over long periods. Therefore, whereas understanding the scheduler logic is pretty easy, choosing the actual numbers requires some traffic analysis, and good guessing to some degree.

The rest of the details about CQ mirror the features in PQ. Figure 4-12 highlights the features of CQ.

Figure 4-12 *CQ Features*



The figure represents the internals of a router, after the routing decision has identified the output interface for the packet. The following list describes each component of the queuing process, with the numbers in the list matching the numbers in the figure:

1. CQ can classify packets using access-control lists (ACLs) for most Layer 3 protocols, matching anything allowed by any of the types of ACLs. CQ can also directly match, without using an ACL, the incoming interface, packet length, and TCP and UDP port numbers. CQ and PQ use the exact same classification options.

2. Tail drop is the only available drop policy.
3. Sixteen queues maximum.
4. You can set the queue length to zero, which means the queue has theoretically infinite length. Default is 20 per queue.
5. Inside a queue, CQ uses FIFO logic.
6. When scheduling among the queues, CQ services the configured byte count per queue during a continuous round-robin through the queues.

CQ works great for networks with no jitter-sensitive applications that also need to reserve general percentages of link bandwidth for different classes of traffic. CQ does not provide low-latency service, like PQ's High queue does, although it does allow each queue to receive some service when the link is congested. Table 4-4 summarizes some of the key features of CQ. For those of you pursuing the QoS exam, look at Appendix B for details of how to configure CQ.

Table 4-4 *CQ Functions and Features*

CQ Feature	Explanation
Classification	Classifies based on matching an ACL for all Layer 3 protocols, incoming interface, packet size, whether the packet is a fragment, and TCP and UDP port numbers.
Drop policy	Tail drop.
Number of queues	16.
Maximum queue length	Infinite; really means that packets will not be tail dropped, but will be queued.
Scheduling inside a single queue	FIFO.
Scheduling among all queues	Services packets from a queue until a byte count is reached; round-robins through the queues, servicing the different byte counts for each queue. The effect is to reserve a percentage of link bandwidth for each queue.

Weighted Fair Queuing (WFQ)

Weighted Fair Queuing differs from PQ and CQ in several significant ways. The first and most obvious difference is that WFQ does not allow classification options to be configured! WFQ classifies packets based on flows. A *flow* consists of all packets that have the same source and destination IP address, and the same source and destination port numbers. So, no explicit matching is configured. The other large difference between WFQ versus PQ and CQ is the scheduler, which simply favors low-volume, higher-precedence flows over large-volume, lower-precedence flows. Also because WFQ is flow based, and each flow uses a different queue, the number of queues becomes rather large—up to a maximum of 4096 queues per interface. And although WFQ uses tail drop, it really uses a slightly modified tail-drop scheme—yet another difference.

IOS provides several variations of WFQ as well. This section concentrates on WFQ, whose longer, more descriptive name is Flow-Based WFQ. For those of you taking the QoS exam, Appendix B covers several variations of WFQ—namely, distributed WFQ (dWFQ), ToS-based dWFQ, and QoS group-based dWFQ. Each has slight variations to the baseline concepts and configuration of WFQ.

Ironically, WFQ requires the least configuration of all the queuing tools in this chapter, yet it requires the most explanation to achieve a deep understanding. The extra work to read through the details will certainly help on the exam, plus it will give you a better appreciation for WFQ, which may be the most pervasively deployed QoS tool in Cisco routers.

WFQ Classification

Flow-Based WFQ, or just WFQ, classifies traffic into flows. Flows are identified by at least five items in an IP packet:

- Source IP address
- Destination IP ADDRESS
- Transport layer protocol (TCP or UDP) as defined by the IP Protocol header field
- TCP or UDP source port
- TCP or UDP destination port

Depending on what document you read, WFQ also classifies based on the ToS byte. In particular, the CCIP QoS exam expects WFQ to classify based on the ToS byte as well as the five items listed previously. The DQOS course, and presumably the DQOS exam, claims that it classifies on the five fields listed previously. Most documentation just lists the five fields in the preceding list. (As with all items that may change with later releases of the exams and courses, look to www.ciscopress.com/1587200589 for the latest information.)

Whether WFQ uses the ToS byte or not when classifying packets, practically speaking, does not matter much. Good design suggests that packets in a single flow ought to have their Precedence or DSCP field set to the same value—so the same packets would get classified into the same flow, regardless of whether WFQ cares about the ToS byte or not for classification.

The term “flow” can have a couple of different meanings. For instance, imagine a PC that is downloading a web page. The user sees the page appear, reads the page for 10 seconds, and clicks a button. A second web page appears, the user reads the page for 10 seconds, and clicks another button. All the pages and objects came from a single web server, and all the pages and objects were loaded using a single TCP connection between the PC and the server. How many different combinations of source/destination, address/port, and transport layer protocol, are used? How many different flows?

From a commonsense perspective, only one flow exists in this example, because only one TCP connection is used. From WFQ's perspective, no flows may have occurred, or three flows existed, and possibly even more. To most people, a single TCP flow exists as long as the TCP connection stays up, because the packets in that connection always have the same source address, source port, destination address, and destination port information. However, WFQ considers a flow to exist only as long as packets from that flow are queued to be sent out of an interface. For instance, while the user is reading the web pages for 10 seconds, the routers finish sending all packets sent by the web server, so the queue for that flow is empty. Because the intermediate routers had no packets queued in the queue for that flow, WFQ removes the flow. Similarly, even while transferring different objects that comprise a web page, if WFQ empties a flow's queue, it removes the queue, because it is no longer needed.

Why does it matter that flows come and go quickly from WFQ's perspective? With class-based schemes, you always know how many queues you have, and you can see some basic statistics for each queue. With WFQ, the number of flows, and therefore the number of queues, changes very quickly. Although you can see statistics about active flows, you can bet on the information changing before you can type the **show queue** command again. The statistics show you information about the short-lived flow—for instance, when downloading the third web page in the previous example, the **show queue** command tells you about WFQ's view of the flow, which began when the third web page was being transferred, not when the TCP connection was formed.

WFQ Scheduler: The Net Effect

Cisco publishes information about how the WFQ scheduler works. Even with an understanding of how the scheduler works, however, the true goals behind the scheduler are not obvious. This section reflects on what WFQ provides, and the following sections describe how WFQ accomplishes the task.

The WFQ scheduler has two main goals. The first is to provide fairness among the currently existing flows. To provide fairness, WFQ gives each flow an equal amount of bandwidth. If 10 flows exist for an interface, and the clock rate is 128 kbps, each flow effectively gets 12.8 kbps. If 100 flows exist, each flow gets 1.28 kbps. In some ways, this goal is similar to a time-division multiplexing (TDM) system, but the number of time slots is not preset, but instead based on the number of flows currently exiting an interface. Also keep in mind that the concept of equal shares of bandwidth for each flow is a goal—for example, the actual scheduler logic used to accomplish this goal is much different from the bandwidth reservation using byte counts with CQ.

With each flow receiving its fair share of the link bandwidth, the lower-volume flows prosper, and the higher-volume flows suffer. Think of that 128-kbps link again, for instance, with 10 flows. If Flow 1 needs 5 kbps, and WFQ allows 12.8 kbps per flow, the queue associated with Flow 1 may never have more than a few packets in it, because the packets will drain quickly. If Flow 2 needs 30 kbps, then packets will back up in Flow 2's queue, because WFQ only gives this queue 12.8 kbps as well. These packets experience more delay and jitter, and possibly loss if the queue fills. Of course, if Flow 1 only needs 5 kbps, the actual WFQ scheduler allows other flows to use the extra bandwidth.

The second goal of the WFQ scheduler is to provide more bandwidth to flows with higher IP precedence values. The preference of higher-precedence flows is implied in the name—"Weighted" implies that the fair share is weighted, and it is weighted based on precedence. With 10 flows on a 128-kbps link, for example, if 5 of the flows use precedence 0, and 5 use precedence 1, WFQ might want to give the precedence 1 flows twice as much bandwidth as the precedence 0 flows. Therefore, 5 precedence 0 flows would receive roughly 8.5 kbps each, and 5 precedence 1 flows would receive roughly 17 kbps each. In fact, WFQ provides a fair share roughly based on the ratio of each flow's precedence, plus one. In other words, precedence 7 flows get 8 times more bandwidth than does precedence 0 flows, because (7 + 1) / (0 + 1) = 8. If you compare precedence 3 to precedence 0, the ratio is roughly (3 + 1) / (0 + 1) = 4.

So, what does WFQ accomplish? Ignoring precedence for a moment, the short answer is lower-volume flows get relatively better service, and higher-volume flows get worse service. Higher-precedence flows get better service than lower-precedence flows. If lower-volume flows are given higher-precedence values, the bandwidth/delay/jitter/loss characteristics improve even more. In a network where most of the delay-sensitive traffic is lower-volume traffic, WFQ is a great solution. It takes one command to enable it, and it is already enabled by default! Its default behavior favors lower-volume flows, which may be the more important flows. In fact, WFQ came out when many networks' most important interactive flows were Telnet and Systems Network Architecture (SNA) encapsulated in IP. These types of flows used much less volume than other flows, so WFQ provided a great default, without having to worry about how to perform prioritization on encapsulated SNA traffic.

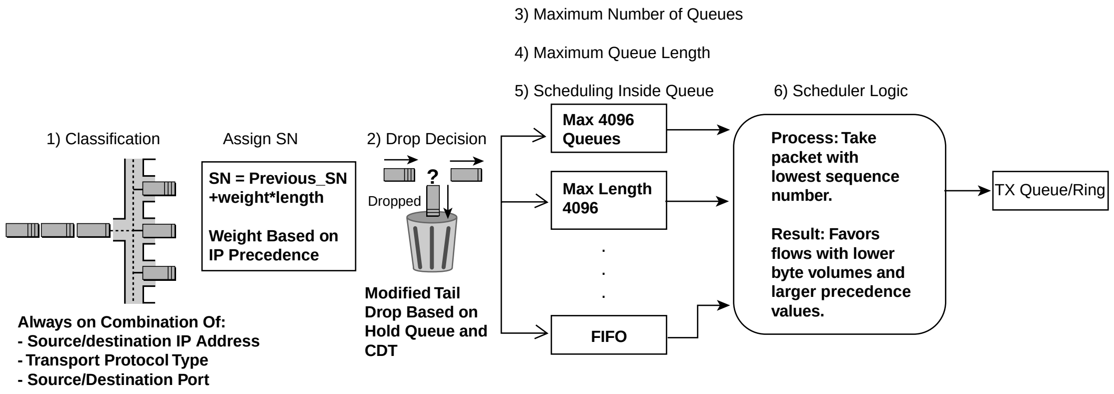
WFQ Scheduling: The Process

WFQ gives each flow a weighted percentage of link bandwidth. However, WFQ does not predefine queues like class-based queuing tools do, because WFQ dynamically classifies queues based on the flow details. And although WFQ ends up causing each flow to get some percentage of link bandwidth, the percentage changes, and changes rapidly, because flows come and go frequently. Because each flow may have different precedence values, the percentage of link bandwidth for each flow will change, and it will change very quickly, as each flow is added or removed. In short, WFQ simply could not be implemented by assigning a percentage of bandwidth, or a byte count, to each queue.

The WFQ scheduler is actually very simple. When the TX Queue/Ring frees a slot, WFQ can move one packet to the TX Queue/Ring, just like any other queuing tool. The WFQ scheduler takes the packet with the lowest sequence number (SN) among all the queues, and moves it to the TX Queue/Ring. The SN is assigned when the packet is placed into a queue, which is where the interesting part of WFQ scheduling takes place.

For perspective on the sequence of events, marking the SN, and serving the queues, examine Figure 4-13.

Figure 4-13 *WFQ: Assigning Sequence Numbers and Servicing Queues*


3) Maximum Number of Queues
4) Maximum Queue Length
5) Scheduling Inside Queue
6) Scheduler Logic
1) Classification
Assign SN
2) Drop Decision
Max 4096 Queues
Max Length 4096
FIFO
SN = Previous_SN +weight*length
Weight Based on IP Precedence
?
Dropped
Process: Take packet with lowest sequence number.
Result: Favors flows with lower byte volumes and larger precedence values.
TX Queue/Ring
Always on Combination Of:
- Source/destination IP Address
- Transport Protocol Type
- Source/Destination Port
Modified Tail Drop Based on Hold Queue and CDT

WFQ calculates the SN before adding a packet to its associated queue. In fact, WFQ calculates the SN before making the drop decision, because the SN is part of the modified tail-drop logic. The WFQ scheduler considers both packet length and precedence when calculating the SN. The formula for calculating the SN for a packet is as follows:

Previous_SN + weight * new_packet_length

The formula considers the length of the new packet, the weight of the flow, and the previous SN. By considering the packet length, the SN calculation results in a higher number for larger packets, and a lower number for smaller packets. The formula considers the SN of the most recently enqueued packet in the queue for the new sequence number. By including the SN of the previous packet enqueued into that queue, the formula assigns a larger number for packets in queues that already have a larger number of packets enqueued.

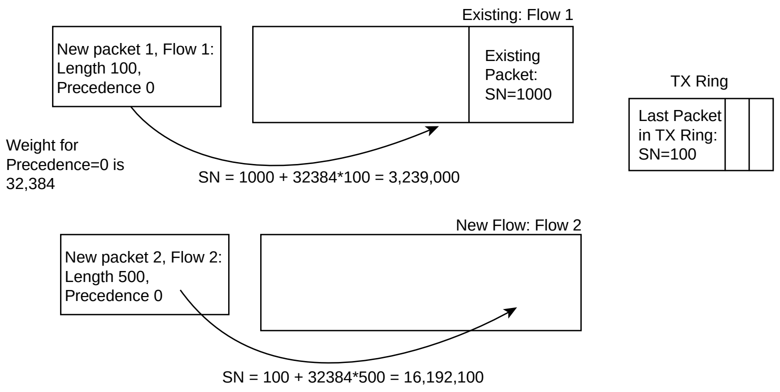
The third component of the formula, the weight, is the most interesting part. We know from the basic scheduling algorithm that the lowest SN is taken next, and we know that WFQ wants to give more bandwidth to the higher-precedence flows. So, the weight values are inversely proportional to the precedence values. Table 4-5 lists the weight values used by WFQ before and after the release of 12.0(5)T/12.1.

Table 4-5 *WFQ Weight Values, Before and After 12.0(5)T/12.1*

Precedence	Before 12.0(5)T/12.1	After 12.0(5)T/12.1
0	4096	32384
1	2048	16192
2	1365	10794
3	1024	8096
4	819	6476
5	682	5397
6	585	4626
7	512	4048

As seen in the table, the larger the precedence value, the lower the weight making the SN lower. An example certainly helps for a fuller understanding. Consider the example in Figure 4-14, which illustrates one existing flow and one new flow.

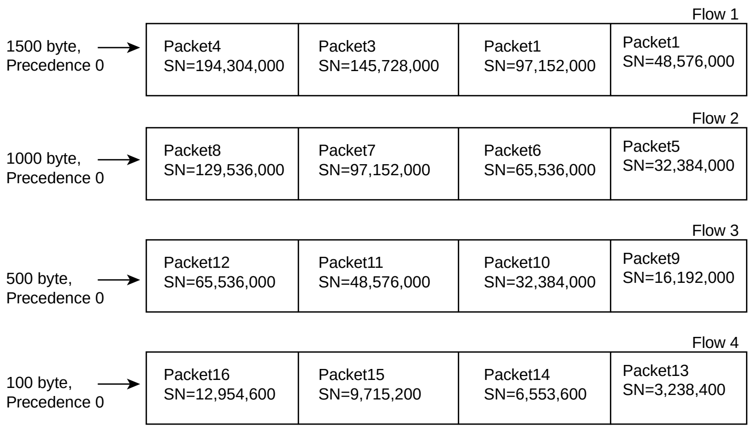
Figure 4-14 *WFQ Sequence Number Assignment Example*


Existing: Flow 1
New packet 1, Flow 1: Length 100, Precedence 0
Existing Packet: SN=1000
TX Ring
Last Packet in TX Ring: SN=100
Weight for Precedence=0 is 32,384
SN = 1000 + 32384*100 = 3,239,000
New Flow: Flow 2
New packet 2, Flow 2: Length 500, Precedence 0
SN = 100 + 32384*500 = 16,192,100

When adding new packet 1 to the queue for Flow 1, WFQ just runs the formula against the length of the new packet (100) and the weight, adding the SN of the last packet in the queue to which the new packet will be added. For new flows, the same formula is used; because there are no other packets in the queue, however, the SN of the most recently sent packet, in this case 100, is used in the formula. In either case, WFQ assigns larger SN values for larger packets and for those with lower IP precedence.

A more detailed example can show some of the effects of the WFQ SN assignment algorithm and how it achieves its basic goals. Figure 4-15 shows a set of four flow queues, each with four packets of varying lengths. For the sake of discussion, assume that the SN of the previously sent packet is zero in this case. Each flow's first packet arrives at the same instant in time, and all packets for all flows arrive before any more packets can be taken from the WFQ queues.

Figure 4-15 *WFQ Sequence Number Assignment Example 2*

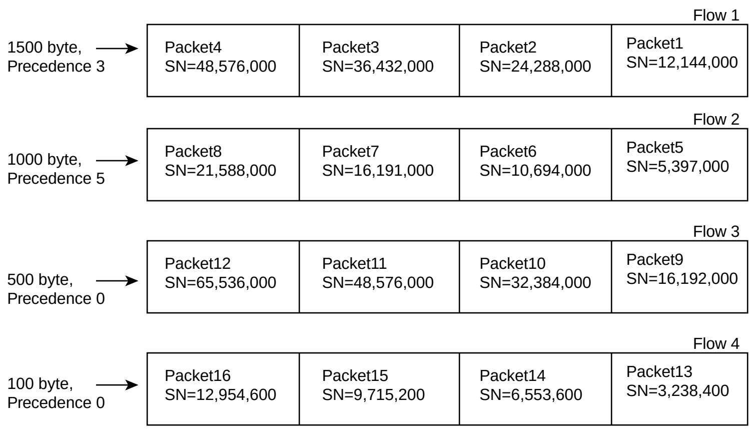


In this example, each flow had four packets arrive, all with a precedence of zero. The packets in Flow 1 were all 1500 bytes in length; in Flow 2, the packets were 1000 bytes in length; in Flow 3, they were 500 bytes; and finally, in Flow 4, they were 100 bytes. With equal precedence values, the Flow 4 packets should get better service, because the packets are much smaller. In fact, all four of Flow 1's packets would be serviced before any of the packets in the other flows. Flow 3's packets are sent before most of the packets in Flow 1 and Flow 2. Thus, the goal of giving the lower-volume flows better service is accomplished, assuming the precedence values are equal.

NOTE For the record, the order the packets would exit the interface, assuming no other events occur, is 13 first, then 14, followed by 15, 16, 9, 5, 19, 1, 11, 6, 12, 2, 7, 8, 3, 4.

To see the effect of different precedence values, look at Figure 4-16, which lists the same basic scenario but with varying precedence values.

Figure 4-16 *WFQ Sequence Number Assignment with Varying Precedence Values*



The SNs for Flow 1 and Flow 2 improve dramatically with the higher precedence values of 3 and 5, respectively. Flow 4 still gets relatively good service, even at precedence 0. Two packets from Flow 2, and one from Flow 1, will be serviced before Flow 4's fourth packet (SN 12,954,600), which is an example of how the higher precedence value gives the packets in this flow slightly better service. So, the lower-volume, but lower-precedence flows will have some degradation in service relative to the higher-volume, but higher-precedence flows.

NOTE For the record, the order the packets would exit the interface, assuming no other events occur, is 13, 5, 14, 15, 6, 1, 16, 7, 9, 2, 8, 3, 4, 11, 12.

Finally, a router using WFQ can experience a phenomenon called *too fair*. With many flows, WFQ will give some bandwidth to every flow. In the previous example, what happens if 200 new flows begin? Each of those new flows will get a relatively low SN, because the SN of the most recently sent packet is used in the formula. The packets that are already in the existing queues will have to wait on all the new packets. In an effort to give each flow some of the link bandwidth, WFQ may actually not give some or most of the flows enough bandwidth for them to survive.

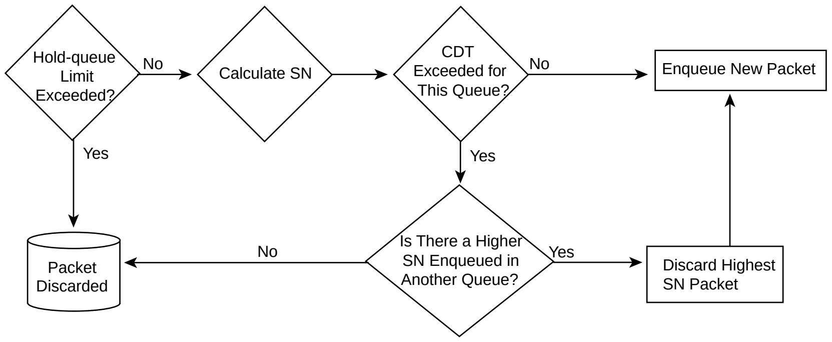
WFQ Drop Policy, Number of Queues, and Queue Lengths

WFQ uses a slightly modified tail-drop policy for choosing when to drop packets. The decision is based on several factors, one being the SN of the packet.

WFQ places an absolute limit on the number of packets enqueued among all queues; this value is called the *hold-queue limit*. If a new packet arrives, and the hold-queue limit has been reached, the packet is discarded. That part of the decision is based not on a single queue, but on the whole WFQ queuing system for the interface.

The next decision is based on an individual queue. If a packet needs to be placed into a queue, and that queue's *congestive discard threshold* (CDT) has been reached, the packet may be thrown away. CDT is a little like a maximum queue length for each flow's queue, but WFQ puts a little twist on how the concept is used (hence the use of another term, instead of just calling it the maximum queue length). To appreciate how the CDT is used, examine Figure 4-17.

Figure 4-17 *WFQ Modified Tail Drop and Congestive Discard Threshold*


Hold-queue Limit Exceeded?
No
Calculate SN
CDT Exceeded for This Queue?
No
Enqueue New Packet
Yes
Yes
Packet Discarded
No
Is There a Higher SN Enqueued in Another Queue?
Yes
Discard Highest SN Packet

The hold-queue size limits the total number of packets in all of the flow or conversation queues. However, CDT limits the number of packets in each individual queue. If CDT packets are already in the queue into which a packet should be placed, WFQ considers discarding the new packet. Normally, the new packet is discarded. If a packet with a larger SN has already been enqueued in a different queue, however, WFQ instead discards the packet with the larger SN! It's like going to Disneyland, getting in line, and then being told that a bunch of VIPs showed up, so you cannot ride the ride, and you will have to come back later. (Hopefully Disney would not take you out of the line and send you to the bit bucket, though!) In short, WFQ can discard a packet in another flow when the queue for a different flow has exceeded CDT but still has lower sequence numbers. You can configure the CDT to a value between 1 and 4096, inclusive.

Finally, WFQ can be configured for a maximum of 4096 queues, but interestingly, the actual value can only be a power of 2 between 16 and 4096, inclusive. The IOS restricts the values because WFQ performs a hash algorithm to classify traffic, and the hash algorithm only works when the number of queues is one of these valid values.

WFQ Configuration

Although WFQ requires a little deeper examination to understand all the underlying concepts, configuration is simple. IOS uses WFQ by default on all serial interfaces with bandwidths set at T/1 and E/1 speeds and below. None of WFQ's parameters can be set for an individual queue, so at most, the WFQ configuration will be one or two lines long. An example configuration for WFQ follows Tables 4-6 and 4-7. Tables 4-6 and 4-7 list the configuration and exec commands related to WFQ respectively.

Table 4-6 *Configuration Command Reference for WFQ*

Command	Mode and Function
fair-queue [*congestive-discard-threshold* [*dynamic-queues* [*reservable-queues*]]]	Interface configuration mode; enables WFQ, sets the CDT, sets maximum number of queues, and sets the number reserved for RSVP use
hold-queue *length* {**in** | **out**}	Interface configuration mode; changes the length of the hold queue

Table 4-7 *Exec Command Reference for WFQ*

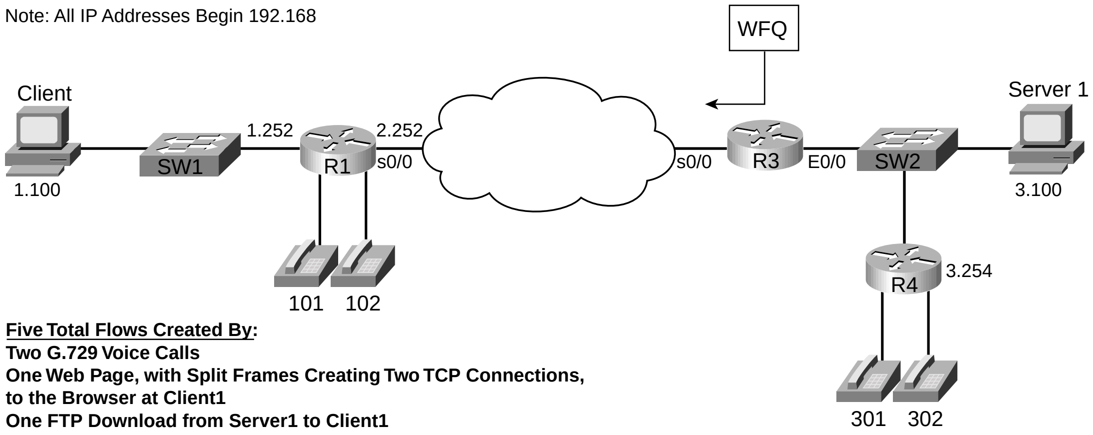
Command	Function
show queue *interface-name interface-number* [**vc** [*vpi*/] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing [**custom** | **fair** | **priority** | **random-detect** [**interface** *atm-subinterface* [vc [[*vpi*/] *vci*]]]]	Lists configuration and statistical information about the queuing tool on an interface

In the next example, R3 uses WFQ on its S0/0 interface. R3 marks the packets as they enter E0/0, using CB marking. Two voice calls, plus one FTP download, and a large web page download generate the traffic. The web page is the same one used throughout the book, with competing frames on the left and right side of the page. Note that each of the two frames in the web page uses two separate TCP connections. The marking logic performed by CB marking is as follows:

- VoIP payload—DSCP EF
- HTTP traffic for web pages with "important" in the URL—DSCP AF21
- HTTP traffic for web pages with "not" in the URL—DSCP AF23
- All other—DSCP BE

Repetitive examples do not help much with WFQ, because there is little to configure. Example 4-3 shows the basic configuration, followed by some **show** commands. After that, it shows a few of the optional parameters being set. The example uses the familiar network diagram, as repeated in Figure 4-18.

Figure 4-18 *Sample WFQ Network—WFQ on R3's S0/0 Interface*


Note: All IP Addresses Begin 192.168
WFQ
Client
1.100
SW1
1.252
R1
2.252
s0/0
101
102
s0/0
R3
E0/0
SW2
Server 1
3.100
R4
3.254
301
302
Five Total Flows Created By:
Two G.729 Voice Calls
One Web Page, with Split Frames Creating Two TCP Connections, to the Browser at Client1
One FTP Download from Server1 to Client1

Example 4-3 *WFQ Configuration and* **show** *Commands*

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int s 0/0
R3(config-if)#fair-queue
R3(config-if)#^Z
R3#sh int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 9/255, rxload 8/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  171, LMI stat recvd 163, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 378/2, interface broadcasts 347
  Last input 00:00:01, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:28:46
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 8249
```

continues

Example 4-3 *WFQ Configuration and* **show** *Commands (Continued)*

```
  Queueing strategy: weighted fair
  Output queue: 126/1000/64/8249 (size/max total/threshold/drops)
     Conversations  6/7/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 52000 bits/sec, 97 packets/sec
  5 minute output rate 58000 bits/sec, 78 packets/sec
     36509 packets input, 2347716 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     1 input errors, 0 CRC, 1 frame, 0 overrun, 0 ignored, 0 abort
     28212 packets output, 2623792 bytes, 0 underruns
     0 output errors, 0 collisions, 5 interface resets
     0 output buffer failures, 0 output buffers swapped out
     10 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
R3#show queueing fair
Current fair queue configuration:

  Interface           Discard    Dynamic  Reserved  Link    Priority
                      threshold  queues   queues    queues  queues
  Serial0/0           64         256      0         8       1
  Serial0/1           64         256      0         8       1

R3#show queueing fair int s 0/0
Current fair queue configuration:

  Interface           Discard    Dynamic  Reserved  Link    Priority
                      threshold  queues   queues    queues  queues
  Serial0/0           64         256      0         8       1

R3# show queue s 0/0
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 11027
  Queueing strategy: weighted fair
  Output queue: 79/1000/64/11027 (size/max total/threshold/drops)
     Conversations  4/8/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

! Next stanza lists info about one of the VoIP calls
  (depth/weight/total drops/no-buffer drops/interleaves) 37/5397/1359/0/0
  Conversation 15, linktype: ip, length: 64
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x013B, ttl: 253,
TOS: 184 prot: 17, source port 16772, destination port 19232
! Next stanza lists info about one of the VoIP calls
  (depth/weight/total drops/no-buffer drops/interleaves) 37/5397/1359/0/0
  Conversation 125, linktype: ip, length: 64
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x0134, ttl: 253,
  TOS: 184 prot: 17, source port 16638, destination port 19476

! Next stanza lists info about one of the HTTP TCP connections
  (depth/weight/total drops/no-buffer drops/interleaves) 1/10794/36/0/0
  Conversation 33, linktype: ip, length: 1404
```

Example 4-3 *WFQ Configuration and* **show** *Commands (Continued)*

```
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xFF50, ttl: 127,
  TOS: 72 prot: 6, source port 80, destination port 1067

! Next stanza lists info about one of the HTTP TCP connections
  (depth/weight/total drops/no-buffer drops/interleaves) 2/10794/34/0/0
  Conversation 34, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xFF53, ttl: 127,
  TOS: 88 prot: 6, source port 80, destination port 1068

! Notice the TOS values versus the weight in the last two stanzas!

R3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int s 0/0
R3(config-if)#fair-queue 100 64 10
R3(config-if)#hold-queue 500 out
R3(config-if)#^Z
!
R3#show interface serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 9/255, rxload 8/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  198, LMI stat recvd 190, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 442/2, interface broadcasts 406
  Last input 00:00:01, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:33:14
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 12474
  Queueing strategy: weighted fair
  Output queue: 95/500/100/12474 (size/max total/threshold/drops)
     Conversations  5/6/64 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

! lines omitted for brevity

R3#show queueing fair
Current fair queue configuration:

  Interface           Discard    Dynamic  Reserved  Link    Priority
                      threshold  queues   queues    queues  queues
  Serial0/0           100        64       10        8       1
  Serial0/1           64         256      0         8       1
R3#sh queue s 0/0
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 13567
```

continues

Example 4-3 *WFQ Configuration and* **show** *Commands (Continued)*

```
  Queueing strategy: weighted fair
  Output queue: 125/500/100/13567 (size/max total/threshold/drops)
     Conversations  5/7/64 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 61/5397/654/0/0
  Conversation 61, linktype: ip, length: 64
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x0134, ttl: 253,
  TOS: 184 prot: 17, source port 16638, destination port 19476

  (depth/weight/total drops/no-buffer drops/interleaves) 61/5397/653/0/0
  Conversation 15, linktype: ip, length: 64
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x013B, ttl: 253,
  TOS: 184 prot: 17, source port 16772, destination port 19232

  (depth/weight/total drops/no-buffer drops/interleaves) 1/10794/15/0/0
  Conversation 34, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0x00A5, ttl: 127,
  TOS: 88 prot: 6, source port 80, destination port 1068

  (depth/weight/total drops/no-buffer drops/interleaves) 1/10794/15/0/0
  Conversation 33, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0x00A7, ttl: 127,
  TOS: 72 prot: 6, source port 80, destination port 1067

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/12/0/0
  Conversation 29, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0x00A1, ttl: 127,
  TOS: 0 prot: 6, source port 1353, destination port 1065
```

To enable WFQ, the **fair-queue** interface subcommand is used. After enabling WFQ, the **show interface** command output shows the fact that WFQ is enabled. (See the highlighted portion of the first **show interface** command in Example 4-3.) Note that the hold queue default size of 1000 is shown in the **show interface** output as well. The **show interface** command and the **show queueing fair** commands both list the CDT (default 64), along with the maximum number of queues (256).

The most interesting **show** command for WFQ is the **show queue** command. Note that a summary section is listed first, followed by a stanza of output for each active flow. Each stanza lists statistics about the current size, number of drops, and so on. Each stanza also lists the details used for classification. For instance, each stanza of the first **show queue** command includes a comment added by me. Knowing that two voice calls, one TCP connection for FTP, and two TCP connections for HTTP were being used, I could look at the source and destination addresses and ports numbers and decide which WFQ flows correlated to each of these user flows.

You can understand the usage of the ToS byte by WFQ with a little closer examination of the output of the **show queue** command. You may recall that the two HTTP transfers were marked with different DSCP values; note that the two HTTP flows in the command output have ToS byte values of 72 and 80. Which of these values corresponds to DSCP AF21 (important HTTP URLs per CB marking) and AF23 ("not" important URLs per CB marking)? Table 4-8 lists the pertinent details needed to correlate DSCP, ToS, and precedence values used in the example.

Table 4-8 *DSCP, ToS Byte, and WFQ Weight Values Used in Example 4-3*

DSCP Name	Type of Traffic Marked in This Example	Binary DSCP, with Precedence Portion in Bold	Binary ToS, with 0s Padded for Last 2 Bits	ToS Byte Decimal Value	Precedence Value (Decimal)
EF	VoIP	**101**110	**101**11000	184	5
AF21	HTTP URLs with "important"	**010**010	**010**01000	72	2
AF23	HTTP URLs with "not"	**010**110	**010**11000	88	2
BE	All else	**000**000	**000**00000	0	0

WFQ always weights the packets based on the first 3 bits of the ToS byte—in other words, based on the Precedence field. However, the **show queue** command output lists the entire contents of the ToS byte, which in this case included 6 bits marked by CB marking, and two trailing binary 0s. Therefore, the ToS byte values in the command lists the same values shown in the table. Even though CB marking marked a different DSCP for each type of HTTP traffic, as far as WFQ is concerned, each receives the same amount of weight. This is because WFQ does not look past the Precedence field when determining the weight.

Finally, the second half of Example 4-3 just shows some configuration changes and the resulting changes in the **show** command output. The configuration changes the CDT, the maximum number of queues, and the length of the hold queue. The highlighted portions of the **show interface**, **show queueing**, and **show queue** commands reflect the details of the configuration changes.

WFQ Summary

WFQ works well for networks where the most delay-sensitive traffic requires less bandwidth than the average flow. Also flows with higher precedence work well, with low-volume, high-precedence flows receiving exceptional treatment. Best of all, WFQ requires no classification configuration. As a result, WFQ provides a great default queuing choice, particularly when traffic characteristics are unpredictable and difficult to qualify.

WFQ works poorly for voice and interactive video traffic, because both need low delay and low jitter. WFQ does not provide a priority queue in order to minimize delay and jitter. Also delay can increase when too many concurrent flows occur, due to WFQ being "too fair," allowing some bandwidth to each flow, which may cause the voice or video flows to not get enough bandwidth.

Table 4-9 summarizes some of the key features of WFQ.

Table 4-9 *WFQ Functions and Features*

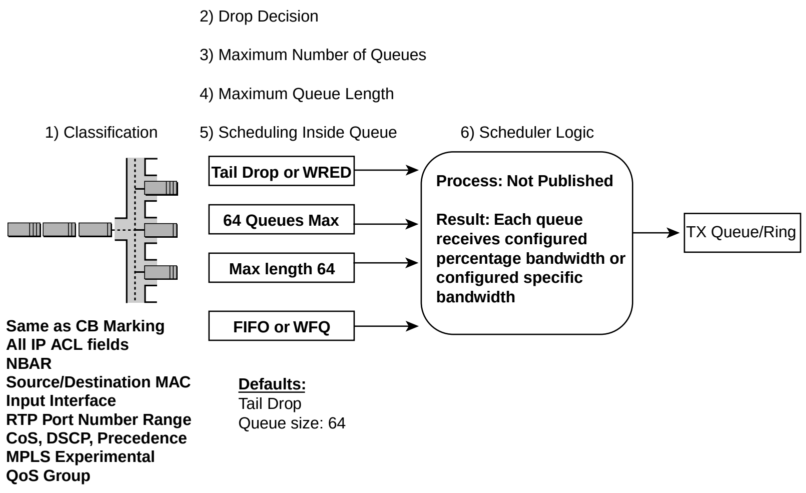
WFQ Feature	Explanation
Classification	Classifies without configuration, based on source/destination IP address/ port, protocol type (TCP|UDP), and ToS.
Drop policy	Modified tail drop.
Number of queues	4096.
Maximum queue length	Congestive discard threshold per queue (max 4096), with an overall limit based on the hold queue for all queues (max 4096).
Scheduling inside a single queue	FIFO.
Scheduling among all queues	Serves lowest sequence number (SN). The SN is assigned when the packet is placed into the queue, as a function of length and precedence.

Class-Based WFQ (CBWFQ)

Like the other queuing tools with WFQ in the name, CBWFQ uses features that are similar to some other queuing tools, and completely different from others. CBWFQ is like CQ, in that it can be used to reserve minimum bandwidth for each queue, but it differs from CQ in that you can configure the actual percentage of traffic, rather than a byte count. CBWFQ is like WFQ in that CBWFQ can actually use WFQ inside one particular queue, but it differs from WFQ in that it does not keep up with flows for all the traffic.

Many people find it difficult to keep the details memorized. To help overcome confusion, the features of CBWFQ are covered in the next several pages. At the end of this section, some summary tables list the key features and compare CBWFQ to some of the other queuing tools. To begin the coverage, examine Figure 4-19, which outlines the typical queuing features in sequence.

Figure 4-19 *CBWFQ—Summary of Main Features*



Starting left to right in the figure, CBWFQ classifies packets using the exact same set of fields that CB marking uses to classify packets. In fact, CBWFQ uses the exact same configuration commands, all of which are part of the Modular QoS CLI (MQC) commands described in Chapter 3, "Classification and Marking." CBWFQ's use of MQC makes learning the configuration for CBWFQ easy, assuming you remember how to configure CB marking from the preceding chapter. And unlike WFQ and dWFQ, which use flow-based classifiers, CBWFQ does not classify based on the flow, but on anything you can match with the MQC commands.

NOTE CBWFQ uses the terms "class" and "queue" to refer to the single queue that is associated with a class that can be defined with CBWFQ. The terms "class" and "queue" are often used interchangeably when describing CBWFQ.

CBWFQ supports two types of drop policy, namely tail drop and WRED. Chapter 6, "Congestion Avoidance Through Drop Policies," covers WRED in detail, but the general idea is to discard packets before the queue actually fills, with the intent of making some TCP connections react to the lost packet and slow down sending packets. By having a few TCP connections slow down, the queue may not fill, reducing congestion.

You can enable WRED on any of the 64 queues available with CBWFQ. However, WRED is a good option for some queues, and not for others. If a queue holds on to VoIP payload, for example, you do not really want to drop any packets, because if you drop voice packets, the voice quality degrades. In queues holding less-drop-sensitive traffic, such as data, WRED is a good option, but WRED works poorly in queues holding voice and video traffic.

CBWFQ supports 64 queues, with a maximum and default queue length of 64. All 64 queues can be configured, but one class queue, called *class-default*, is automatically configured. If the explicitly configured classification does not match a packet, IOS places the packet into the class-default class. You are allowed to change the configuration details regarding this default class, but this one class always exists.

So far, the other queuing tools in this chapter supported only FIFO logic inside a single queue. In fact, some of you may have been wondering why "step 5" was included in illustrations such as the one shown in Figure 4-19. Currently, CBWFQ can use either FIFO or WFQ inside the class-default queue. With Flow-Based WFQ in the class-default queue, when CBWFQ decides to take one or more packets from the queue, it takes the packet with the best sequence number (SN)—just like WFQ normally does.

CBWFQ provides a great advantage by allowing WFQ to be used in the class-default queue. You may recall that WFQ is actually a very good default choice for queuing, because it treats low-volume flows well, and many low-volume flows are also interactive flows. WFQ also treats packets with high precedence well. So, with CBWFQ, for the traffic you know about, you classify it, and reserve the right amount of bandwidth for the class. For the traffic you cannot characterize, you let it default into the class-default queue, where you can dynamically apply some fairness to the default traffic by using WFQ. The capability to reserve bandwidth for some packets, and fairly assign the rest of the bandwidth with WFQ, makes CBWFQ a very powerful queuing tool.

Finally, Cisco does tell us the general idea behind how the CBWFQ scheduler works. The scheduler gives a percentage of the bandwidth to each class, based on the configured values. For instance, four classes, including class-default, may be configured with bandwidth percentages that total 100 percent. The scheduler ensures that each queue receives that percentage of bandwidth. If some queues do not need their bandwidth for a short period, the bandwidth is spread across the other classes. Cisco does not really offer more details about how the scheduler works—so you do not need to worry more about how CDWFQ works for the exams!

Table 4-10 summarizes some of the key features of CBWFQ.

Table 4-10 *CBWFQ Functions and Features*

CBWFQ Feature	Description
Classification	Classifies based on anything that MQC commands can match, just like CB marking. Includes all extended IP ACL fields, NBAR, incoming interface, CoS, precedence, DSCP, source/destination MAC, MPLS Experimental, QoS group, and RTP port numbers
Drop policy	Tail drop or WRED, configurable per queue.
Number of queues	64.

Table 4-10 *CBWFQ Functions and Features (Continued)*

CBWFQ Feature	Description
Maximum queue length	64.
Scheduling inside a single queue	FIFO on 63 queues; FIFO or WFQ on class-default queue.
Scheduling among all queues	Algorithm is not published. The result of the scheduler provides a percentage guaranteed bandwidth to each queue.

CBWFQ Configuration

CBWFQ configuration uses the same MQC commands as does CB marking, with a few additional commands added. If you remember how to configure CB marking from Chapter 3, CBWFQ configuration will be easy for you. The commands used for CP marking and CBWFQ configuration are repeated here in Tables 4-11 and 4-12.

Table 4-11 *Command Reference for CBWFQ*

Command	Mode and Function
class-map *class-map-name*	Global config; names a class map, where classification options are configured.
match …	Class map subcommand; defines specific classification parameters.
match access-group {*access-group* | **name** *access-group-name*}	Access-control list (ACL).
match source-address mac *address*	Source MAC address.
match ip precedence *ip-precedence-value* [*ip-precedence-value ip-precedence-value ip-precedence-value*]	IP precedence.
match mpls experimental *number*	MPLS Experimental.
match cos *cos-value* [*cos-value cos-value cos-value*]	CoS.
match destination-address mac *address*	Destination MAC address.
match input-interface *interface-name*	Input interface.
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	IP DSCP.
match ip rtp *starting-port-number port-range*	RTP’s UDP port number range.
match qos-group *qos-group-value*	QoS group.

continues

Table 4-11 *Command Reference for CBWFQ (Continued)*

Command	Mode and Function
match protocol *protocol-name*	NBAR protocol types.
match protocol citrix [**app** *application-name-string*].	NBAR Citrix applications.
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	Host name and URL string.
match any	Matches any and all packets.
policy-map *policy-map-name*	Global config; names a policy, which is a set of actions to perform.
class *name*	Policy map subcommand; identifies the packets to perform QoS actions on by referring to the classification logic in a class map
bandwidth {*bandwidth-kbps* | **percent** *percent*}	Class subcommand; sets literal or percentage bandwidth for the class. Must use either use actual bandwidth or percent on all classes in a single policy map.
fair-queue [**queue-limit** *queue-value*]	Class subcommand; enables WFQ in the class (class-default only).
random-detect dscp *dscpvalue min-threshold max-threshold* [*mark-probability-denominator*]	Class subcommand; enables DSCP-based WRED in the class.
random-detect precedence *precedence min-threshold max-threshold mark-prob-denominator*	Class subcommand; enables precedence-based WRED in the class.
max-reserved-bandwidth *percent*	Interface subcommand; defines the percentage of link bandwidth that can be reserved for CBWFQ queues besides class-default.

Table 4-12 *Exec Command Reference for CBWFQ*

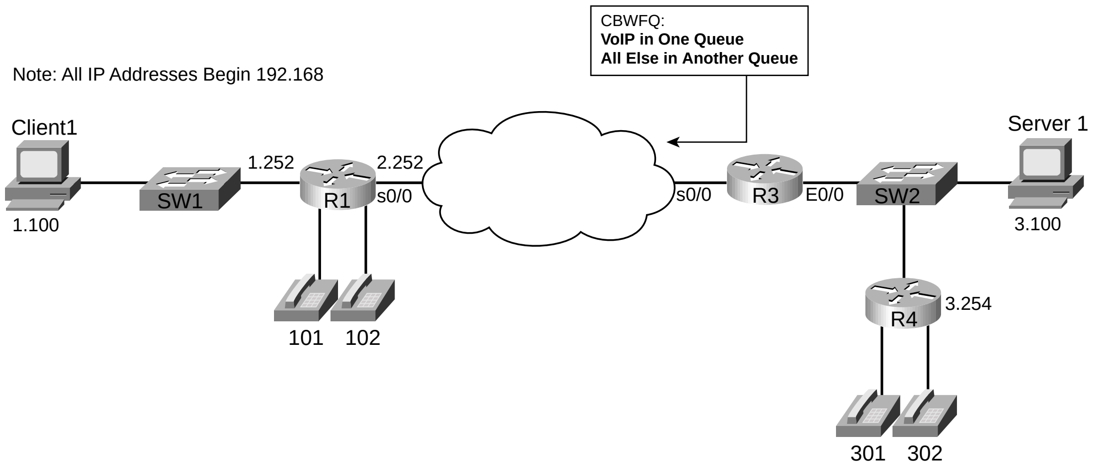
Command	Function
show policy-map *policy-map-name*	Lists configuration information about all MQC-based QoS tools
show policy-map *interface-spec* [**input** | **output**] [**class** *class-name*]	Lists statistical information about the behavior of all MQC-based QoS tools

The remainder of this section includes several familiar lab scenarios, with example configurations and **show** commands. In the first CBWFQ example, R3 uses CBWFQ on its S0/0 interface. The engineer configuring R3 decided that voice traffic could benefit from being placed into a different queue than all other traffic, so a simple QoS policy has been devised, which includes the following:

- All VoIP payload traffic is placed in a queue.
- All other traffic is placed in another queue.
- Give the VoIP traffic 50 percent of the bandwidth.
- WFQ and WRED should be used on the non-VoIP traffic.

Figure 4-20 reminds you of the now-familiar example network, which shows the router in which the configuration is applied. Example 4-4 shows the configuration and **show** commands.

Figure 4-20 *Network Used with CBWFQ Configuration Examples*



Example 4-4 *CBWFQ, VoIP in High Queue, Everything Else in Normal Queue*

```
R3#show running-config
Building configuration...

! Portions omitted for brevity
!
ip cef
!
class-map match-all voip-rtp
  match ip rtp 16384 16383
```

continues

Example 4-4 *CBWFQ, VoIP in High Queue, Everything Else in Normal Queue (Continued)*

```
!
!
policy-map queue-voip
  class voip-rtp
   bandwidth percent 50
  class class-default
   fair-queue

! Portions omitted for brevity
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 bandwidth 128
 service-policy output queue-voip
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
! Portions omitted for brevity

R3#show policy-map int s 0/0

 Serial0/0

  Service-policy output: queue-voip

    Class-map: voip-rtp (match-all)
      136435 packets, 8731840 bytes
      30 second offered rate 51000 bps, drop rate 0 bps
      Match: ip rtp 16384 16383
      Weighted Fair Queueing
        Output Queue: Conversation 265
        Bandwidth 50 (%) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 48550/3107200
        (depth/total drops/no-buffer drops) 14/0/0

    Class-map: class-default (match-any)
      1958 packets, 1122560 bytes
      30 second offered rate 59000 bps, drop rate 0 bps
      Match: any
      Weighted Fair Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 256
        (total queued/total drops/no-buffer drops) 15/0/0
```

Example 4-4 *CBWFQ, VoIP in High Queue, Everything Else in Normal Queue (Continued)*

```
R3#show policy-map int s 0/0 output class class-default Serial0/0
  Service-policy output: queue-voip

    Class-map: class-default (match-any)
      2217 packets, 1417985 bytes
      30 second offered rate 71000 bps, drop rate 0 bps
      Match: any
      Weighted Fair Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 256
        (total queued/total drops/no-buffer drops) 10/0/0

R3#show policy-map
  Policy Map queue-voip
    Class voip-rtp
      Weighted Fair Queueing
            Bandwidth 50 (%) Max Threshold 64 (packets)
    Class class-default
      Weighted Fair Queueing
            Flow based Fair Queueing Max Threshold 64 (packets)

R3#show interface s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 9/255, rxload 8/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  132, LMI stat recvd 132, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 0/64, broadcasts sent/dropped 307/0, interface broadcasts 285
  Last input 00:00:02, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:22:02
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 16/1000/64/0 (size/max total/threshold/drops)
     Conversations  4/8/256 (active/max active/max total)
     Reserved Conversations 1/1 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  30 second input rate 52000 bits/sec, 102 packets/sec
  30 second output rate 59000 bits/sec, 101 packets/sec
     126301 packets input, 8141304 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     126197 packets output, 8550371 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

continues

Example 4-4 *CBWFQ, VoIP in High Queue, Everything Else in Normal Queue (Continued)*

```
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
R3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#policy-map queue-voip
R3(config-pmap)#class class-default
R3(config-pmap-c)#no fair-queue
R3(config-pmap-c)#^Z

R3#show policy-map
  Policy Map queue-voip
    Class voip-rtp
      Weighted Fair Queueing
            Bandwidth 50 (%) Max Threshold 64 (packets)
    Class class-default
```

The configuration for CBWFQ requires many commands, but the logic is straightforward. **class-map voip-rtp** matches all VoIP payload by matching with the **match ip rtp 16384 16383** command. The **policy-map queue-voip** policy assigns VoIP traffic 50 percent of the bandwidth based on the **bandwidth percent 50** command. But 50 percent of what? Well, the actual bandwidth is derived from the percentage of the bandwidth configured on the **bandwidth interface** subcommand. In this case, the bandwidth is set to 128, so the voip-rtp class gets 64 kbps, which is enough for the two G.729a VoIP calls. Just like with CB marking, the **service-policy output** command enables the policy on the interface, in this case S0/0.

The **show policy-map interface serial 0/0** command lists the most interesting statistical information about CBWFQ. (This paragraph, and the rest of the text that describes Example 4-4, explains the commands in the example in the same sequence as they are shown in the example.) It lists a stanza of information for each class, listing the configured matching parameters and bandwidth percentages. The offered rate of traffic that has been classified into each queue is listed, along with drop statistics. These values are useful when monitoring the configuration to decide whether the configuration values should be changed. Also note that in class-default, the output implies that Flow-Based WFQ is in use inside the queue.

The command also has other options that reduce the amount of output, which can be large. For instance, the **show policy-map int s 0/0 output class class-default** shown in the example lists only information about class class-default.

The **show policy-map** command just lists a summary of the configured policy maps. Interestingly, the class-default stanza of the command output lists Weighted Fair Queuing, and then says it is “flow based.” In this case, the command output is reminding you that inside the class-default queue, Flow-Based WFQ is applied to each flow. For comparison, at the end of the example, the configuration has been changed to disable WFQ in the class-default queue. The **show policy-map** command no longer lists Flow-Based WFQ for class-default.

NOTE Although some of the CBWFQ command output references WFQ in sections describing queues other than class-default, CBWFQ does not use Flow-Based WFQ inside any of these queues. CBWFQ can only use Flow-Based WFQ inside one queue—the class-default queue.

Good QoS design calls for the marking of packets close to the source of the packet. Example 4-5 accomplishes the same queuing goals as the preceding example, but CBWFQ relies on the fact that the packets have been marked before reaching R3's S0/0 interface. In a real network, the packets could be marked on one of the LAN switches, or in an IP Phone, or by the computers in the network. This example shows the packets being marked upon entering R3's E0/0 interface, just like Example 3-1 in Chapter 3. Example 4-5 shows the revised configuration based on the following criteria:

- All VoIP payload traffic has been marked with DSCP EF; place this traffic in a queue.
- All other traffic has been marked with DSCP BE; place this traffic in a different queue.
- Give the VoIP traffic 58 kbps of the bandwidth on the link.
- Use WRED and WFQ on the non-VoIP traffic.

Example 4-5 *CBWFQ, DSCP EF in One Queue, Everything Else in Another Queue*

```
R3#show running-config
class-map match-all voip-rtp
  match ip rtp 16384 16383
!
class-map match-all dscp-ef
  match ip dscp ef
!
!
policy-map voip-and-be
  class voip-rtp
   set ip dscp 46
  class class-default
   set ip dscp 0
!
policy-map queue-on-dscp
  class dscp-ef
   bandwidth 58
 class class-default
  random-detect dscp-based
  fair-queue
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 service-policy input voip-and-be
```

continues

Example 4-5 *CBWFQ, DSCP EF in One Queue, Everything Else in Another Queue (Continued)*

```
!
interface serial0/0
 clock rate 128000
 bandwidth 128
 service-policy output queue-on-dscp
R3#
R3#show policy-map interface serial 0/0
 Serial0/0

  Service-policy output: queue-on-dscp

    Class-map: dscp-ef (match-all)
      8654 packets, 553856 bytes
      30 second offered rate 51000 bps, drop rate 0 bps
      Match: ip dscp ef
      Weighted Fair Queueing
        Output Queue: Conversation 41
        Bandwidth 58 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 8442/540288
        (depth/total drops/no-buffer drops) 8/0/0

    Class-map: class-default (match-any)
      673 packets, 779357 bytes
      30 second offered rate 71000 bps, drop rate 0 bps
      Match: any
      Weighted Fair Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 32
        (total queued/total drops/no-buffer drops) 14/0/0
         exponential weight: 9

dscp     Transmitted       Random drop      Tail drop    Minimum Maximum  Mark
          pkts/bytes        pkts/bytes       pkts/bytes    thresh  thresh  prob
af11         0/0               0/0              0/0           32      40  1/10
af12         0/0               0/0              0/0           28      40  1/10
af13         0/0               0/0              0/0           24      40  1/10
af21         0/0               0/0              0/0           32      40  1/10
af22         0/0               0/0              0/0           28      40  1/10
af23         0/0               0/0              0/0           24      40  1/10
af31         0/0               0/0              0/0           32      40  1/10
af32         0/0               0/0              0/0           28      40  1/10
af33         0/0               0/0              0/0           24      40  1/10
af41         0/0               0/0              0/0           32      40  1/10
af42         0/0               0/0              0/0           28      40  1/10
af43         0/0               0/0              0/0           24      40  1/10
cs1          0/0               0/0              0/0           22      40  1/10
cs2          0/0               0/0              0/0           24      40  1/10
cs3          0/0               0/0              0/0           26      40  1/10
cs4          0/0               0/0              0/0           28      40  1/10
cs5          0/0               0/0              0/0           30      40  1/10
cs6          23/2219           0/0              0/0           32      40  1/10
cs7          0/0               0/0              0/0           34      40  1/10
```

Example 4-5 *CBWFQ, DSCP EF in One Queue, Everything Else in Another Queue (Continued)*

```
ef           0/0             0/0             0/0             36      40  1/10
rsvp         0/0             0/0             0/0             36      40  1/10
default      9/117           0/0             0/0             20      40  1/10

R3#show policy-map
  Policy Map voip-and-be
    Class voip-rtp
      set ip dscp ef
    Class class-default
      set ip dscp default

  Policy Map queue-on-dscp
    Class dscp-ef
      Weighted Fair Queueing
            Bandwidth 58 (kbps) Max Threshold 64 (packets)
    Class class-default
      Weighted Fair Queueing
            Flow based Fair Queueing
            exponential weight 9
            dscp    min-threshold    max-threshold    mark-probability
            ----------------------------------------------------------

            af11      -                -                1/10
            af12      -                -                1/10
            af13      -                -                1/10
            af21      -                -                1/10
            af22      -                -                1/10
            af23      -                -                1/10
            af31      -                -                1/10
            af32      -                -                1/10
            af33      -                -                1/10
            af41      -                -                1/10
            af42      -                -                1/10
            af43      -                -                1/10
            cs1       -                -                1/10
            cs2       -                -                1/10
            cs3       -                -                1/10
            cs4       -                -                1/10
            cs5       -                -                1/10
            cs6       -                -                1/10
            cs7       -                -                1/10
            ef        -                -                1/10
            rsvp      -                -                1/10
            default   -                -                1/10
```

The configuration is again detailed, but straightforward. **policy-map voip-and-be** marks packets as they enter E0/0, matching and marking VoIP packets with DSCP EF, and matching and marking all other packets, marking them with DSCP BE. The **class-map match-all dscp-ef** command creates a class that matches all DSCP EF traffic. **Policy-map queue-on-dscp** refers to **class dscp-ef** in order to match all VoIP traffic, giving the voice traffic 58 kbps with the **bandwidth 58** command. CBWFQ uses tail drop in each class by default, so under the **class class-default** command, the **random-detect dscp-based** command is used to enable WRED in the default class. In addition, of course, the **service-policy** command enables the service policy on the interface.

The **show** commands do not provide much new information, other than the statistics about WRED operation. The details about this part of the output are covered in Chapter 6.

You may have found it strange to see a configuration with classification and marking happening as the packet entered E0/0 in R3, and then another round of classification for packets exiting R3's S0/0 interface. One of the great advantages of classification and marking is that after the packets are marked, the configuration of the tools providing each per-hop behavior (PHB) is simple. Another advantage is that after the packet has been marked, other devices can perform the simpler matching of just looking for DSCP or IP precedence.

If the QoS policy calls for applying PHBs only for packets exiting WAN interfaces, and the policy does not call for PHBs between packets entering and exiting LAN interfaces, the classification and marking, and the queuing, may be performed with a single configuration. Example 4-6 shows a similar configuration to the preceding example, but with both the marking and queuing performed for packets exiting R3's S0/0 interface.

Example 4-6 *CBWFQ and CB Marking Combined*

```
R3#show running-config
class-map match-all voip-rtp
  match ip rtp 16384 16383
!
policy-map mark-and-queue
  class voip-rtp
   set ip dscp ef
   bandwidth 58
  class class-default
   set ip dscp 0
   random-detect dscp
   fair-queue
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
! No service policy on E0/0!
!
interface serial0/0
 clock rate 128000
 bandwidth 128
 service-policy output mark-and-queue
```

Two classes are used in this example: voip-rtp, which matches all VoIP payload UDP ports; and class-default, which matches all other packets. Inside **policy-map mark-and-queue**, each class includes a **set** command to set the DSCP value. The voip-rtp class includes a **bandwidth** command to reserve bandwidth; class-default automatically gets 25 percent of the bandwidth. The **service-policy output mark-and-queue** command enables the policy for packets exiting R3's S0/0 interface. In this example, you get the benefits of classification and marking, a shorter configuration, and only one classification step inside this router, reducing overhead. Also keep in mind that CEF must be enabled with the **ip cef** global command for CB marking to work at all.

The configuration section for CBWFQ ends with a final example that has a larger number of classes. The same familiar traffic flows are used—two VoIP calls, a NetMeeting video conference, HTTP with two different frames (important.jpg and not-so.jpg), and an FTP download. The configuration shows CB marking on ingress of R3's E0/0, and CBWFQ on egress at R3's S0/0. The criteria for each type of traffic is as follows:

- R3's S0/0 is clocked at 128 kbps.
- VoIP payload is marked with DSCP EF, and placed in its own queue, using tail drop. This class gets 58 kbps.
- NetMeeting voice and video from Server1 to Client1 is marked with DSCP AF41, and placed in its own queue, using tail drop. This class gets 22 kbps.
- Any HTTP traffic whose URL contains the string "important" anywhere in the URL is marked with AF21, and placed in its own queue, using WRED. This class gets 20 kbps.
- Any HTTP traffic whose URL contains the string "not-so" anywhere in the URL is marked with AF23, and placed in its own queue, using WRED. This class gets 8 kbps.
- All other traffic is marked with DSCP BE, and placed in its own queue, using WRED and WFQ. This class gets the remaining 20 kbps.

Example 4-7 shows the configuration.

Example 4-7 *CBWFQ—Classes for VoIP, NetMeeting, HTTP "important," HTTP "not-so" Important, and Everything Else*

```
R3#show running-config
Building configuration...

! Portions omitted for brevity
!
ip cef
!
class-map match-all dscp-ef
  match ip dscp ef
class-map match-all dscp-af41
  match ip dscp af41
class-map match-all dscp-af21
  match ip dscp af21
class-map match-all dscp-af23
  match ip dscp af23
```

continues

Example 4-7 *CBWFQ—Classes for VoIP, NetMeeting, HTTP "important," HTTP "not-so" Important, and Everything Else (Continued)*

```
!
class-map match-all http-impo
  match protocol http url "*important*"
class-map match-all http-not
  match protocol http url "*not-so*"
class-map match-all voip-rtp
  match ip rtp 16384 16383
class-map match-all NetMeet
  match access-group 101
!
!
policy-map laundry-list
  class voip-rtp
   set ip dscp ef
  class NetMeet
   set ip dscp af41
  class http-impo
   set ip dscp af21
  class http-not
   set ip dscp af23
  class class-default
   set ip dscp default
policy-map queue-on-dscp
  class dscp-ef
   bandwidth 58
  class dscp-af41
   bandwidth 22
  class dscp-af21
   bandwidth 20
   random-detect dscp-based
  class dscp-af23
   bandwidth 8
   random-detect dscp-based
  class class-default
   fair-queue
   random-detect dscp-based
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 ip nbar protocol-discovery
 half-duplex
 service-policy input laundry-list
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 bandwidth 128
 no ip address
 encapsulation frame-relay
 load-interval 30
```

Example 4-7 *CBWFQ—Classes for VoIP, NetMeeting, HTTP "important," HTTP "not-so" Important, and Everything Else (Continued)*

```
 max-reserved-bandwidth 85
 service-policy output queue-on-dscp
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
! Portions omitted for brevity
!
access-list 101 permit udp host 192.168.3.100 range 16384 32767 192.168.1.0
  0.0.0.255 range 16384 32767
!
```

The QoS policy requires the various queues to be assigned a total of 108 kbps of the 128 kbps of link bandwidth, which is about 84 percent of the link bandwidth. By default, CBWFQ only allows 75 percent of the link bandwidth to be explicitly assigned, reserving 25 percent for class class-default. Cisco recommends that you not change the setting, but you can with the **max-reserved-bandwidth** command, as shown in the configuration. By changing the **max-reserved-bandwidth** to 85 percent, the amounts of bandwidth that were suggested in the policy statements can be assigned. The danger is that the class-default, where important overhead such as keepalives and routing updates are queued, may not get enough bandwidth.

CBWFQ Summary

CBWFQ combines some of the best features of the various queuing tools into a single tool. Like CQ, CBWFQ can reserve a guaranteed amount of bandwidth per queue, but without the negative side effects of the CQ scheduler. CBWFQ can use WFQ as the default behavior for unclassified traffic. Packet loss behavior can take advantage of WRED, which reduces the possibilities of global synchronization. In addition, of all the queuing tools, CBWFQ has the largest variety of directly matchable fields for classifying packets.

CBWFQ does have some drawbacks, however. Most drawbacks are minor, but one negative is the lack of a PQ-like feature. Delay- and jitter-sensitive traffic still suffers, even when enough bandwidth has been reserved by CBWFQ, because the CBWFQ scheduler can serve other queues when a VoIP or video packet is waiting in a queue. The next feature covered in the book, namely Low Latency Queuing (LLQ), overcomes this problem.

Table 4-13 summarizes some of the additional key comparison features among the WFQ family of queuing tools.

Table 4-13 *CBWFQ Comparison to Other WFQ Tools*

Tool	Classification	Drop Policy	Weighting	Max # of Queues
WFQ	Flow based	Modified tail drop	Based on IP Precedence	4096
CBWFQ	Class based, with very flexible options based on MQC	Tail drop or WRED per queue	Configured as % of link bandwidth per queue, or just as actual bandwidth	**64**

Low Latency Queuing (LLQ)

Low Latency Queuing sounds like the best queuing tool possible, just based on the name. What packet wouldn't want to experience low latency? As it turns out, for delay (latency) sensitive traffic, LLQ is indeed the queuing tool of choice.

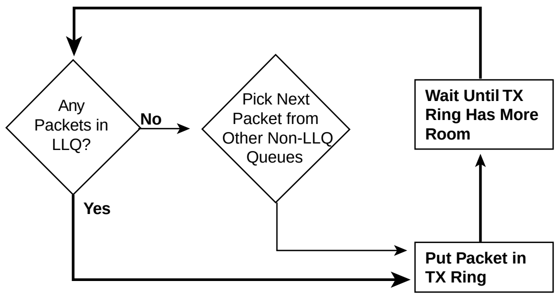
LLQ is simple to understand and simple to configure, assuming you already understand CBWFQ. LLQ is not really a separate queuing tool, but rather a simple option of CBWFQ applied to one or more classes. CBWFQ treats these classes as strict-priority queues. In other words, CBWFQ always services packets in these classes if a packet is waiting, just as PQ does for the High queue.

NOTE This section uses examples with only a single LLQ class in most cases. However, you can have more than one low-latency priority queue at the same time. It is very important that you read the section titled "LLQ with More Than One Priority Queue," just before the section about configuring LLQ. This section not only explains why you might want more than one low-latency queue, but it also covers some important information for the exam.

LLQ introduces some new lingo that you may find a little tricky. From one perspective, something like PQ has been added to CBWFQ, so you can expect to read or hear phrases that refer to the low-latency queue as "the PQ." Someone might say, "What did you put in the PQ?" What he really wants to know is what type of packets you classified and placed into the queue in which you enabled the LLQ feature of CBWFQ. In addition, the queue in which LLQ is enabled is sometimes just called "the LLQ." Therefore, if you use CBWFQ, and use the **priority** command to enable LLQ in one of the classes, you are really using LLQ, and that one class with the **priority** command is "the LLQ" or "the PQ."

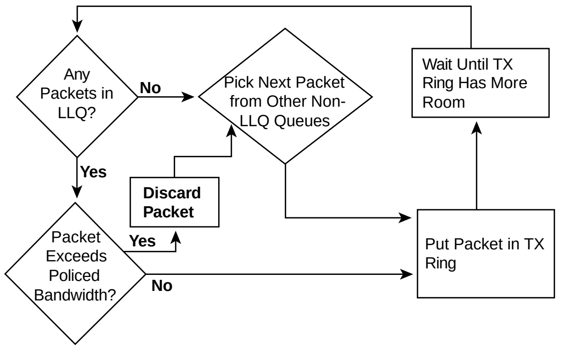
Terminology aside, the simple addition of LLQ logic to CBWFQ is depicted in Figure 4-21.

Figure 4-21 *Servicing Queues with LLQ and CBWFQ*


Any Packets in LLQ?
No
Pick Next Packet from Other Non-LLQ Queues
Wait Until TX Ring Has More Room
Yes
Put Packet in TX Ring

Note that like PQ, the LLQ scheduler always checks the low-latency queue first, and takes a packet from that queue. If there are no packets in the low-latency queue, the normal, unpublished scheduler logic applies to the other non-low-latency queue queues, giving them their guaranteed bandwidth. For delay-sensitive traffic, the addition of a low-latency queue overcomes the one big negative of CBWFQ. In fact, with all the other queuing tools covered in this chapter so far, only PQ gave voice traffic the best quality. Of course, PQ had the negative side effect of almost destroying the performance of the lower-priority applications when the link was congested. With LLQ, you get the best of both worlds—low latency for the traffic in one queue, and guaranteed bandwidth for the traffic in other queues. Notice the thicker lines in Figure 4-21. If you follow these lines, you can see a path through the logic for LLQ in which only the low-latency queue gets any service. How can LLQ guarantee the other queues their respective bandwidths, with logic that never lets those queues get serviced? Well, the real answer is that Figure 4-21 is only part of the story. To prevent LLQ from having the same problem as PQ, where packets in the highest-priority queue could dominate and take all the available bandwidth, LLQ's scheduler actually operates as shown in Figure 4-22.

Figure 4-22 *Services Queues with LLQ and CBWFQ—The Real Story*


Any Packets in LLQ?
No
Pick Next Packet from Other Non-LLQ Queues
Wait Until TX Ring Has More Room
Yes
Discard Packet
Packet Exceeds Policed Bandwidth?
Yes
No
Put Packet in TX Ring

LLQ actually polices the PQ based on the configured bandwidth. By doing so, the packets in the queue that are forwarded still have very low latency, but LLQ also prevents the low-latency traffic from consuming more than its configured amount of bandwidth. By discarding excess traffic, LLQ can still provide bandwidth guarantees to the non-priority queues. The policing function works like policing as described in Chapter 5, but it is automatic in the low-latency queue—no additional policing configuration is required.

The policing function of LLQ takes care of protecting the other queues from the low-latency queue, but it does discard packets to accomplish that goal. Take a moment to reflect on the types of traffic that need to be classified into the low-latency queue. VoIP traffic, and in most cases, video traffic, need the low-latency, low-jitter performance of the low-latency queue. However, these are the same types of traffic that are most sensitive to dropped packets. So, although putting voice and interactive video into the low-latency queue may be good for queuing, discarding packets that exceed the configured rate for the queue would be harmful to those types of traffic. (Remember, interactive video needs low latency, but one-way video does not.)

The solution to the LLQ policing feature's bad effect on VoIP and interactive video traffic lies outside the control of LLQ. The solution requires the engineer to use whatever means necessary to prevent more than the reserved bandwidth for a low-latency queue from getting introduced into the network. If the low-latency queue has 30 kbps reserved, for example, a single G.729 call will never cause the policer to discard a packet. If a second call occurs, the policer will discard packets, and both voice calls will sound bad. The solution requires some engineering, and some use of call admission control (CAC) tools, to prevent the low-latency queue from being oversubscribed.

LLQ Configuration

LLQ configuration requires one more command in addition to the commands used for CBWFQ configuration. Instead of using the **bandwidth** command on a class, use the **priority** command. This single additional command is listed in Table 4-24. The syntax for this command is as follows:

```
priority {bandwidth-kbps | percent percentage} [burst]
```

This class subcommand enables LLQ in this class, reserves bandwidth, and enables the policing function. You can also configure the burst for the policer with this command, and it defaults to 20 percent of the configured policing rate.

The **priority** command sets the guaranteed minimum bandwidth, which is also the maximum bandwidth! As mentioned earlier, LLQ polices the traffic in a class that uses the **priority** command and discards excess traffic. The **burst** parameter works just like bursts do for policing tools described in Chapter 5; refer to Chapter 5 for more details on the concepts behind policing.

In the following example, the final lab scenario used in the CBWFQ section is repeated, except that LLQ is also enabled. The class with VoIP traffic has reserved 58 kbps again, but this time using the **priority** command. With two VoIP calls, the voice sounds fine. The same familiar traffic flows are used—two VoIP calls, a NetMeeting video conference, HTTP with two different frames (important.jpg and not-so.jpg), and an FTP download. The configuration shows CB marking on ingress of R3's E0/0, and CBWFQ on egress at R3's S0/0. The criteria for each type of traffic is as follows:

- R3's S0/0 is clocked at 128 kbps.
- VoIP payload is marked with DSCP EF, and placed in its own queue, using tail drop. This class gets 58 kbps.
- NetMeeting voice and video from Server1 to Client1 is marked with DSCP AF41, and placed in its own queue, using tail drop. This class gets 22 kbps.
- Any HTTP traffic whose URL contains the string "important" anywhere in the URL is marked with AF21, and placed in its own queue. This class gets 20 kbps.
- Any HTTP traffic whose URL contains the string "not-so" anywhere in the URL is marked with AF23, and placed in its own queue. This class gets 8 kbps.
- All other traffic is marked with DSCP BE, and placed in its own queue, using WRED and WFQ. This class gets the remaining 20 kbps.

Example 4-8 shows the configuration.

Example 4-8 *LLQ for VoIP, CBWFQ for NetMeeting, HTTP "important," HTTP "not-so" Important, and Everything Else*

```
R3#show running-config
Building configuration...
!
!Portions omitted for brevity
!
ip cef
!
class-map match-all dscp-ef
  match ip dscp ef
class-map match-all dscp-af41
  match ip dscp af41
class-map match-all dscp-af21
  match ip dscp af21
class-map match-all http-impo
  match protocol http url "*important*"
class-map match-all dscp-af23
  match ip dscp af23
class-map match-all http-not
  match protocol http url "*not-so*"
class-map match-all voip-rtp
  match ip rtp 16384 16383
class-map match-all NetMeet
  match access-group 101
!
!
```

continues

Example 4-8 *LLQ for VoIP, CBWFQ for NetMeeting, HTTP "important," HTTP "not-so" Important, and Everything Else (Continued)*

```
policy-map laundry-list
  class voip-rtp
   set ip dscp ef
  class NetMeet
   set ip dscp af41
  class http-impo
   set ip dscp af21
  class http-not
   set ip dscp af23
  class class-default
   set ip dscp default
policy-map queue-on-dscp
  class dscp-ef
    priority 58
  class dscp-af41
   bandwidth 22
  class dscp-af21
   bandwidth 20
   random-detect dscp-based
  class dscp-af23
   bandwidth 8
   random-detect dscp-based
  class class-default
   fair-queue
   random-detect dscp-based
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 ip nbar protocol-discovery
 half-duplex
 service-policy input laundry-list
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 bandwidth 128
 no ip address
 encapsulation frame-relay
 load-interval 30
 max-reserved-bandwidth 85
 service-policy output queue-on-dscp
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
access-list 101 permit udp host 192.168.3.100 range 16384 32767 192.168.1.0
  0.0.0.255 range 16384 32767
```

Example 4-8 *LLQ for VoIP, CBWFQ for NetMeeting, HTTP "important," HTTP "not-so" Important, and Everything Else (Continued)*

```
! portions omitted for brevity
R3#show policy-map queue-on-dscp
    Policy Map queue-on-dscp
    Class dscp-ef
      Weighted Fair Queueing
            Strict Priority
            Bandwidth 58 (kbps) Burst 1450 (Bytes)
    Class dscp-af41
      Weighted Fair Queueing
            Bandwidth 22 (kbps) Max Threshold 64 (packets)
    Class dscp-af21
      Weighted Fair Queueing
            Bandwidth 20 (kbps) Max Threshold 64 (packets)
    Class dscp-af23
      Weighted Fair Queueing
            Bandwidth 8 (kbps) Max Threshold 64 (packets)
    Class class-default
      Weighted Fair Queueing
            Flow based Fair Queueing
            exponential weight 9
            dscp    min-threshold    max-threshold    mark-probability
            ----------------------------------------------------------

            af11      -                -                1/10
            af12      -                -                1/10
            af13      -                -                1/10
            af21      -                -                1/10
            af22      -                -                1/10
            af23      -                -                1/10
            af31      -                -                1/10
            af32      -                -                1/10
            af33      -                -                1/10
            af41      -                -                1/10
            af42      -                -                1/10
            af43      -                -                1/10
            cs1       -                -                1/10
            cs2       -                -                1/10
            cs3       -                -                1/10
            cs4       -                -                1/10
            cs5       -                -                1/10
            cs6       -                -                1/10
            cs7       -                -                1/10
            ef        -                -                1/10
            rsvp      -                -                1/10
            default   -                -                1/10


R3#show policy-map interface s 0/0 output class dscp-ef
 Serial0/0
```

continues

Example 4-8 *LLQ for VoIP, CBWFQ for NetMeeting, HTTP "important," HTTP "not-so" Important, and Everything Else (Continued)*

```
  Service-policy output: queue-on-dscp

    Class-map: dscp-ef (match-all)
      227428 packets, 14555392 bytes
      30 second offered rate 52000 bps, drop rate 0 bps
      Match: ip dscp ef
      Weighted Fair Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 58 (kbps) Burst 1450 (Bytes)
        (pkts matched/bytes matched) 12194/780416
        (total drops/bytes drops) 0/0

R3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#policy-map queue-on-dscp
R3(config-pmap)#class dscp-ef
R3(config-pmap-c)#priority 48
R3(config-pmap-c)#^Z
R3#show policy-map interface s 0/0 output class dscp-ef
 Serial0/0

  Service-policy output: queue-on-dscp

    Class-map: dscp-ef (match-all)
      279830 packets, 17909120 bytes
      30 second offered rate 51000 bps, drop rate 2000 bps
      Match: ip dscp ef
      Weighted Fair Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 48 (kbps) Burst 1200 (Bytes)
        (pkts matched/bytes matched) 64402/4121728
        (total drops/bytes drops) 97/6208
R3#
```

The only change to this configuration, when compared with the CBWFQ configuration in Example 4-8 is that in the dscp-ef class, inside **policy-map queue-on-dscp**, the **priority** command rather than the **bandwidth** command was used to reserve bandwidth. As seen in the output from the **show policy-map** command, IOS now performs strict-priority queuing on the traffic in class dscp-ef. Also note that the **show policy-map** output shows a burst value was defined (1450 bytes), which is used by the policing function of LLQ. The default burst size is equal to 200 milliseconds of traffic; 58000 bits/second * .2 seconds equals 11600 bits, or 1450 bytes.

Note also the drops experienced by the voice traffic as shown with the **show policy-map interface s 0/0 output class dscp-ef** command. The low-latency queue in this example has experienced zero drops, while providing the lowest possible latency for the voice traffic—which is exactly what you want for voice.

At the end of the example, the **priority** command was changed to reserve only 48 kbps. The two G.729 calls need about 52 kbps, and we know that LLQ polices. Notice that some packets have been dropped according to the final **show policy** command; this particular command was issued less than 10 seconds after changing the **priority** command to use only 48 kbps! In the time it took me to write this paragraph, the number of dropped packets had increased to 2000. Thus, the counters reinforce the fact that LLQ does indeed police the traffic. Therefore, you would need to use CAC mechanisms to prevent oversubscription of the low-latency queue.

LLQ with More Than One Priority Queue

Some Cisco documentation claims that you can only have one low-latency queue inside a single policy map. In other words, only one class can use the **priority** command, making it a low-latency queue. Other Cisco documentation claims that you can have more than one low-latency queue in a single policy map.

As it turns out, you can have multiple low-latency queues in a single policy map. Why would you need more than one low-latency queue in a policy map, and how would it work? Well, it's actually pretty simple, now that you know how to configure LLQ.

First, imagine a policy map that has one low-latency queue configured with the **priority 400** command. This queue needs 320 kbps for a single video conference call, plus three G.729 voice calls totaling about 80 kbps. If only the three voice calls and the single video-conference call occur, the LLQ configuration works fine, the policer does not drop any packets, and the voice and video packets are processed as FIFO within that LLQ class. As always, packets in the low-latency queue get serviced ahead of packets in the other non-LLQ classes.

Compare that configuration to a policy map with two low-latency queues defined—one for voice with a **priority 80** command, and another for video conferencing with **priority 320** configured. What's really different about this than the first configuration? The policing, but not the queuing.

With multiple low-latency queues, each class is policed at the configured rate. For instance, with all voice calls going into the class with **priority 80**, three G.729 calls are supported, but not more than that. With video-conferencing traffic going into the class with **priority 320**, only that single video call is supported.

The fact that the different types of traffic are policed separately in the second example provides the motivation to use multiple low-latency queues. Suppose that a fourth voice call were made, and the CAC tools in place did not prevent the call, meaning that more than 80 kbps of voice traffic was being sent. With a single low-latency queue, both video and some voice packets would be discarded due to policing, because the cumulative rate for all traffic would exceed 400 kbps. The policer would have no way to decide to discard just the extra voice, but not video, because the policer acts on all traffic in the class. However, with the two lower-latency queues configuration, only the voice calls would lose packets due to policing, and the video conference would not have any packets dropped by the policer.

In effect, with multiple low-latency queues, you get more granularity in what you police. In fact, the most typical reason to use multiple low-latency queues is to support voice in one queue, and video in another.

Queuing does not differ when comparing using a single low-latency queue with multiple low-latency queues in a single policy map. IOS always takes packets from the low-latency queues first, as compared with the non-low-latency queues (those with a **bandwidth** command), just like before. However, IOS does not reorder packets between the various low-latency queues inside the policy map. In other words, IOS treats all traffic in all the low-latency queues with FIFO logic.

In short, using multiple low-latency queues in one policy map does enable you to police traffic more granularly, but it does not reorder packets among the various low-latency queues.

Unfortunately, the DQOS and QoS courses and exams disagree with each other about where you can have more than one low-latency queue. The DQOS course states that only one low-latency queue is allowed in a single policy map, whereas the QoS course states that multiple low-latency queues are allowed. So, you may talk to colleagues who have seen some Cisco documentation, or taken the DQOS class, and swear that only one low-latency queue is allowed. Thankfully, the chances are low that you would see a question distinguishing this fine point. However, you should be aware of the differences in documentation and the courses, so we made it a point to draw attention to the differences here.

IP RTP Priority

Before Cisco created LLQ, Cisco created two other tools that added a single priority queue to another queuing tool. IP RTP Reserve was the first of these two tools to be introduced, followed by the IP RTP Priority feature. With current IOS releases, Cisco recommends not using these older tools, but instead suggests using LLQ. However, because IP RTP Priority is currently on QoS exams, a brief explanation is warranted.

IP RTP Priority provides many features like LLQ, but with a few differences. It adds a PQ to either WFQ or to CBWFQ. It classifies VoIP payload traffic by classifying packets in a range of UDP port numbers, from which it only selects traffic from the even-numbered (VoIP payload) ports. Like LLQ, the PQ created by IP RTP Priority always gets serviced first, but the bandwidth reserved for this queue is policed to prevent it from taking over all bandwidth. Table 4-14 summarizes the main features and compares the features with LLQ.

Table 4-14 *Comparison of LLQ and IP RTP Priority Features*

Feature	LLQ	IP RTP Priority
Adds a priority queue to WFQ	No	Yes
Adds a priority queue to CBWFQ	Yes	Yes
Can classify on even UDP ports in a specified range	Yes	Yes
Can classify on anything MQC can use to classify	Yes	No
Reserves a configured amount of bandwidth	Yes	Yes
Bandwidth is policed, so priority queue cannot exceed the configured bandwidth	Yes	Yes
Currently recommended best queuing tool for low latency	Yes	No

IP RTP Priority Configuration

To configure IP RTP Priority, you just add one additional configuration command to an interface that already uses WFQ. There are no additional **show** commands to list information about IP RTP Priority beyond the commands that apply to the underlying queuing tools (WFQ and CBWFQ). Table 4-15 lists the two configuration commands, and Example 4-9 lists several example configurations.

Table 4-15 *Command Reference for IP RTP Priority*

Command	Mode and Function
ip rtp priority *starting-rtp-port-number port-number-range bandwidth*	Interface configuration mode; enables IP RTP Priority on a subinterface or interface
frame-relay ip rtp priority *starting-rtp-port-number port-number-range bandwidth*	FRTS (Frame Relay traffic shaping) map-class configuration mode; enables IP RTP Priority in an FRTS map class, which is then enabled on a subinterface or DLCI (data-link connection identifier)

Example 4-9 *IP RTP Priority Configuration Examples*

```
! Example 1
interface serial 1/0
 description this interface will use WFQ and IP RTP Priority, with 30 kbps reserved
 encapsulation hdlc
 fair-queue
```

continues

Example 4-9 *IP RTP Priority Configuration Examples (Continued)*

```
 ip rtp-priority 16384 16383 30
!
! Example 2
interface serial 1/1
 description this interface will use CBWFQ and IP RTP Priority, with 30 kbps reserved
 encapsulation frame-relay
 service-policy output my-policy
 ip rtp-priority 16384 16383 30
!
! Example 3
interface serial 1/3
 encapsulation frame-relay
 frame-relay traffic-shaping
interface serial 1/3.1 point-to-point
 description this subinterface will use FRTS per DLCI, plus WFQ and IP RTP Priority
 frame-relay interface-dlci 100
  class my-shape
!
map-class my-shape
 frame-relay cir 128000
 Frame-relay fair-queue
 Frame-relay ip rtp-priority 16384 16383 30
```

Example 4-9 lists three different configurations for IP RTP Priority. The first example, configured on serial interface 1/0, shows a simple configuration on a point-to-point serial link using High-level Data Link Control (HDLC) encapsulation. The **ip rtp-priority 16384 16383 30** command enables the feature, matching even-numbered UDP ports between 16384 and 32767. (The **16384** parameter is the base port number, and the **16383** parameter is the number of ports starting at the base port number.) The last parameter, **30**, sets the guaranteed minimum bandwidth and the policing rate, just as the **priority** command did for LLQ.

The second example just shows IP RTP Priority enabled on a serial interface using Frame Relay. The **ip rtp priority** command enables the function on Frame Relay interfaces as well as HDLC interfaces. Also in the second example, RTP Priority is configured to work with CBWFQ, as evidenced by the **service-policy** subcommand under **serial 1/1**.

Finally, the third example shows RTP Priority along with Frame Relay traffic shaping (FRTS). Chapter 5 explains the details behind FRTS configuration. FRTS parameters are configured as subcommands of a command called **map-class**. Queuing, including IP RTP Priority, can be configured inside an FRTS **map-class**, which is then enabled on an interface, subinterface, or DLCI.

Cisco recommends LLQ be used for delay-sensitive traffic such as VoIP and interactive video traffic. However, in cases where CBWFQ and LLQ are not an option (for instance, when the IOS release is earlier than 12.0(7)T/12.1 Mainline), IP RTP Priority provides a good alternative for queuing delay-sensitive traffic.

Summary of Queuing Tool Features

The classification feature and the scheduler are the two most important features of a queuing tool. The drop policy, and the maximum number of queues supported, is also very important. Although this chapter covers these features in detail, a brief summary may be useful to avoid losing some of the details.

Table 4-16 lists the queuing tools covered in detail in this chapter, cross-referenced with the classification fields that each supports.

Table 4-16 *Classification Fields Used by Classification and Marking Tools*

Classification Field*	WFQ	CBWFQ	LLQ	RTP Priority
Extended IP ACLs		X	X	
Incoming interface		X	X	
Packet length				
Fragment				
TCP/UDP port numbers		X	X	
Flow (S/D address and port, plus protocol	X			
ToS byte	X**			
QoS group		X	X	
Even-numbered UDP ports, in a prespecified range		X	X	X
IP Precedence		X	X	
IP DSCP		X	X	
CoS		X	X	
NBAR		X	X	
MPLS Experimental bits	X	X	X	
Input interface	X	X	X	

* Some Fields can be matched via ACL as well as being matched directly. For instance, DSCP can be matched with an ACL. In the table, if an X is listed, it means the tool can directly match that field.

** As listed in the chapter, some documents claim that WFQ also classifies based on the ToS byte.

Table 4-17 lists the tools covered in this chapter, along with a brief synopsis of the scheduler, drop, and maximum queues supported.

Table 4-17 *Summary of Scheduler, Drop, and Number of Queues*

Tool	Scheduler	Drop Policy	Max # of Queues
FIFO	Services packets in the same order that they arrived.	Tail drop	1
PQ	Always services higher-priority queues first; the result is great service for the High queue, with potential for 100% of link bandwidth. Service degrades quickly for lower-priority queues.	Tail drop	4
CQ	Services packets from a queue until a byte count is reached; round-robins through the queues, servicing the different byte counts for each queue. The effect is to reserve a percentage of link bandwidth for each queue.	Tail drop	16
WFQ	Services lowest sequence number (SN). SNs assigned when packet placed into queue, as a function of length and precedence.	Modified tail drop*	4096
CBWFQ	Algorithm is not published. The result of the scheduler provides a percentage guaranteed bandwidth to each queue.	Tail drop or WRED	64
LLQ	Always services low-latency queue first, but each low-latency queue is policed to prevent it from dominating the link.	Tail drop or WRED	64
IP RTP Priority	Always services low-latency queue first, but queue is policed to prevent it from dominating the link.	Tail drop	1

* WFQ's modified tail drop includes a per-queue limit, an aggregate limit for all queues, with the ability to dequeue a previously enqueued packet if the new packet has a better SN.

Certainly, one of the difficulties in passing any QoS exam is memorizing some of the details that, frankly, you would never need to memorize to do your job well. Take the time to read the "Foundation Summary," review the example configurations scattered throughout this book, and focus on tables such as the two in this summary section to memorize the details long enough to pass the exam!

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final preparation before the exam, these tables and figures are a convenient way to review the day before the exam.

Table 4-18 outlines the key features of queuing tools, with a brief definition of each feature.

Table 4-18 *Key Concepts When Comparing Queuing Tools*

Feature	Definition	QoS Characteristic Affected
Classification	The capability to examine packets to determine into which queue the packet should be placed. Many options are available.	None
Drop policy	When the queue has been determined, the drop policy defines the rules by which the router chooses to drop the packet. Tail drop, modified tail drop, WRED (Weighted Random Early Detect), and FRED (Flow-Based Random Early Detect) are the main options.	Loss
Scheduling inside a single queue	Inside a single queue, packets can be reordered. In most cases, however, FIFO logic is used for packets inside each queue.	Bandwidth, delay, jitter, and loss
Scheduling between different queues	The logic that defines how queuing chooses the next packet to take from the output queues and place it in the TX Queue (Transmit Queue).	Bandwidth, delay, jitter, and loss
Maximum number of queues	The maximum number of different queues the queuing tools support, which in turn implies the maximum number of traffic classifications that can be treated differently by the queuing method.	None
Maximum queue length	The maximum number of packets in a single queue.	Loss, delay

Figure 4-23 depicts the TX Queue, along with a single FIFO Queuing output queue.

Figure 4-23 *Single FIFO Output Queue, with a Single TX Queue*

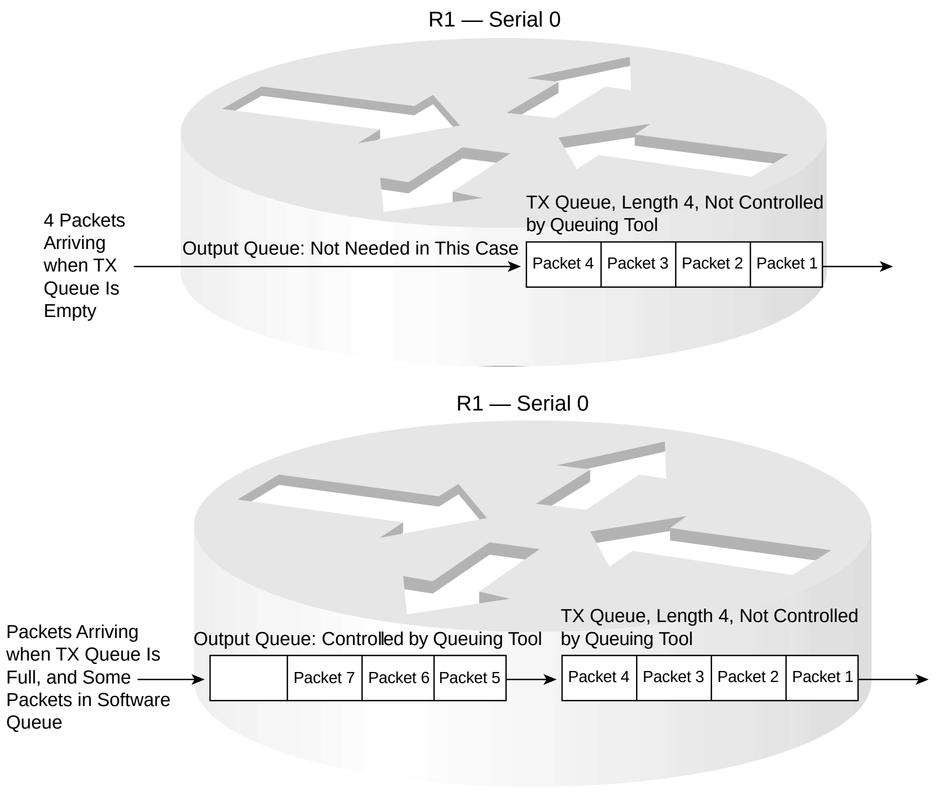
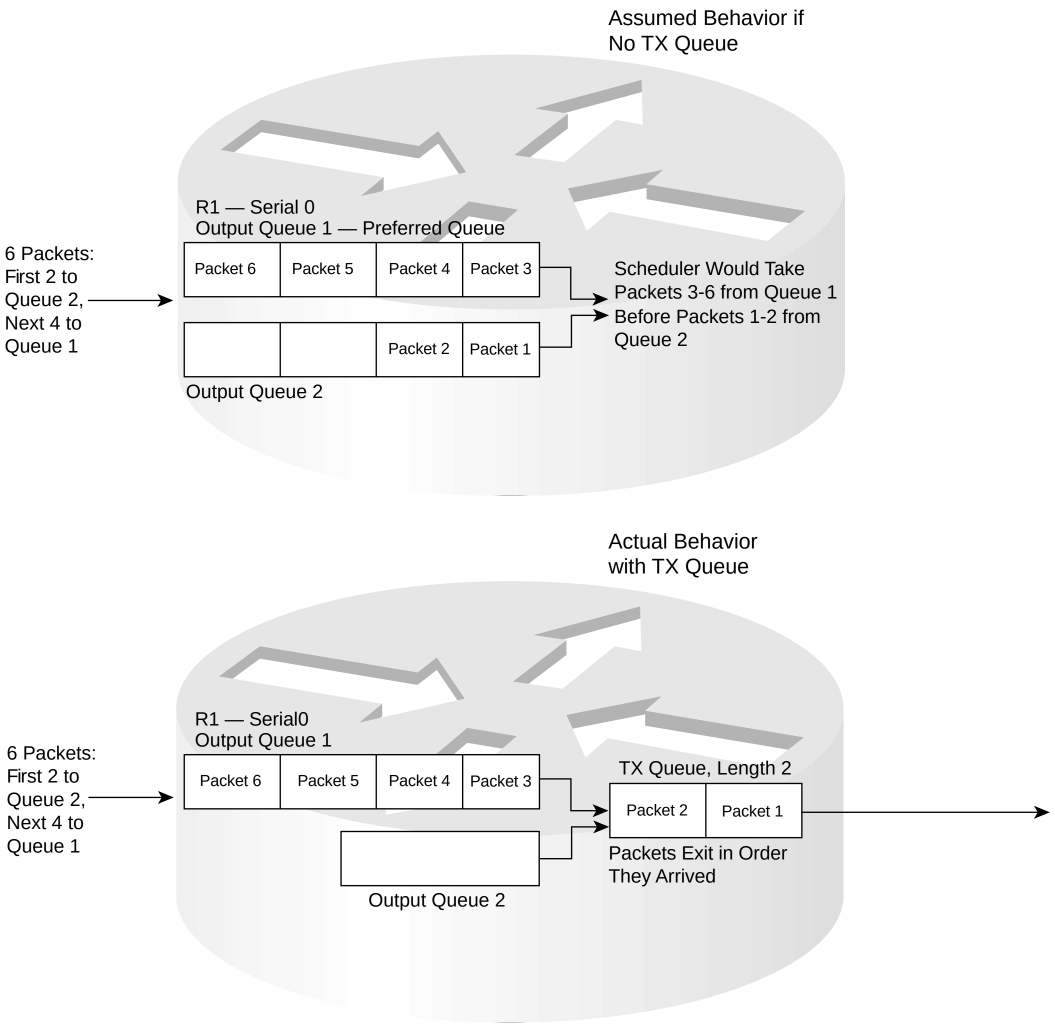

R1 — Serial 0
4 Packets Arriving when TX Queue Is Empty
Output Queue: Not Needed in This Case
TX Queue, Length 4, Not Controlled by Queuing Tool
Packet 4
Packet 3
Packet 2
Packet 1
R1 — Serial 0
Packets Arriving when TX Queue Is Full, and Some Packets in Software Queue
Output Queue: Controlled by Queuing Tool
Packet 7
Packet 6
Packet 5
TX Queue, Length 4, Not Controlled by Queuing Tool
Packet 4
Packet 3
Packet 2
Packet 1

Figure 4-24 shows how TX Queues affect queuing. With queuing configured with two queues, seven packets arrive, numbered in the order in which they arrive. The output queuing configuration specifies that the first two packets (1 and 2) should be placed into Queue 2, and the next four packets (numbered 3 through 6) should be placed into Queue 1.

Figure 4-24 *Two Output Queues, with Scheduler Always Servicing Queue 1 Rather Than Queue 2 When Packets Are in Queue 1*


Assumed Behavior if No TX Queue
R1 — Serial 0
Output Queue 1 — Preferred Queue
6 Packets: First 2 to Queue 2, Next 4 to Queue 1
Packet 6
Packet 5
Packet 4
Packet 3
Scheduler Would Take Packets 3-6 from Queue 1 Before Packets 1-2 from Queue 2
Packet 2
Packet 1
Output Queue 2
Actual Behavior with TX Queue
R1 — Serial0
Output Queue 1
6 Packets: First 2 to Queue 2, Next 4 to Queue 1
Packet 6
Packet 5
Packet 4
Packet 3
TX Queue, Length 2
Packet 2
Packet 1
Packets Exit in Order They Arrived
Output Queue 2

The following list summarizes the key points about TX Rings and TX Queues in relation to their effect on queuing:

- The TX Queue/TX Ring always performs FIFO scheduling, and cannot be changed.
- The TX Queue/TX Ring uses a single queue, per interface.

- IOS shortens the interface TX Queue/TX Ring automatically when an output queuing method is configured
- You can configure the TX Ring/TX Queue length to a different value.

To delay the traffic, traffic shaping places the packets into the queue associated with the subinterface or DLCI and drains the traffic from the shaping queue at the shaped rate. Figure 4-25 shows the structure of the queues on a subinterface, interface, and the TX Queue, when shaping is enabled.

Figure 4-25 *Shaping Queues, Interface Queues, and TX Ring*

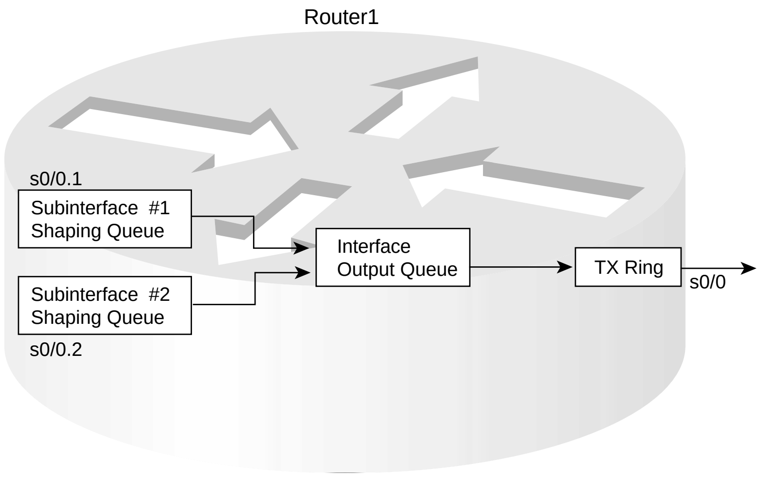


Table 4-19 summarizes some of the key features of PQ.

Table 4-19 *PQ Functions and Features*

PQ Feature	Explanation
Classification	Classifies based on matching an ACL for all Layer 3 protocols, incoming interface, packet size, whether the packet is a fragment, and TCP and UDP port numbers.
Drop policy	Tail drop.
Maximum number of queues	4.
Maximum queue length	Infinite; really means that packets will not be tail dropped, but will be queued.

Table 4-19 *PQ Functions and Features (Continued)*

PQ Feature	Explanation
Scheduling inside a single queue	FIFO.
Scheduling among all queues	Always service higher-priority queues first; result is great service for the High queue, with potential for 100% of link bandwidth. Service degrades quickly for lower-priority queues.

Table 4-20 summarizes some of the key features of CQ.

Table 4-20 *CQ Functions and Features*

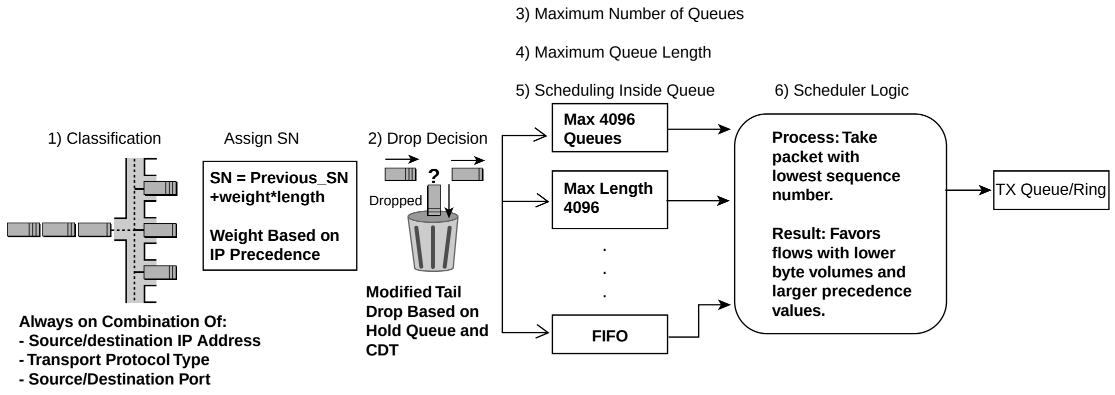
CQ Feature	Explanation
Classification	Classifies based on matching an ACL for all Layer 3 protocols, incoming interface, packet size, whether the packet is a fragment, and TCP and UDP port numbers.
Drop policy	Tail drop.
Number of queues	16.
Maximum queue length	Infinite; really means that packets will not be tail dropped, but will be queued.
Scheduling inside a single queue	FIFO.
Scheduling among all queues	Services packets from a queue until a byte count is reached; round-robins through the queues, servicing the different byte counts for each queue. The effect is to reserve a percentage of link bandwidth for each queue.

Flow-Based WFQ, or simply WFQ, classifies traffic into flows. Flows are identified by at least five items in an IP packet.

- Source IP address
- Destination IP address
- Transport layer protocol (TCP or UDP) as defined by the IP Protocol header field
- TCP or UDP source port
- TCP or UDP destination port

For perspective on the sequence of events for WFQ, marking the sequence number, and serving the queues, examine Figure 4-26.

Figure 4-26 *WFQ—Assigning Sequence Numbers and Servicing Queues*


3) Maximum Number of Queues
4) Maximum Queue Length
5) Scheduling Inside Queue
6) Scheduler Logic
1) Classification
Assign SN
2) Drop Decision
SN = Previous_SN +weight*length
Weight Based on IP Precedence
Dropped
?
Modified Tail Drop Based on Hold Queue and CDT
Max 4096 Queues
Max Length 4096
FIFO
Process: Take packet with lowest sequence number.
Result: Favors flows with lower byte volumes and larger precedence values.
TX Queue/Ring
Always on Combination Of:
- Source/destination IP Address
- Transport Protocol Type
- Source/Destination Port

WFQ calculates the sequence number (SN) before adding a packet to its associated queue. The formula for calculating the SN for a packet is as follows:

Previous_SN + weight * new_packet_ength

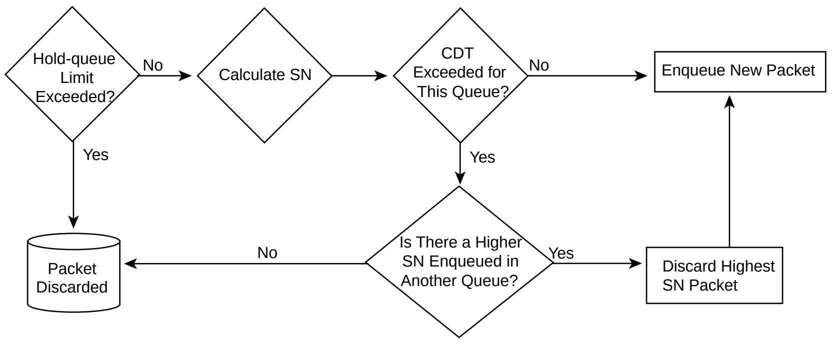
Table 4-21 lists the weight values used by WFQ before and after the release of 12.0(5)T/12.1.

Table 4-21 *Weight Values Used by WFQ*

Precedence	Before 12.0(5)T/12.1	After 12.0(5)T/12.1
0	4096	32384
1	2048	16192
2	1365	10794
3	1024	8096
4	819	6476
5	682	5397
6	585	4626
7	512	4048

WFQ discards some packet when a queue's *congestive discard threshold* (CDT) has been reached. To appreciate how the CDT is used, examine Figure 4-27.

Figure 4-27 *WFQ Modified Tail Drop and Congestive Discard Threshold*


Hold-queue Limit Exceeded?
No
Calculate SN
CDT Exceeded for This Queue?
No
Enqueue New Packet
Yes
Yes
Packet Discarded
No
Is There a Higher SN Enqueued in Another Queue?
Yes
Discard Highest SN Packet

Tables 4-22 and 4-23 list the configuration and exec commands related to WFQ.

Table 4-22 *Configuration Command Reference for WFQ*

Command	Mode and Function
fair-queue [*congestive-discard-threshold* [*dynamic-queues* [*reservable-queues*]]]	Interface configuration mode; enables WFQ, sets the CDT, sets maximum number of queues, and sets the number reserved for RSVP use
hold-queue *length* {**in** | **out**}	Interface configuration mode; changes the length of the hold queue

Table 4-23 *Exec Command Reference for WFQ*

Command	Function
show queue *interface-name interface-number* [**vc** [*vpi/*] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing [**custom** | **fair** | **priority** | **random-detect** [**interface** *atm-subinterface* [vc [[*vpi*/] *vci*]]]]	Lists configuration and statistical information about the queuing tool on an interface

Table 4-24 summarizes some of the key features of WFQ.

Table 4-24 *WFQ Functions and Features*

WFQ Feature	Explanation
Classification	Classifies without configuration, based on source/destination IP address/ port, protocol type (TCP|UDP), and ToS.
Drop policy	Modified tail drop.
Number of queues	4096.
Maximum queue length	Congestive discard threshold per queue (max 4096), with an overall limit based on the hold queue for all queues (max 4096).
Scheduling inside a single queue	FIFO.
Scheduling among all queues	Serves lowest sequence number (SN). The SN is assigned when the packet is placed into the queue, as a function of length and precedence.

Table 4-25 summarizes some of the key features of CBWFQ.

Table 4-25 *CBWFQ Functions and Features*

CBWFQ Feature	Description
Classification	Classifies based on anything that MQC commands can match, just like CB marking. Includes all extended IP ACL fields, NBAR, incoming interface, CoS, precedence, DSCP, source/destination MAC, MPLS Experimental, QoS group, and RTP port numbers
Drop policy	Tail drop or WRED, configurable per queue.
Number of queues	64.
Maximum queue length	64.
Scheduling inside a single queue	FIFO on 64 queues; FIFO or WFQ on class-default queue.
Scheduling among all queues	Algorithm is not published. The result of the scheduler provides a percentage guaranteed bandwidth to each queue.

All the commands for CBWFQ are repeated for reference in Tables 4-26 and 4-27.

Table 4-26 *Command Reference for CBWFQ*

Command	Mode and Function
class-map *class-map-name*	Global config; names a class map, where classification options are configured.
match …	Class map subcommand; defines specific classification parameters.

Table 4-26 *Command Reference for CBWFQ (Continued)*

Command	Mode and Function
match access-group {*access-group* | **name** *access-group-name*}	Access-control list (ACL).
match source-address mac *address*	Source MAC address.
match ip precedence *ip-precedence-value* [*ip-precedence-value ip-precedence-value ip-precedence-value*]	IP precedence.
match mpls experimental *number*	MPLS Experimental.
match cos *cos-value* [*cos-value cos-value cos-value*]	CoS.
match destination-address mac *address*	Destination MAC address.
match input-interface *interface-name*	Input interface.
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	IP DSCP.
match ip rtp *starting-port-number port-range*	RTP's UDP port number range.
match qos-group *qos-group-value*	QoS group.
match protocol *protocol-name*	NBAR protocol types.
match protocol citrix [**app** *application-name-string*].	NBAR Citrix applications.
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	Host name and URL string.
match any	Matches any and all packets.
policy-map *policy-map-name*	Global config; names a policy, which is a set of actions to perform.
class *name*	Policy map subcommand; identifies the packets to perform QoS actions on by referring to the classification logic in a class map
bandwidth {*bandwidth-kbps* | **percent** *percent*}	Class subcommand; sets literal or percentage bandwidth for the class. Must use either use actual bandwidth or percent on all classes in a single policy map.
fair-queue [**queue-limit** *queue-value*]	Class subcommand; enables WFQ in the class (class-default only).
random-detect dscp *dscpvalue min-threshold max-threshold* [*mark-probability-denominator*]	Class subcommand; enables DSCP-based WRED in the class.

continues

Table 4-26 *Command Reference for CBWFQ (Continued)*

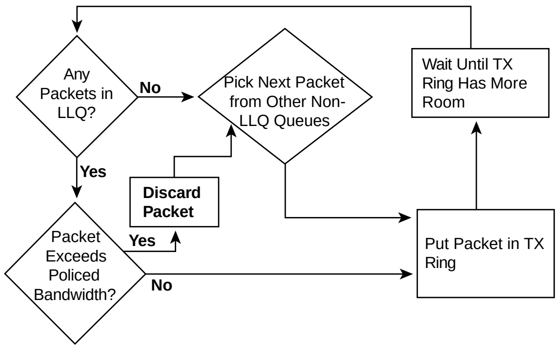
Command	Mode and Function
random-detect precedence *precedence min-threshold max-threshold mark-prob-denominator*	Class subcommand; enables precedence-based WRED in the class.
max-reserved-bandwidth *percent*	Interface subcommand; defines the percentage of link bandwidth that can be reserved for CBWFQ queues besides class-default.

Table 4-27 *Exec Command Reference for CBWFQ*

Command	Function
show policy-map *policy-map-name*	Lists configuration information about all MQC-based QoS tools
show policy-map *interface-spec* [**input** | **output**] [**class** *class-name*]	Lists statistical information about the behavior of all MQC-based QoS tools

To prevent LLQ from having the same problem as PQ, where packets in the highest-priority queue could dominate, LLQ's scheduler actually works as shown in Figure 4-28.

Figure 4-28 *Servicing Queues with LLQ and CBWFQ—The Real Story*


Any Packets in LLQ?
No
Pick Next Packet from Other Non-LLQ Queues
Wait Until TX Ring Has More Room
Yes
Discard Packet
Yes
Packet Exceeds Policed Bandwidth?
No
Put Packet in TX Ring

The single additional configuration command for LLQ is listed in Table 4-28.

Table 4-28 *Command Reference for LLQ*

Command	Mode and Function
priority{*bandwidth-kbps* | **percent** *percentage*} [*burst*]	Class subcommand; enables LLQ in this class, reserves bandwidth, and enables the policing function. The burst for the policer can also be configured with this command.

Table 4-29 summarizes the main features of IP RTP Priority, and compares the features with LLQ.

Table 4-29 *Comparison of LLQ and IP RTP Priority Features*

Feature	LLQ	IP RTP Priority
Adds a priority queue to WFQ	No	Yes
Adds a priority queue to CBWFQ	Yes	Yes
Can classify on even UDP ports in a specified range	Yes	Yes
Can classify on anything MQC can use to classify	Yes	No
Reserves a configured amount of bandwidth	Yes	Yes
Bandwidth is policed, so priority queue cannot exceed the configured bandwidth	Yes	Yes
Currently recommended best queuing tool for low latency	Yes	No

Table 4-30 lists the two configuration commands used with IP RTP Priority.

Table 4-30 *Command Reference for IP RTP Priority*

Command	Mode and Function
ip rtp priority *starting-rtp-port-number port-number-range bandwidth*	Interface configuration mode; enables IP RTP Priority on a subinterface or interface
frame-relay ip rtp priority *starting-rtp-port-number port-number-range bandwidth*	FRTS (Frame Relay traffic shaping) map-class configuration mode; enables IP RTP Priority in an FRTS map class, which is then enabled on a subinterface or DLCI (data-link connection identifier)

Table 4-31 summarizes the details of the scheduler, drop, and maximum queues supported for the queuing tools covered in detail in this chapter.

Table 4-31 *Summary of Scheduler, Drop, and Number of Queues*

Tool	Scheduler	Drop Policy	Max # of Queues
FIFO	Services packets in the same order that they arrived.	Tail drop	1
PQ	Always services higher-priority queues first; the result is great service for the High queue, with potential for 100% of link bandwidth. Service degrades quickly for lower-priority queues.	Tail drop	4

continues

Table 4-31 *Summary of Scheduler, Drop, and Number of Queues (Continued)*

Tool	Scheduler	Drop Policy	Max # of Queues
CQ	Services packets from a queue until a byte count is reached; round-robins through the queues, servicing the different byte counts for each queue. The effect is to reserve a percentage of link bandwidth for each queue.	Tail drop	16
WFQ	Services lowest sequence number (SN). SNs assigned when packet placed into queue, as a function of length and precedence.	Modified tail drop*	4096
CBWFQ	Algorithm is not published. The result of the scheduler provides a percentage guaranteed bandwidth to each queue.	Tail drop or WRED	64
LLQ	Always services low-latency queue first, but each low-latency queue is policed to prevent it from dominating the link.	Tail drop or WRED	64
IP RTP Priority	Always services low-latency queue first, but queue is policed to prevent it from dominating the link.	Tail drop	1

* WFQ's modified tail drop includes a per-queue limit, an aggregate limit for all queues, with the ability to dequeue a previously enqueued packet if the new packet has a better SN.

Also make sure and review Tables 4-16 and 4-17 immediately preceding the Foundation Summary.

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD exam to include, or not include, the topics that are only on the CCIP QoS.

Conceptual Questions

1. Describe the benefits of having a single FIFO output queue.
2. Explain the effects of changing a single FIFO queue's length to twice its original value. Include comments about how the change affects bandwidth, delay, jitter, and loss.
3. Explain the purpose of a TX Ring and TX Queue in a Cisco router.
4. Explain how a long TX Ring might affect the behavior of a queuing tool.
5. Describe the command output that identifies the length of the TX Ring or TX Queue, and whether the length was automatically lowered by IOS.
6. Explain under what circumstances the TX Ring, interface output queues, and subinterface output queues both fill and drain, and to where they drain.

Priority Queuing and Custom Queuing

7. Assume a queuing tool has been enabled on interface S0/0. Describe the circumstances under which the queuing tool would actually be used.
8. Explain the circumstances under which it would be useful to enable a queuing tool on a subinterface.
9. Describe the classification feature of Priority Queuing, including the list of items that can be examined for classification decisions.
10. Describe the classification feature of Custom Queuing, including the list of items that can be examined for classification decisions.
11. List the classification options available to Custom Queuing that are not also available to Priority Queuing.

12 Describe the process and end result of the scheduling feature of Priority Queuing.

13 Describe the process and end result of the scheduling feature of Custom Queuing.

14 List the maximum number of queues used by Priority Queuing and Custom Queuing.

WFQ

15 Characterize the effect the WFQ scheduler has on different types of flows.

16 Describe the WFQ scheduler process. Include at least the concept behind any formulas, if not the specific formula.

17 You previously disabled WFQ on interface S0/0. List the minimum number of commands required to enable WFQ on S0/0.

18 What commands list statistical information about the performance of WFQ?

19 Define what comprises a flow in relation to WFQ.

20 You just bought and installed a new 3600 series router. Before adding any configuration to the router, you go ahead and plug in the new T1 Frame Relay access link to interface S0/0. List the minimum number of commands required to enable WFQ on S0/0.

CBWFQ, LLQ, IP RTP Priority

21 Describe the CBWFQ scheduler process, both inside a single queue and among all queues.

22 Describe how LLQ allows for low latency while still giving good service to other queues.

23 Compare and contrast IP RTP Priority and LLQ. In particular, mention what other queuing tools can be used concurrently with each, how each classifies packets, and which is recommended by Cisco.

24 Compare and contrast the CBWFQ command that configures the guaranteed bandwidth for a class with the command that enables LLQ for a class.

25 Describe the CBWFQ classification options. List at least five fields that can be matched without using an ACL.

26 Name the two CBWFQ global configuration commands that define classification options, and then the per-hop behaviors, respectively. Also list the command that enables CBWFQ on an interface.

27 List the command used to configure IP RTP Priority on a serial link.

28 Characterize the type of traffic that can be queued using both IP RTP priority and LLQ. List specific port numbers and IP addresses as applicable, and describe the type of traffic.

29 Examine the following configuration (see Example 4-10). Which of the five policy maps would certainly enable LLQ for voice payload traffic, based only of the information in the configuration?

Example 4-10 *Exhibit for CBWFQ Configuration Questions*

```
!
class-map match-all class1
  match ip rtp 16384 16383
class-map match-all class2
  match access-group 101
class-map match-all class3
  match ip rtp 16384 32767
class-map match-all class4
  match ip dscp ef
class-map match-all class4
  match access-group 102
!
policy-map pmap1
 class class1
  priority 60
policy-map pmap2
 class class2
  priority 60
policy-map pmap3
 class class3
  priority 60
policy-map pmap4
 class class4
  priority 60
policy-map pmap5
 class class5
  priority 60
!
interface Serial0/0
 service-policy output ?????
!
access-list 101 permit udp any gt 16383 any gt 16383
access-list 102 permit udp any range 16383 32767 any range 16383 32767
!
```

30 Using the same exhibit as in the preceding example, describe what must also be true for pmap4 to queue voice payload traffic successfully, and only voice payload traffic, in a low-latency queue.

Comparing Queuing Tool Options

31 Which of the following queuing tools allows for WRED inside a single queue? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

32 Which of the following queuing tools can always service a particular queue first, even when other queues have packets waiting? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

33 Which of the following queuing tools allows for a percentage bandwidth to be assigned to each queue? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

34 Which queuing tools could be configured to provide the lowest possible latency for voice traffic? Of these, which does Cisco recommend as the best option for voice queuing today?

35 Which of the following queuing tools can use flow-based classification? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

36 Which of the following queuing tools uses the Modular QoS CLI? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

37 Which of the following queuing tools allows for a value to be configured, which then results in a specific number of bytes being taken from each queue during a round-robin pass through the queues? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

38 Which of the following queuing tools allow the largest number of queues for a flow-based tool? For a class-based tool? What are the maximum values? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

39 Which queuing tools could be configured to provide the lowest possible latency for voice signaling traffic?

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Topics

- Describe the difference between policing and shaping and how each ones relates to QoS.
- Describe the various mechanisms for policing, when to apply each, and how to configure them.
- Identify the various types of traffic shaping, their differences, and how to apply each.
- Configure the different types of traffic shaping.

QoS Exam Objectives

- Describe shaping and policing mechanisms available in Cisco IOS.
- Describe the benefits and drawbacks of traffic shaping and policing mechanisms.
- Describe Generic Traffic Shaping (GTS) on Cisco routers.
- Configure GTS on Cisco routers.
- Describe Frame Relay Traffic Shaping (FRTS) and its differences from GTS.
- Configure Frame Relay Traffic Shaping (FRTS) on Cisco routers.
- Describe the Committed Access Rate (CAR) mechanism including its benefits and drawbacks.
- Configure Committed Access Rate (CAR) on Cisco Routers.

CHAPTER 5

Traffic Policing and Shaping

Traffic policing allows devices in one network to enforce a traffic contract. *Traffic contracts* define how much data one network can send into another, typically expressed as a committed information rate (CIR) and a committed burst (Bc). *Policing* measures the flow of data, and discards packets that exceed the traffic contract.

Similarly, traffic shaping allows devices to conform to a traffic contract. In cases where packets that exceed the traffic contract might be discarded, the sending device may choose just to slow down its sending rate, so that the packets are not discarded. The process of sending the traffic more slowly than it could be sent, to conform to a traffic contract, is called shaping.

In short, policing typically drops out-of-contract traffic, whereas shaping typically delays it.

Shaping and policing share several concepts and mechanisms. Both need to measure the rate at which data is sent or received, and take action when the rate exceeds the contract. Often when policing is used for packets entering a network, shaping is also used on devices sending into that network. Although shaping and policing are not always used in the same networks, there are more similarities than differences, so both are covered in this single chapter.

"Do I Know This Already?" Quiz

The purpose of the "Do I Know This Already?" quiz is to help you decide whether you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 12-question quiz, derived from the major sections in "Foundation Topics" section of this chapter, helps you determine how to spend your limited study time.

Table 5-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 5-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Foundations Topics Section	Questions Covered in This Section
Shaping and Policing Concepts	1–4
Policing with CAR and CB Policer	5–8
Shaping with FRTS, GTS, DTS, and CB Shaping	9–12

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **10 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary," and the "Q&A" sections.
- **11 or 12 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, move to the next chapter.

Shaping and Policing Concepts Questions

1 Explain the points during the process of a single router receiving and forwarding traffic at which shaping and policing can be enabled on a router.

2 Compare and contrast the actions that shaping and policing take when a packet exceeds a traffic contract.

3 If a router has a shaping tool configured, with a shaping rate of 256 kbps, and a Bc of 16,000 bits, what Tc value does the shaping tool use?

4 Define the terms Tc, Bc, Be, and CIR.

Policing with CAR and CB Policer Questions

5 List the command, with the correct syntax, that sets a policed rate of 512 kbps, a Bc of 1 second's worth of traffic, and a Be of an additional 0.5 second's worth of traffic, when using CAR. Do not assume any defaults; explicitly set the values in the command. You can choose any other settings needed for the command.

6 Which policing tools allow for three categories of actions to take?

7 Explain the concept behind re-marking policed packets versus discarding the packets.

8 CB policing has been configured under subinterface s0/0.1. What **show** command lists statistics for CB policing behavior just for that subinterface?

Shaping with FRTS, GTS, DTS, and CB Shaping

9 Along with the **class-map**, **policy-map**, and **service-policy** commands, CB shaping requires one specific command that actually sets values used for the shaping function. List the command, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using CB shaping. Do not assume any defaults; explicitly set the values in the command.

10 Compare and contrast the use of the **class-map** and **map-class** commands in terms of how each is used by FRTS and CB shaping.

11 DTS has been configured under subinterface s0/0.1. What **show** command displays statistics for DTS behavior just for that subinterface?

12 Which of the traffic-shaping tools can be enabled on each VC on a Frame Relay multipoint subinterface?

Foundation Topics

Traffic shaping solves some of the most important issues relating to quality of service (QoS) in networks today. Even when policing is not also used, traffic shaping solves a category of delay and loss problems called *egress blocking*, which can occur in all multiaccess WANs, such as Frame Relay and ATM networks. Traffic shaping is covered extensively in CCNP and CCIE exams and labs, so the coverage in this chapter will help you with other exams as well.

Policing solves specific problems relating to network capacity and traffic engineering. Suppose, for example, that an Internet service provider (ISP) engineers their network to effectively forward packets at rate *x*. Suppose further that the Sales department sells enough access so that the customers all together pay for *x* capacity. However, the customers can collectively send 10*x* into the ISP's network, so everyone suffers. Policing just gives a network engineer the ability to "enforce the law" by discarding excess traffic, much like a real policeman just enforces the law of the local community. Policing can also prevent a single customer from taking too much bandwidth, even if the provider has enough capacity to handle the extra traffic.

This chapter first explains the core concepts of traffic shaping and policing. Following that, each of four traffic-shaping tools—GTS, CB shaping, DTS, and FRTS—are covered, including additional concepts specific to each tool. The two traffic-policing tools, CAR and CB policing, are covered at the end of the chapter.

Traffic-Policing and Traffic-Shaping Concepts

Traffic shaping and traffic policing both measure the rate at which data is sent or received. *Policing* discards excess packets, so that the overall policed rate is not exceeded. *Shaping* enqueues the excess packets, which are then drained from the queue at the shaping rate. In either case, both policing and shaping prevent the traffic from exceeding the bit rate defined to the policer or shaper.

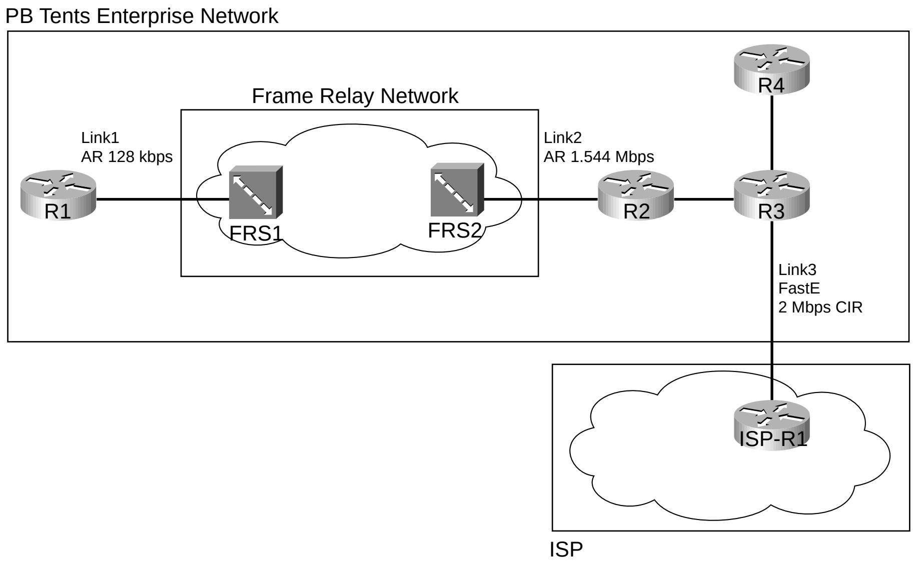
This section covers concepts related to both shaping and policing. It starts with some of the motivations for using shaping and policing. One classic reason to choose to shape occurs when the device at the other end of the link is policing. Suppose, for instance, that R1 sits in an enterprise, and R2 is inside an ISP. R1 sends packets to R2, and R2 polices traffic, discarding any traffic beyond *x* bits per second (bps). The ISP might have chosen to police at R2 to protect the network from accepting too much traffic. R1 could be configured to shape the traffic it sends to R2 to the same rate as the policer at R2, instead of having the excess packets discarded by R2. Other less-obvious reasons for both shaping and policing exist. The upcoming section discusses these.

This section also discusses the mechanisms shaping and policing use to perform their functions. For instance, both policers and shapers must measure bit rates. To measure a rate, a number of bits or bytes over a time period must be observed and calculated. To keep the process simple, shaping and policing use similar mechanisms to account for the numbers of bits and bytes sent over time. First, however, we start with the motivations for using shaping and policing.

When and Where to Use Shaping and Policing

Most implementations of shaping and policing occur at the edges between two different networks. For instance, consider Figure 5-1, which illustrates the two typical cases for shaping and policing. The figure shows PB Tents Enterprise network, with a Frame Relay service, and Internet connectivity using ISP1.

Figure 5-1 *Connection to an ISP, and to a Frame Relay Network*



In this case, PB Tents has three separate boundaries between different networks. Link1, between R1 and the Frame Relay network switch labeled FRS1, is the first boundary. The second boundary is between the switch FRS2 and R2. Finally, a boundary exists between R3 and ISP-R1, over Link3.

For each boundary, the legal documents that detail the agreement between PB Tents and the Frame Relay service provider, and the documents that detail the agreement with ISP1, include something called a *traffic contract*. The need for the contract makes the most sense in the context of Frame Relay, but it also applies to the Internet connection. For instance, R2 uses a T/1 access link into the Frame Relay network. However, the virtual circuit (VC) between R2 and R1 may only include a committed information rate (CIR) of 64 kbps. Similarly, R1 has a 128-kbps access link, and the CIR of the VC to R2 still has a CIR of 64 kbps. When R1 sends a packet, the packet is transmitted at 128 kbps—that's the only speed that physically works! Likewise, R2 must send at 1.5 Mbps. However, the traffic contract may just state that the VC from R1 to R2 allows only 64 kbps in each direction.

Similarly, PB Tents and ISP1 may agree to install a link that is faster than PB Tents actually needs right now, expecting that PB Tent's Internet traffic loads will grow. When PB Tents needs more capacity to the Internet, each party can just agree to a new traffic contract, and PB Tents will pay ISP1 more money. For instance, metro Ethernet/Fast Ethernet/Gigabit Ethernet services have become more common; most sites do not really need 100 Mbps of bandwidth to the Internet. If PB Tents connected to ISP1 using a Fast Ethernet connection, but the traffic contract stated that PB Tents gets only 2 Mbps of service, however, the same mismatch between physical capability and legal ability, as seen with Frame Relay, occurs on the Internet connection.

In short, policing and shaping can play a role in cases where a router can send more traffic than the traffic contract allows. Shaping just slows the rate of sending packets so that the traffic contract is not exceeded. Policing discards some packets so that the traffic contract is not exceeded.

Policing—When and Where?

Whenever the physical clock rate exceeds the traffic contract, policing may be needed. Suppose, for instance, that ISP1 has 1000 customers, just like PB Tents, each with a 100-Mbps connection, and a contract for support of 2 Mbps. What happens over time? Well, without something to prevent it, each customer will send and receive more and more traffic. For a while, all the customers are happy, because their packets make it through the overbuilt ISP1 core. Even if ISP1 has enough capacity to support 10 Mbps of traffic from every customer, eventually, ISP1's network will become overrun, because their customers keep sending more and more traffic, so eventually all traffic will suffer. Queues become congested frequently, causing dropped packets. Multimedia traffic suffers through the poor performance as a result of high delay and jitter. TCP sessions continually decrease their window sizes because of the lost packets, causing synchronization effects inside ISP1. ISP1 can add capacity, but that probably means that ISP1 should start charging more to their customers, who may not be willing to upgrade to a higher-traffic contract.

In actual ISP networks, the network engineers design the core of the network expecting some degree of oversubscription. The term "oversubscription" means that the customer has sent and received more traffic than was contracted, or subscribed. As in the example of ISP1 in the preceding paragraph, ISPs and Frame Relay providers build their network expecting some oversubscription. However, they certainly do not build the core expecting every customer to send traffic at full access rate, all the time.

Policing protects a network from being overrun by traffic. If ISP1 just policed traffic from each customer, discarding packets that exceed the traffic contract, it would protect itself from being overrun. However, the decision to add policing to a network can be politically difficult. Suppose that ISP1 has these 1000 customers, each of whom contracted for 2 Mbps of traffic. Each customer sends and receives more, averaging 10 Mbps, so that ISP1's network is becoming too congested. ISP1 chooses to implement policing, using the contracted rate, discarding packets that exceed 2 Mbps of traffic. Of course, most of their customers will be very unhappy! Such a move may be a career-ending, if not business-ending, choice.

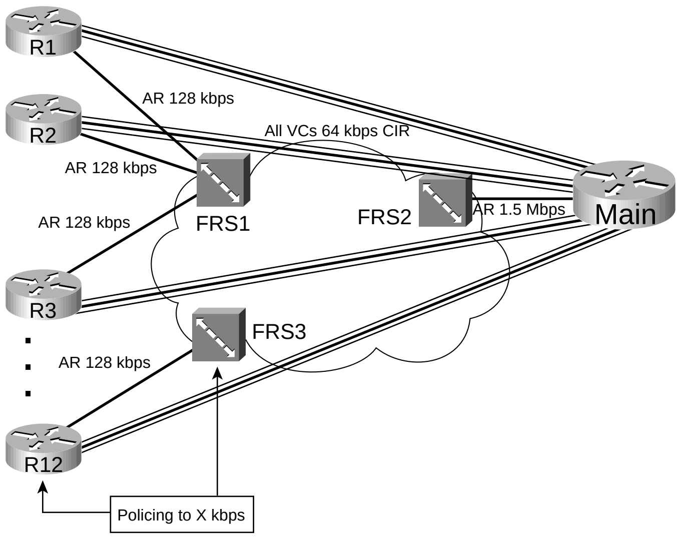
Policers can also just mark down the traffic, instead of discarding it. To do so, the policer marks the packet with a different IP precedence or DSCP value when the traffic rate is exceeded, but it still lets the packet through. Later QoS functions, including policers and packet-drop tools such as Weighted Random Early Detection (WRED), can more aggressively discard marked-down packets as compared with those that have not been marked down. Essentially, the policer can increase the chance that a packet will get discarded somewhere else in the network if that packet causes the traffic rate to be exceeded. Generally speaking, when policers mark down packets, if the network is not currently congested, the packet can get through the network; if congested, the packet is much more likely to be discarded.

ISPs make the business choice of whether to police, and how aggressively to police. The options reduce to the following three basic options:

- Do not police. To support the traffic, build the network to support the traffic as if all customers will send and receive data at the clock rate of the access link. From a sales perspective, close deals by claiming that no policing will be done, but encourage customers who exceed their contracts to pay for more bandwidth.
- Police at the contracted rate. To support these traffic levels, the network only needs to be built to support the collective contracted rates, although the core would be overbuilt to support new customers. From a sales perspective, encourage customers that are beginning to exceed their contracts to upgrade, and give incentives.
- Police somewhere in between the contracted rate and the access-link clock rate. For instance, ISP1 might police PB Tents at 5 Mbps, when the contract reads 2 Mbps. The network can be built to support the collective policed rates. The sales team can encourage customers to buy a larger contracted rate when they consistently exceed the contracted rate, but keep customer satisfaction higher by pointing out their generosity by only policing at rates much higher than the contracted rates.

Policing can be useful in multiaccess WANs (Frame Relay and ATM networks) for the same reason that it was useful for the ISP connection described earlier. Whenever data can be sent faster than the contracted rate, the danger exists that a network will be overrun when many sites exceed their contract at the same time. An example will help you understand a few of the issues. Figure 5-2, the network diagram for PB Tents network, has been expanded to show 12 branches, with a single central site.

Figure 5-2 *PB Tents Network, 12 Frame Relay Branches, 1 Central Site*



Each branch can send traffic at 128 kbps, but each branch only has a contracted 64-kbps CIR on their respective VCs to the main site. If all 12 sites conform to their CIRs, the Frame Relay network should be able to handle the load. If all 12 sites offer 128 kbps of traffic for long periods, however, the provider may still go ahead and try to forward all the traffic, because most Frame Relay providers overbuild their core networks. They also like to imply in their sales pitch that the customer gets to send excess packets for free.

Of course, at some point, if every customer of this provider sent traffic at full line rates for a period of time, the network would probably congest. The same options exist for the Frame Relay network as for an ISP—not to police but build more capacity; police to CIR, and deal with the sales and customer satisfaction issues; or police at something over CIR, and deal with the sales and customer satisfaction issues in slightly different ways.

To police the network in Figure 5-2, the Frame Relay switches can be configured to perform the policing, or the routers can be used. Traditionally, policing is performed as packets enter a network, which would suggest policing as packets enter the Frame Relay switches from the customer. If the service provider actually controls the edge routers in the enterprise network, however, the policing feature can be performed as packets exit the routers, going toward the Frame Relay cloud. If the customer controls the routers at the edge of the cloud, policing in these routers may be risky for the service provider, just because of the possibility that some customers might turn off policing to get more capacity for free.

The Cisco QoS exams cover policing in IOS routers. The exams do not cover policing in Frame Relay switches, or in LAN switches, although the basic concepts are the same.

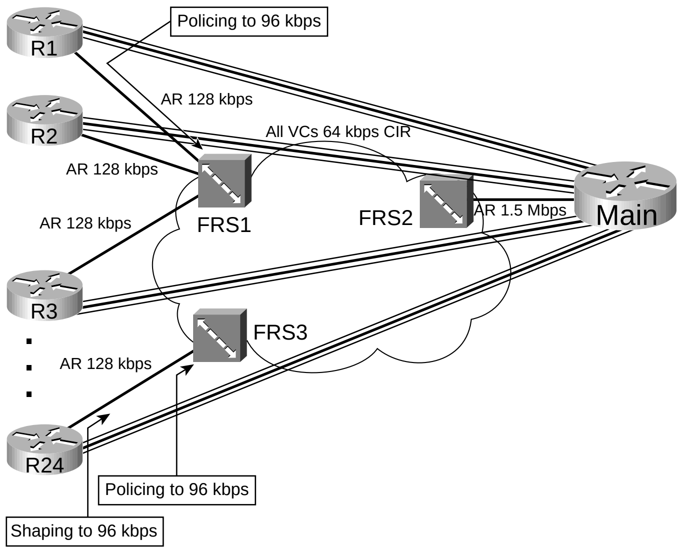
Traffic Shaping—When and Where?

Networks use traffic shaping for two main reasons:

- To shape the traffic at the same rate as policing (if the service provider polices traffic)
- To avoid the effects of egress blocking

For instance, consider Branches 1 and 24 in Figure 5-3. Branch 1 does not shape, whereas Branch 24 does shape to 96 kbps. In both cases, the Frame Relay switches they are connected to police packets at a 96-kbps rate. (The CIR in each case is 64 kbps. Therefore, the service provider is not policing aggressively. The PB Tents engineer wants to get as much bandwidth as possible out of the service, so he shapes at 96 kbps rather than the 64-kbps CIR.)

Figure 5-3 *PB Tents Network, Policing and Shaping, Versus Just Policing*



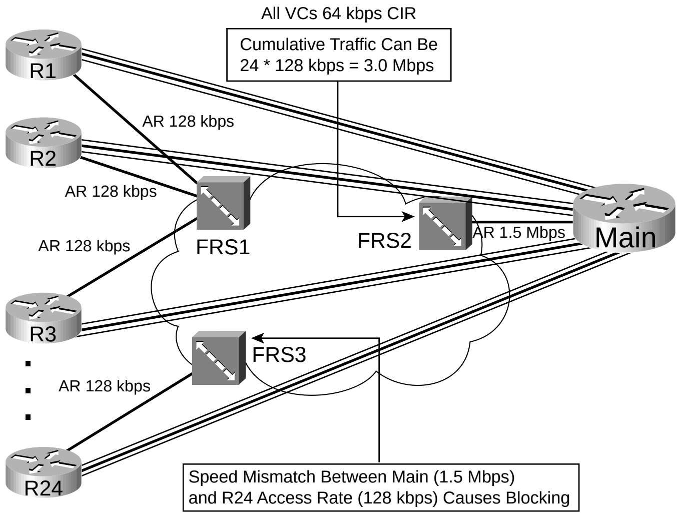
For Branch 1, the absence of shaping ensures that R1 will not artificially delay any packets. However, the policing performed at FRS1 will discard some packets when R1 sends more than 96-kbps worth of traffic. Therefore, some packets will be dropped, although the packets that are not dropped will not experience extra shaping delay. This strategy makes sense when the traffic from Branch 1 is not drop sensitive, but may be delay and jitter sensitive.

For Branch 24, the presence of shaping ensures that R1 will artificially delay some packets. However, the policing performed at FRS3 will not discard packets, because R1 will not send more than 96-kbps worth of traffic. Therefore, no packets will be dropped, although some packets will experience more delay and jitter. This strategy makes sense when the traffic from Branch 24 is drop sensitive, but not delay and jitter sensitive.

The other reason to use shaping is to avoid the effects of egress blocking. *Egress blocking* occurs when packets try to exit a multiaccess WAN, such as Frame Relay and ATM, and cannot exit the network because of congestion. Automobile traffic patterns cause the same kinds of behavior as egress blocking. In the morning, for instance, everyone in the state may try to commute to the same small, downtown area of a big city. Even though an eight-lane highway leads into the city, it may seem that everyone living in the surrounding little towns tries to get off at the few exits of the highway between 7 and 8 a.m. each morning. The highway and exits in the downtown area become congested. Similarly, in the afternoon, if everyone tries to reach the suburbs through one exit off the highway at 5:30 p.m., the eight-lane highway feeding into the two-lane exit road becomes congested. Likewise, although plenty of capacity may exist in a network, egress blocking can occur for packets trying to exit the network.

Figure 5-4 illustrates what happens with egress blocking, using a Frame Relay network as an example.

Figure 5-4 *PB Tents Network, Egress Blocking*


All VCs 64 kbps CIR
Cumulative Traffic Can Be 24 * 128 kbps = 3.0 Mbps
R1
R2
R3
R24
AR 128 kbps
AR 128 kbps
AR 128 kbps
AR 128 kbps
FRS1
FRS2
FRS3
AR 1.5 Mbps
Main
Speed Mismatch Between Main (1.5 Mbps) and R24 Access Rate (128 kbps) Causes Blocking

Suppose that all 24 branches shape at 64 kbps. The cumulative traffic sent by the branches to the main site is 1.5 Mbps, if each branch simultaneously sends 64 kbps. Because the Main router has a T/1 installed, FRS2 should not experience congestion when forwarding packets out of the access link to the Main router. However, what if shaping were not used at the branches? If all 24 branches were to send traffic at 128 kbps (access rate) for a period of time, the cumulative offered load would be about 3 Mbps. Packets would begin to queue trying to exit FRS2's interface connected to the Main router. The packets would experience more delay, more jitter, and eventually more packet drops as the FRS2 output queue filled. Notice that the service provider did not do any policing—egress blocking still occurred, because the branches could collectively overload the egress link between the cloud and the main site.

Interestingly, even if policing were used, and shaping at the branches, egress blocking could still occur. In Figure 5-3, shaping and policing were configured at 96 kbps, because the service provider did not want to be too aggressive in enforcing the traffic contract. With all 24 branches sending 96 kbps at the same time, about 2.25 Mbps of traffic needs to exit FRS2 to get to the Main router. Again, egress blocking can occur, even with policing and shaping enabled!

Similarly, egress blocking can occur right to left in the figure as well. Imagine that the Main router receives 11 consecutive 1500-byte packets from a LAN interface, destined to Branch 24. It takes the Main router roughly 100 milliseconds to send the packets into the Frame Relay network, because its access link is a T/1. When the frames arrive in FRS1, they need to be sent out the access link to R24. However, this access link runs at 128 kbps. To send these 11 packets, it takes slightly more than 1 second just to serialize the packets over the link! Most of the packets then wait in the output queue on FRS3, waiting their turn to be sent. This simple case is another example of egress blocking, sometimes just referred to as a *speed mismatch*.

One solution to the egress blocking problem is to shape the traffic. In the example network, shaping all VCs at the branches to 64 kbps would ensure that the cumulative offered load did not exceed the access rate at the main site. Similarly, if the Main router shaped the VC to R1 to 64 kbps, or even 128 kbps, the egress blocking problem on FRS1 would be solved.

In both cases, however, delay and jitter occurs as a result of the shaping function. Instead of having more queuing delay in the Frame Relay switches, shaping delays occur in the router because packets wait in router output queues. With the queuing occurring in the routers, however, the features of IOS queuing tools can be used to better manipulate the traffic, and give better delay characteristics to delay-sensitive traffic. For instance, with the queues forming in a router, the router can use Low Latency Queuing (LLQ) to dequeue Voice over IP (VoIP) packets first. A Frame Relay switch cannot perform complicated queuing, because the Frame Relay switch does not examine fields outside the Frame Relay or IP header when making forwarding and queuing decisions.

Table 5-2 summarizes some of the key points about the rationale behind when you should use policing and shaping.

Table 5-2 *Policing and Shaping: When to Use Them, and Where*

Topic	Rationale
Why police?	If a neighboring network can send more traffic than the traffic contract specifies, policing can be used to enforce the contract, protecting the network from being overrun with too much traffic.
Where to police?	Typically, policing is performed as packets enter the first device in a network. Egress policing is also supported, although it is less typical.

continues

Table 5-2 *Policing and Shaping: When to Use Them, and Where (Continued)*

Topic	Rationale
Why shape?	The first of two reasons for shaping is when the neighboring network is policing. Instead of waiting for the neighboring policer to discard traffic, a shaper can instead delay traffic so that it will not be dropped. The second reason has to do with the effects of egress blocking. By shaping, egress blocking can be avoided, or minimized, essentially moving the queues from inside the service provider cloud, and back into the enterprise routers. By doing so, the router queuing tools can selectively give better QoS performance to particular types of traffic.
Where to shape?	Shaping is always an egress function. Typically, shaping is performed on packets exiting a router, going into another network. This may be the edge between a router and a multiaccess WAN, or possibly just a link to an ISP.

How Shaping Works

Shaping only makes sense when the physical clock rate of a transmission medium exceeds a traffic contract. The most typical case for shaping involves a router connected to a Frame Relay or ATM network. More often today, however, connections to ISPs use a point-to-point serial link or an Ethernet link between an enterprise and the ISP, with a traffic contract defining lower traffic volumes than the physical link. Starting with a physical installation using a higher-speed link allows for simple growth by just increasing the traffic contract.

Routers can only send bits out an interface at the physical clock rate. To have the average bit rate, over time, be lower than the clock rate, the router just has to send some packets for some specified time period, and then not send any packets for another time period. To average sending at a packet rate of half the physical link speed, the router should send packets half of the time, and not send the other half of the time. To make the average rate equal to 1/4 of the physical link speed, the router should send 1/4 of the time, and not send packets 3/4 of the time. Over time, it looks like a staccato series of sending, and silence.

You can understand traffic-shaping logic by reviewing just a few simple examples. Of course, you need to know a few more details for the exams! However, the basics follow these simple examples: If R1 has a 128-kbps access rate, and a 64-kbps CIR, and the engineer wants to shape the traffic to match CIR (64 kbps), R1 just has to send traffic on the link half of the time. If, over time, R1 sends traffic half of the time, at 128 kbps (because that's the only rate it can actually send traffic), the average over that time is 64 kbps. The concept is that simple!

A few more simple examples here emphasize the point. Referring to Figure 5-4, assume R1 wants to shape at 96 kbps, because the Frame Relay switch is policing at 96 kbps. With a 128-kbps access rate, to shape to 96 kbps, R1 should send 3/4 of the time, because 96/128 = 3/4.

Again from Figure 5-4, if the Main router wants to shape the VC connecting it to R24 at 128 kbps, to avoid the egress-blocking problem, the Main router needs to send packets 128/1536 (actual available bit rate for T/1 is 1.536 Mbps), or 1/12 of the time. If the Main router wants to shape that same VC to 64 kbps, to match the CIR, the Main router should send packets over that VC at 64/1536, or 1/24, of the time.

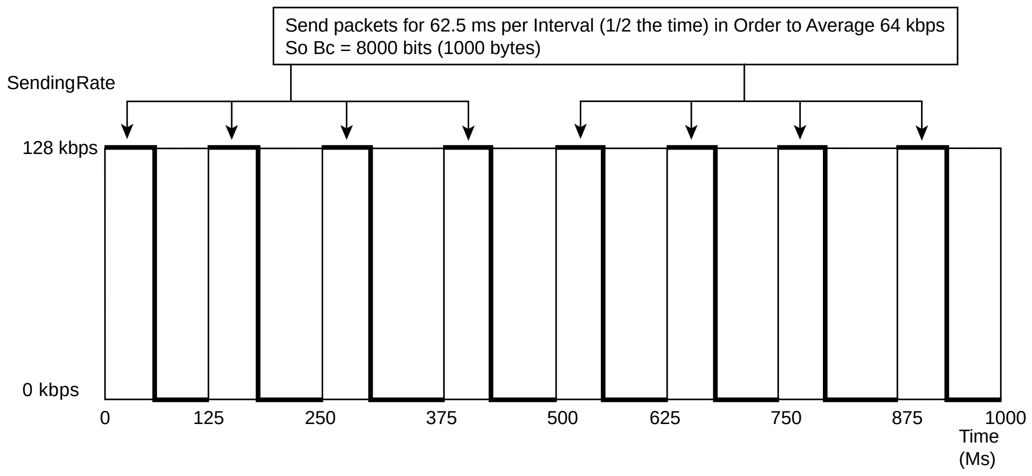
Traffic shaping implements this basic logic by defining a measurement interval, and a number of bits that can be sent in that interval, so that the overall shaped rate is not exceeded. Examples help, but first, Table 5-3 lists some definitions.

Table 5-3 *Shaping Terminology*

Term	Definition
Tc	Time interval, measured in milliseconds, over which the committed burst (Bc) can be sent. With many shaping tools, Tc = Bc/CIR.
Bc	Committed burst size, measured in bits. This is the amount of traffic that can be sent over the interval Tc. Typically also defined in the traffic contract.
CIR	Committed information rate, in bits per second, defines the rate defined in the traffic contract.
Shaped Rate	The rate, in bits per second, to which a particular configuration wants to shape the traffic. In some cases, the shaped rate is set to the same value as CIR; in others, the shaped rate is set to a larger value, with the hope of sending more traffic through the network.
Be	Excess burst size, in bits. This is the number of bits beyond Bc that can be sent after a period of inactivity.

The actual processes used by traffic shaping, and the terms in Table 5-3, will make much more sense to you with a few examples. The first example, as outlined in Figure 5-5, shows the familiar case where R1 shapes to 64 kbps, with a 128-kbps access link.

Figure 5-5 *Mechanics of Traffic Shaping—128-kbps AR, 64-kbps Shaped Rate*



The router should send literally half of the time to average sending 64 kbps on a 128-kbps link. Traffic shaping accomplishes this by sending up to half of the time in each Tc.

As shown in the figure, R1 sends at line rate for 62.5 ms, and then is silent for 62.5 ms, completing the first interval. (The Tc defaults to 125 ms for many shaping tools.) As long as packets are queued and available, R1 repeats the process during each interval. At the end of 1 second, for instance, R1 would have been sending for 62.5 ms in 8 intervals, or 500 ms—which is .5 second. By sending for half of the second at 128 kbps, R1 will have sent traffic at an average rate of 64 kbps.

IOS traffic shaping does not actually start a timer for 62.5 ms, and then stop sending when the timer stops. IOS actually calculates, based on the configuration, how many bits could be sent in each interval so that the shaped rate would be met. This value is called the *committed burst* (Bc) for each interval. It is considered a burst, because the bits actually flow at the physical line rate. The burst is committed, because if you send this much every interval, you are still conforming to the traffic contract. In this example, the Bc value is set to 8000 bits, and the actual process allows the shaper to send packets in each interval until 8000 bits have been sent. At that point, the shaper waits until the Tc has ended, and another interval starts, with another Bc worth of bits sent in the next interval. With an interval of 125 ms, and 8000 bits per interval, a 64-kbps shaped rate is achieved.

The Bc value is calculated using the following formulas:

Bc = Tc * CIR

Or

Bc = Tc * Shaped rate

In the first formula, which is scattered throughout the Cisco courses, the formula assumes that you want to shape at the CIR. In some cases, however, you want to shape to some other rate, so the second formula gives the more exact formula. With the default of 125 ms for Tc, and with a shaped rate of 64 kbps, the Bc would be

Bc = .125 seconds * 64000 bits/second = 8000 bits

When configuring shaping, you typically configure the shaping rate and optionally the Bc. If you configure both values, IOS changes the Tc so that the formula is met; you never actually configure the value of Tc. If you just configure the shaping rate, IOS assumes a 125-ms Tc, and calculates the Bc. The formulas IOS uses to calculate Tc when you configure both the shaping rate and the Bc are as follows:

Tc = Bc/CIR

Or

Tc = Bc/Shaped rate

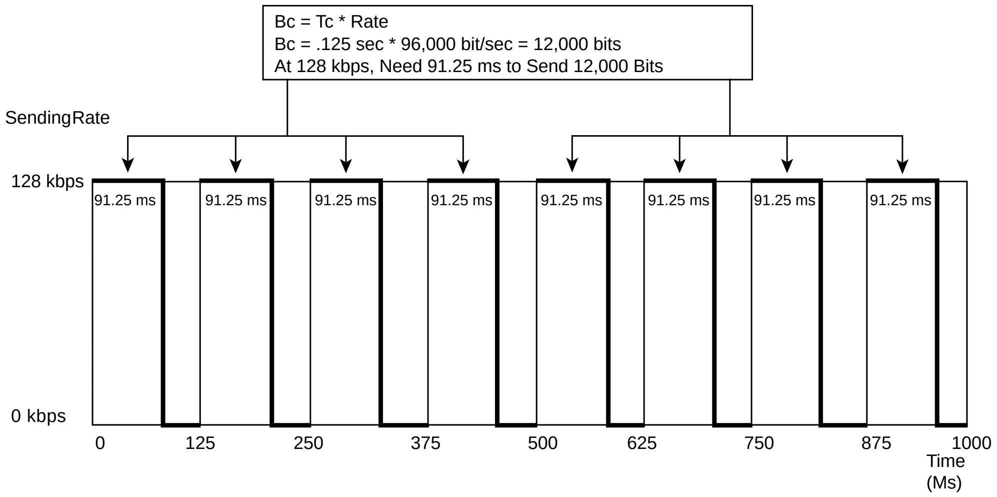
Again, the formula referring to CIR assumes that you shape to the CIR value, whereas the second formula refers to the shaping rate, because you can configure shaping to shape at a rate different from CIR.

Additional examples should bring the concepts together. Previously you read the example of the PB Tents company shaping at 96 kbps over a link using a 128-kbps clock rate, because the Frame Relay provider policed at 96 kbps. If the shaping function is configured with a shaping rate of 96 kbps, with no Bc value, the formulas specify the following:

Bc = .125 sec * 96,000 bits/sec = 12,000 bits

Figure 5-6 shows what happens in this example.

Figure 5-6 *Mechanics of Traffic Shaping—128-kbps Access Rate, 96-kbps Shaped Rate*

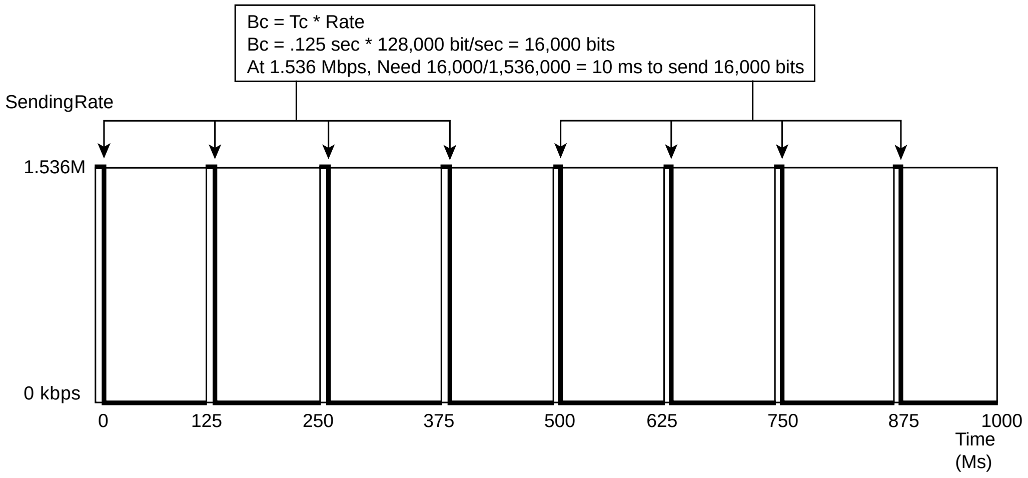


For each interval, shaping can release 12,000 bits, which takes 91.25 ms. 91.25/125 = 3/4, implying that the router will average sending bits at 3/4 of the clock rate, or 96 kbps.

Traffic shaping uses the idea of a number of bits per interval for implementation because it's much more efficient than calculating rates all the time. The shaper just grabs the next packet, decrements the Bc values by the number of bits in the packet, and keeps doing that until the Bc value is consumed. At that point, shaping waits until the Tc has expired, when shaping gets to send another Bc worth of bits.

The length of Tc may have some impact on the delay and jitter characteristics of the packets being shaped. Consider another example, with the Main router sending packets to R24, shaping at 128 kbps, but with a T/1 access link. Figure 5-7 shows the shaping details.

Figure 5-7 *Mechanics of Traffic Shaping—Main Router with 1.536 Access Rate, 128-kbps Shaped Rate*



Simply put, at T/1 speeds, it does not take long to send the Bc worth of bits per interval. However, The Tc value may be a poor choice for delay-sensitive traffic. Suppose that a VoIP packet arrives at Main, and it needs to be sent to R24. Main uses LLQ, and classifies VoIP into the low-latency queue, so the new VoIP packet will be sent next. That's true—unfortunately, the packet sent just prior to the new VoIP packet's arrival consumed all of Bc for this Tc. How long does the VoIP packet have to wait before the current Tc will end and a new one will begin? Well, it only took 10 ms to send the Bc worth of bits, so another 115 ms must pass before the current Tc ends, and the VoIP packet can be sent! With one-way delay budgets of 150 to 200 ms, a single shaping delay of 115 ms just will not work. Therefore, Cisco recommends that when you have delay-sensitive traffic, configure Bc such that Tc is 10 ms or less. In this same example, if the Bc were configured to 1280 bits, Tc = 1280/128,000 = .010 seconds, or 10 ms.

NOTE Many of you might be concerned about the relatively small Bc of 1280 bits, or only 160 bytes! Most packets exceed that length. Well, as it turns out, you will also typically use fragmentation in the exact same cases. To accommodate the same delay-sensitive traffic, the fragments will be of similar size—in fact, as you will read in Chapter 7, "Link Efficiency," the fragmentation size will likely be 160 bytes in this particular example. Therefore, with delay-sensitive traffic, you will drive Tc down to about 10 ms by lowering Bc, and the Bc value will essentially allow a single fragment per Tc. By doing so, you reduce the shaping delay waiting on the next Tc to occur, and you reduce the serialization delay by fragmenting packets to smaller sizes.

The next several sections continue the discussion of how traffic shaping works, covering excess burst, queuing, adaption, and some concepts about enabling shaping.

Traffic Shaping, Excess Burst, and Token Buckets

Traffic shaping includes the capability to send more than Bc in some intervals. The idea is simple: Data traffic is bursty, so after a period of inactivity, it would be nice if you could send more than Bc in the first interval after traffic occurs again. This extra number of bits is called the burst excess, or Be. Traffic-shaping tools allow Be as an option.

The underlying operation of traffic shaping to allow for Be requires a little more insight into how traffic shaping works, and it also requires us to understand the concept of token buckets. Token buckets can be used to describe how shaping and policing are implemented.

Ignoring Be for a moment, imagine a bucket filled with tokens, like subway tokens. In the token-bucket scenario, each token lets you buy the right to send 1 bit. One token bucket is used for formal operation of traffic shaping as discussed earlier; this bucket has a size of Bc. The bucket is filled to the brim, but no more, at the beginning of each Tc. Every time a packet is sent, traffic shaping spends tokens from the token bucket to buy the right to send the packet. If the packet is 1000 bits long, 1000 tokens are removed from the bucket. When traffic shaping tries to send a packet, and the bucket does not have enough tokens in it to buy the right to send the packet, traffic shaping must wait until the next interval, when the token bucket is refilled.

An analogy of token bucket is a child and the allowance the child receives every Saturday morning. For the sake of argument, assume the weekly allowance is $10. The child may spend the money every week; if the child doesn't spend it, he may save up to buy something more expensive. Imagine that the child's parents are looking at the child's piggybank every Saturday morning, however, and if they find some leftover money, they just add a little more money so that the child always starts Saturday morning with $10! After a few weeks of this practice, the child would likely try to spend all the money each week, knowing that he would never be able to save any more than $10. Similarly, the Bc of bits, or the tokens in the bucket if you prefer, are only usable in that individual Tc interval, and the next Tc (interval) always starts with Bc tokens in the bucket, but never any more.

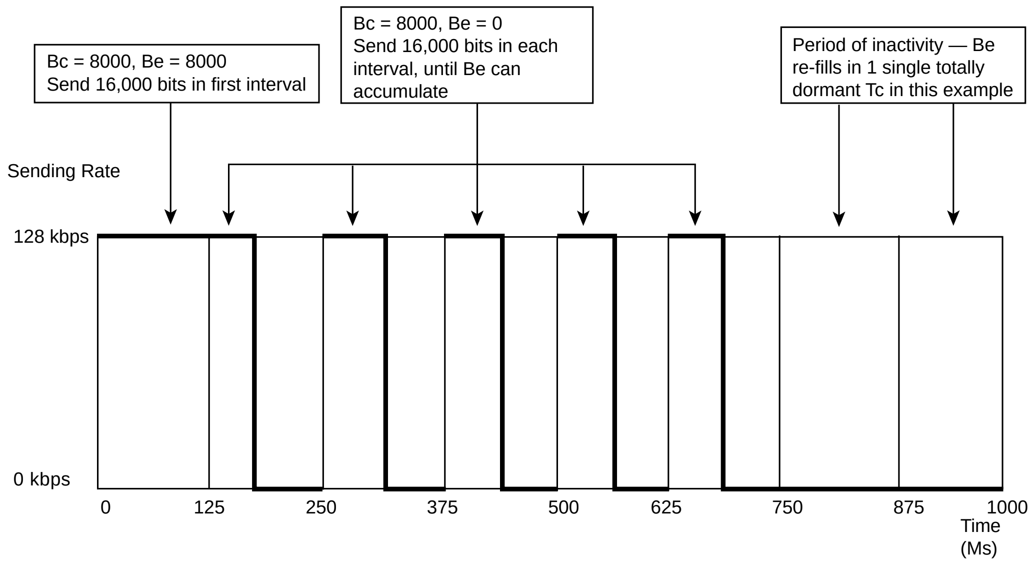
Traffic shaping implements Be by using a second bucket that can be used to accumulate any unused tokens from the first bucket. If in interval one, traffic shaping did not need all Bc worth of tokens in the first bucket, traffic shaping would move these tokens into the second bucket. So long as each successive interval does not use all Bc of the tokens in the first bucket, the second bucket would continue filling. Of course, the second bucket has a finite size as well, set at Be bits or tokens. If traffic shaping doesn't need Bc bits (tokens) in each interval, the second bucket will fill, with any additional extra tokens wasted—in other words, you can only save up Be worth of "extra" tokens, because your second bucket is a set size.

So, the dual token bucket is similar to having two piggybanks—any saved money from each week is dumped from the first piggybank into the second piggybank, and the first piggybank is refilled with $10 each week. Continuing the analogy, the Mom and Dad in our scenario would also monitor the second piggybank so that it does not get too full.

Traffic shaping uses the Be so that more than Bc can be sent after a period of low activity. Traffic shaping can spend the tokens in the first bucket, and try to take the next packet, finding it does not have enough tokens in the first bucket. Because a Be has been configured, traffic shaping looks in the second token bucket, and takes enough tokens from there to buy the right to send the next packet. In fact, in a single interval, all of Bc and Be can be consumed. Before Be can be used again, traffic shaping would need some intervals with less than Bc traffic, so that the unused tokens from some intervals could be used to refill the second bucket.

Figure 5-8 shows a graph of how shaping works when using Be. The shaper represented by the graph shapes to a CIR of 64 kbps, over a 128-kbps link. The Bc has been set to 8000 bits, with an additional 8000 bits for Be.

Figure 5-8 *Bc and Be, After a Period of Inactivity (Both Buckets Are Full)*



This example assumes that enough inactive or slow periods have occurred, just prior to this example, so that Be has 8000 bits in it. In other words, the second token bucket holds 8000 tokens, each representing a bit. A large amount of traffic shows up, so traffic shaping sends as fast as it can for several consecutive time intervals. In the first interval, traffic shaping can send a total of 16,000 bits, because the actual Bc at the start of the interval is 8000, and the actual Be at the start of the first interval is also 8000 bits. On a 128-kbps link, over the default 125-ms Tc, it takes all 125 ms to send 16,000 bits! Effectively, in this particular case, after a period of inactivity, R1 sends continuously for 187.5 ms until traffic shaping artificially slows down the traffic. Thus, the goal of allowing a burst of data traffic to get started quickly is accomplished with Be.

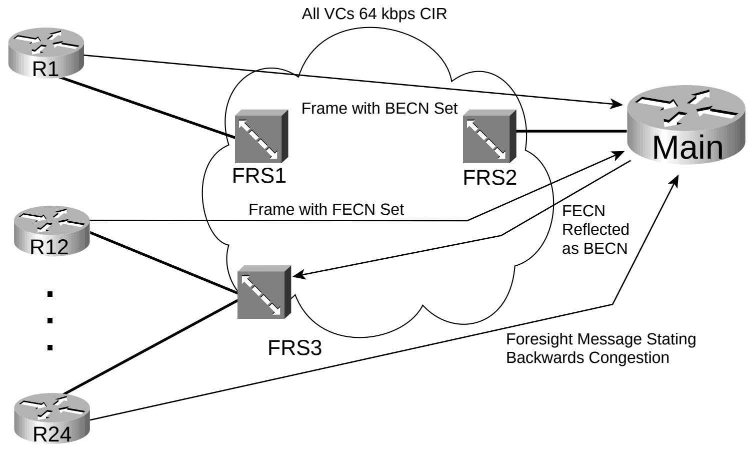
Traffic-Shaping Adaption

The rate at which the shaping function shapes traffic can vary over time. The adaption or adaptation process causes the shaper to recognize congestion and reduce the shaping rate temporarily, to help reduce congestion. Similarly, adaption notices when the congestion abates and returns the shaping rate to the original rate.

Two features define how adaption works. First, the shaper must somehow notice when congestion occurs, and when it does not occur. Second, the shaper must adjust its rate downward and upward as the congestion occurs and abates.

Figure 5-9 represents three different ways in which the main router can notice congestion. Three separate lines represent three separate frames sent to the main router, signifying congestion. Two of the frames are data frames with the Frame Relay backward explicit congestion notification (BECN) bit set. This bit can be set inside any Frame Relay frame header, signifying whether congestion has occurred in the direction opposite to the direction of the frame with the bit set. The third (bottom) message is a Foresight message. Stratacom, and later Cisco after they acquired Stratacom, defined Foresight as a signaling protocol in Frame Relay and ATM networks, used to signal information about the network, such as congestion information. If the Frame Relay network consists of Cisco/Stratacom WAN switches, the switch can send Foresight messages, and Cisco routers can react to those messages. Following the figure, each of the three variations for the Main router to recognize that congestion is occurring is explained in detail.

Figure 5-9 *FECN, BECN, and Foresight Feedback*



First consider the BECN frame. Backward means that the congestion exists in the opposite, or backward, direction, as compared with the direction of the frame. Therefore, if FRS1 notices congestion trying to send frames to R1 (right to left in the figure), on the next frame sent by R1 (left to right in the figure), FRS1 can mark the BECN bit. In fact, any device can set the forward explicit congestion notification (FECN) and BECN bits—however, in some networks, the Frame Relay switches do set the bits, and in some, they do not.

If the BECN bit is set, the Main router, if using adaptive shaping, reduces its shaping rate on the VC to R1. Because the congestion occurs right to left, as signaled by a BECN flowing left to right, router Main knows it can slow down and help reduce the congestion. If Main receives another frame with BECN set, Main slows down more. Eventually, Main slows down the shaping rate until it reaches a minimum rate, sometimes called the *minimum information rate* (MIR), and other times called the *mincir.*

Similarly, if Main receives a Frame from R12 with FECN set, the congestion is occurring left to right. It does not help for Main to slow down. It does help for R12 to slow down. Therefore, the Main router can "reflect" the FECN, by marking the BECN bit in the next frame it sends on the VC to R12. R12, receiving a BECN, can reduce the shaping rate.

Finally, Foresight messages are separate, nondata signaling frames. Therefore, when the congestion occurs, Foresight does not need to wait on a data frame to signal congestion. In addition, Foresight sends messages toward the device that needs to slow down. For instance, a switch notices congestion right to left on the VC between Main and R24. The switch generates and sends a Foresight message to Main, using that same VC, so Main knows it needs to slow down its shaping rate on that VC temporarily.

When configuring adaptive shaping, you configure the minimum and maximum shaping rate. The configuration commands refer to the minimum rate as mincir, and the maximum rate as CIR, with mincir defaulting to 50 percent of CIR.

With no congestion, shaping uses the maximum rate. When the shaper receives a BECN or Foresight message, it slows down by 25 percent of the maximum rate. It continues to slow down by 25 percent of the maximum rate per Tc, until the minimum rate is reached. After 16 consecutive intervals occur without a BECN or Foresight congestion message, the shaping rate grows by 1/16 of the maximum rate during each Tc, until the maximum rate is reached again.

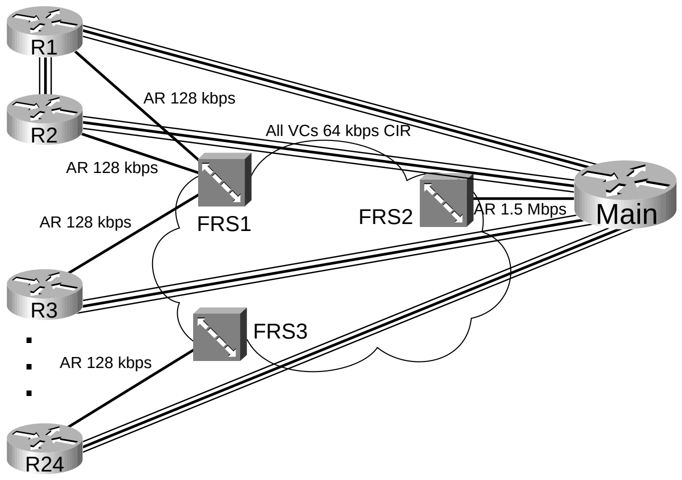
Where to Shape: Interfaces, Subinterfaces, and VCs

Shaping can be applied to the physical interface, a subinterface, or in some cases, to an individual VC. Depending on the choice, the configuration causes traffic shaping to occur separately for each VC, or it shapes several VCs together. In most cases, engineers want to shape each VC individually.

When shaping is applied to an interface for which VCs do not exist, shaping is applied to the main interface, because there are no subinterfaces or VCs on those interfaces. On Frame Relay and ATM interfaces, however, some sites have multiple VCs terminating in them, which means that subinterfaces will most likely be used. In some cases, more than one VC is associated with a single multipoint subinterface; in other cases, point-to-point subinterfaces are used, with a single VC associated with the subinterface. The question becomes this: To shape per VC, where do you enable traffic shaping?

First, consider a typical branch office, such as R24 in Figure 5-10.

Figure 5-10 *PB Tents Network: Shaping on Subinterfaces and VCs*



R24 has a single VC to the Main site at PB Tents. Because R24 only has the single VC, the configuration on R24 may not even use subinterfaces at all. If the configuration does not use subinterfaces on R24's serial link, traffic shaping can be configured on the physical interface. If the configuration includes a subinterface, you can enable traffic shaping on the physical interface, or on the subinterface. Because there is only one VC, it does not really matter whether shaping is enabled on the physical interface, or the subinterface—the behavior is the same.

Now consider the Main router. It has a VC to each remote site. (Also notice that a VC has been added between R1 and R2, just to make things interesting.) So, on the main router, point-to-point subinterfaces are used for the VCs to branches 3 through 24, and a multipoint subinterface is used for the two VCs to R1 and R2. To shape each VC to branches 3 through 24 separately, shaping can be configured on the subinterface. However, shaping applied to a multipoint subinterface shapes all the traffic on all VCs associated with the subinterface. To perform shaping on each VC, you need to enable shaping on each individual data-link connection identifier (DLCI).

In summary, most QoS policies call for shaping on each VC. The configuration commands used to enable shaping differ slightly based on the number of VCs, and how they are configured. Table 5-4 summarizes the options.

Table 5-4 *Options of How to Enable Shaping for per-VC Shaping*

Location	Requirements for Shaping per VC
No VCs, for example, point-to-point links	Shape on the main interface. Shaping occurs for all traffic on interface.
Physical interface, 1 VC, no subinterfaces	Shaping shapes the individual VC associated with this interface. Shaping can be enabled on the physical interface.
Physical interface, 1 VC, 1 subinterface	Shaping shapes the individual VC associated with this interface. Shaping can be enabled on the physical interface, the subinterface, or the VC (DLCI).
Multiple VCs on 1 interface, point-to-point subinterfaces only	Shaping can be enabled on the subinterface, or per DLCI. Both methods work identically.
Multiple VCs on 1 interface, some multipoint subinterfaces with > 1 VC per subinterface	Must enable shaping on each DLCI to shape per VC.

Queuing and Traffic Shaping

Shaping tools support a variety of queuing tools that can be applied to the packets waiting in the shaping queue(s). At the same time, IOS supports queuing tools for the interface output queue(s) associated with the physical interface. Deciding when to use queuing tools on shaping queues, when to use them on the interface, and how the configurations differ in each case, can be a little confusing. This section clears up some of that confusion.

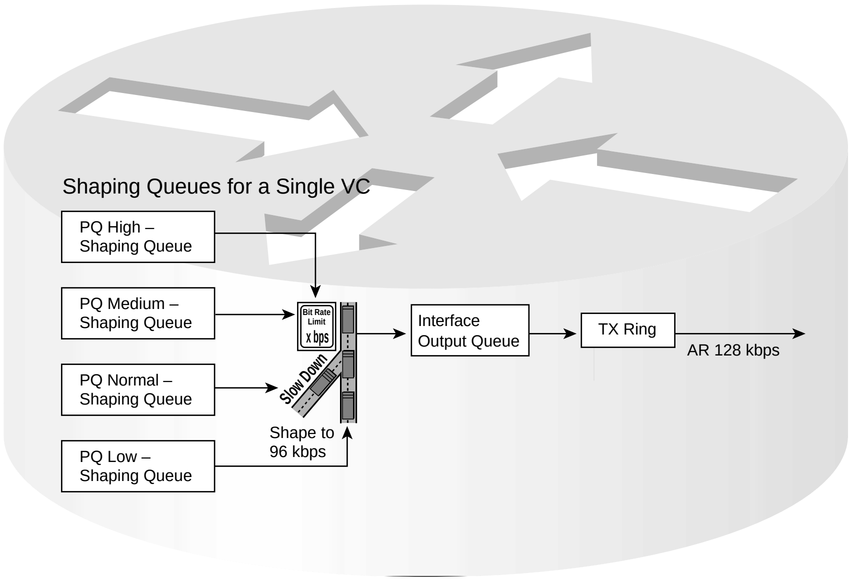
To begin, Table 5-5 lists the traffic-shaping tools, and the queuing tools supported by each for the shaping queues.

Table 5-5 *Options for Queuing in Traffic-Shaping Tools*

Shaping Tool	Queuing Tools Supported for the Shaping Queue(s)
GTS	WFQ
CB shaping	FIFO, WFQ, CBWFQ, LLQ
DTS	FIFO, WFQ, CBWFQ, LLQ
FRTS	FIFO, WFQ, CBWFQ, LLQ, PQ, CQ

When a shaper uses a queuing tool, instead of having a single shaping queue, multiple shaping queues exist. If FRTS were configured to use Priority Queuing (PQ), for instance, PQ would create four queues for shaping, named High, Medium, Normal, and Low. Figure 5-11 shows the basic idea, with shaping enabled on the physical interface, FIFO Queuing on the physical interface, and PQ configured for shaping the only VC.

Figure 5-11 *FRTS, with FIFO Queuing for the Physical Interface, Plus PQ for the Shaping Queue*



The shaping queues exist separately from the interface output queues, as seen in the figure. With PQ applied to the shaper, four shaping queues exist for this VC. When the shaper decides to allow another packet to be sent, it takes the next packet from the PQ shaping queues, according to PQ scheduler logic. Those packets are placed into queues associated with the physical interface and then forwarded out the interface.

In some cases, the shaping queues are bypassed, and in other cases, the interface output queues are bypassed. To understand why, consider Figure 5-12, which demonstrates part of the logic behind the decision for determining when each queue should be used.

Packets are held in a shaping queue or interface output queue only if there is some reason why the packet must wait to take the next step. For instance, you already know that if the TX Ring is not full, packets are immediately placed into the TX Ring, bypassing the interface output queue. Likewise, if shaping decides that a packet does not need to be delayed, it can go directly to the interface output queue, or even to the TX Ring.

Many QoS designs call for shaping per VC, as mentioned in the preceding section. Suppose that a router has two 64-kbps CIR VCs sharing an access link, each configured on a separate point-to-point subinterface. Shaping queues will be created for each VC. A single set of interface output queues will be created, too. Figure 5-13 depicts the overall idea.

Figure 5-12 *Decision Logic for Queuing with Shaping Enabled*

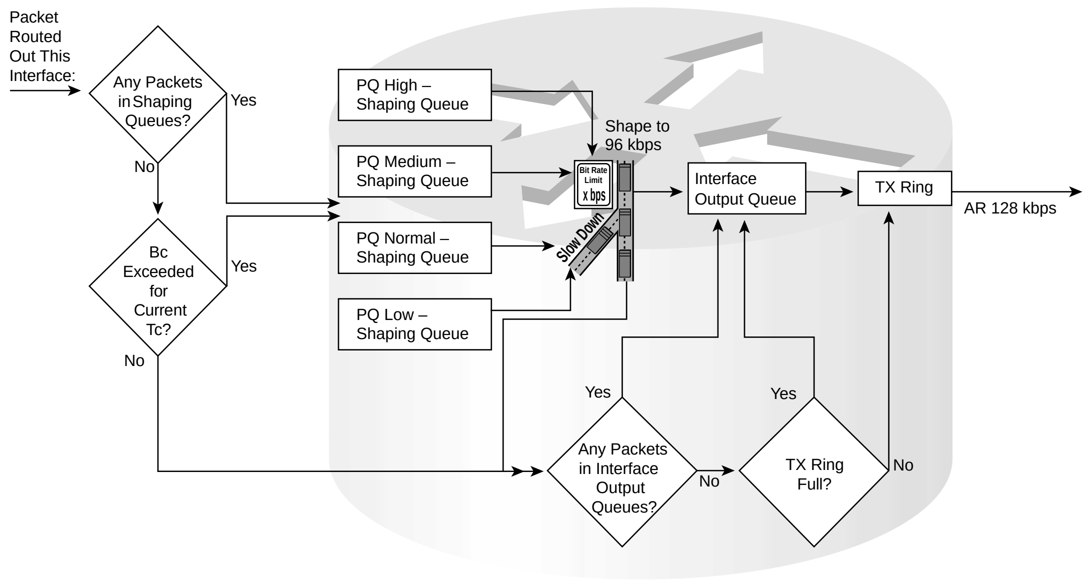
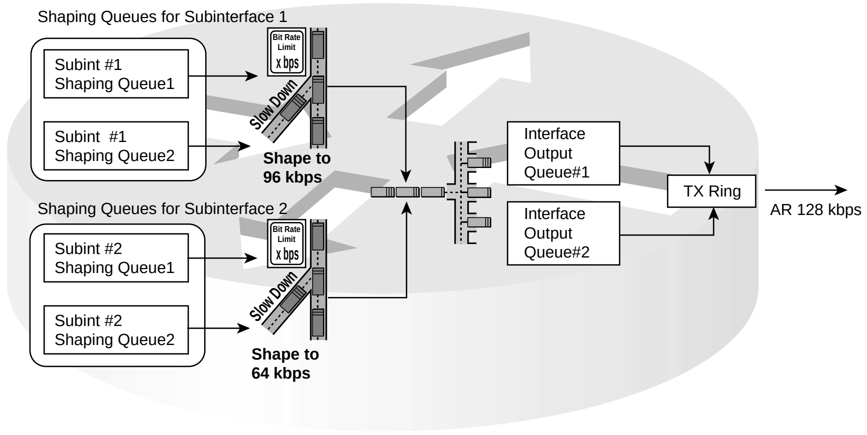

Packet Routed Out This Interface:
Any Packets in Shaping Queues?
Yes
No
Bc Exceeded for Current Tc?
Yes
No
PQ High – Shaping Queue
PQ Medium – Shaping Queue
PQ Normal – Shaping Queue
PQ Low – Shaping Queue
Shape to 96 kbps
Bit Rate Limit x bps
Slow Down
Interface Output Queue
TX Ring
AR 128 kbps
Yes
Yes
Any Packets in Interface Output Queues?
No
TX Ring Full?
No

Figure 5-13 *Fancy Queuing for the Physical Interface and for Two Sets of Shaping Queues*



The shaping tool creates a set of queues for each subinterface or VC, based on the queuing tool configured for use by the shaper. IOS creates only one set of output interface queues for the physical interface, based on the queuing configuration on the physical interface, as covered in Chapter 4, “Congestion Management.” In Figure 5-13, two sets of shaping queues have been created, one per VC. Both VCs feed the single set of interface output queues.

Finally, in this particular example, congestion can occur at the physical interface. The total of the two shaping rates listed in Figure 5-13 is 160 kbps, which exceeds the access rate. Because interface output queues can fill, it helps to apply a queuing tool to the interface output queues in this case.

How Policing Works

The internal processing logic used by IOS when implementing CAR differs slightly from the logic used when implementing class-based (CB) policing. Regardless of the internal processes, each policing tool wants to classify a packet relative to whether it is within the traffic contract. Based on that classification, the packet can be allowed to pass, be dropped, or be re-marked with a different IP precedence or IP DSCP (differentiated services code point) value.

Single and dual token-bucket models are used so that you fully appreciate the internal processing of the policers. Remember, the token-bucket mechanism is used to describe the concepts, but not necessarily to describe how the software is actually written. When no Be is configured, a single bucket is used, which represents Bc, and when Be is used, a second bucket is used, representing Be.

CAR internals are covered in the following sections, followed by CB policing concepts.

CAR Internals

Details aside, what CAR does is decide whether the packet conforms to or exceeds a traffic contract. CAR then takes action on the packet based on the configuration, which tells CAR whether to pass, discard, or re-mark conforming packets and exceeding packets. The actions themselves are simple and obvious; the hard part about CAR is understanding how CAR decides whether a packet conforms to the contract, or exceeds the contract.

CAR uses one or two token buckets to determine whether each packet conforms to or exceeds a traffic contract. CAR fills the buckets with tokens, with each token representing the right to send or receive 1 byte. When a packet is policed, CAR checks the buckets to decide whether enough tokens are in the right bucket to consider the packet as conforming. (The details of how CAR makes that decision are covered in the next few pages.) To understand CAR, you need to understand how CAR fills the buckets with tokens, and how CAR decides, based on the packet and the tokens in the buckets, whether the packet conforms or exceeds the traffic contract.

CAR and CB policing work slightly differently with and without a Be specified. The following sections explain how CAR works without Be, and how it works with Be.

CAR Without Be

Just like most descriptions of policing logic, this book uses token buckets to describe CAR logic. In this discussion, one bucket is called Bucket1, and the other is ingeniously called

Bucket2. Bucket1 begins with Bc worth of tokens in it, and Bucket2 begins with Be worth of tokens in it. Of course, only Bucket1 is used when Be = 0, but both Bucket1 and Bucket2 are used when Be is greater than 0. In some explanations found in other documents, you might hear Bucket1 referred to as the Bc bucket, or just as Bc. Because Bc is a static, configured value, and the number of tokens in the buckets change, however, I refer to the buckets as Bucket1 and Bucket2.

First, CAR fills Bucket1 with Bc tokens. Remember, each token represents a single byte, and Bc is configured in bytes. As time passes, CAR replenishes Bucket1 at regular intervals, using a Tc value calculated with the same formula used by shaping. The configuration sets the policing rate and the Bc value, with CAR calculating the Tc value based on the following familiar formula:

$$Tc = \frac{Bc}{\text{Policed rate}}$$

CAR replenishes Bucket1 every Tc by adding Bc tokens to the bucket. Because Bucket1 has a maximum capacity of Bc tokens, and CAR replenishes Bc tokens every Tc, CAR effectively completely refills Bucket1 every Tc.

CAR categorizes each packet as “conforming” or “exceeding” the traffic contract as it arrives at the policer. CAR compares the number of bytes in the packet to the number of tokens in Bucket1. (Remember, Be = 0 for now, so the policer does not even think about Bucket2 in this case.) CAR’s decision is simple:

- If the number of bytes in the packet is less than or equal to (<=) the number of tokens in Bucket1, the packet conforms. CAR removes tokens from Bucket1 equal to the number of bytes in the packet, and performs the action for packets that conform to the contract.
- If the number of bytes in the packet is greater than (>) the number of tokens in Bucket1, the packet exceeds. CAR does not remove tokens from Bucket1, and performs the action for packets that exceed the contract.

So, without a Be configured, CAR logic is pretty easy. All the packets policed in a single Tc interval conform, so long as the packets cumulative number of bytes does not exceed Bc. If more than Bc bytes need to be sent, CAR decides those packets exceed the contract.

Car with Be

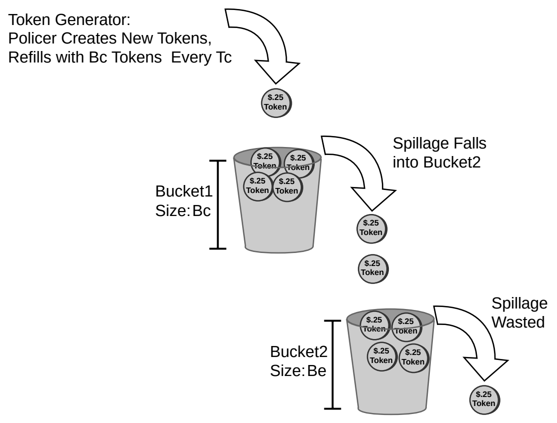
The previous discussion revolved around the simpler case, in which there was no Be. With a Be configured, CAR uses a slightly more complex algorithm to decide whether a packet conforms to or exceeds the traffic contract. This more complex algorithm has a simple goal—to soften the blow when the policer must discard packets. CAR’s algorithm recognizes when the extra bytes defined by Be are being consumed, and classifies some packets as “exceed,” and some packets as “conform,” before all of Be has been consumed! The strategy behind discarding a few packets, but not all, when consuming the tokens in Bucket2 follows the same concept as WRED. By discarding a few packets before all the Be capacity has been consumed, maybe the senders of the traffic may reduce the rate of sending packets, reducing congestion, and avoiding the day when all packets exceed the policed rate.

NOTE For the remaining discussion of how policing works with Be, assume that the action for conforming packets is to forward the packet, and the action for nonconforming packets is to discard the packet. Other options, such as re-marking the packets, are available; however, continually referring to those options during this discussion just confuses the discussion.)

To understand how CAR works is to understand how the buckets are filled and drained. Initially, CAR fills Bucket1 with Bc tokens, and Bucket2 with Be tokens. Remember, each token represents a single byte, and Bc and Be are configured in bytes.

CAR replenishes Bucket1 every Tc by adding Bc tokens to the bucket, but CAR does not directly replenish Bucket2. When CAR adds Bc tokens to Bucket1, if any tokens are spilled, those tokens fill Bucket2. If Bc = 10,000, and Bucket1 has 2000 tokens in it when CAR refills Bucket1, for example, 2000 tokens spill into Bucket2. Bucket1 has a maximum capacity of Bc tokens, which effectively means that Bucket1 gets refilled to its maximum every Tc. Bucket2 has a maximum capacity of Be tokens, but it may not get filled to capacity each Tc—that depends on how many tokens “spill” out of Bucket1 when it is refilled. Figure 5-14 outlines the process.

Figure 5-14 *Refilling Dual Token Buckets with CAR*


Token Generator:
Policer Creates New Tokens,
Refills with Bc Tokens Every Tc
$.25 Token
Spillage Falls into Bucket2
Bucket1 Size: Bc
Bucket2 Size: Be
Spillage Wasted

With Bc and Be configured, CAR uses dual token buckets, and the CAR algorithm gets more complicated. The algorithm is easy and familiar:

1. If the number of bytes in the packet is less than or equal to (<=) the number of tokens in Bucket1, the packet conforms. CAR also removes tokens from Bucket1 equal to the number of bytes in the packet and performs the action for packets that conform to the contract.
2. If the number of bytes in the packet is greater than (>) the number of tokens in Bucket1, CAR uses the *debt calculations* to determine whether the packet conforms or exceeds. CAR removes tokens from Bucket2 equal to the number of bytes in the packet if the packet conforms based on this rule.

As noted in part two of the algorithm, CAR uses the concept of debt. This means that if the packet size of a new packet is more than the number of tokens in Bucket1, CAR can borrow from Bucket2. CAR needs to replenish, or repay, those tokens after an interval when Bc is not completely used, as was described earlier.

The term “actual debt” (Da) defines the most obvious concept related to debt calculations. Imagine that Bucket1 has been depleted down to 0 tokens, and Bucket2 has 10,000 tokens in it. A new 1000-byte packet arrives that needs to be policed. CAR compares the packet length to Bucket1, and decides that Bucket1 does not have enough tokens. CAR then looks at Bucket2, which does have more tokens (10,000) than the packet size (1000). Therefore, CAR considers the packet as conforming, and it decrements Bucket2 to 9000. CAR also now considers Da to be 1000—the number of actual tokens “borrowed” from Be or Bucket2. Imagine 2 more 1000-byte packets arrive before CAR replenishes tokens into Bucket1. These packets also conform, and CAR decrements Bucket2 down to 8000 tokens after the second packet, and then to 7000 tokens after the third packet. Similarly, Da is incremented to 2000, and then 3000.

Dc is the other key component of the CAR algorithm. The compounded debt (Dc) variable is also calculated by CAR as Be is being consumed. Each time Da is incremented, Dc is also incremented. Dc grows more quickly than Da, however, because the formula used to calculate Dc is Dc = old_Dc + new_Da. Table 5-6 lists the same three packets described in the explanation of Da, plus 2 more 1000-byte packets, with the computed Da and Dc values. In this case, Be is set to 8000 bytes. Keep in mind that the Dc calculation does not begin until Bc (Bucket1) has been consumed during a particular interval.

Table 5-6 *Example Comparing Growth in Da and Dc with Five Consecutive 1000-Byte Packets*

Packet	Da = Old Da + New_Packet_Length	Dc = Old_Dc + New_Da	Dc
1st 1000-byte packet	1000	0 + 1000	1000
2nd 1000-byte packet	2000	1000 + 2000	3000
3rd 1000-byte packet	3000	3000 + 3000	6000
4th 1000-byte packet	4000	4000 + 6000	0*
5th 1000-byte packet	5000	0 + 5000	5000

* Dc would have been 10,000, but because Be is 8000 in this example, the Dc value is reset to 0.

Dc grows until it passes Be. When Dc is calculated and happens to be above Be, CAR considers the packet to exceed the traffic contract. At that point, CAR resets Dc to 0, as shown in Table 5-6. Note that the fourth 1000-byte packet causes the Dc value to exceed Be, so the packet is considered to exceed the contract, and the Dc value is reduced to 0.

Now that Da and Dc have been defined, the full logic of deciding whether each packet conforms or exceeds can be outlined, as shown here:

1 If the number of bytes in the packet is less than or equal to (<=) the number of tokens in Bucket1, the packet conforms. CAR removes tokens from Bucket1 equal to the number of bytes in the packet, and performs the action for packets that conform to the contract.

2 If the number of bytes in the packet is greater than (>) the number of tokens in Bucket1, but the number of bytes in the packet is less than or equal to (<=) the number of tokens in Bucket2, use the following debt computations:

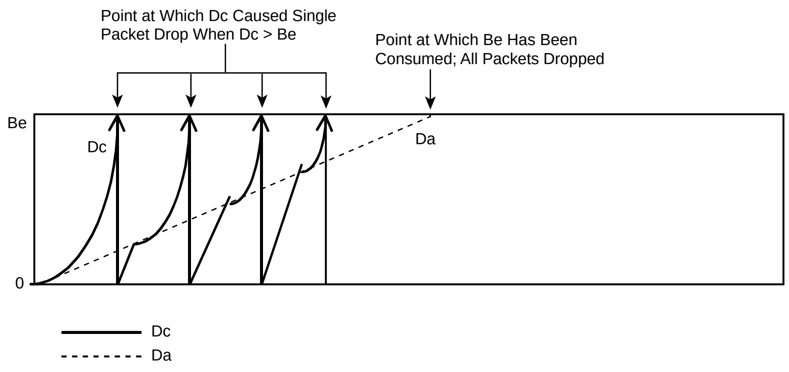
   a. If Dc is less than or equal to (<=) Be, the packet conforms to the contract. CAR removes tokens from Bucket2 equal to the number of bytes in the packet, and performs the action for packets that conform to the contract.

   b. If Dc is greater than (>) Be, the packet exceeds the contract. CAR removes no tokens from either bucket, and performs the action for packets that exceed the contract. CAR also resets Dc to 0.

3 If the number of bytes in the packet is greater than (>) the number of tokens in Bucket1 or Bucket2, the packet exceeds the contract. CAR removes no tokens from either bucket, and performs the action for packets that exceed the contract.

During a particular interval, as long as there are plenty of tokens in Bucket1, the packets conform (Step 1). If there are not enough tokens in Bucket1 or Bucket2, the packet exceeds (Step 3). At Step 2, however, while consuming Bucket2's tokens, a few packets can actually exceed the contract, and others will actually conform to the contract. Figure 5-15 depicts the idea behind how Da and Dc grow with CAR.

Figure 5-15 *Actual Debt and Compounded Debt with CAR*



NOTE Some explanations describe CAR as using only a single token bucket. For the purposes of the exam, if you understand how the process works, as described in this section, you will know more than enough to answer the exam questions. Regardless of whether the actual software uses one or two buckets, the process works as generally described in this chapter.

CB Policing Internals

CB policing internals worked just like CAR internals before IOS 12.1(5)T, but with IOS 12.1(5)T, the underlying logic was changed. This section explains how CB policing works as of 12.1(5)T and beyond. As with the CAR discussion, this section first describes CB policing without Be, and then with Be.

CB Policing Without Be

CB policing also uses token buckets. Bucket1 is the only token bucket used when Be = 0; Bucket1 holds the tokens granted based on the configured committed burst. Bucket2 is used when Be is greater than (>) 0, holding the tokens granted by Be.

Unlike CAR, CB policing replenishes tokens in Bucket1 in response to policing a packet, as opposed to every Tc seconds. Every time a packet is policed, CB policing puts some tokens back into Bucket1. The number of tokens placed into Bucket1 is calculated as follows:

$$\frac{(\text{Current\_packet\_arrival\_time} - \text{Previous\_packet\_arrival\_time}) \times \text{Police\_rate}}{8}$$

NOTE Note that the arrival times' units are in seconds, and the police rate is in bits per second, with the result being a number of tokens. Each token represents the right to send 1 byte.

The idea behind the formula is simple—essentially, a small number of tokens are replenished before each packet is policed, with an end result of having tokens replenished at the policing rate. Suppose, for instance, that the police rate is 128,000 bps (which happens to equal 16,000 bytes per second.) If 1 second has elapsed since the last packet arrived, CB policing would replenish the bucket with 16,000 tokens. If 0.1 seconds had passed since the last packet had arrived, it would replenish the bucket with 0.1 seconds worth of tokens, or 1600 tokens. If .01 seconds had passed, it would replenish 160 tokens at that time. Essentially, Bucket1 is replenished with a prorated number of tokens based on how long ago it was last replenished.

CB policing (with no Be) works just like CAR (with no Be) regarding the choice of whether a packet conforms or exceeds. CB policing compares the number of bytes in the packet to the number of packets in Bucket1. (Remember, Be = 0 right now, meaning Bucket2 is not used.) CB policing's decision is simple, as noted here:

- If the number of bytes in the packet is less than or equal to (<=) the number of tokens in Bucket1, the packet conforms. CB policing removes tokens from Bucket1 equal to the number of bytes in the packet, and performs the action for packets that conform to the contract.
- If the number of bytes in the packet is greater than (>) the number of tokens in Bucket1, the packet exceeds the contract. CB policing does not remove tokens from Bucket1, and performs the action for packets that exceed the contract.

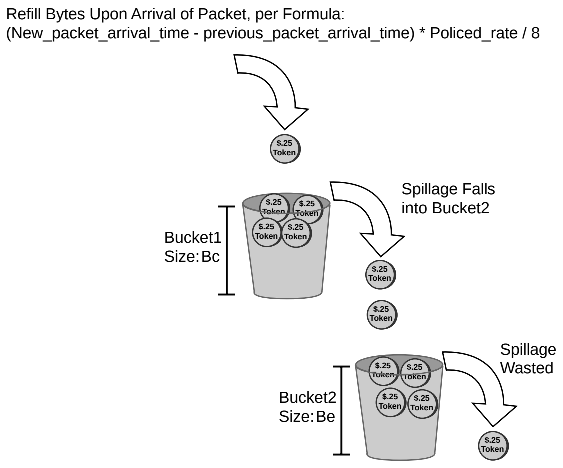
Therefore, the logic with no Be in use is simple. Bucket1 gets replenished with tokens. If the packet conforms, CB policing either forwards, discards, or re-marks the packet, and some tokens are then removed from Bucket1. If the packet exceeds, CB policing either forwards, discards, or re-marks the packet, but no tokens are removed from Bucket1.

CB Policing with Be

CB policing uses a simpler algorithm than CAR when a Be value has been configured. CB policing categorizes packets into three groups: conform, exceed, and violate. By using three categories, a very simple and very useful algorithm can be used.

Just like with CAR, to understand how CB policing works is to understand how the buckets are filled and drained. CB policing continues to replenish Bucket1 when a packet arrives. If Bucket1 is full, the extra, or spilled tokens, replenish Bucket2. If Bucket2 fills, the excess tokens spill onto the ground and are wasted. Figure 5-16 shows the basic process:

Figure 5-16 *Refilling Dual Token Buckets with CB Policing*


Refill Bytes Upon Arrival of Packet, per Formula:
(New_packet_arrival_time - previous_packet_arrival_time) * Policed_rate / 8
$.25 Token
Spillage Falls into Bucket2
Bucket1 Size: Bc
Bucket2 Size: Be
Spillage Wasted

With Bc and Be configured, CB policing uses dual token buckets, and the algorithm is simple:

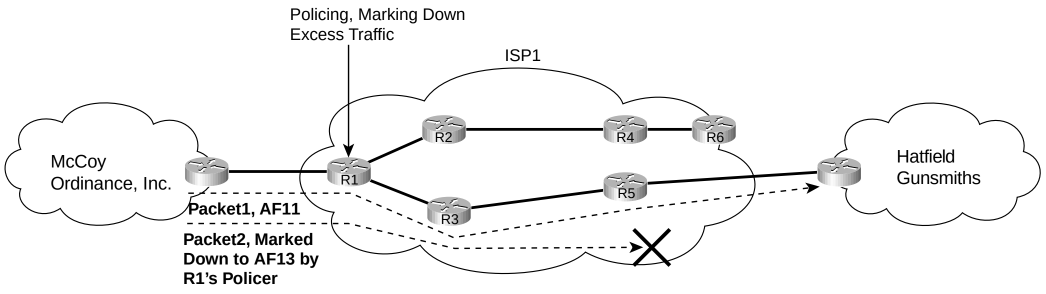
1. If the number of bytes in the packet is less than or equal to (<=) the number of tokens in Bucket1, the packet *conforms*. CB policing removes tokens from Bucket1 equal to the number of bytes in the packet, and performs the action for packets that conform to the contract.
2. If the packet does not conform, and the number of bytes in the packet is less than or equal to (<=) the number of tokens in Bucket2, the packet *exceeds*. CB policing removes tokens from Bucket2 equal to the number of bytes in the packet, and performs the action for packets that exceed the contract.
3. If the packet neither conforms nor exceeds, it *violates* the traffic contract. CB policing does not remove tokens from either bucket, and performs the action for packets that violate the contract.

Essentially, packets that fit within Bc conform, those that require the extra bytes allowed by Be exceed, and those that go beyond even Be are considered to violate the traffic contract.

Policing, but Not Discarding

Shapers queue excess packets, and policers discard excess packets. However, policers allow a sort of compromise, where the packets are not discarded, but they are marked so that if congestion occurs later, this particular packet is more likely to be discarded. Consider Figure 5-17, for instance, and the policing function on R1.

Figure 5-17 *Marking Down Packets with Policing*


Policing, Marking Down Excess Traffic
ISP1
McCoy Ordinance, Inc.
R1
R2
R3
R4
R5
R6
Hatfield Gunsmiths
Packet1, AF11
Packet2, Marked Down to AF13 by R1's Policer

In the figure, two packets travel over a route marked with dotted lines. Each is marked with DSCP AF11 as they enter R1. R1's policer decides that Packet1 conforms, but that Packet2 exceeds the policing rate. R1's policer re-marks Packet2 as AF13. DiffServ suggests that AF13 should be in the same queuing class as AF11, but with a higher drop preference than AF11. If no congestion occurs, both packets get through the network. If some congestion occurs, however, Packet2 has a higher chance of being discarded, because it has a DSCP that implies a higher preference to be dropped than does Packet1.

Policing by marking down the packet provides a compromise option. The ISP can protect the network from being overrun by aggressively discarding marked-down packets at the onset of congestion. Customer satisfaction can be improved by letting the customers take advantage of the capacity when it is available.

The same concept, on a limited basis, can be applied in Frame Relay and ATM networks. Frame Relay headers include the discard eligibility (DE) bit, and ATM headers include the cell loss priority (CLP) bit. Each of these single bits can be used to imply that the frame or cell is a better frame or cell to discard when congestion occurs.

Traffic-Shaping Tools

In Chapter 4, you read about how each different queuing tool has significant differences with how their schedulers work. Conversely, the four shaping tools covered next all basically work the same way. In fact, the four shaping tools share much of the underlying shaping code in IOS. Although some features differ, and the configurations certainly differ, most of the core functions behave the same.

Because the shaping tools behave the same for the most part, to prepare for the QoS exams, pay particular attention when the tools do differ. For instance, Frame Relay traffic shaping (FRTS) allows FRF.12 fragmentation at the same time as shaping, but the other shaping tools do not. Distributed traffic shaping (DTS) distributes processing to Versatile Interface Processor (VIP) cards in 7500 routers. Generic traffic shaping (GTS) and class-based shaping (CB shaping) both support a wide variety of interfaces on all Cisco router platforms. The similarities and differences are outlined in each shaping section in the chapter; make sure to review the tables comparing the tools as you prepare for the exams.

The following sections cover GTS first, CB shaping, DTS, and then FRTS.

Generic Traffic-Shaping Configuration

GTS performs traffic shaping using the same logic and features discussed in the introductory section of this chapter. It can be enabled on a large variety of interfaces. It can also adapt the rate based on BECN signals, and reflect BECNs on a VC after receiving a FECN. It supports a single queuing tool for the shaping queues: Weighted Fair Queuing (WFQ). It can also classify traffic, performing shaping on a subset of the traffic on an interface by classifying packets based on access-control lists (ACLs).

The only feature of GTS not already covered to some depth, other than configuration, is the concept of shaping a subset of the traffic on an interface or subinterface. GTS can classify traffic with an ACL; traffic permitted by the ACL is shaped based on the parameters specified on the same command. For instance, you could shape all FTP traffic to 32 kbps, and web traffic to 64 kbps.

Other than the material already covered, the only other thing really left to think about with GTS is how to configure it. Tables 5-7 and 5-8 list the configuration and **show** commands pertinent to GTS.

Table 5-7 *Command Reference for Generic Traffic Shaping*

Command	Mode and Function
traffic-shape rate *bit-rate* [*burst-size* [*excess-burst-size*]]	Interface configuration mode; enables GTS for a shaped rate, with optional Bc and Be settings, in bits
traffic-shape group *access-list bit-rate* [*burst-size* [*excess-burst-size*]]	Interface configuration mode; enables GTS for a shaped rate, only for traffic permitted by the referenced ACL, with optional Bc and Be settings, in bits
traffic-shape adaptive *bit-rate*	Interface configuration mode; enables adaptive shaping, and sets the minimum shaped rate
traffic-shape fecn-adapt	Interface configuration mode; enables the reflection of BECN signals upon receipt of an FECN

Table 5-8 *Exec Command Reference for Generic Traffic Shaping*

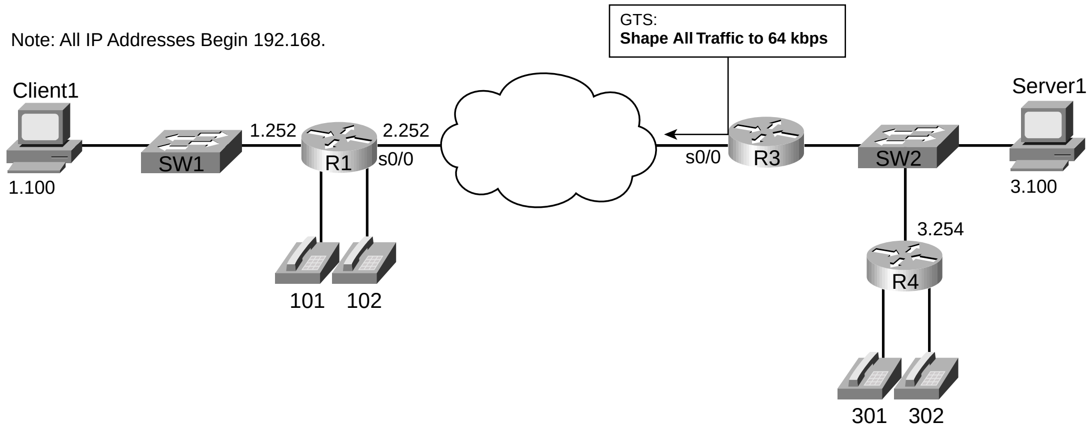
Command	Function
show traffic-shape [*interface-type interface-number*]	Lists information about the configuration details
show traffic-shape queue [*interface-number* [**dlci** *dlci-number*]]	Lists statistics about the queuing tool used on the shaping queue
show traffic-shape statistics [*interface-type interface-number*]	Lists statistics about shaping operation

The examples in this section use a familiar network diagram, as shown in Figure 5-18. The configuration shows R3, with a 128-kbps access rate, and a 64-kbps Frame Relay VC connecting to R1. Traffic from R3 to R1 is Shaped using the following criteria:

- Shape all traffic at a 64 kbps rate.
- Use the default setting for Tc.
- Enable the configuration per VC.
- Use WFQ on the physical interface output queues.
- Use WFQ on the shaping queues (only choice available).

In each example, the client downloads one web page, which has two frames inside the page. The web page uses two separate TCP connections to download two separate large JPG files that are shown in each of the two frames inside the browser window. The client also downloads a file using FTP. Additionally, a VoIP call is placed between extension 302 and 102. Example 5-1 shows the configuration and some sample **show** commands.

Figure 5-18 *Sample Network Used for GTS Configuration Examples*



Example 5-1 *CB Shaping on R3, 64-kbps Shape Rate*

```
interface serial0/0
 bandwidth 64
 load-interval 30
 fair-queue
interface serial0/0.1
 traffic-shape 64000

R3#show traffic-shape

Interface   Se0/0.1
       Access Target    Byte    Sustain   Excess     Interval  Increment Adapt
VC     List   Rate      Limit   bits/int  bits/int   (ms)      (bytes)   Active
-             64000     2000    8000      8000       125       1000      -

R3#show traffic-shape statistics
                  Access Queue   Packets   Bytes     Packets   Bytes     Shaping
I/F               List   Depth                       Delayed   Delayed   Active
Se0/0.1                  65      4285      687360    4248      680272    yes

R3#show traffic-shape statistics serial 0/0.1
                  Access Queue   Packets   Bytes     Packets   Bytes     Shaping
I/F               List   Depth                       Delayed   Delayed   Active
Se0/0.1                  41      4720      752636    4683      745548    yes

R3#show queue serial 0/0
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 4227
  Queueing strategy: weighted fair
```

continues

Example 5-1 *CB Shaping on R3, 64-kbps Shape Rate (Continued)*

```
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/7/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

R3#show traffic-shape queue serial 0/0.1
Traffic queued in shaping queue on Serial0/0.1
  Queueing strategy: weighted fair
  Queueing Stats: 64/1000/64/768 (size/max total/threshold/drops)
     Conversations  3/4/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 64 kilobits/sec

! Next one is a web connection
  (depth/weight/total drops/no-buffer drops/interleaves) 61/32384/734/0/0
  Conversation 14, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xAA21, ttl: 127,
  TOS: 0 prot: 6, source port 80, destination port 3041

! Next one is VoIP
  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 0, linktype: ip, length: 188
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x0060, ttl: 254,
TOS: 0 prot: 17, source port 19273, destination port 17641

! Next one is Passive FTP, not requiring port 20 on either side
  (depth/weight/total drops/no-buffer drops/interleaves) 2/32384/35/0/0
  Conversation 11, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xAA24, ttl: 127,
  TOS: 0 prot: 6, source port 4904, destination port 3043
```

The configuration itself is rather simple. You can configure GTS on the subinterface, or on the main interface in this case, because only one VC runs out of the interface. Because most installations tend to enable GTS on subinterfaces, the **traffic-shape rate 64000** command was added to interface s0/0.1 to enable GTS for a shaping rate of 64 kbps. GTS only allows WFQ on the shaping queues, so no other configuration command is needed to enable WFQ. The **fair-queue** command on the physical interface enables WFQ just for the interface output queues.

Three **show** commands list information about GTS. First, immediately after the configuration, the **show traffic-shape** command lists basic configuration information, and the derived values for Bc and Be of 8000 bits each. Bc was derived by IOS using the formula Bc = Tc * CIR, which is just a variation on the Tc = Bc/CIR formula. The Tc value defaults to 125 ms, but interestingly, GTS can use different default Tc values, based on the configuration settings you use.

GTS uses WFQ for the shaping queues; the next command in the example, **show traffic-shape queue**, supplies the same output that the **show queue** command does for WFQ on an interface.

Finally, the **show traffic-shape statistics** command lists basic statistics about GTS. The last column of output is particularly interesting, because it tells you whether shaping is currently active. IOS does not need to shape all the time, so this column just lists the current state when the command was issued. Shaping activates for three reasons:

- If adaptive shaping is currently reacting to BECNs by lowering the shaped rate
- If Bc and Be have been exceeded, causing packets to be delayed
- If Frame Relay fragmentation is enabled

Queuing occurs on the physical interface when the traffic load exceeds the interface clock rate, and shaping queues form when the traffic load exceeds the shaping rate. In this case, with a single VC, shaped at 64 kbps, and a 128-kbps clock rate on the physical interface, no queues should form on the physical interface. The offered load on the physical link should not exceed 64 kbps, because that is how much traffic the GTS will allow to pass through the only subinterface in the example. The output of the **show queue** and **show traffic-shape queue** commands in the example support these facts. The **show queue** command lists information about WFQ on the physical interface; it does not list any flows, because no congestion is occurring there. However, the **show traffic-shape queue** command lists information about WFQ's performance with the shaping queues. In this case, it lists information about a VoIP call, and one FTP download.

This particular example reminds us of the importance of the queuing methods supported by the shaping tool. Because GTS only supports WFQ in conjunction with shaping, there is no low-latency option, such as LLQ. The VoIP call in this particular example had a completely unusable quality level.

A second GTS configuration example is shown next. For the second configuration, imagine that the Frame Relay provider actually polices the VC from R3 to R1 at 96 kbps. You engineer the network to support a single G.729 VoIP call, which takes about 28 kbps. You decide that you want to be very careful about packet drops, latency, and jitter for this voice traffic, so you decide to shape all traffic except voice. To avoid drops inside the cloud, you shape the rest of the traffic to a rate of 64 kbps (so that the combined single VoIP call, and the shaped rate of 64 kbps, do not exceed the policing rate in the Frame Relay network). The next example shows the configuration, and the criteria for the configuration is as follows:

- Shape non-VoIP traffic at 64 kbps.
- Choose values so Tc is 50 ms.
- Enable the configuration on the subinterface.
- Use WFQ on the physical interface output queues.
- Use WFQ on the shaping queues (the only choice available).

The traffic generated for this second example is the same type of traffic generated for the first—a single VoIP call, one web page downloaded with two frames inside the page (causing two

TCP connections), and an FTP get. Example 5-2 shows the configuration and some sample **show** commands.

Example 5-2 *GTS on R3, 64 kbps for Non-Voice Traffic, Tc = 50 ms*

```
interface serial0/0
 bandwidth 64
 load-interval 30
 fair-queue
interface serial0/0.1
 traffic-shape group 101 64000 3200
!
access-list 101 deny udp any range 16384 32767 any range 16384 32767
access-list 101 permit ip any any

R3#show traffic-shape

Interface   Se0/0.1
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
-      101    64000     800    3200      3200      50        400       -

R3#show traffic-shape statistics
                  Access Queue   Packets   Bytes     Packets   Bytes     Shaping
I/F               List   Depth                       Delayed   Delayed   Active
Se0/0.1           101    0       94        120156    59        77376     no

R3#show traffic-shape queue
Traffic queued in shaping queue on Serial0/0.1
 Traffic shape group: 101
  Queueing strategy: weighted fair
  Queueing Stats: 3/1000/64/0 (size/max total/threshold/drops)
     Conversations  1/2/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 64 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 3/32384/0/0/0
  Conversation 11, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xB077, ttl: 127,
  TOS: 0 prot: 6, source port 4904, destination port 3043

R3#show access-lists
Extended IP access list 101
    deny udp any range 16384 32767 any range 16384 32767 (18638 matches)
    permit ip any any (1257 matches)
```

The configuration enables GTS by using the **traffic-shape group 101 64000 3200** command. This command refers to ACL 101, which permits all traffic except VoIP traffic, implying that shaping occurs for all traffic except VoIP. The shaped rate of 64 kbps implies that the traffic permitted by the ACL will be shaped to 64 kbps. Also note that, according to the **traffic-shape** command, the Bc value was set to 3200, which should give a Tc of 3200/64,000, or 50 ms. The

show traffic-shape command confirms the calculated Tc value, listing it as 50 ms. Also note that the Be value was set to 3200, because GTS defaults Be to be equal to Bc.

NOTE All IOS shapers use bits as the unit when setting Bc and Be; both policers use bytes as the unit.

The **show access-lists** command lists the number of packets permitted and denied by ACL 101. Because ACL 101 is used for matching by GTS, this also tells us how many packets were shaped by GTS, and how many were not.

GTS supports several useful traffic-shaping features. It can be used on many types of interfaces, and it can shape a subset of the traffic by referencing ACLs. It can also adapt to BECNs, and reflect FECNs. However, GTS does have several drawbacks. It only supports WFQ in the shaping queues, and it cannot support FRF.12 fragmentation on Frame Relay subinterfaces. Table 5-9 summaries the key points for comparison between the various traffic-shaping tools, and lists whether GTS supports each.

Table 5-9 *Comparison of Traffic-Shaping Tools: GTS*

Feature	GTS
Supports ATM, FR, HDLC, PPP, and LAN interfaces	Yes
Can be enabled on interfaces and subinterfaces	Yes
Can be enabled per FR DLCI to support per-VC shaping on multipoint interfaces	No
Supports adaptive shaping	Yes
Supports concurrent FRF.12 Frame Relay fragmentation	No
Queuing methods allowed in shaping queue	WFQ
Concurrent queuing methods on physical interface	All
Can be configured using MQC commands	No
Can classify traffic to shape a subset of the traffic on an interface/VC	Yes
Default Tc	Variable
Distributes shaping processing to VIPs in 7500 series routers	No

Class-Based Shaping Configuration

The underlying processes of class-based (CB) shaping are the same as GTS. Just like GTS, CB shaping can be enabled on a large variety of interfaces. It can also adapt the rate based on BECN signals, and reflect BECNs on a VC after receiving an FECN. It can also perform shaping on a subset of the traffic on an interface. Finally, the configuration is simple, because like all other QoS features starting with the words "class based," CB shaping uses the Modular QoS command-line interface (MQC) for configuration.

However, CB shaping provides some significant functional improvements over GTS. First, CB shaping supports several queuing methods for the shaping queues, namely FIFO, WFQ, CBWFQ, and LLQ. With these additional queuing tools, multiservice traffic can be better supported. For instance, LLQ can be applied to the shaping queues, providing a low-latency service for voice and video traffic.

CB shaping can be configured to work just like GTS, but it has an option with which you can tell it to send even more traffic during each interval. To work just like GTS, you would configure CB shaping using the **shape average** command, with which you configure the shaping rate, and optionally the Bc and Be values, in bits. CB shaping sends Bc bits per Tc, or Bc + Be bits after periods of low activity, just like GTS.

Alternatively, you can tell CB shaping to send Bc + Be bits in *every* interval. To do so, just use the **shape peak** command rather than **shape average**.

The biggest difference between GTS and CB shaping lies in the MQC configuration. Tables 5-10 and 5-11 list the configuration and **show** commands pertinent to CB shaping.

Table 5-10 *Command Reference for Class-Based Shaping*

Command	Mode and Function
shape [**average** | **peak**] *mean-rate* [[*burst-size*] [*excess-burst-size*]]	Policy-map class configuration mode; enables shaping for the class, setting the shaping rate, and optionally Bc and Be. The average option causes shaping as normal; the peak option causes Bc + Be to be sent per Tc.
Shape adaptive *min-rate*	Policy-map class configuration mode; enables the minimum rate for adaptive shaping. The maximum rate is configured with the **shape average** or **shape peak** command.
Shape fecn-adapt	Policy-map class configuration mode; enables reflection of BECN signals upon receipt of an FECN.
service-policy {**input** | **output**} *policy-map-name*	Interface or subinterface configuration mode; enables CB shaping on the interface.
class-map *class-map-name*	Global config; names a class map, where classification options are configured.
Match . . .	**class-map** subcommand; defines specific classification parameters.
match access-group {*access-group* | **name** *access-group-name*}	**class-map** subcommand; references either numbered or named ACL.

Table 5-10 *Command Reference for Class-Based Shaping (Continued)*

Command	Mode and Function
match source-address mac *address*	**class-map** subcommand; references the source MAC address that forwarded the packet to this router, typically the previous-hop router.
match ip precedence *ip-precedence-value* [*ip-precedence-value ip-precedence-value ip-precedence-value*]	**class-map** subcommand; references one or more IP precedence values. A packet with any of the listed values matches.
match mpls experimental *number*	**class-map** subcommand; references MPLS Experimental bits.
match cos *cos-value* [*cos-value cos-value cos-value*]	**class-map** subcommand; references one or more class of service (CoS) values. A packet with any of the listed values matches.
match destination-address mac *address*	**class-map** subcommand; references the destination MAC address of the device to which the packet will be forwarded next.
match input-interface *interface-name*	**class-map** subcommand; matches based on the interface in which the packet was received.
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	**class-map** subcommand; references one or more IP DSCP values. A packet with any of the listed values matches.
match ip rtp *starting-port-number port-range*	**class-map** subcommand; references a range of UDP ports, only matching the even numbered ports, which carry voice payload.
match qos-group *qos-group-value*	**class-map** subcommand; matches based on QoS group.
match protocol *protocol-name*	**class-map** subcommand; matches based on NBAR-defined protocol types.
match protocol citrix [**app** *application-name-string*].	**class-map** subcommand; matches based on NBAR-defined Citrix application types.
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	**class-map** subcommand; matches based on NBAR-discovered details inside the host name or URL string.
match any	**class-map** subcommand; matches all packets.
policy-map *policy-map-name*	Global config; names a policy, which is a set of actions to perform.
class *name*	**policy-map** subcommand; identifies which packets on which to perform some action by referring to the classification logic in a class map.

Table 5-11 *Exec Command Reference for Class-Based Shaping*

Command	Function
show policy-map *policy-map-name*	Lists configuration information about all MQC-based QoS tools
show policy-map *interface-spec* [**input** | **output**] [**class** *class-name*]	Lists statistical information about the behavior of all MQC-based QoS tools

For the sake of comparison, the following two examples use requirements very similar to the two examples for GTS. The first configuration shows R3, with a 128-kbps access rate, and a 64-kbps Frame Relay VC connecting to R1. The criteria for the configuration is as follows:

- Shape all traffic at a 64-kbps rate.
- Use the default setting for Tc.
- Enable the configuration on the subinterface.
- Use WFQ on the physical interface.
- Use the default queuing method for the shaping queues.

In each example, the client downloads one web page, which has two frames inside the page. The web page uses two separate TCP connections to download two separate large JPG files. The client also downloads a file using FTP get. In addition, a VoIP call is placed between extension 302 and 102. Example 5-3 shows the configuration and some sample **show** commands.

Example 5-3 *CB Shaping on R3, 64-kbps Shape Rat*

```
policy-map shape-all
  class class-default
   shape average 64000
!
interface serial0/0
 bandwidth 64
 load-interval 30
 fair-queue
interface serial0/0.1
 service-policy output shape-all

R3#show queue serial 0/0
  Input queue: 0/75/1/0 (size/max/drops/flushes); Total output drops: 5965
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/90 (size/max total/threshold/drops)
     Conversations  0/7/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

R3#show policy-map
  Policy Map shape-all
    Class class-default
```

Example 5-3 *CB Shaping on R3, 64-kbps Shape Rat (Continued)*

```
      Traffic Shaping
         Average Rate Traffic Shaping
                 CIR 64000 (bps) Max. Buffers Limit 1000 (Packets)

R3#show policy-map interface s0/0.1

 Serial0/0.1

  Service-policy output: shape-all

    Class-map: class-default (match-any)
      7718 packets, 837830 bytes
      30 second offered rate 69000 bps, drop rate 5000 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
            64000/64000     2000   8000      8000      125       1000

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      56        6393      692696    6335      684964    yes
```

The CB shaping configuration uses the default class (class-default), and a **policy-map** (shape-all), which is enabled on serial 0/0.1 using the **service-policy output shape-all** command. The command **class-map class-default** matches all packets. The command **policy-map shape-all** only refers to the class-default class—essentially, classification is configured, but it matches all traffic. Inside **class class-default**, the **shape average 64000** command shapes the traffic to 64 kbps.

Just like with other MQC-based tools, the **show policy-map** and **show policy-map interface serial0/0.1** commands provide all the interesting information about this example. In fact, the highlighted output from these commands in Example 5-3 looks strikingly similar to the output from the **show traffic-shape** commands used with GTS; remember, much of the underlying code is the same code. Note that **show policy-map** just lists the same information in the configuration, whereas **show policy-map interface** lists statistics, tells you whether shaping is currently active, and lists the computed values, such as Bc and Be in this case. Bc = Tc * CIR, or .125 seconds * 64,000 bps, or 8000 bits. Be defaults to the same size as Bc.

In this example, packets are enqueued on the subinterface due to shaping, but few packets are enqueued on the physical interface. With a 128-kbps clock rate, and a shaping rate for all traffic on the interface of 64 kbps, no congestion should occur on the physical interface.

By default, CB shaping uses simple FIFO Queuing for the shaping queue. A common misconception about CB shaping is that because the MQC commands are used, CBWFQ is automatically used—but that is not true! In this example, FIFO Queuing is used for the shaping queue.

A second CB shaping configuration example is shown next; the criteria and motivation are the same as the second GTS example. For the second configuration, imagine that the Frame Relay provider actually polices the VC from R3 to R1 at 96 kbps. You engineer the network to support a single G.729 VoIP call, which takes about 28 kbps. You decide that you want to be very careful about packet drops, latency, and jitter for this voice traffic, so you decide to shape all traffic except voice. To avoid drops inside the cloud, you shape the rest of the traffic to a rate of 64 kbps (so that the combined single VoIP call, and the Shaped rate of 64 kbps, do not exceed the policing rate in the Frame Relay network). The next example shows the configuration, and the criteria for the configuration is as follows:

- Shape non-VoIP traffic at 64 kbps.
- Choose values so Tc is 50 ms.
- Enable the configuration on the subinterface.

In this case, a single VoIP call and one web page connection with two frames inside the page are used, plus an FTP get. Example 5-4 shows the configuration and some sample **show** commands.

Example 5-4 *CB Shaping on R3, 64-kbps for Non-Voice Traffic, Tc = 50 ms*

```
ip cef
!

class-map match-all voip-rtp
  match ip rtp 16384 16383
!
!
policy-map shape-non-voip
  class voip-rtp
  class class-default
   shape average 64000 3200
!
interface serial0/0
 bandwidth 64
 load-interval 30
 fair-queue
interface serial0/0.1
 service-policy output shape-non-voip

R3#show policy-map
  Policy Map shape-non-voip
    Class voip-rtp
    Class class-default
      Traffic Shaping
         Average Rate Traffic Shaping
                 CIR 64000 (bps) Max. Buffers Limit 1000 (Packets)
            Bc 3200

R3#show policy-map interface serial 0/0.1
```

Example 5-4 *CB Shaping on R3, 64-kbps for Non-Voice Traffic, Tc = 50 ms (Continued)*

```
Serial0/0.1

  Service-policy output: shape-non-voip

    Class-map: voip-rtp (match-all)
      50379 packets, 3224256 bytes
      30 second offered rate 25000 bps
      Match: ip rtp 16384 16383

    Class-map: class-default (match-any)
      5402 packets, 6634217 bytes
      30 second offered rate 66000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
            64000/64000     800    3200      3200      50        400

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      8         31        40168     30        38764     yes

R3#show queue serial 0/0
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 6083
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/7/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
```

The configuration in the example is relatively simple, but detailed. It begins with some familiar MQC commands configuring LLQ. The **class-map match-all voip-rtp** command creates a new class map, with parameters to match all VoIP payload traffic. The **policy-map shape-non-voip** command refers to **class voip-rtp**, with no parameters. With no parameters, no actions are taken for packets in the class, which includes no shaping action. **Class class-default** refers to the class that matches all traffic, with the **shape average 64000 3200** command shaping to 64 kbps, with a Bc of 3200 bits. (CB shaping then calculates Tc as 3200/64000, or 50 ms.) Note that because **class voip-rtp** occurs first in the policy map, all VOIP traffic matches that class, and is not shaped.

The **show policy-map** command just restates the information in the configuration. Note the absence of commands under **class voip-rtp**—remember, no commands were added under the command **class voip-rtp**—effectively creating a "do nothing" class. The **class class-default** command matches all other traffic, shaping to 64 kbps. In the **show policy-map interface s0/0.1** command, you see that shaping is enabled only for **class class-default**, but not for **class voip-rtp**.

This example has repeated the goals of the second GTS example. However, CB shaping can be used with much better effect, given the base requirements, by taking advantage of LLQ, which is not available with GTS. In the preceding example, the motivation to shape was based on the Frame Relay provider's policing of the VC at 96 kbps. If the VoIP call was not active, the data still only gets 64 kbps worth of bandwidth! Although that's great for the single voice call, it may not be a good choice for the data. The real requirement is for high-quality voice, with good treatment of the data as a secondary goal, knowing that the service provider is policing at 96 kbps.

A better solution is to use CB shaping's capability to include CBWFQ or LLQ as the queuing tools for the shaping queues. In the first two examples, the shaping queue was a single FIFO queue, because no queuing was configured. In the next example, CB shaping will shape all traffic, including voice, at 96 kbps, but with LLQ used on the shaping queues to ensure good voice quality. To that end, this configuration forces Tc down to 10 ms, which is the recommended value for Tc when delay-sensitive voice and video traffic is shaped. The revised requirements are as follows:

- Ensure high-quality VoIP for one G.729 call, through low loss, low delay, and low jitter.
- Achieve VoIP loss goal by not exceeding the policing rate of the service provider.
- Achieve VoIP delay and jitter goal by using LLQ.
- Choose values so Tc is 10 ms.
- Shape all traffic, including voice, because LLQ will always take the voice traffic next anyway.

Example 5-5 shows the configuration.

Example 5-5 *CB Shaping on R3, 96-kbps Shape Rate, with LLQ for Shaping Queues*

```
ip cef
!

class-map match-all voip-rtp
  match ip rtp 16384 16383
!
!
! The following Policy map is used for queuing (LLQ)
!
policy-map queue-voip
  class voip-rtp
   priority 32
  class class-default
   fair-queue
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! The following Policy map is used for Shaping;
! It refers to policy-map queue-voip
!
policy-map shape-all
  class class-default
   shape average 96000 960
   service-policy queue-voip
!
```

Example 5-5 *CB Shaping on R3, 96-kbps Shape Rate, with LLQ for Shaping Queues (Continued)*

```
interface serial0/0
 bandwidth 64
 load-interval 30
 fair-queue
interface serial0/0.1
 service-policy output shape-all

R3#show policy-map
  Policy Map shape-all
    Class class-default
      Traffic Shaping
         Average Rate Traffic Shaping
                 CIR 96000 (bps) Max. Buffers Limit 1000 (Packets)
            Bc 960
      service-policy queue-voip

  Policy Map queue-voip
    Class voip-rtp
      Weighted Fair Queueing
            Strict Priority
            Bandwidth 32 (kbps) Burst 800 (Bytes)
    Class class-default
      Weighted Fair Queueing
            Flow based Fair Queueing Max Threshold 64 (packets)

R3#show policy-map interface serial 0/0.1

 Serial0/0.1

  Service-policy output: shape-all

    Class-map: class-default (match-any)
      5189 packets, 927835 bytes
      30 second offered rate 91000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
            96000/96000     1200   960      960      10        120

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      17        5172      910975    4002      831630    yes

      Service-policy : queue-voip
        Class-map: voip-rtp (match-all)
          4623 packets, 295872 bytes
          30 second offered rate 25000 bps, drop rate 0 bps
          Match: ip rtp 16384 16383
          Weighted Fair Queueing
            Strict Priority
```

continues

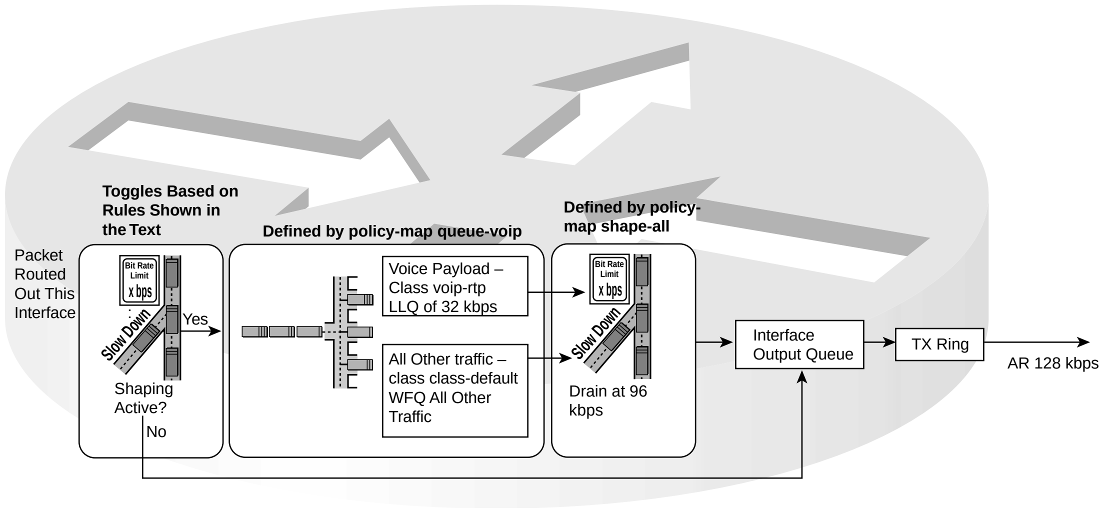
Example 5-5 *CB Shaping on R3, 96-kbps Shape Rate, with LLQ for Shaping Queues (Continued)*

```
            Output Queue: Conversation 24
            Bandwidth 32 (kbps) Burst 800 (Bytes)
            (pkts matched/bytes matched) 3528/225792
            (total drops/bytes drops) 0/0

        Class-map: class-default (match-any)
          566 packets, 631963 bytes
          30 second offered rate 65000 bps, drop rate 0 bps
          Match: any
          Weighted Fair Queueing
            Flow Based Fair Queueing
            Maximum Number of Hashed Queues 16
        (total queued/total drops/no-buffer drops) 17/0/0
```

Example 5-5 contains a lot of interesting command output. It begins with the configuration, which is followed by several **show** commands. The configuration contains two policy maps; one configures shaping, and the other configures LLQ, which is applied to the packets held because of shaping.

Because the interaction between the two policy maps can be confusing, Figure 5-19 shows the general idea behind what is happening in the configuration.

Figure 5-19 *Interaction Between Shaping Policy Map shape-all and Queuing Policy Map queue-voip*


Toggles Based on Rules Shown in the Text
Packet Routed Out This Interface
Bit Rate Limit x bps
Slow Down
Yes
Shaping Active?
No
Defined by policy-map queue-voip
Voice Payload – Class voip-rtp LLQ of 32 kbps
All Other traffic – class class-default WFQ All Other Traffic
Defined by policy-map shape-all
Bit Rate Limit x bps
Slow Down
Drain at 96 kbps
Interface Output Queue
TX Ring
AR 128 kbps

Scanning the figure from left to right, packets are first routed out the subinterface. Then the IOS checks to see whether shaping is active. Shaping becomes active when a single packet exceeds the traffic contract; shaping only becomes inactive when all the shaping queues are drained, and the ensuing packets are not exceeding the traffic contract. Therefore, the shaper must be involved at this step to decide whether to try to queue the packet into the shaping queue.

If the packet exceeds the contract, the shaper needs to queue the packet. In this case, instead of a single queue, you have a queuing system for the shaping queues as defined by policy map queue-voip. This policy map defines two queues, with one being a 32-kbps low-latency queue into which all payload is placed. This queue was created with the **class voip-rtp** command inside policy map queue-voip. Because queue-voip defines queuing, and no other details have been configured, all the rest of the packets go into a second queue, associated with the class-default class. (WFQ is applied to the packets inside the class-default queue, by default, but it can be disabled.)

The next step in the process really shows how integrated the queuing and shaping functions are in with this configuration. After packets have been enqueued in one of the shaping queues, CB shaping must decide *when* to take a packet from a queue. However, the shaper does not decide *which* packet to take—the queuing logic as defined in policy map queue-voip determines which packet to dequeue next. Therefore, the **shape average 96000 960** command tells CB shaping the rate and Bc values to use when deciding when it can next dequeue packets from the shaping queues. When CB shaping releases the next packet from the shaping queues, the packet is placed into the interface output queue, and it finally exits the serial interface.

The interaction between CB shaping, and the queuing tool used for the shaping queues (LLQ in this case) can be a bit confusing. In particular, the shaper must make the decision about whether to put a packet into the shaping queue, and then the shaper decides when the next packet can be taken from a shaping queue, and then the queuing scheduler decides which packet to service next from the shaping queues.

Now taking a closer look at the configuration. The **policy-map shape-all** command creates the shaping configuration, with the **shape average 96000 960** command enabling shaping and defining a Bc so that Tc = 960/96,000, or 10 ms. Although a Bc of 960 bits may seem smaller than a typical packet, you would typically also use a fragmentation tool on this interface when VoIP is in the network. Fragmentation would happen to cause the largest frame size to be shorter than 960 bits. In addition, the **service-policy queue-voip** command enables LLQ inside the class-default class inside policy map shape-all—thereby enabling LLQ for the shaping queues.

A closer look at policy map queue-voip reveals two classes. The **class voip-rtp** command matches all VoIP payload, and is assigned 32 kbps with the **priority 32** command, making it the low-latency queue. Remember, LLQ actually polices the traffic in the low-latency queue at this rate, so this queue can send only 32 kbps. Because all traffic is shaped at 96 kbps, 64 kbps remains, which is guaranteed for queuing; the **class class-default** command matches all other traffic, guaranteeing the remaining 64 kbps to the class.

Finally, the **service-policy output shape-all** subcommand on interface s 0/0.1 enables the policy on the subinterface, which enables CB shaping. Therefore, the CB shaping logic, and related LLQ logic, is applied to all packets exiting subinterface serial 0/0.1.

The **show policy-map** commands reflect the details in the configuration. The command lists **policy-map shape-all** as performing traffic shaping, and **policy-map queue-voip** as performing LLQ. The **show policy-map interface s0/0.1** command lists the shaping parameters and statistics, just like it did in the earlier examples. However, halfway through the command output, a new section has appeared, with details about **policy-map shape-all**'s use of the LLQ **policy-map queue-voip**. The same information seen in CBWFQ and LLQ **show policy-map interface** commands in Chapter 4 appears in this area, including packet rates and byte rates.

This example allows the G.729 call to work well, because LLQ will always service the voip-rtp class when a packet is waiting. If no VoIP calls are up, however, the rest of the traffic can send at 96 kbps, the maximum sending rate for shaping. In other words, the rest of the traffic is guaranteed 64 kbps, with a maximum limit of the shaping rate. With LLQ, the VoIP always receives good treatment when the call is up, unless the VoIP traffic exceeds the LLQ policed rate of 32 kbps.

CB shaping supports all the features that GTS supports, plus a few others. It can support CBWFQ and LLQ for the shaping queues, and of course MQC commands are used to configure it. However, neither CB shaping nor GTS support FRF.12 fragmentation on Frame Relay subinterfaces, and neither can be enabled per-VC on multipoint Frame Relay subinterfaces.

NOTE Fragmentation is very important on slower-speed links when supporting voice and video as well as data over a link. The lack of support for Frame Relay fragmentation by GTS, CB shaping, and DTS seemingly makes FRTS the only reasonable option for shaping in networks with multiservice traffic. However, GTS, CB shaping, and DTS can support another form of fragmentation over Frame Relay, called "LFI using Multilink Point-to-Point Protocol over Frame Relay." This feature is not covered on either of the QoS exams, but it is a reasonable alternative in real life, which also makes GTS, DTS, and CB shaping reasonable alternatives to FRTS on Frame Relay interfaces.

Table 5-12 summaries the key points for comparison between GTS and CB shaping.

Table 5-12 *Comparison of Traffic Shaping Tools: GTS and CB Shaping*

Feature	GTS	CB Shaping
Supports ATM, FR, HDLC, PPP, LAN interfaces	Yes	**Yes**
Can be enabled on interfaces and subinterfaces	Yes	**Yes**
Can be enabled per FR DLCI to support per-VC shaping on multipoint interfaces	No	**No**
Supports adaptive shaping	Yes	**Yes**
Supports concurrent FRF.12 Frame Relay fragmentation	No	**No**

Table 5-12 *Comparison of Traffic Shaping Tools: GTS and CB Shaping (Continued)*

Feature	GTS	CB Shaping
Queuing methods allowed in shaping queue	WFQ	**FIFO, WFQ**, **CBWFQ**, **LLQ**
Concurrent queuing methods on physical interface	All	**All**
Can be configured using MQC commands	No	**Yes**
Can classify traffic to shape a subset of the traffic on an interface/VC	Yes	**Yes**
Default Tc	Variable	**125 ms**
Distributes shaping processing to VIPs in 7500 series routers	No	**No**

Distributed Traffic Shaping (DTS) Configuration

DTS shapes traffic identically to CB shaping, but with processing distributed to VIPs in a 7500 series router. In fact, DTS and CB shaping configurations are very similar, but with a few extra requirements. Distributed CEF must be configured for the interfaces on which DTS should operate. DTS also has a few idiosyncrasies that are not mentioned in either of the QoS courses, so they are unlikely to be on the exam.

If you want to read further about DTS, look for the "Cisco AVVID Enterprise Quality of Service Infrastructure Quality of Service Design" document at www.cisco.com/warp/customer/771/srnd/qos_srnd.pdf. On pages 4 through 18 of that document, the text outlines some of the unique requirements for DTS that are beyond the scope of the QoS exams, but important in real networks. This document is a very good reference for QoS design recommendations across all QoS functions and Cisco product lines.

To configure DTS, just complete these listed tasks:

Step 1 Ensure VIP2-40 or better line cards are installed.

Step 2 For E3 or faster interfaces, use VIP2-50s or better.

Step 3 Configure dCEF globally and on the necessary interfaces.

Step 4 Configure CB shaping as shown in the preceding section.

Table 5-13 lists the comparison features for the shaping tools, with DTS highlighted.

Table 5-13 *Comparison of Traffic Shaping Tools: GTS, CB Shaping, and DTS*

Feature	GTS	CB Shaping	DTS
Supports ATM, FR, HDLC, PPP, LAN interfaces	Yes	Yes	**Yes**
Can be enabled on interfaces and subinterfaces	Yes	Yes	**Yes**

continues

Table 5-13 *Comparison of Traffic Shaping Tools: GTS, CB Shaping, and DTS (Continued)*

Feature	GTS	CB Shaping	DTS
Can be enabled per FR DLCI to support per-VC shaping on multipoint interfaces	No	No	**No**
Supports adaptive shaping	Yes	Yes	**Yes**
Supports concurrent FRF.12 Frame Relay fragmentation	No	No	**No**
Queuing methods allowed in shaping queue	WFQ	FIFO, WFQ, CBWFQ, LLQ	**FIFO, WFQ, CBWFQ, LLQ**
Concurrent queuing methods on physical interface	All	All	**All**
Can be configured using MQC commands	No	Yes	**Yes**
Can classify traffic to shape a subset of the traffic on an interface/VC	Yes	Yes	**Yes**
Default Tc	Variable	125 ms	**125 ms**
Distributes shaping processing to VIPs in 7500 series routers	No	No	**Yes**

Frame Relay Traffic Shaping (FRTS) Configuration

Frame Relay traffic shaping (FRTS) differs from the other three shaping tools in several significant ways. The most obvious difference is the most important—FRTS applies to Frame Relay only. But the basic shaping function is the same, with the same parameters—a shaped rate, which is often set to CIR; a Tc interval, which defaults to 125 ms; and the Bc value is either set, or calculated based on the Tc = Bc/CIR formula.

The first big difference between FRTS and the other shapers has to do with queuing tool support. FRTS does not allow any IOS queuing tool to be used on the physical interface when FRTS is configured for VCs on the interface. Even if a queuing tool has already been configured, IOS removes the configuration from the physical interface when FRTS is enabled. FRTS does supply the largest list of options for queuing tools for the shaping queue: FIFO, PQ, CQ, CBWFQ, LLQ, and WFQ are all available.

NOTE For you exam takers, be aware that at the time this book went to press the Cisco QoS course book incorrectly claims that FRTS supports WFQ on the physical interface; the DQOS course book does not say anything about queuing on the physical interface with FRTS.

The next important difference is that FRTS supports concurrent Frame Relay fragmentation (FRF) using Frame Relay Forum Implementation Agreement 12, also known as FRF.12. With FRF.12, large packets are fragmented, with "large" being defined with configuration commands. Small packets are interleaved, so that a small packet does not have to wait on the long serialization delay associated with the longer original packets. Interestingly, to perform the interleaving feature, FRF uses two queues on the physical interface, with one of the queues holding small, unfragmented packets, and the other holding the fragments of large packets. The queue holding the unfragmented packets is treated like a low-latency queue, always being serviced first. Therefore, although FRTS does not allow any queuing tools on the physical interface, FRF.12 supplies the added benefit of at least a two-queue system, called dual-FIFO, to the physical interface.

FRTS, unlike the other shaping tools, cannot shape a subset of the traffic on an interface. Each of the other three shapers can be configured on one subinterface, and not the other, essentially enabling shaping on only some of the traffic leaving an interface. The other three shapers can also configure classification parameters, and shape only part of the traffic on a subinterface. Unlike the other shapers, when FRTS is enabled on the physical interface, all traffic on the interface is shaped in some way. In fact, with FRTS enabled, each VC is shaped separately. However, you cannot enable FRTS for only a subset of the VCs on an interface, nor for a subset of the traffic exiting a single VC.

FRTS shapes all VCs on an interface after it has been enabled on that interface. To enable FRTS, add the **frame-relay traffic-shape** command under the physical interface. If you add no other configuration commands, FRTS uses default settings and shapes each individual VC. If you include additional configuration per VC, FRTS uses those parameters rather than the defaults. In any case, FRTS always shapes each VC separately after it has been enabled on the physical interface.

Unlike the other three shapers, FRTS can dynamically learn the CIR, Bc, and Be values configured per VC at the Frame Relay switch and use those settings for shaping. Cisco's WAN switching products (from the Stratacom acquisition in the mid-1990s) use an Enhanced LMI (ELMI) feature, which IOS understands. Using ELMI, the switch just announces the CIR, Bc, and Be for each VC to the router. So, if you want to use FRTS only to shape to CIR, and the Frame Relay network uses Cisco switches, you can just enable FRTS and ELMI on the interface, and the rest (Bc, CIR, and so on) will be learned dynamically for each VC.

Finally, the biggest difference relates to FRTS configuration. The commands used differ totally from the other three tools. Tables 5-14 and 5-15 list the configuration and **show** commands pertinent to FRTS, respectively.

Table 5-14 *Command Reference for Frame Relay Traffic Shaping*

Command	Mode and Function
frame-relay traffic-shaping	Interface subcommand; enables FRTS on the interface.
class *name*	Interface DLCI subcommand; enables a specific FRTS map class for the DLCI.
frame-relay class *name*	Interface or subinterface command; enables a specific FRTS map class for the interface or subinterface.
map-class frame-relay *map-class-name*	Global configuration mode; names a map class, and places CLI into map-class configuration mode
frame-relay priority-group *list-number*	Map-class configuration mode; enables PQ for the shaping queues associated with this map class
frame-relay custom-queue-list *list-number*	Map-class configuration mode; enables CQ for the shaping queues associated with this map class
frame-relay fair-queue [*congestive_discard_threshold* [*number_dynamic_conversation_queues* [*number_reservable_conversation_queues* [*max_buffer_size_for_fair_queues*]]]]	Map-class configuration mode; enables WFQ for the shaping queues associated with this map class
service-policy {**input** | **output**} *policy-map-name*	Map-class configuration mode; enables LLQ or CBWFQ on the shaping queues associated with the map class.
frame-relay traffic-rate *average* [*peak*]	Map-class configuration mode; sets the shaped rate, and the EIR*. Bc and Be are calculated from these, based on Tc of 125ms.
frame-relay bc {**in** | **out**} *bits*	Map-class configuration mode; sets the Bc value. Alternative configuration option to **frame-relay traffic-rate**.
frame-relay be {**in** | **out**} *bits*	Map-class configuration mode; sets the Be value. Alternative configuration option to **frame-relay traffic-rate**.
frame-relay cir {**in** | **out**} *bps*	Map-class configuration mode; sets the CIR value. Alternative configuration option to **frame-relay traffic-rate**.
frame-relay adaptive-shaping {**becn** | **foresight**}	Map-class configuration mode; enables adaptive shaping, specifying either BECN or Foresight for signaling.
frame-relay mincir {**in** | **out**} *bps*	Map-class configuration mode; sets the minimum CIR used for adaptive shaping.

Table 5-14 *Command Reference for Frame Relay Traffic Shaping (Continued)*

Command	Mode and Function
frame-relay tc *milliseconds*	Map-class configuration mode; for 0 CIR VCs, sets the Tc value.
frame-relay qos-autosense	Interface configuration mode; uses ELMI to automatically discover CIR, Bc, and Be settings for each VC.

* EIR = excess information rate

Table 5-15 *Exec Command Reference for Frame Relay Traffic Shaping*

Command	Function
show frame-relay pvc [**interface** *interface*] [*dlci*]	Shows PVC statistics, including shaping statistics
show traffic-shape [*interface-type interface-number*]	Shows information about FRTS configuration per VC
show traffic-shape queue [*interface-number* [**dlci** *dlci-number*]]	Shows information about the queuing tool used with the shaping queue
show traffic-shape statistics [*interface-type interface-number*]	Shows traffic-shaping statistics

For the sake of comparison, the first FRTS example follows the same requirements as the first GTS and CB shaping examples. The configuration shows R3, with a 128-kbps access rate, and a 64-kbps Frame Relay VC connecting to R1. The criteria for the configuration is as follows:

- Shape all traffic at a 64-kbps rate.
- Use the default setting for Tc.
- Do not use a Be.
- Enable the configuration on the subinterface.
- Do not specify a particular queuing method for the shaping queue.

In each example, the client downloads one web page, which has two frames inside the page. The web page uses two separate TCP connections to download two separate large JPG files. The PC also downloads a file using FTP get. In addition, a VoIP call is placed between extension 302 and 102. Example 5-6 shows the configuration and some sample **show** commands.

Example 5-6 *FRTS Configuration, 64 kbps, with Mostly Defaults*

```
!
! Portions of Configuration omitted for Brevity
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 no fair-queue
 clockrate 128000
 frame-relay traffic-shaping
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay class shape-all-64
 frame-relay interface-dlci 101
!
map-class frame-relay shape-all-64
 frame-relay traffic-rate 64000 64000

R3#show frame-relay pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

              Active     Inactive      Deleted       Static
  Local          2            0            0            0
  Switched       0            0            0            0
  Unused         0            0            0            0

DLCI = 101, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.1

  input pkts 135486        output pkts 129093       in bytes 8648302
  out bytes 17252015       dropped pkts 5102        in pkts dropped 0
  out pkts dropped 7876             out bytes dropped 1227684
  late-dropped out pkts 7876        late-dropped out bytes 1227684
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 1441      out bcast bytes 117294
  pvc create time 01:45:46, last time pvc status changed 01:30:01

R3#show frame-relay pvc 101

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 101, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.1

  input pkts 135152        output pkts 128787       in bytes 8626938
  out bytes 17210463       dropped pkts 5060        in pkts dropped 0
  out pkts dropped 7834             out bytes dropped 1212912
  late-dropped out pkts 7834        late-dropped out bytes 1212912
```

Example 5-6 *FRTS Configuration, 64 kbps, with Mostly Defaults (Continued)*

```
  in FECN pkts 0          in BECN pkts 0          out FECN pkts 0
  out BECN pkts 0         in DE pkts 0            out DE pkts 0
  out bcast pkts 1440     out bcast bytes 117230
  pvc create time 01:45:40, last time pvc status changed 01:29:54
  cir 64000     bc 64000     be 0         byte limit 1000   interval 125
  mincir 32000     byte increment 1000  Adaptive Shaping none
  pkts 96272     bytes 11677007  pkts delayed 58459     bytes delayed 9188174
  shaping active
  traffic shaping drops 2774
  Queueing strategy: fifo
  Output queue 3/40, 678 drop, 3777 dequeued

R3#show traffic-shape

Interface   Se0/0.1
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
101           64000     1000   64000     0         125       1000      -


R3#show traffic-shape statistics
                  Access Queue   Packets   Bytes     Packets   Bytes     Shaping
I/F               List   Depth                       Delayed   Delayed   Active
Se0/0.1                  39      97123     11816731  59238     9317918   yes

R3#show traffic-shape statistics  serial 0/0.1
                  Access Queue   Packets   Bytes     Packets   Bytes     Shaping
I/F               List   Depth                       Delayed   Delayed   Active
Se0/0.1                  6       97584     11884637  59643     9379560   yes

R3#show traffic-shape queue
Traffic queued in shaping queue on Serial0/0.1 dlci 101
  Queueing strategy: fcfs
  Queueing Stats: 23/40/959 (size/max total/drops)
Packet 1, linktype: ip, length: 64, flags: 0x10000088
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x0384, ttl: 253,
  TOS: 0 prot: 17, source port 16704, destination port 19472
    data: 0x4140 0x4C10 0x0028 0x0000 0x8012 0x16FD 0x4D91
          0x0E64 0x22FC 0x03FE 0x1E90 0xC796 0x387A 0x5193

Packet 2, linktype: ip, length: 64, flags: 0x10000088
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x0384, ttl: 253,
  TOS: 0 prot: 17, source port 16704, destination port 19472
    data: 0x4140 0x4C10 0x0028 0x0000 0x8012 0x16FE 0x4D91
          0x0F04 0x22FC 0x03FE 0x144B 0x2EE8 0xC8C2 0x1483

Packet 3, linktype: ip, length: 64, flags: 0x10000088
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x0384, ttl: 253,
  TOS: 0 prot: 17, source port 16704, destination port 19472
```

continues

Example 5-6 *FRTS Configuration, 64 kbps, with Mostly Defaults (Continued)*

```
    data: 0x4140 0x4C10 0x0028 0x0000 0x8012 0x16FF 0x4D91
          0x0FA4 0x22FC 0x03FE 0xDCEA 0xB529 0x916A 0x5254
!
! Many lines  omitted for brevity
!
```

FRTS configuration typically involves three separate steps, although only the first step is actually required.

1. FRTS must be enabled on the physical interface with the **frame-relay traffic-shape** interface subcommand. This is the only required step.
2. The shaping parameters—the shaping rate, Bc, and Be—need to be configured inside a FRTS **map-class** command.
3. The shaping parameters defined in the **map-class** should be enabled on the interface, subinterface, or DLCI using the **frame-relay class** *class-name* command.

All three steps were used in Example 5-6. The **frame-relay traffic-shape** interface subcommand, under serial 0/0, enables FRTS. Next, the **frame-relay class shape-all-64** subinterface subcommand tells the router to use the shaping parameters in the **map-class** called **shape-all-64**. Finally, the **map-class frame-relay shape-all-64** command creates a map class, with the **frame-relay traffic-rate 64000 64000** command specifying the shaping rate of 64,000 bps. From the first **64000** parameter, FRTS calculates the Bc and Tc values. The second **64000** in the command sets the excess information rate (EIR), from which the Be is calculated; to have a burst greater than zero, the excess rate must be larger than the shaping rate.

The **show frame-relay pvc** command, which follows the configuration in the example, lists statistics about each Frame Relay permanent virtual circuit (PVC). However, the **show frame-relay pvc 101** command, with a specific VC listed, gives some basic information about FRTS operation on the VC. In this case, the output shows that shaping is active, which means that FRTS was actively shaping packets when this command was issued. (As with other shapers, FRTS shaping is active when packets are in the shaping queues, or as soon as a packet exceeds the traffic contract so that it should be placed in the queues.) The command also lists that the default queuing type of FIFO is used, along with some statistics about the number of packets tail dropped from the shaping queue.

The same **show traffic-shape** commands used with GTS also provide useful information for FRTS. The **show traffic-shape** command output shown in Example 5-6 lists the basic shaping settings. Remember, the **frame-relay traffic-rate 64000 64000** command did not explicitly set Bc, Be, or Tc, but Bc and Be are shown in the **show traffic-shape** command output. The logic to derive the values works like this, with CIR representing the configured shaping rate:

- Tc defaults to 125 ms
- Bc = Tc * CIR (in this example, Bc = .125 * 64000 = 8000)
- Be = Tc * (EIR – CIR)

In this example, the shaping parameters are set as follows:

- Tc defaults to 125 ms
- Bc = .125 * 64000 = 8000
- Be = .125 * (64000 – 64000) = 0

For each Tc of 125 ms, FRTS allows 8000 bits, so the overall rate becomes 64,000 bps.

Those of you who are thoroughly reading the command output may have noticed that the **show traffic-shape** command actually lists Bc as 64,000, not the 8000 bits suggested previously. Interestingly, when using the **frame-relay traffic-rate** command, the **show traffic-shape** "Sustained Bits/interval" heading lists the bits per second. Internally, a 125-ms Tc value is really used, and a Bc of 8000 is really used—but the output of the command lists the number of bits that can be sent in a full second. The value of 1000 bytes under the heading "Increment (Bytes)" accurately lists the real Bc value used. (I did not believe it either; check out www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fwan_r/frcmds/wrffr3.htm#xtocid24, and look for the **frame-relay traffic-rate** command, for some confirming comments.)

The **show traffic-shape statistics** and the **show traffic-shape queue** commands list the same basic information for FRTS as they did for GTS. One piece of terminology not seen with GTS, however, is that the default queuing type of FRTS shaping tools is called "FCFS" in the **show** command output. FCFS stands for first-come, first-served, which is just another term for FIFO.

You can configure basic FRTS shaping parameters in two ways. The first example, Example 5-6, used the **traffic-shape rate** command. Alternatively, you can use the **frame-relay cir**, **frame-relay Bc**, and **frame-relay Be** commands to set FRTS shaping parameters inside an FRTS map class. In fact, if you want to manipulate Tc down to a smaller value, which you typically should do to support voice and video traffic, you must use these alternative commands. Remember, Tc = Bc/shaped rate, or Tc = Bc/CIR if you are shaping at the CIR. Example 5-7 lists two examples that use these additional FRTS commands, with **show** commands listing the changed Tc and Bc values. The commands are applied to R3, the same router as in Example 5-6.

Example 5-7 *FRTS Configuration by Setting CIR, Bc to Manipulate Tc*

```
R3#show running-config
!
! Portions of Configuration omitted for Brevity
!
map-class frame-relay shape-all-64-long
 frame-relay cir 64000
 frame-relay bc 8000

! Portions of the configuration omitted for brevity
!
R3#
```

continues

Example 5-7 *FRTS Configuration by Setting CIR, Bc to Manipulate Tc (Continued)*

```
R3(config)#interface serial 0/0.1
R3(config-subif)#frame class shape-all-64-long
R3(config-subif)#^Z

R3#show traffic-shape

Interface   Se0/0.1
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
101           64000     1000    8000     0         125       1000      -

R3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
!
R3(config)#map-class frame-relay shape-all-64-shortTC
R3(config-map-class)#frame-relay cir 64000
R3(config-map-class)#frame-relay bc 640
R3(config-map-class)#int s 0/0.1
R3(config-subif)#frame-relay class shape-all-64-shortTC
R3(config-subif)#^Z

R3#show traffic-shape

Interface   Se0/0.1
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
101           64000     80     640       0         10        80        -
```

Example 5-7 lists two configurations, with **show** command output listed after each configuration. In each case, the configuration commands enable you to explicitly set CIR, Bc, and Be, with Tc being calculated with the familiar Tc = Bc/CIR. In **map-class frame-relay shape-all-64-long**, CIR and Bc are set to the same values as in the first FRTS configuration shown in Example 5-6. After the configuration section at the beginning of Example 5-7, this map class is enabled on interface S0/0.1; the **show traffic-shape** command now accurately lists the Bc value of 8000, and the shaping rate (as set by the **frame-relay cir** command) of 64,000 bps.

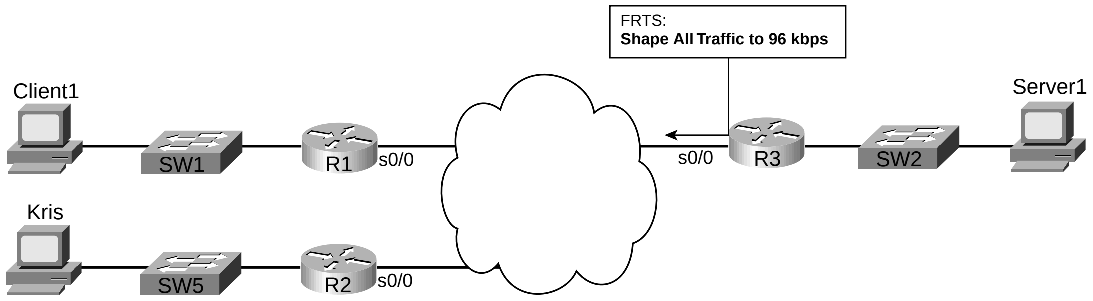
The second configuration in Example 5-7 uses the **map-class frame-relay shape-all-64-shortTC** command to set Bc to 1/100 of the CIR value, which yields a Tc = 640/64,000, or 10 ms. This map class shows how you would set values to lower Tc, which is particularly useful to reduce the delay waiting for the next Tc if you have multiservice traffic. The example shows the configuration being changed to use **map-class shape-all-64-shortTC** by adding the **frame-relay class shape-all-64-shortTC** command. The **show traffic-shape** command follows, listing a Tc value of 10 ms.

In Example 5-8, the lab network has a new remote site added, with a PC named Kris, and a router (R2) with a 64-kbps CIR VC to R3. Suppose that the Frame Relay provider actually polices each VC at 96 kbps. The criteria for the configuration is summarized as follows:

- Shape all traffic from R3 to R1, and from R3 to R2, at 96 kbps.
- Allow burst rates of 112 kbps on each VC.
- Use default settings for Bc and Be.

In this case, traffic to site R1 consists of a single VoIP call, and one web connection with two frames inside the page. At site R2, PC Kris FTP transfers a large file from the FTP server near R3. Figure 5-20 shows the network, and Example 5-8 shows the configuration and some sample **show** command output.

Figure 5-20 *Example Network with VCs from R3 to Two Remote Sites*



Example 5-8 *R3 FRTS Configuration on Two Different VCs, with Identical Settings*

```
!
! Many lines omitted for brevity
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 no fair-queue
 clockrate 128000
 frame-relay class shape-all-96
 frame-relay traffic-shaping
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
```

continues

Example 5-8 *R3 FRTS Configuration on Two Different VCs, with Identical Settings (Continued)*

```
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
interface Serial0/0.2 point-to-point
 description point-to-point subint connected to DLCI 102 (R2)
 ip address 192.168.23.253 255.255.255.0
 frame-relay interface-dlci 102
!
map-class frame-relay shape-all-96
 frame-relay traffic-rate 96000 112000
 no frame-relay adaptive-shaping
!
! Lines omitted for brevity
!

R3#show frame pvc 101

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 101, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.1

  input pkts 251070        output pkts 239522       in bytes 16004601
  out bytes 34597106       dropped pkts 15567       in pkts dropped 0
  out pkts dropped 15567            out bytes dropped 3005588
  late-dropped out pkts 15567       late-dropped out bytes 3005588
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 1951      out bcast bytes 158000
  pvc create time 02:22:11, last time pvc status changed 02:06:26
  cir 96000     bc 96000     be 16000     byte limit 3500   interval 125
  mincir 48000     byte increment 1500  Adaptive Shaping none
  pkts 33292     bytes 4753889   pkts delayed 19315     bytes delayed 3460327
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: fifo
  Output queue 0/40, 747 drop, 18449 dequeued

R3#show frame pvc 102

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 102, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.2

  input pkts 4063          output pkts 3820         in bytes 250312
  out bytes 3863342        dropped pkts 1206        in pkts dropped 1206
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 587       out bcast bytes 50380
  pvc create time 02:22:13, last time pvc status changed 02:05:29
  cir 96000     bc 96000     be 16000     byte limit 3500   interval 125
```

Example 5-8 *R3 FRTS Configuration on Two Different VCs, with Identical Settings (Continued)*

```
  mincir 48000     byte increment 1500  Adaptive Shaping none
  pkts 2916      bytes 3802088   pkts delayed 1120      bytes delayed 1468724
  shaping active
  traffic shaping drops 0
  Queueing strategy: fifo
  Output queue 0/40, 0 drop, 1120 dequeued

R3#show traffic-shape

Interface   Se0/0.1
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
101           96000     3500   96000     16000     125       1500      -

Interface   Se0/0.2
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
102           96000     3500   96000     16000     125       1500      -

R3#show traffic-shape statistics
                  Access Queue   Packets   Bytes     Packets   Bytes     Shaping
I/F               List   Depth                       Delayed   Delayed   Active
Se0/0.1                  18      34730     4920740   19737     3510580   yes
Se0/0.2                  0       3103      4065016   1191      1563968   yes
```

The FRTS configuration in this example sets FRTS parameters in a map class, which is then enabled on a physical interface. FRTS always performs shaping on each VC separately; therefore, in this case, the shaping parameters per VC will be picked up from the map class that has been enabled on the physical interface. Notice that a new map class, **map-class shape-all-96**, is configured with a **frame-relay traffic-rate 96000 112000** command to set the CIR and EIR values. The **frame-relay class shape-all-96** command has been added to the physical interface, and not to the individual subinterfaces. FRTS includes a feature that I call the *inheritance* feature, which just means that if a subinterface does not have a **frame-relay class** command, it uses the frame-relay class configured on the physical interface. Similarly, on multipoint subinterfaces, if a map class has not been configured on a particular DLCI, it inherits the FRTS parameters from the map class configured on the subinterface.

The first two **show** commands after the configuration (**show frame pvc 101** and **show frame pvc 102**) list both shaping parameters and operational statistics. The parameters are the same, because each subinterface picked up its parameters from the map class (shape-all-96) that was enabled on the physical interface. However, the operational statistics differ, because FRTS shapes each VC separately. The **show traffic-shape** commands that follow confirm the same settings are used on each of the two subinterfaces as well. And in case you still think that FRTS may be shaping all the subinterfaces together, the **show traffic-shape statistics** command lists the varying statistics for shaping on each VC at the end of the example.

FRTS uses a default setting of CIR = 56kbps, and Tc = 125 ms, if the **frame-relay class** command does not appear on the interface. In other words, if you enable FRTS on the physical interface with the **frame-relay traffic-shape** command, but do not enable a map class, FRTS still shapes each VC individually—but it does so with default parameters. So, be careful—pick a good set of default settings, put them in a map class, and enable it on the physical interface as in Example 5-8, just to be safe.

In Example 5-8, the G.729 voice call between R1 and R3 suffered, mainly due to the fact that shaping increases delay, and no effort was made to service the voice traffic more quickly. Suppose that the network engineer notices that IOS supports LLQ as an option for queuing in the shaping queues for FRTS. Therefore, he wants to solve the problem of poor voice quality by putting the voice call into a low-latency queue. With a shaping rate of 96 kbps, and with a single G.729 call, the voice call quality should improve. The criteria for the configuration is summarized as follows:

- Shape all traffic from R3 to R1, and from R3 to R2, at 96 kbps, respectively.
- Use LLQ for queuing on the VC to R1, with 30 kbps maximum in the low-latency queue.
- Configure Bc so that Tc = 10 ms.
- Use Be = 0.

In this case, traffic to site R1 consists of a single VoIP call, and one web connection with two frames inside the page. At site R2, PC Kris FTP transfers a large file from the FTP server near R3. Example 5-9 shows the configuration and some sample **show** commands.

Example 5-9 *FRTS to Two Sites, with LLQ Used in the Shaping Queue to Site 1*

```
R3#show running-config
!
! Many lines omitted for brevity
!
class-map match-all voip-rtp
  match ip rtp 16384 16383
!
policy-map voip-and-allelse
  class voip-rtp
    priority 30
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 no fair-queue
 clockrate 128000
 frame-relay class shape-all-96
 frame-relay traffic-shaping
```

Example 5-9 *FRTS to Two Sites, with LLQ Used in the Shaping Queue to Site 1 (Continued)*

```
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay class shape-with-LLQ
 frame-relay interface-dlci 101
!
interface Serial0/0.2 point-to-point
 description point-to-point subint connected to DLCI 102 (R2)
 ip address 192.168.23.253 255.255.255.0
 frame-relay interface-dlci 102
!
! Note – No frame-relay class command on previous VC!
!
!
map-class frame-relay shape-all-96

 frame-relay cir 96000
 frame-relay bc 960
 frame-relay be 0
!
map-class frame-relay shape-with-LLQ
 frame-relay cir 96000
 frame-relay bc 960
 frame-relay be 0
 service-policy output voip-and-allelse
!
R3#show frame-relay pvc 101
PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 101, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.1

  input pkts 18487         output pkts 17749        in bytes 1184282
  out bytes 2639555        dropped pkts 863         in pkts dropped 0
  out pkts dropped 863              out bytes dropped 115649
  late-dropped out pkts 863         late-dropped out bytes 115649
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 364       out bcast bytes 30272
  pvc create time 00:26:08, last time pvc status changed 00:24:49
  cir 96000     bc 96000     be 16000     byte limit 3500   interval 125
  mincir 48000     byte increment 1500  Adaptive Shaping none
  pkts 17718     bytes 2621259   pkts delayed 15671     bytes delayed 2238337
  shaping active
  traffic shaping drops 0
  service policy voip-and-allelse
 Serial0/0.1: DLCI 101 -

  Service-policy output: voip-and-allelse
```

continues

Example 5-9 *FRTS to Two Sites, with LLQ Used in the Shaping Queue to Site 1 (Continued)*

```
    Class-map: voip-rtp (match-all)
      5101 packets, 326464 bytes
      30 second offered rate 25000 bps, drop rate 0 bps
      Match: ip rtp 16384 16383
      Weighted Fair Queueing
        Strict Priority
        Output Queue: Conversation 24
        Bandwidth 30 (kbps) Burst 750 (Bytes)
        (pkts matched/bytes matched) 4468/285952
        (total drops/bytes drops) 0/0

    Class-map: class-default (match-any)
      386 packets, 412201 bytes
      30 second offered rate 31000 bps, drop rate 0 bps
      Match: any
  Output queue size 42/max total 600/drops 0

R3#show traffic-shape queue serial 0/0.1
Traffic queued in shaping queue on Serial0/0.1 dlci 101
  Queueing strategy: weighted fair
  Queueing Stats: 15/600/64/0 (size/max total/threshold/drops)
     Conversations  4/8/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 18 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 5/0/0/0/0
  Conversation 24, linktype: ip, length: 64
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x012F, ttl: 253,
  TOS: 0 prot: 17, source port 19018, destination port 17766

  (depth/weight/total drops/no-buffer drops/interleaves) 3/32384/0/0/0
  Conversation 2, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0x177B, ttl: 127,
  TOS: 0 prot: 6, source port 80, destination port 1148

  (depth/weight/total drops/no-buffer drops/interleaves) 2/32384/0/0/0
  Conversation 1, linktype: ip, length: 1404
  source: 192.168.3.100, destination: 192.168.1.100, id: 0x1775, ttl: 127,
  TOS: 0 prot: 6, source port 80, destination port 1147

  (depth/weight/total drops/no-buffer drops/interleaves) 5/32384/0/0/0
  Conversation 12, linktype: ip, length: 1244
  source: 192.168.3.100, destination: 192.168.1.100, id: 0x1758, ttl: 127,
  TOS: 0 prot: 6, source port 1176, destination port 1146

R3#sh traffic-shape  queue s 0/0.2
Traffic queued in shaping queue on Serial0/0.2 dlci 102
  Queueing strategy: fcfs
```

Example 5-9 begins with the new FRTS and LLQ configuration. The **policy-map voip-and-allelse** command defines a policy map that puts all even RTP ports into a low-latency queue, with 32 kbps maximum, and all other traffic into the class-default class. LLQ is enabled inside a new FRTS **map-class shape-with-LLQ** command, with the **service-policy output voip-and-allelse** command enabling LLQ inside the map class. Any VC that uses the shape-with-LLQ map class will use the settings in that map class, including the LLQ configuration. In this case, the single VC on subinterface s 0/0.1 uses LLQ for the shaping queue because of the **frame-relay class shape-with-LLQ** command.

The VC on subinterface S0/0.2 does not use **map-class voip-and-allelse**. Because no **frame-relay class** command is configured on subinterface 0/0.2, FRTS uses the shaping parameters from **map-class shape-all-64**, because it is configured on physical interface s 0/0.

Immediately following the configuration, the **show frame-relay pvc 101** command lists a large amount of new information. Essentially, IOS lists the same kinds of information normally seen with the **show policy-map interface** command in the **show frame-relay pvc 101** command. Information about the MQC classes defined, and statistics about the packets in each class, is listed. Also note that the **show traffic-shape queue s 0/0.2** at the end of the example reminds us that FCFS is used on the other subinterface.

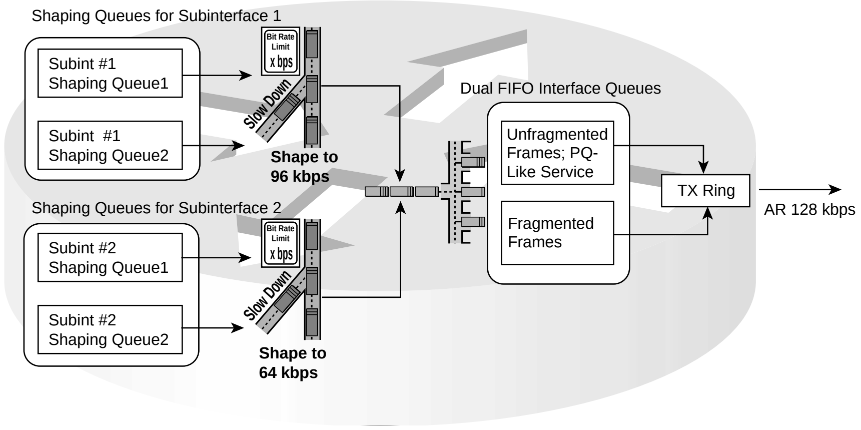
If you take a step back from the configuration and **show** commands for a moment, it may be obvious that the two VCs, shaped at 96 kbps each, oversubscribe R3's access link, which is clocked at 128 kbps. Because FRTS only supports FIFO Queuing on the physical interface, congestion still occurs there. Although adding LLQ to subinterface S0/0.1 helped the quality of the voice call, call quality still suffered due to drops and jitter caused by the oversubscribed FIFO output queue on the physical interface s 0/0.

The final solution to the voice quality problem in this case is to take advantage of the queuing feature introduced by FRF.12 fragmentation. Frame Relay fragmentation (FRF) can be used with FRTS. FRF actually creates a set of two queues on the physical interface, called dual-FIFO queues. Packets that are larger than the fragmentation size are fragmented and placed into one queue. Packets that are equal to, or smaller than, the fragmentation size are not fragmented, and placed into the other queue. IOS treats the nonfragmented frame queue as a priority queue—in other words, it is always serviced first. Therefore, if you want to give great service to small packets such as VoIP, FRF can provide the traditional benefits of fragmentation, and provide a priority queue for all small packets, including VoIP. Figure 5-21 outlines the basic idea, with FRTS on two subinterfaces.

Using FRF to create a two-queue PQ system works well with voice traffic, because the packets are generally small. However, video traffic includes too many larger packets to benefit substantially from FRF's queuing feature, because the larger packets are fragmented and placed in the lower-priority queue. Chapter 7 shows the configuration for adding FRF to this network.

Other FRTS configuration items that might be on the exam include how to configure adaptive shaping, and how to enable FRTS parameters on a VC on a multipoint subinterface. Example 5-10 lists a simple map class configuration that enables adaptive shaping, with the configuration added to DLCI 101, but not DLCI 102, on multipoint subinterface S0/0.33.

Figure 5-21 *Interaction Between Shaping Queues, and Frame Relay Fragmentation Queues*



Example 5-10 *FRTS Adaptive Shaping and per-DLCI Configuration*

```
R3#show running-config
!
! Many lines omitted for brevity
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 no fair-queue
 clockrate 128000
 frame-relay class shape-all-96
 frame-relay traffic-shaping
!
interface Serial0/0.33 multipoint
 description multipoint subint, w/ DLCIs 101 and 102
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
  class my-adapt-shape-class
 frame-relay interface-dlci 102
!
map-class frame-relay shape-all-96
 frame-relay traffic-rate 96000 112000
!
map-class frame-relay my-adapt-shape-class
 frame-relay traffic-rate 96000 112000
 frame-relay adaptive-shaping becn
 frame-relay mincir 64000
```

The **map-class frame-relay my-adapt-shape-class** command creates a new map class with adaptive FRTS enabled. With adaptive shaping, FRTS uses the shape rate as the maximum rate, and the rate configured on the **frame-relay mincir** command as the minimum rate. The new **map-class my-adapt-shape-class** command enables adaptive shaping with the **frame-relay adaptive-shaping becn** command, and sets the mincir with the **frame-relay mincir 64000** command. The map class is enabled on DLCI 101, but not DLCI 102, on subinterface s0/0.33.

The example also shows how to enable a map class for a single DLCI. To enable an FRTS map class per DLCI, first enter subinterface configuration mode, and then issue the **frame-relay interface-dlci** command. This command places you into DLCI configuration mode, where the **class my-adapt-shape-class** command enables map class my-adapt-shape-class just for this DLCI. Note that after the **frame-relay interface-dlci 102** command, there is no **class** command. So, on DLCI 102, the FRTS parameters in class shape-all-96, as configured on the physical interface, are used.

Table 5-16 summarizes the key points for comparison between the various traffic-shaping tools, highlighting FRTS.

Table 5-16 *Comparison of Traffic Shaping Tools: GTS, CB Shaping, DTS, and FRTS*

Feature	GTS	CB Shaping	DTS	FRTS
Supports ATM, FR, HDLC, PPP, LAN interfaces	Yes	Yes	Yes	**No**
Can be enabled on interfaces and subinterfaces	Yes	Yes	Yes	**Yes**
Can be enabled per Frame Relay DLCI to support per-VC shaping on multipoint interfaces	No	No	No	**Yes**
Supports adaptive shaping	Yes	Yes	Yes	**Yes**
Supports concurrent FRF.12 Frame Relay fragmentation	No	No	No	**Yes**
Queuing methods in shaping queue	WFQ	FIFO, WFQ, CBWFQ, LLQ	FIFO, WFQ, CBWFQ, LLQ	**FIFO**, **WFQ**, **CBWFQ**, **LLQ**, **PQ**, **CQ**
Concurrent queuing methods on Physical interface	All	All	All	**FIFO**, **FRF***
Can be configured using MQC commands	No	Yes	Yes	**No**
Can classify traffic to shape a subset of the traffic on an interface/VC	Yes	Yes	Yes	**No**

continues

Table 5-16 *Comparison of Traffic Shaping Tools: GTS, CB Shaping, DTS, and FRTS (Continued)*

Feature	GTS	CB Shaping	DTS	FRTS
Default Tc	Variable	125 ms	125 ms	**125 ms**
Distributes shaping processing to VIPs in 7500 series routers	No	No	Yes	**No**

* The Cisco QoS course claims WFQ is supported on the physical interface. In addition, FRF is not technically a queuing tool, although its feature of using two queues does achieve the same effect.

Traffic-Policing Tools

IOS supports two traffic-policing tools—committed access rate (CAR) and class-based policing (CB policing). Both tools perform the same basic function, but each has its differences that dictate when you can use them. CAR has two categories for packets: conform and exceed. CB policing has three categories: conform, exceed, and violate. CAR allows cascading or nesting of policing statements, which enables you to police all traffic on an interface at one high rate, and police subsets of the traffic at lower rates. CB policer uses MQC commands, and CAR does not.

Both tools support classification of packets, which allows each tool to police subsets of the traffic on an interface or subinterface. If you memorized all the classification options for CAR and CB marking from Chapter 3, you already know about the available classification options for CAR and CB policing, because they use the same options. Of course, no one needs to memorize such things except to cram for an exam, so Table 5-17 lists the matchable fields for classification with both tools.

Table 5-17 *Classification Fields Used by CAR and CB Policing*

Field	Tool	Comments
Anything matched with an IP ACL	CAR, CB policing	N/A
Source MAC Address	CAR, CB policing	CAR uses a special access-rate ACL; CB marking uses the match command.
IP Precedence	CAR, CB policing	CAR uses a special access-rate ACL specific to CAR; CB marking uses the **match** command; Both can match a subset of values.
MPLS Experimental	CAR, CB policing	CAR uses a special access-rate ACL specific to CAR; CB marking uses the **match** command. Both can match a subset of values.
IP DSCP	CAR, CB policing	Can check for multiple values using multiple **match** commands.
QoS Group	CAR, CB policing	The QoS Group field is used to tag packets internal to a single router.

Table 5-17 *Classification Fields Used by CAR and CB Policing (Continued)*

Field	Tool	Comments
Class of Service (CoS)	CB policing	Checks incoming ISL/802.1P CoS bits. Can match multiple values.
Destination MAC Address	CB policing	N/A
Input interface	CB policing	N/A
RTP's UDP port-number range	CB policing	RTP uses even numbered UDP ports from 16384–32767 for voice payload. This matching option allows matching a subset of the port numbers, and it matches only the even-numbered ports.
NBAR protocol types	CB policing	Refer to the coverage in Chapter 3 for more details.
NBAR Citrix applications	CB policing	NBAR can recognize different types of Citrix applications; CB marking can use NBAR to classify based on these application types.
Host Name and URL string	CB policing	NBAR can match URL strings using regular expressions, including the host name. CB marking can use NBAR to match these strings for classification.

Each policer defines an action to take when a packet conforms, exceeds, or violates a traffic contract. Both support either dropping or transmitting the packet, of course. Both tools allow re-marking of QoS fields. In particular, note that CB policing allows for marking of the ATM CLP bit, which CAR does not. Table 5-18 lists the various actions supported by each policer.

Table 5-18 *Policing Actions Used by CAR and CB Policing*

Action Keyword	Meaning	CAR?	CB Policer?
drop	Discards the packet	Yes	Yes
transmit	Forwards the packet	Yes	Yes
set-prec-transmit	Forwards the packet after marking the IP precedence value.	Yes	Yes
set-qos-transmit	Forwards the packet after marking the QoS group	Yes	Yes
set-dscp-transmit	Forwards the packet after marking the IP DSCP value	Yes	Yes
set-mpls-exp-transmit	Forwards the packet after marking the MPLS Experimental bits	Yes	Yes

continues

Table 5-18 *Policing Actions Used by CAR and CB Policing (Continued)*

Action Keyword	Meaning	CAR?	CB Policer?
set-frde-transmit	Forwards the packet after marking the Frame Relay discard eligibility (DE) bit	No	Yes
set-clp-transmit	Forwards the packet after marking the ATM cell loss priority (CLP) bit	No	Yes
set-prec-continue	Marks the IP precedence value, and continues to the next nested (cascaded) CAR command	Yes	No
set-dscp-continue	Marks the QoS group, and continues to the next nested (cascaded) CAR command	Yes	No
set-mpls-exp-continue	Marks the IP precedence value, and continues to the next nested (cascaded) CAR command	Yes	No
set-qos-continue	Marks the QoS group, and continues to the next nested (cascaded) CAR command	Yes	No
continue	Just continues to the next nested (cascaded) CAR command	Yes	No

The following sections show several examples of CB policing and CAR, respectively, along with some **show** commands.

Class-Based Policing Configuration

CB policing performs policing using three separate actions for packets that conform, exceed, or violate the traffic contract. (The exact meanings of each of these three categories were covered in the "How Policing Works" section earlier in this chapter.) Generally speaking, CB policing considers packets that happen to arrive when enough Bc tokens are available as "conforming" packets. Packets that arrive when Bc is consumed, but Be is not, are considered "exceeding"; and packets that arrive after Bc and Be have been consumed are considered "violating" packets.

For each category (conform, exceed, violate), CB policing can use a variety of actions. Table 5-18 lists the action keywords used in the **police** command. In general, the choices are to drop the packet, transmit the packet, or to first re-mark some QoS field, and then transmit the packet.

CB policing uses MQC commands for configuration. Because it is class based, CB policing can police subsets of the traffic on the interface or subinterface on which it is enabled. CB policing uses the same familiar MQC classification commands that all the other MQC-based tools use; again, you only need to learn one more MQC command to know how to configure another MQC QoS feature.

The **police** command configures CB policing inside a policy map. On the **police** command, you define the policing rate in bps, the **burst-normal** in bytes, and the **burst-max** in bytes. Note that although Bc is represented by **burst-normal** in the command, the configured **burst-max** value is actually Bc + Be. To have no Be, you configure the **burst-normal** and **burst-max** values to the same number. To have a Be of *x*, configure the **burst-max** parameter to a value of *x* more than **burst-normal**. If neither **burst-normal** nor **burst-max** is configured, both values default to 1.5 second's worth of traffic at the police rate; because the two settings default to the same number, no excess burst capability exists with default settings. (These settings become more obvious after looking at **show** command output.)

Two examples of configuration for CB policing follow Tables 5-19 and 5-21. Table 5-19 lists the CB policing configuration commands. Table 5-20 lists the actions that you can configure on the **police** command. Table 5-21 lists the CB policing **show** commands.

Table 5-19 *Command Reference for Class-Based Policing*

Command	Mode and Function
police *bps burst-normal burst-max* **conform-action** *action* **exceed-action** *action* [**violate-action** *action*]	**policy-map** class subcommand; enables policing for the class, setting the police rate, Bc, and Bc + Be values, and actions taken. Actions are **drop**, **set-clp-transmit**, **set-dscp-transmit**, **set-prec-transmit**, **set-qos-transmit**, **transmit**.
service-policy {**input** | **output**} *policy-map-name*	Interface or subinterface configuration mode; enables CB shaping on the interface.
class-map *class-map-name*	Global config; names a class map, where classification options are configured.
Match …	**class-map** subcommand; defines specific classification parameters.
match access-group {*access-group* | **name** *access-group-name*}	Access-control list (ACL).
match source-address mac *address-destination*	Source MAC address.
match ip precedence *ip-precedence-value* [*ip-precedence-value ip-precedence-value ip-precedence-value*]	IP precedence.
match mpls experimental *number*	MPLS Experimental.
match cos *cos-value* [*cos-value cos-value cos-value*]	CoS.
match destination-address mac *address*	Destination MAC address.
match input-interface *interface-name*	Input interface.

continues

Table 5-19 *Command Reference for Class-Based Policing (Continued)*

Command	Mode and Function
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	IP DSCP.
match ip rtp *starting-port-number port-range*	RTP's UDP port-number range.
match qos-group *qos-group-value*	QoS group.
match protocol *protocol-name*	NBAR protocol types.
match protocol citrix [**app** *application-name-string*].	NBAR Citrix applications.
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	Host name and URL string.
match any	All packets.
policy-map *policy-map-name*	Global config; names a policy, which is a set of actions to perform.
class *name*	**policy-map** subcommand; identifies which packets on which to perform some action by referring to the classification logic in a class map.

Table 5-20 *Options for Actions Taken with the* **police** *Command*

Command	Mode and Function
drop	Drops the packet
set-dscp-transmit	Sets the DSCP and transmits the packet
set-prec-transmit	Sets the IP precedence (0 to 7) and sends the packet
set-qos-transmit	Sets the QoS group ID (1 to 99) and sends the packet
set-clp-transmit	Sets the ATM CLP bit (ATM interfaces only) and sends the packet
transmit	Sends the packet

Table 5-21 *Exec Command Reference for Class-Based Policing*

Command	Function
show policy-map *policy-map-name*	Lists configuration information about all MQC-based QoS tools
show policy-map *interface-spec* [*input* | *output*] [**class** *class-name*]	Lists statistical information about the behavior of all MQC-based QoS tools

Before diving in to the first configuration example, you may find it helpful to briefly review the basics of policing. CB policing categories traffic into three groups when policing—one when packets conform, one when packets exceed, and one when packets violate the traffic contract. The policer considers packets that happen to arrive when enough tokens exist in the Bc token bucket to conform. Packets that arrive and require tokens from the Be bucket are considered to exceed, and packets for which there are not enough tokens in either Bc or Be are considered to violate the contract. For each grouping, you can configure a separate action.

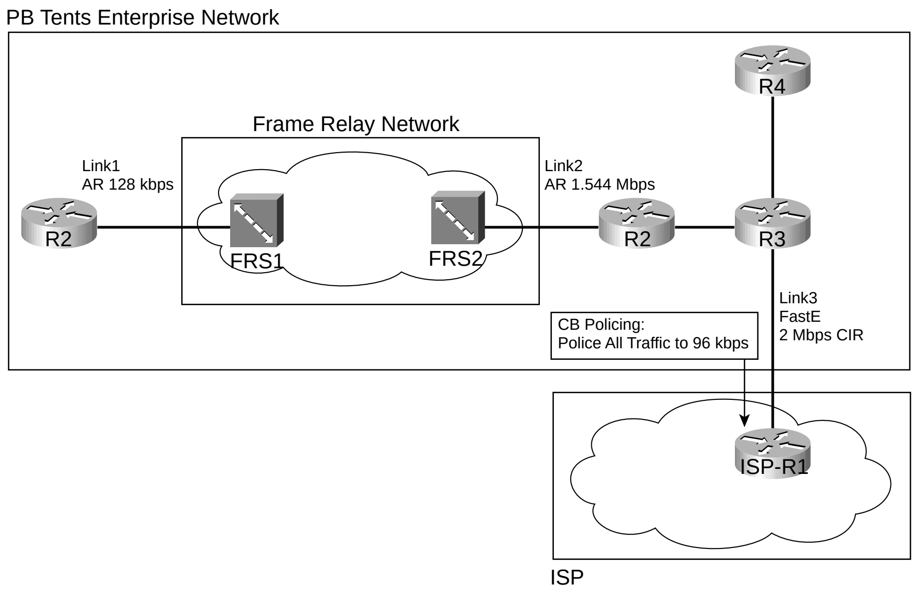
The actions to take on each packet, whether it conforms, exceeds, or violates the contract, boil down to either dropping the packet, transmitting the packet, or re-marking and transmitting the packet. The **drop** and **transmit** options are pretty obvious. However, CB policing includes keywords such as **set-prec-transmit** and **set-dscp-transmit**, which allow the policer to transmit the packet, but first mark the IP Precedence or DSCP field with a lower value. You may recall from the "How Policing Works" section that marking down a packet can be useful because the marked-down packet can have a higher likelihood for discard later, but if no congestion occurs, the packet can be delivered.

You can use CB policing to police all traffic entering or exiting an interface. In the first example, router ISP-edge polices ingress traffic from an enterprise network. The criteria for the first CB policing example is as follows:

- All traffic policed at 96 kbps at ingress to the ISP-edge router.
- Bc of 1 second's worth of traffic is allowed.
- Be of 0.5 second's worth of traffic is allowed.
- Traffic that violates the contract is discarded.
- Traffic that exceeds the contract is marked down to DSCP Be.
- Traffic that conforms to the contract is forwarded with no re-marking.

Figure 5-22 shows the network in which the configuration is applied, and Example 5-11 shows the configuration.

Figure 5-22 *Example Network for Policing Examples*



Example 5-11 *CB Policing at 96 kbps at ISP-edge Router*

```
ISP-edge#show running-config
Building configuration...
!
!Lines omitted for brevity
!
ip cef
!
policy-map police-all
  class class-default
! note: the police command wraps around to a second line.
     police cir 96000 bc 12000 be 18000 conform-action transmit exceed-action set-dscp-
transmit 0 violate-action drop
!
interface Serial1/0
 description connected to FRS port S1. Single PVC to R3.
 no ip address
 encapsulation frame-relay
 load-interval 30
 service-policy input police-all
 no fair-queue
 clockrate 1300000
!
```

Example 5-11 *CB Policing at 96 kbps at ISP-edge Router (Continued)*

```
interface Serial1/0.1 point-to-point
 description point-point subint global DLCI 101, connected via PVC to Ent-edge
 ip address 192.168.2.251 255.255.255.0
 frame-relay interface-dlci 103
!
! Lines omitted for brevity
!

ISP-edge#show policy-map
  Policy Map police-all
    Class class-default
! note: the next output lines describes the police command, which
! wraps around to a second line.
      police cir 96000 conform-burst 12000 excess-burst 18000 conform-action transmit
exceed-action set-dscp-transmit 0 violate-action drop

ISP-edge#show  policy-map  interface s 1/0

 Serial1/0

  Service-policy input: police-all

    Class-map: class-default (match-any)
      8375 packets, 1446373 bytes
      30 second offered rate 113000 bps, drop rate 15000 bps
      Match: any
      police:
        cir 96000 bps, conform-burst 12000, excess-burst 18000
        conformed 8077 packets, 1224913 bytes; action: transmit
        exceeded 29 packets, 17948 bytes; action: set-dscp-transmit 0
        violated 269 packets, 203512 bytes; action: drop
        conformed 95000 bps, exceed 0 bps violate 20000 bps
```

The example takes advantage of the fact that in any policy map, all traffic that does not match a class gets placed into the class-default class. Because one design goal was to police all traffic, no explicit class maps were needed—all traffic matches the class-default class inside every policy map by default. Therefore, inside new policy map police-all, the **police cir 96000 conform-burst 12000 excess-burst 18000 conform-action transmit exceed-action set-dscp-transmit 0 violate-action drop** command enables policing for the class-default class.

The parameters of the **police** command set the policing rate to 96,000 bps, with 12,000 *bytes* of burst capability. The shaping tools configure Bc and Be as a number of bits; IOS policers configure these values as a number of bytes. The requirements for this example stated 1 second's worth of Bc, and 12,000 bytes can be sent in 1 second with a CIR of 96,000 bps. The **excess-burst** configuration parameter actually defines the Bc and Be value combined, in bytes. So, the value is configured at 18,000, which is 6000 bytes more than Bc. The stated goals asked for a Be of 0.5 seconds of traffic, and it does indeed take 0.5 seconds to send 6000 bytes at 96 kbps.

NOTE All IOS shapers use bits as the unit when setting Bc and Be; both policers use bytes as the unit.

In Example 5-11, the **police** command transmits packets that *conform*, marks down packets that *exceed* to a DSCP value of zero, and drops packets that *violate* the values. The **show policy-map** command repeats the same details as shown in the configuration command, as highlighted in the example. The **show policy-map interface s1/0** command lists statistics about the number of packets that conformed, exceeded, and violated the contract.

One of the advantages of CB policing is the ability to perform policing per class. The next example, Example 5-12, shows CB policing with web traffic classified and policed differently than the rest of the traffic. The criteria for the first CB policing example is as follows:

- Police web traffic at 80 kbps at ingress to the ISP-edge router. Transmit conforming and exceeding traffic, but discard violating traffic.
- Police all other traffic at 16 kbps at ingress to the ISP-edge router. Mark down exceeding and violating traffic to DSCP 0.
- Bc of 1 second's worth of traffic is allowed.
- Be of 0.5 second's worth of traffic is allowed.

Example 5-12 shows the configuration.

Example 5-12 *CB Policing 80 kbps for Web Traffic, 16 kbps for the Rest with Markdown to Be, at ISP-edge Router*

```
ISP-edge#show running-config
Building configuration...
!
!Lines omitted for brevity
!
ip cef
!
class-map match-all match-web
  match protocol http
!
policy-map police-web
  class match-web
     police cir 80000 bc 10000 be 15000 conform-action transmit exceed-action transmit
violate-action drop
  class class-default
     police cir 16000 bc 2000 be 3000 conform-action transmit exceed-action
transmit violate-action set-dscp-transmit 0
!
interface Serial1/0
 description connected to FRS port S1. Single PVC to R3.
 no ip address
 ip nbar protocol-discovery
```

Example 5-12 *CB Policing 80 kbps for Web Traffic, 16 kbps for the Rest with Markdown to Be, at ISP-edge Router (Continued)*

```
 encapsulation frame-relay
 load-interval 30
 service-policy input police-web
 no fair-queue
 clockrate 1300000
!
interface Serial1/0.1 point-to-point
 description point-point subint global DLCI 101, connected via PVC to DLCI 103 (R3)
 ip address 192.168.2.251 255.255.255.0
 frame-relay interface-dlci 103
!
!
!Lines omitted for brevity
!
ISP-edge#show policy-map
  Policy Map police-web
    Class match-web
! note: the police command wraps around to a second line.
      police cir 80000 conform-burst 10000 excess-burst 15000 conform-action
transmit exceed-action transmit violate-action drop
    Class class-default
! note: the police command wraps around to a second line.
      police cir 16000 conform-burst 2000 excess-burst 3000 conform-action
transmit exceed-action transmit violate-action set-dscp-transmit 0

ISP-edge#show policy-map interface s 1/0

 Serial1/0

  Service-policy input: police-web

    Class-map: match-web (match-all)
      736 packets, 900505 bytes
      30 second offered rate 90000 bps, drop rate 14000 bps
      Match: protocol http
      police:
        cir 80000 bps, conform-burst 10000, excess-burst 15000
        conformed 625 packets, 748645 bytes; action: transmit
        exceeded 13 packets, 14268 bytes; action: transmit
        violated 98 packets, 137592 bytes; action: drop
        conformed 75000 bps, exceed 0 bps violate 17000 bps

    Class-map: class-default (match-any)
      3751 packets, 241636 bytes
      30 second offered rate 26000 bps, drop rate 0 bps
      Match: any
      police:
        cir 16000 bps, conform-burst 2000, excess-burst 3000
```

continues

Example 5-12 *CB Policing 80 kbps for Web Traffic, 16 kbps for the Rest with Markdown to Be, at ISP-edge Router (Continued)*

```
          conformed 2330 packets, 149928 bytes; action: transmit
          exceeded 46 packets, 2944 bytes; action: transmit
          violated 1376 packets, 88808 bytes; action: set-dscp-transmit 0
          conformed 16000 bps, exceed 0 bps violate 9000 bps
```

If you are becoming comfortable with MQC configurations now, this configuration should be relatively easy to decipher. The **class-map match-all match-web** command creates a new class, which matches all web traffic using NBAR. The **policy-map police-web** command creates a new policy map, which uses **class match-web** to classify web traffic, and **class class-default** to classify all other traffic. Inside each class, a **police** command is used, setting the parameters as outlined in the stated goals. For instance, the **police cir 80000 bc 10000 be 15000 conform-action transmit exceed-action transmit violate-action drop** command sets the rate at 80 kbps, with a 1-second Bc value of 10,000 bytes, and a configured Be of 15,000 bytes, which means that the actual Be, in terms of the additional amount beyond Bc, is 5000 bytes.

The **show policy-map interface s1/0** command lists statistics as always, in this case showing the two classes in the policy map police-web. As you would expect, separate policing statistics are shown per class, because CB policing is enabled per class.

Table 5-22 summarizes the features of CB policing.

Table 5-22 *CB Policing Features*

Feature	CB Policing
Allows conform and exceed action categories	Yes
Allows violate action category	Yes
Polices either all traffic, or a subset through classification	Yes
Uses MQC for configuration	Yes
Allows nested or cascaded policing logic	No
Can be enabled per subinterface	Yes
Can be enabled per DLCI on multipoint subinterfaces	No

Committed Access Rate (CAR) Configuration

CAR has more similarities than differences when compared to CB policing. Both perform policing on all traffic on either an interface or subinterface. Both can classify traffic to police a subset of traffic as well. Both use the same units when configuring policing parameters—bits per second for the policing rate, bytes for the normal and Be values, with the configured Be value actually representing Bc + Be.

CAR differs from CB policing regarding four main features. The most obvious is that CAR uses the **rate-limit** command, which is not part of the MQC set of commands. CAR also uses only two categories for actions—conform and exceed—as opposed to the three categories (conform, exceed, and violate) supported by CB policing. The most significant difference is that CAR has a feature called cascaded or nested **rate-limit** commands. Finally, CAR does not use MQC for configuration. Each of these differing features are covered in the example configurations.

Most QoS tools that classify packets operate with logic similar to ACLs in that, when a packet is matched, the action(s) related to that matched statement are taken. With all MQC features, such as CB marking, CBWFQ, CB policing, and CB shaping, after a particular class has been matched, the action associated with that class inside the policy map is performed. For instance, all MQC policy maps end with the class-default class, which matches all packets; however, packets may have matched an earlier class, so that a packet would never fall through to the class-default class.

With CAR, a single packet can match multiple statements. By doing so, you can actually police progressively more specific subsets of the traffic on the interface or subinterface. For example, you can create logic such as the following:

- Police all traffic on the interface at 500 kbps; but before sending this traffic on its way . . .
  - — Police all web traffic at 400 kbps.
  - — Police all FTP traffic at 150 kbps
  - — Police all VoIP traffic at 200 kbps.

In other words, you can police a larger group of traffic, but also prevent one particular subset of that group from taking over all the available bandwidth. In the preceding example (Example 5-12), web traffic can only take 400 kbps of the traffic, but the overall rate can be 500 kbps. This section ends with a configuration example that polices a larger set of traffic, and subsets of the larger set.

Table 5-23 lists the configuration commands used with CAR, and Table 5-24 lists the options for the actions to be taken when CAR decides a packet either conforms to or exceeds the traffic contract. Table 5-25 lists the CAR **show** commands.

Table 5-23 *Configuration Command Reference for CAR*

Command	Mode and Function
rate-limit {**input** | **output**} [**access-group** [**rate-limit**] *acl-index*] *bps burst-normal burst-max* **conform-action** *conform-action* **exceed-action** *exceed-action*	Interface mode; configures classification, marking, policing, and enables CAR on the interface
access-list rate-limit *acl-index* {*precedence* | *mac-address* | *exp* **mask** *mask*}	Global mode; creates a CAR ACL, which can match IP precedence, MAC addresses, and MPLS Experimental bits

Table 5-24 *Options for Actions Taken with the* **rate-limit** *Command*

Command	Mode and Function
continue	Evaluates the next **rate-limit** command
drop	Drops the packet
set-dscp-continue	Sets the differentiated services code point (DSCP) (0 to 63) and evaluates the next **rate-limit** command
set-dscp-transmit	Sends the DSCP and transmits the packet
set-mpls-exp-continue	Sets the MPLS Experimental bits (0 to 7) and evaluates the next **rate-limit** command
set-mpls-exp-transmit	Sets the MPLS Experimental bits (0 to 7) and sends the packet
set-prec-continue	Sets the IP precedence (0 to 7) and evaluates the next **rate-limit** command
set-prec-transmit	Sets the IP precedence (0 to 7) and sends the packet
set-qos-continue	Sets the QoS group ID (1 to 99) and evaluates the next **rate-limit** command
set-qos-transmit	Sets the QoS group ID (1 to 99) and sends the packet
transmit	Sends the packet

Table 5-25 *Exec Command Reference for CAR*

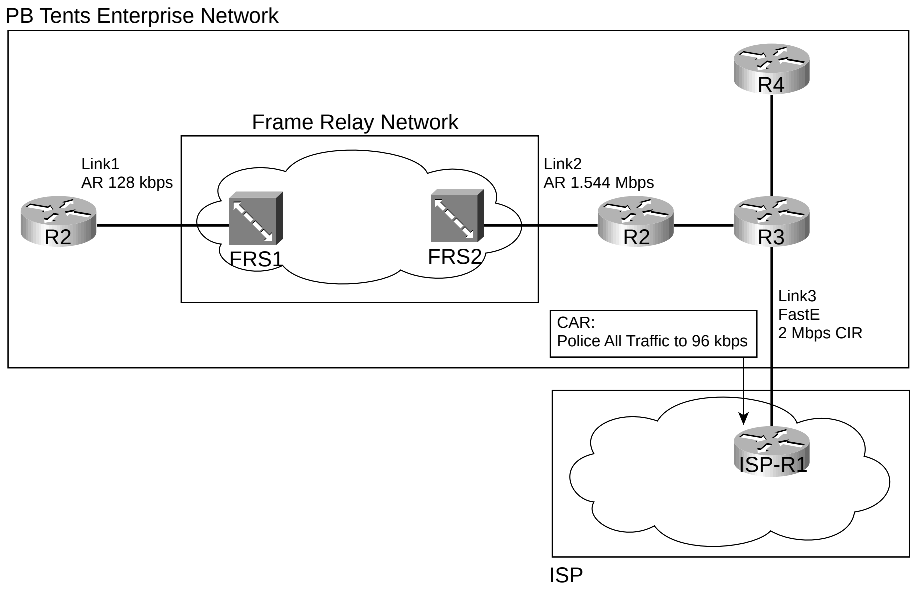
Command	Function
show interfaces [*interface-type interface-number*] **rate-limit**	Displays CAR statistics on the interface specified, or on all interfaces if the interface is not specified
show access-lists rate-limit [*acl-index*]	Lists information about the configuration of rate-limit ACLs

Like CB policing, you can use CAR to police all traffic entering or exiting an interface. In Example 5-13, router ISP-edge polices ingress traffic from an enterprise network. The criteria for the first CB policing example is as follows:

- All traffic policed at 96 kbps at ingress to the ISP-edge router.
- Bc of 1 second’s worth of traffic is allowed.
- Be of 0.5 second’s worth of traffic is allowed.
- Traffic that exceeds the contract is discarded.
- Traffic that conforms to the contract is forwarded with precedence reset to zero.

Figure 5-23 shows the network in which the configuration is applied, and Example 5-13 shows the configuration.

Figure 5-23 *Example network for Policing Examples*



Example 5-13 *CB Policing at 96 kbps at ISP-edge Router*

```
ISP-edge#show running-config
!
!Lines omitted for brevity
!
interface Serial1/0
 description connected to FRS port S1. Single PVC to R3.
 no ip address
 encapsulation frame-relay
 load-interval 30
 no fair-queue
 clockrate 1300000
!
interface Serial1/0.1 point-to-point
 description point-point subint global DLCI 101, connected via PVC to DLCI 103 (R3)
 ip address 192.168.2.251 255.255.255.0
! note: the rate-limit command wraps around to a second line.
```

continues

Example 5-13 *CB Policing at 96 kbps at ISP-edge Router (Continued)*

```
 rate-limit input 96000 12000 18000 conform-action set-prec-transmit 0
  exceed-action drop
 frame-relay interface-dlci 103
!
!Lines omitted for brevity
!
ISP-edge#show interfaces s 1/0.1 rate-limit
Serial1/0.1 point-point subint global DLCI 101, connected via PVC to DLCI 103 (R3)
  Input
    matches: all traffic
      params:  96000 bps, 12000 limit, 18000 extended limit
      conformed 2290 packets, 430018 bytes; action: set-prec-transmit 0
      exceeded 230 packets, 67681 bytes; action: drop
      last packet: 0ms ago, current burst: 13428 bytes
      last cleared 00:02:16 ago, conformed 25000 bps, exceeded 3000 bps
ISP-edge#
```

The configuration requires a single **rate-limit** command under serial 1/0.1 on router ISP-edge. All the parameters are typed in the single command line: **rate-limit input 96000 12000 18000 conform-action set-prec-transmit 0 exceed-action drop**. The rate of 96 kbps is listed with a Bc of 12,000 bytes, and a Be of 6000 bytes. (Remember, the **burst-excess** parameter of 18,000 is actually Bc + Be.)

The **show interfaces s1/0.1 rate-limit** command lists the operational statistics, including numbers of bytes and packets that conformed and exceeded the contract. Interestingly, the two measured rates (conform and exceed) over time do not total more than the policing rate; it appears that the preemptive discarding of packets with the debt process during Be processing is having some good effect. In this particular network, only three concurrent TCP connections were used to create traffic, so just a few packets lost would reduce the TCP windows, and reduce traffic quickly.

Example 5-14 exhibits how to classify traffic with CAR using rate-limit ACLs, and how to use CAR with cascaded **rate-limit** commands. To classify traffic, CAR requires the use of either a normal ACL, or a *rate-limit* ACL. A rate-limit ACL can match MPLS Experimental bits, IP precedence, or MAC address. For CAR to match other IP header fields, you must use an IP ACL. In Example 5-14, the CAR configuration meets the requirements of the example for cascaded statements mentioned in the introduction to this section, repeated in the following list.

- Police all traffic on the interface at 496 kbps; but before sending this traffic on its way . . .
  - — Police all web traffic at 400kbps.
  - — Police all FTP traffic at 160kbps
  - — Police all VoIP traffic at 200 kbps.

- Choose Bc and Be so that Bc has 1 second's worth of traffic, and Be provides no additional burst capability over Bc.

Example 5-14 shows the configuration.

Example 5-14 *Cascaded CAR* **rate-limit** *Commands, with Subclassifications*

```
!
! Next ACL matches all web traffic
!
Access-list 101 permit tcp any eq www any
Access-list 101 permit tcp any any eq www
!
! Next ACL matches all FTP traffic
!
access-list 102 permit tcp any eq ftp any
access-list 102 permit tcp any any eq ftp
access-list 102 permit tcp any eq ftp-data any
access-list 102 permit tcp any any eq ftp-data
!
! Next ACL matches all VoIP traffic
!
access-list 103 permit udp any range 16384 32767 any range 16384 32767
!
interface s 0/0
rate-limit input 496000 62000 62000 conform-action continue exceed-action drop
rate-limit input access-group 101 400000 50000 50000 conform-action transmit exceed-
  action drop
rate-limit input access-group 102 160000 20000 20000 conform-action transmit exceed-
  action drop
rate-limit input access-group 103 200000 25000 25000 conform-action transmit exceed-
  action drop
```

The CAR configuration needs to refer to IP ACLs to classify the traffic, using three different IP ACLs in this case. ACL 101 matches all web traffic, ACL 102 matches all FTP traffic, and ACL 103 matches all VoIP traffic.

Under subinterface S1/0.1, four **rate-limit** commands are used. The first sets the rate for all traffic, dropping traffic that exceeds 496 kbps. However, the conform action is listed as "continue". This means that packets conforming to this statement are now compared to the next rate-limit statements, and when matching a statement, some other action is taken. For instance, web traffic matches the second **rate-limit** command, with a resulting action of either transmit or drop. VoIP traffic is actually compared with the next three **rate-limit** commands before matching the last **rate-limit** command. The following list characterizes the types of traffic, and which **rate-limit** commands they match, in the example.

- All traffic matches the first **rate-limit** command, and is either dropped or passed to the second **rate-limit** command.
- All web traffic matches the second **rate-limit** command, and is either transmitted or dropped.

- All FTP traffic matches the third **rate-limit** command, and is either transmitted or dropped.
- All VoIP traffic matches the fourth **rate-limit** command, and is either transmitted or dropped.
- All other traffic is transmitted, because it did not match any more **rate-limit** commands.

You also may have noticed that the policing rates used in this example did not exactly match the values in the original problem statement at the beginning of this section. For instance, originally the requirement stated 500 kbps for all traffic; the configuration uses 496 kbps. CAR requires that the policing rate be a multiple of 8000, so the requirements were adjusted accordingly.

Table 5-26 summarizes the CAR features, comparing them with CB policing.

Table 5-26 *CAR and CB Policing Features Compared*

Feature	CB Policing	CAR
Allows conform and exceed action categories	Yes	Yes
Allows violate action category	Yes	No
Polices either all traffic, or a subset through classification	Yes	Yes
Uses MQC for configuration	Yes	No
Allows nested or cascaded policing logic	No	Yes
Can be enabled per subinterface	Yes	Yes
Can be enabled per DLCI on multipoint subinterfaces	No	No
Can set ATM CLP bit	Yes	No
Can set FR DE bit	Yes	No

Foundation Summary

The “Foundation Summary” is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures are a convenient way to review the day before the exam.

ISPs make the business choice of whether to police, and how aggressively to police. The options reduce to the following three basic options:

- Do not police. To support the traffic, build the network to support the traffic as if all customers will send and receive data at the clock rate of the access link. From a sales perspective, close deals by claiming that no policing will be done, but encourage customers who exceed their contracts to pay for more bandwidth.
- Police at the contracted rate. To support these traffic levels, the network only needs to be built to support the collective contracted rates, although the core would be overbuilt to support new customers. From a sales perspective, encourage customers that are beginning to exceed their contracts to upgrade, and give incentives.
- Police somewhere in between the contracted rate and the access-link clock rate. For instance, ISP1 might police PB Tents at 5 Mbps, when the contract reads 2 Mbps. The network can be built to support the collective policed rates. The sales team can encourage customers to buy a larger contracted rate when they consistently exceed the contracted rate, but keep customer satisfaction higher by pointing out their generosity by only policing at rates much higher than the contracted rates.

Figure 5-24 points out two cases of egress blocking, using a Frame Relay network as an example.

Figure 5-24 *PB Tents Network, Egress Blocking*

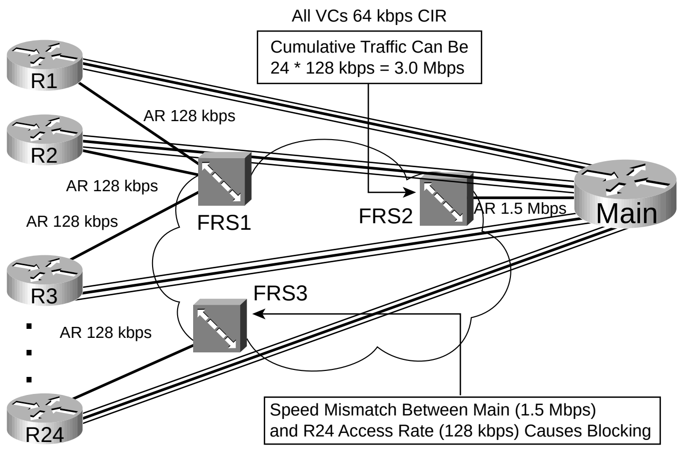


Table 5-27 summarizes some of the key points about when and where you should consider using policing and shaping.

Table 5-27 *Policing and Shaping: When to Use Them, and Where*

Topic	Rationale
Why police?	If a neighboring network can send more traffic than the traffic contract specifies, policing can be used to enforce the contract, protecting the network from being overrun with too much traffic.
Where to police?	Typically, policing is performed as packets enter the first device in a network. Egress policing is also supported, although it is less typical.
Why shape?	The first of two reasons for shaping is when the neighboring network is policing. Instead of waiting for the neighboring policer to discard traffic, a shaper can instead delay traffic so that it will not be dropped. The second reason has to do with the effects of egress blocking. By shaping, egress blocking can be avoided, or minimized, essentially moving the queues from inside the service provider cloud, and back into the enterprise routers. By doing so, the router queuing tools can selectively give better QoS performance to particular types of traffic.
Where to shape?	Shaping is always an egress function. Typically, shaping is performed on packets exiting a router, going into another network. This may be the edge between a router and a multiaccess WAN, or possibly just a link to an ISP.

Traffic shaping implements this basic logic by defining a measurement interval, and a number of bits that can be sent in that interval, so that the overall shaped rate is not exceeded. Table 5-28 lists some related definitions.

Table 5-28 *Shaping Terminology*

Term	Definition
Tc	Time interval, measured in milliseconds, over which the committed burst (Bc) can be sent. With many shaping tools, Tc = Bc/CIR.
Bc	Committed burst size, measured in bits. This is the amount of traffic that can be sent over the interval Tc. Typically also defined in the traffic contract.
CIR	Committed information rate, in bits per second, defines the rate defined in the traffic contract.
Shaped Rate	The rate, in bits per second, to which a particular configuration wants to shape the traffic. In some cases, the shaped rate is set to the same value as CIR; in others, the shaped rate is set to a larger value, with the hope of sending more traffic through the network.
Be	Excess burst size, in bits. This is the number of bits beyond Bc that can be sent after a period of inactivity.

The formulas IOS uses to calculate Tc when you configure both the shaping rate and the Bc are as follows:

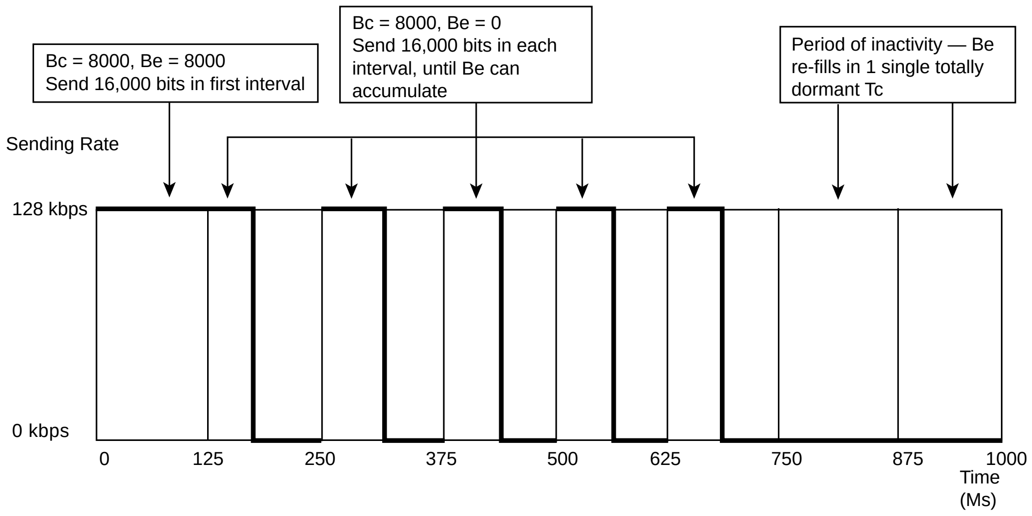
Tc = Bc/CIR

Or

Tc = Bc/Shaped rate

Figure 5-25 lists the overall process used by shaping, in terms of Tc, Bc, and Be.

Figure 5-25 *Bc and Be, After a Period of Inactivity (Both Buckets Full)*



In summary, most QoS policies call for shaping on each VC. The number of VCs, and how they are configured, dictates where the shaping tool needs to be enabled. Table 5-29 summarizes the options.

Table 5-29 *Options of How to Enable Shaping for per-VC Shaping*

Location	Requirements for Shaping per VC
No VCs	Shape on the main interface. Shaping occurs for all traffic on interface.
Physical interface, 1 VC, no subinterfaces	Shaping shapes the individual VC associated with this interface. Shaping is enabled on the physical interface.
Physical interface, 1 VC, 1 subinterface	Shaping shapes the individual VC associated with this interface. Shaping can be enabled on the physical interface, the subinterface, or the VC (DLCI).

continues

Table 5-29 *Options of How to Enable Shaping for per-VC Shaping (Continued)*

Location	Requirements for Shaping per VC
Multiple VCs on 1 interface, point-to-point subinterfaces only	Shaping can be enabled on the subinterface, or per DLCI. Both methods work identically.
Multiple VCs on 1 interface, some multipoint subinterfaces with > 1 VC per subinterface	Must enable shaping on each DLCI to shape per VC.

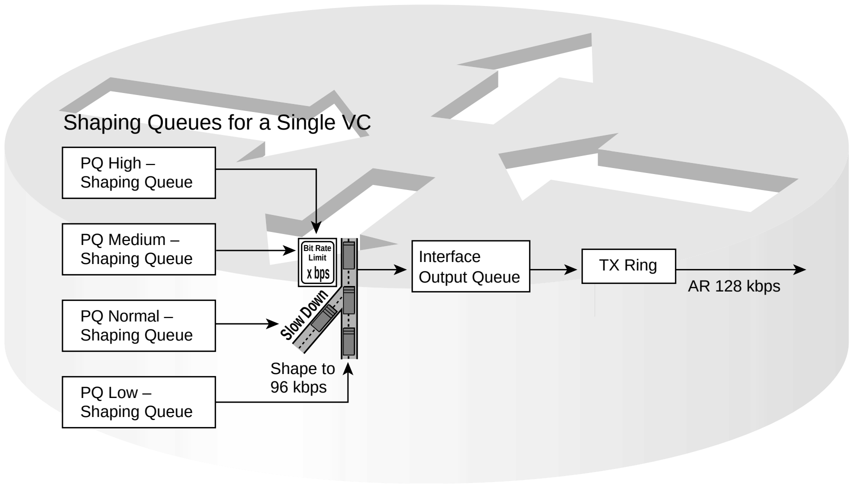
Table 5-30 lists the traffic-shaping tools, and the queuing tools supported by each for the shaping queues.

Table 5-30 *Options for Queuing in Traffic-Shaping Tools*

Shaping Tool	Queuing Tools Supported for the Shaping Queue(s)
GTS	WFQ
CB shaping	FIFO, WFQ, CBWFQ, LLQ
DTS	FIFO, WFQ, CBWFQ, LLQ
FRTS	FIFO, WFQ, CBWFQ, LLQ, PQ, CQ,

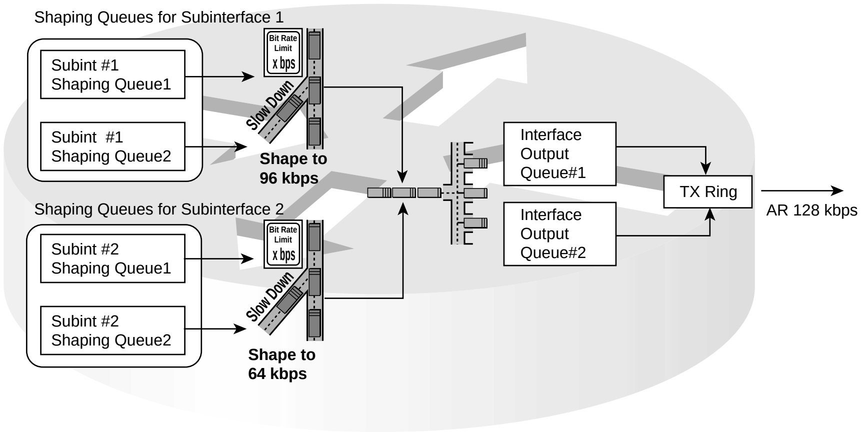
When a shaper uses a queuing tool, instead of having a single shaping queue, multiple shaping queues exist. If FRTS were configured to use PQ, for example, up to four queues could be created for shaping. Figure 5-26 shows the basic idea, with shaping enabled on the physical interface, FIFO Queuing on the physical interface, and PQ configured for the shaping queue.

Figure 5-26 *FIFO Queuing for the Physical Interface, Plus PQ for the Shaping Queue*



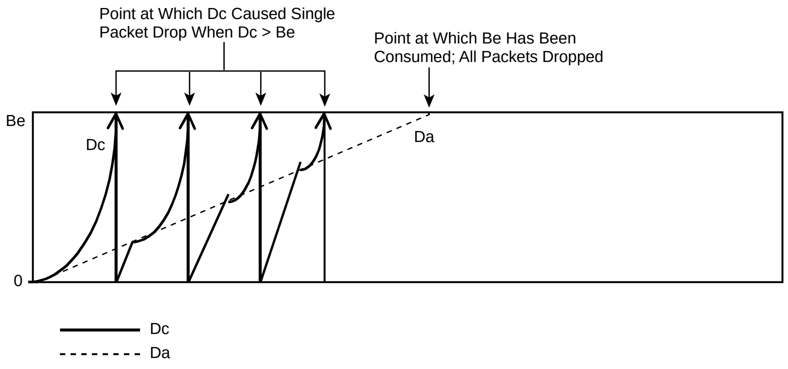
Many QoS designs call for shaping per VC. For the same router, with two 64-kbps CIR VCs (each configured on a separate point-to-point subinterface) shaping queues are created for each subinterface, with the familiar output queues as well. Figure 5-27 depicts the overall idea.

Figure 5-27 *Fancy Queuing for the Physical Interface as well as for Two Sets of Shaping Queues*



One goal of policing when Be is being used is to discard some packets, hoping to avoid the point where all packets are discarded. Figure 5-28 shows the general idea of how policers accomplish that goal, using Da and Dc values.

Figure 5-28 *Actual Debt and Compounded Debt with Policing*



Tables 5-31 and 5-32 list the configuration and **show** commands pertinent to GTS.

Table 5-31 *Command Reference for Generic Traffic Shaping*

Command	Mode and Function
traffic-shape rate *bit-rate* [*burst-size* [*excess-burst-size*]]	Interface configuration mode; enables GTS for a shaped rate, with optional Bc and Be settings, in bits
traffic-shape group *access-list bit-rate* [*burst-size* [*excess-burst-size*]]	Interface configuration mode; enables GTS for a shaped rate, only for traffic permitted by the referenced ACL, with optional Bc and Be settings, in bits
traffic-shape adaptive *bit-rate*	Interface configuration mode; enables adaptive shaping, and sets the minimum shaped rate
traffic-shape fecn-adapt	Interface configuration mode; enables the reflection of BECN signals upon receipt of an FECN

Table 5-32 *Exec Command Reference for Generic Traffic Shaping*

Command	Function
show traffic-shape [*interface-type interface-number*]	Lists information about the configuration details
show traffic-shape queue [*interface-number* [**dlci** *dlci-number*]]	Lists statistics about the queuing tool used on the shaping queue
show traffic-shape statistics [*interface-type interface-number*]	Lists statistics about shaping operation

Tables 5-33 and 5-34 list the configuration and show commands pertinent to CB shaping.

Table 5-33 *Command Reference for Class-Based Shaping*

Command	Mode and Function
shape [**average** | **peak**] *mean-rate* [[*burst-size*] [*excess-burst-size*]]	Policy-map class configuration mode; enables shaping for the class, setting the shaping rate, and optionally Bc and Be. The average option causes shaping as normal; the peak option causes Bc + Be to be sent per Tc.
Shape adaptive *min-rate*	Policy-map class configuration mode; enables the minimum rate for adaptive shaping. The maximum rate is configured with the **shape average** or **shape peak** command.
Shape fecn-adapt	Policy-map class configuration mode; enables reflection of BECN signals upon receipt of an FECN.
service-policy {**input** | **output**} *policy-map-name*	Interface or subinterface configuration mode; enables CB shaping on the interface.

Table 5-33 *Command Reference for Class-Based Shaping (Continued)*

Command	Mode and Function
class-map *class-map-name*	Global config; names a class map, where classification options are configured.
Match . . .	**class-map** subcommand; defines specific classification parameters.
match access-group {*access-group* | **name** *access-group-name*}	**class-map** subcommand; references either numbered or named ACL.
match source-address mac *address*	**class-map** subcommand; references the source MAC address that forwarded the packet to this router, typically the previous-hop router.
match ip precedence *ip-precedence-value* [*ip-precedence-value ip-precedence-value ip-precedence-value*]	**class-map** subcommand; references one or more IP precedence values. A packet with any of the listed values matches.
match mpls experimental *number*	**class-map** subcommand; references MPLS Experimental bits.
match cos *cos-value* [*cos-value cos-value cos-value*]	**class-map** subcommand; references one or more class of service (CoS) values. A packet with any of the listed values matches.
match destination-address mac *address*	**class-map** subcommand; references the destination MAC address of the device to which the packet will be forwarded next.
match input-interface *interface-name*	**class-map** subcommand; matches based on the interface in which the packet was received.
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	**class-map** subcommand; references one or more IP DSCP values. A packet with any of the listed values matches.
match ip rtp *starting-port-number port-range*	**class-map** subcommand; references a range of UDP ports, only matching the even numbered ports, which carry voice payload.
match qos-group *qos-group-value*	**class-map** subcommand; matches based on QoS group.
match protocol *protocol-name*	**class-map** subcommand; matches based on NBAR-defined protocol types.
match protocol citrix [**app** *application-name-string*].	**class-map** subcommand; matches based on NBAR-defined Citrix application types.
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	**class-map** subcommand; matches based on NBAR-discovered details inside the host name or URL string.

continues

Table 5-33 *Command Reference for Class-Based Shaping (Continued)*

Command	Mode and Function
match any	**class-map** subcommand; matches all packets.
policy-map *policy-map-name*	Global config; names a policy, which is a set of actions to perform.
class *name*	**policy-map** subcommand; identifies which packets on which to perform some action by referring to the classification logic in a class map.

Table 5-34 *Exec Command Reference for Class-Based Shaping*

Command	Function
show policy-map *policy-map-name*	Lists configuration information about all MQC-based QoS tools
show policy-map *interface-spec* [**input** | **output**] [**class** *class-name*]	Lists statistical information about the behavior of all MQC-based QoS tools

Tables 5-35 and 5-36 list the configuration and **show** commands pertinent to FRTS.

Table 5-35 *Command Reference for Frame Relay Traffic Shaping*

Command	Mode and Function
frame-relay traffic-shaping	Interface subcommand; enables FRTS on the interface.
class *name*	Interface DLCI subcommand; enables a specific FRTS map class for the DLCI.
frame-relay class *name*	Interface or subinterface command; enables a specific FRTS map class for the interface or subinterface.
map-class frame-relay *map-class-name*	Global configuration mode; names a map class, and places CLI into map-class configuration mode
frame-relay priority-group *list-number*	Map-class configuration mode; enables PQ for the shaping queues associated with this map class
frame-relay custom-queue-list *list-number*	Map-class configuration mode; enables CQ for the shaping queues associated with this map class
frame-relay fair-queue [*congestive_discard_threshold* [*number_dynamic_conversation_queues* [*number_reservable_conversation_queues* [*max_buffer_size_for_fair_queues*]]]]	Map-class configuration mode; enables WFQ for the shaping queues associated with this map class

Table 5-35 *Command Reference for Frame Relay Traffic Shaping (Continued)*

Command	Mode and Function
service-policy {**input** | **output**} *policy-map-name*	Map-class configuration mode; enables LLQ or CBWFQ on the shaping queues associated with the map class.
frame-relay traffic-rate *average* [*peak*]	Map-class configuration mode; sets the shaped rate, and the EIR*. Bc and Be are calculated from these, based on Tc of 125ms.
frame-relay bc {**in** | **out**} *bits*	Map-class configuration mode; sets the Bc value. Alternative configuration option to **frame-relay traffic-rate**.
frame-relay be {**in** | **out**} *bits*	Map-class configuration mode; sets the Be value. Alternative configuration option to **frame-relay traffic-rate**.
frame-relay cir {**in** | **out**} *bps*	Map-class configuration mode; sets the CIR value. Alternative configuration option to **frame-relay traffic-rate**.
frame-relay adaptive-shaping {**becn** | **foresight**}	Map-class configuration mode; enables adaptive shaping, specifying either BECN or Foresight for signaling.
frame-relay mincir {**in** | **out**} *bps*	Map-class configuration mode; sets the minimum CIR used for adaptive shaping.
frame-relay tc *milliseconds*	Map-class configuration mode; for 0 CIR VCs, sets the Tc value.
frame-relay qos-autosense	Interface configuration mode; uses ELMI to automatically discover CIR, Bc, and Be settings for each VC.

* EIR = excess information rate

Table 5-36 *Exec Command Reference for Frame Relay Traffic Shaping*

Command	Function
show frame-relay pvc [**interface** *interface*] [*dlci*]	Shows PVC statistics, including shaping statistics
show traffic-shape [*interface-type interface-number*]	Shows information about FRTS configuration per VC
show traffic-shape queue [*interface-number* [**dlci** *dlci-number*]]	Shows information about the queuing tool used with the shaping queue
show traffic-shape statistics [*interface-type interface-number*]	Shows traffic-shaping statistics

Frame Relay fragmentation (FRF) can be used with FRTS. Figure 5-29 outlines the basic idea, with FRTS on two subinterfaces.

Figure 5-29 *Interaction Between Shaping Queues and Frame Relay Fragmentation Queues*

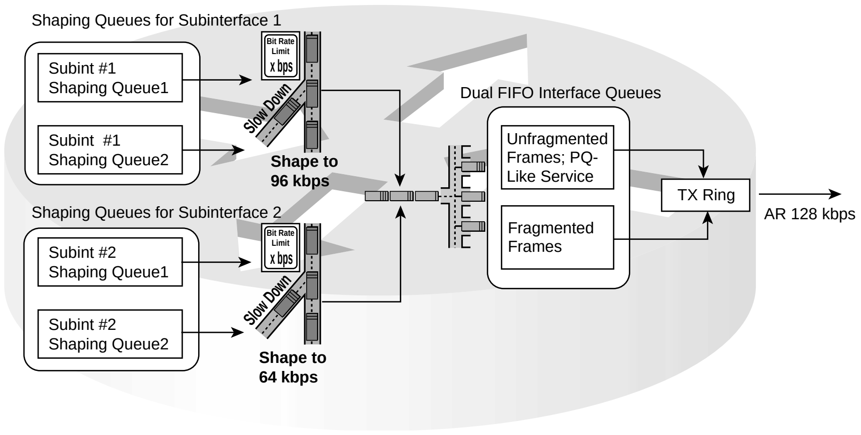


Table 5-37 summaries the key points for comparison between the various traffic-shaping tools.

Table 5-37 *Comparison of Traffic Shaping Tools: GTS, CB Shaping, DTS, and FRTS*

Feature	GTS	CB Shaping	DTS	FRTS
Supports ATM, FR, HDLC, PPP, LAN interfaces	Yes	Yes	Yes	No
Can be enabled on interfaces and subinterfaces	Yes	Yes	Yes	Yes
Can be enabled per Frame Relay DLCI to support per-VC shaping on multipoint interfaces	No	No	No	Yes
Supports adaptive shaping	Yes	Yes	Yes	Yes
Supports concurrent FRF.12 Frame Relay fragmentation	No	No	No	Yes
Queuing methods in shaping queue	WFQ	FIFO, WFQ, CBWFQ, LLQ	FIFO, WFQ, CBWFQ, LLQ	FIFO, WFQ, CBWFQ, LLQ, PQ, CQ
Concurrent queuing methods on Physical interface	All	All	All	FIFO, FRF*

Table 5-37 *Comparison of Traffic Shaping Tools: GTS, CB Shaping, DTS, and FRTS (Continued)*

Feature	GTS	CB Shaping	DTS	FRTS
Can be configured using MQC commands	No	Yes	Yes	No
Can classify traffic to shape a subset of the traffic on an interface/VC	Yes	Yes	Yes	No
Default Tc	Variable	125 ms	125 ms	125 ms
Distributes shaping processing to VIPs in 7500 series routers	No	No	Yes	No

* The Cisco QoS course claims WFQ is supported on the physical interface. In addition, FRF is not technically a queuing tool, although its feature of using two queues does achieve the same effect.

Table 5-38 lists the matchable fields for classification for CB policing and CAR.

Table 5-38 *Classification Fields Used by CAR and CB Policing*

Field	Tool	Comments
Anything matched with an IP ACL	CAR, CB policing	N/A
Source MAC Address	CAR, CB policing	CAR uses a special access-rate ACL; CB marking uses the match command.
IP Precedence	CAR, CB policing	CAR uses a special access-rate ACL specific to CAR; CB marking uses the **match** command; Both can match a subset of values.
MPLS Experimental	CAR, CB policing	CAR uses a special access-rate ACL specific to CAR; CB marking uses the **match** command. Both can match a subset of values.
IP DSCP	CAR, CB policing	Can check for multiple values using multiple **match** commands.
QoS Group	CAR, CB policing	The QoS Group field is used to tag packets internal to a single router.
Class of Service (CoS)	CB policing	Checks incoming ISL/802.1P CoS bits. Can match multiple values.
Destination MAC Address	CB policing	N/A
Input interface	CB policing	N/A

continues

Table 5-38 *Classification Fields Used by CAR and CB Policing (Continued)*

Field	Tool	Comments
RTP's UDP port-number range	CB policing	RTP uses even numbered UDP ports from 16384–32767 for voice payload. This matching option allows matching a subset of the port numbers, and it matches only the even-numbered ports.
NBAR protocol types	CB policing	Refer to the coverage in Chapter 3 for more details.
NBAR Citrix applications	CB policing	NBAR can recognize different types of Citrix applications; CB marking can use NBAR to classify based on these application types.
Host Name and URL string	CB policing	NBAR can match URL strings using regular expressions, including the host name. CB marking can use NBAR to match these strings for classification.

Table 5-39 lists the various actions associated with CB policing and CAR.

Table 5-39 *Policing Actions Used by CAR and CB Policing*

Action Keyword	Meaning	CAR?	CB Policer?
drop	Discards the packet	Yes	Yes
transmit	Forwards the packet	Yes	Yes
set-prec-transmit	Forwards the packet after marking the IP precedence value.	Yes	Yes
set-qos-transmit	Forwards the packet after marking the QoS group	Yes	Yes
set-dscp-transmit	Forwards the packet after marking the IP DSCP value	Yes	Yes
set-mpls-exp-transmit	Forwards the packet after marking the MPLS Experimental bits	Yes	Yes
set-frde-transmit	Forwards the packet after marking the Frame Relay discard eligibility (DE) bit	No	Yes
set-clp-transmit	Forwards the packet after marking the ATM cell loss priority (CLP) bit	No	Yes
set-prec-continue	Marks the IP precedence value, and continues to the next nested (cascaded) CAR command	Yes	No
set-dscp-continue	Marks the QoS group, and continues to the next nested (cascaded) CAR command	Yes	No

Table 5-39 *Policing Actions Used by CAR and CB Policing (Continued)*

Action Keyword	Meaning	CAR?	CB Policer?
set-mpls-exp-continue	Marks the IP precedence value, and continues to the next nested (cascaded) CAR command	Yes	No
set-qos-continue	Marks the QoS group, and continues to the next nested (cascaded) CAR command	Yes	No
continue	Just continues to the next nested (cascaded) CAR command	Yes	No

Tables 5-40 and 5-41 list the CB policing configuration and **show** commands, respectively.

Table 5-40 *Command Reference for Class-Based Policing*

Command	Mode and Function
police *bps burst-normal burst-max* **conform-action** *action* **exceed-action** *action* [**violate-action** *action*]	Policy-map class subcommand; enables policing for the class, setting the police rate, Bc, and Bc + Be values, and actions taken. Actions are **drop**, **set-clp-transmit**, **set-dscp-transmit**, **set-prec-transmit**, **set-qos-transmit**, **transmit**.
service-policy {**input** | **output**} *policy-map-name*	Interface or subinterface configuration mode; enables CB shaping on the interface.
class-map *class-map-name*	Global config; names a class map, where classification options are configured.
Match …	Class-map subcommand; defines specific classification parameters.
match access-group {*access-group* | **name** *access-group-name*}	Access-control list (ACL).
match source-address mac *address-destination*	Source MAC address.
match ip precedence *ip-precedence-value* [*ip-precedence-value ip-precedence-value ip-precedence-value*]	IP precedence.
match mpls experimental *number*	MPLS Experimental.
match cos *cos-value* [*cos-value cos-value cos-value*]	CoS.
match destination-address mac *address*	Destination MAC address.
match input-interface *interface-name*	Input interface.

continues

Table 5-40 *Command Reference for Class-Based Policing (Continued)*

Command	Mode and Function
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	IP DSCP.
match ip rtp *starting-port-number port-range*	RTP's UDP port-number range.
match qos-group *qos-group-value*	QoS group.
match protocol *protocol-name*	NBAR protocol types.
match protocol citrix [**app** *application-name-string*].	NBAR Citrix applications.
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	Host name and URL string.
match any	All packets.
policy-map *policy-map-name*	Global config; names a policy, which is a set of actions to perform.
class *name*	**policy-map** subcommand; identifies which packets on which to perform some action by referring to the classification logic in a class map.

Table 5-41 *Exec Command Reference for Class-Based Policing*

Command	Function
show policy-map *policy-map-name*	Lists configuration information about all MQC-based QoS tools
show policy-map *interface-spec* [*input* | *output*] [**class** *class-name*]	Lists statistical information about the behavior of all MQC-based QoS tools

Tables 5-42 and 5-43 list the CAR configuration and **show** commands, respectively.

Table 5-42 *Configuration Command Reference for CAR*

Command	Mode and Function
rate-limit {**input** | **output**} [**access-group** [**rate-limit**] *acl-index*] *bps burst-normal burst-max* **conform-action** *conform-action* **exceed-action** *exceed-action*	Interface mode; configures classification, marking, policing, and enables CAR on the interface
access-list rate-limit *acl-index* {*precedence* | *mac-address* | *exp* **mask** *mask*}	Global mode; creates a CAR ACL, which can match IP precedence, MAC addresses, and MPLS Experimental bits
The remaining entries describe the possible actions in the **rate-limit** command:	
continue	Evaluates the next **rate-limit** command
drop	Drops the packet
set-dscp-continue	Sets the differentiated services code point (DSCP) (0 to 63) and evaluates the next **rate-limit** command.
set-dscp-transmit	Sends the DSCP and transmits the packet
set-mpls-exp-continue	Sets the MPLS Experimental bits (0 to 7) and evaluates the next **rate-limit** command
set-mpls-exp-transmit	Sets the MPLS Experimental bits (0 to 7) and sends the packet
set-prec-continue	Sets the IP precedence (0 to 7) and evaluates the **next rate-limit** command
set-prec-transmit	Sets the IP precedence (0 to 7) and sends the packet.
set-qos-continue	Sets the QoS group ID (1 to 99) and evaluates the next **rate-limit** command
set-qos-transmit	Sets the QoS group ID (1 to 99) and sends the packet
transmit	Sends the packet

Table 5-43 *Exec Command Reference for CAR*

Command	Function
show interfaces [*interface-type interface-number*] **rate-limit**	Displays CAR statistics on the interface specified, or on all interfaces if the interface is not specified
show access-lists rate-limit [*acl-index*]	Lists information about the configuration of rate-limit ACLs

Table 5-44 summarizes the CAR features, comparing them with CB policing.

Table 5-44 *CAR and CB Policing Features Compared*

Feature	CB Policing	CAR
Allows conform and exceed action categories	Yes	Yes
Allows violate action category	Yes	No
Polices either all traffic, or a subset through classification	Yes	Yes
Uses MQC for configuration	Yes	No
Allows nested or cascaded policing logic	No	Yes
Can be enabled per subinterface	Yes	Yes
Can be enabled per DLCI on multipoint subinterfaces	No	No
Can set ATM CLP bit	Yes	No
Can set FR DE bit	Yes	No

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD exam to include, or not include, the topics that are only on the CCIP QoS.

Shaping and Policing Concepts

1. Explain the points during the process of a single router receiving and forwarding traffic at which shaping and policing can be enabled on a router.
2. Compare and contrast the actions that shaping and policing take when a packet exceeds a traffic contract.
3. Compare and contrast the effects that shaping and policing have on bandwidth, delay, jitter, and loss.
4. Describe the typical locations to enable shaping and policing in an internetwork.
5. Describe the reasons behind egress blocking in a Frame Relay network with a T1 access link at the main site, 128-kbps access links at each of 20 remote sites, with 64-kbps CIR VCs from the main site to each remote site.
6. If a router has a shaping tool configured, with a shaping rate of 256 kbps, and a Bc of 16,000 bits, what Tc value does the shaping tool use?
7. If a router has a shaping tool configured, with a shaping rate of 512 kbps, and a Be of 16,000 bits, what Tc value does the shaping tool use?
8. Define the terms Tc, Bc, Be, and CIR.
9. Explain the goal of the Da and Dc debt processes when using CAR.
10. Describe the concept of traffic-shaping adaption, explaining the two triggers that cause shaping to adapt.
11. Describe the difference between interface output queues and shaping queues, and explain where the queues could exist on a router with 1 physical interface and 20 subinterfaces.

Traffic Shaping

12 What do the following intialisms stand for? FRTS, GTS, DTS, and CB shaping

13 List the command, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using GTS. Do not assume any defaults; explicitly set the values in the command.

14 Along with the **class-map**, **policy-map**, and **service-policy** commands, CB shaping requires one specific command that actually sets values used for the shaping function. List the command, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using CB shaping. Do not assume any defaults; explicitly set the values in the command.

15 Along with the **class-map**, **policy-map**, and **service-policy** commands, DTS requires one specific command that actually sets values used for the shaping function. List the command, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using DTS. Do not assume any defaults; explicitly set the values in the command.

16 Many commands are needed to configure FRTS, but the actual shaping parameters, such as CIR and Bc, are set under the **map-class frame-relay** command. List the **map-class** subcommands, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using FRTS, and using multiple commands. Do not assume any defaults; explicitly set the values in the commands.

17 Compare and contrast the use of the **class-map** and **map-class** commands in terms of how each is used by FRTS and CB shaping.

18 Describe the context inside the configuration mode under which the **service-policy** command can be used to enable LLQ on an FRTS shaping queue. (“Context” means what part of configuration mode—for instance, global-configuration mode, interface configuration mode, and so on.)

19 Explain the context inside the configuration mode under which the **service-policy** command can be used to enable LLQ on a CB shaping queue. (“Context” means what part of configuration mode—for instance, global-configuration mode, interface configuration mode, and so on.)

20 Explain the context inside the configuration mode under which the **service-policy** command can be used to enable LLQ on a GTS shaping queue. (“Context” means what part of configuration mode—for instance, global-configuration mode, interface configuration mode, and so on.)

21 GTS has been configured under subinterface s0/0.1. What **show** command lists statistics for GTS behavior just for that subinterface?

22 DTS has been configured under subinterface s0/0.1. What **show** command lists statistics for DTS behavior just for that subinterface?

23 CB shaping has been configured under subinterface s0/0.1. What **show** command lists statistics for CB shaping behavior just for that subinterface?

24 FRTS has been configured under subinterface s0/0.1. What **show** command lists statistics for FRTS behavior just for that subinterface?

25 Which of the traffic-shaping tools can be enabled on PPP interfaces?

26 Which of the traffic-shaping tools can be enabled on each VC on a Frame Relay multipoint subinterface?

27 Which of the traffic-shaping tools support adaptive shaping?

28 For which of the traffic-shaping tools does IOS perform shaping processing on 7500 VIP cards?

29 Which of the traffic-shaping tools can classify traffic to shape subsets of the traffic on a subinterface?

30 Which shaping tools do not support WFQ and PQ to be used for the shaping queues?

31 Which shaping tools do not allow WFQ or LLQ to be used for the interface output queuing at the same time as the shaping tool is enabled on the same interface?

Traffic-Policing Tools

32 List the command, with the correct syntax, that sets a policed rate of 512 kbps, a Bc of 1 second's worth of traffic, and a Be of an additional 0.5 second's worth of traffic, when using CAR. Do not assume any defaults; explicitly set the values in the command. You can choose any other settings needed for the command.

33 Explain the use of the **continue** keyword as part of CAR policing actions, and the feature CAR provides through the keyword.

34 Which policing tools allow for three categories of actions to take?

35 Explain the concept behind re-marking policed packets versus discarding the packets.

36 CAR has been configured under subinterface s0/0.1. What **show** command lists statistics for CAR behavior just for that subinterface?

37 CB policing has been configured under subinterface s0/0.1. What **show** command lists statistics for CB policing behavior just for that subinterface?

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Topics

- Explain how TCP responds to congestion.
- Explain tail drop and global synchronization.
- Identify and differentiate between the following Cisco IOS Software congestion-avoidance tools: RED, WRED, and FRED.
- Configure Cisco IOS Software congestion-avoidance features.

QoS Exam Objectives

- Describe Random Early Detection (RED).
- Describe the Weighted Random Early Detection (WRED) mechanism.
- Configure Weighted Random Early Detection (WRED) on Cisco routers.
- Describe the Flow-based WRED mechanism.

CHAPTER 6

Congestion Avoidance Through Drop Policies

Quality of service (QoS) congestion-avoidance tools help prevent congestion before it occurs. These tools monitor queue depth, and before the queue fills, they drop some packets. The computers sending the packets might reduce the frequency of sending packets in reaction to the packet loss, particularly if the application sending the data uses TCP. In the moments after the congestion-avoidance tool discards packets, congestion is reduced, because less traffic is sent into the network.

Cisco congestion-avoidance tools rely on the behavior of TCP to reduce congestion. TCP flows do slow down after packet loss. By discarding some TCP packets before congestion gets bad, congestion-avoidance tools may actually reduce the overall number of packets dropped, and reduce congestion, thereby indirectly reducing delay and jitter.

The first section of this chapter begins with a review of TCP and a description of how TCP slows down after packet loss. Following that, several of the problems that can be solved using congestion-avoidance tools are described, namely tail drop, global synchronization, and TCP starvation. The concepts section of this chapter ends with coverage of Random Early Detection (RED), which defines several algorithms, which are the basis of IOS congestion-avoidance tools.

Weighted RED (WRED) and Flow-Based WRED (FRED) are the two congestion-avoidance tools available in IOS. These are covered in sequence with the differences between the two highlighted at the end of the chapter.

"Do I Know This Already?" Quiz

The purpose of the "Do I Know This Already?" quiz is to help you decide whether you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 12-question quiz, derived from the major sections in "Foundation Topics" section of this chapter, helps you determine how to spend your limited study time.

Table 6-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Use Table 6-1 to record your score.

Table 6-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions
1	Congestion-Avoidance Concepts and RED	1 to 4
2	WRED	5 to 8
3	FRED	9 to 12
All questions		1 to 12

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **10 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary,", and "Q&A" sections.
- **11 or 12 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, move to the next chapter.

Congestion-Avoidance Concepts and RED Questions

1 Describe the process of TCP slow start and discuss when it occurs.

2 Describe the meaning of the term "global synchronization," and discuss what causes it.

3 Define the meaning of the term "tail drop."

4 Does RED compare the actual queue depth or the average queue depth to queue thresholds when deciding whether it should discard a packet? Why this one, and not the other?

WRED Questions

5 List the queuing tools that can enable WRED for use with some or all of their queues, effectively enabling WRED concurrently with the queuing tool.

6 Describe how WRED "weights" packets.

7 Taking as many defaults as possible, list the configuration commands needed to configure precedence-based WRED on interface S1/1.

8 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based WRED inside **class class1**, inside **policy map my-policy**. (You can assume that the CBWFQ configuration has already been completed, and you just entered global configuration mode. Assume that you need just to enable WRED in **class class1**.)

FRED Questions

9 Identify the most significant difference between FRED operation and WRED operation.

10 List the three categories of flows defined by FRED, and identify which category has its packets discarded most aggressively.

11 Taking as many defaults as possible, list the configuration commands needed to configure precedence-based FRED on interface S1/1.

12 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based FRED on interface S1/1.

Foundation Topics

The term "congestion avoidance" describes a small set of IOS tools that help queues avoid congestion. Queues fill when the cumulative offered load from the various senders of packets exceeds the line rate of the interface or the shaping rate if shaping is enabled. When more traffic needs to exit the interface than the interface can support, queues form. Queuing tools help us manage the queues; congestion-avoidance tools help us reduce the level of congestion in the queues by selectively dropping packets.

Once again, a QoS tool gives you the opportunity to make tradeoffs between QoS characteristics—in this case, packet loss versus delay and jitter. However, the tradeoff is not so simple in this case. It turns out that by selectively discarding some packets before the queues get completely full, cumulative packet loss can be reduced, and queuing delay and queuing jitter can also be reduced! When you prune a plant, you kill some of the branches, but the plant gets healthier and more beautiful through the process. Similarly, congestion-avoidance tools discard some packets, but in doing so achieve the overall effect of a healthier network.

This chapter begins by explaining the core concepts that create the need for congestion-avoidance tools. Following this discussion, the underlying algorithms, which are based RED, are covered. Finally, the chapter includes coverage of configuration and monitoring for the two IOS congestion-avoidance tools, WRED and FRED.

Congestion-Avoidance Concepts and Random Early Detection (RED)

Congestion-avoidance tools rely on the behavior of TCP to reduce congestion. A large percentage of Internet traffic consists of TCP traffic, and TCP senders reduce the rate at which they send packets after packet loss. By purposefully discarding a percentage of packets, congestion-avoidance tools cause some TCP connections to slow down, which reduces congestion.

This section begins with a discussion of User Datagram Protocol (UDP) and TCP behavior when packets are lost. By understanding TCP behavior in particular, you can appreciate what happens as a result of tail drop, which is covered next in this section. Finally, to close this section, RED is covered. The two IOS congestion-avoidance tools both use the underlying concepts of RED.

TCP and UDP Reactions to Packet Loss

UDP and TCP behave very differently when packets are lost. UDP, by itself, does not react to packet loss, because UDP does not include any mechanism with which to know whether a packet was lost. TCP senders, however, slow down the rate at which they send after recognizing that a packet was lost. Unlike UDP, TCP includes a field in the TCP header to number each TCP segment (sequence number), and another field used by the receiver to confirm receipt of the packets (acknowledgment number). When a TCP receiver signals that a packet was not received, or if an acknowledgment is not received at all, the TCP sender assumes the packet was lost, and resends the packet. More importantly, the sender also slows down sending data into the network.

TCP uses two separate window sizes that determine the maximum window size of data that can be sent before the sender must stop and wait for an acknowledgment. The first of the two different windowing features of TCP uses the Window field in the TCP header, which is also called the *receiver window* or the *advertised window.* The receiver grants the sender the right to send *x* bytes of data before requiring an acknowledgment, by setting the value *x* into the Window field of the TCP header. The receiver grants larger and larger windows as time goes on, reaching the point at which the TCP sender never stops sending, with acknowledgments arriving just before a complete window of traffic has been sent.

The second window used by TCP is called the *congestion window*, or CWND, as defined by RFC 2581. Unlike the advertised window, the congestion window is not communicated between the receiver and sender using fields in the TCP header. Instead, the TCP sender calculates CWND. CWND varies in size much more quickly than does the advertised window, because it was designed to react to congestion in networks.

The TCP sender always uses the lower of the two windows to determine how much data it can send before receiving an acknowledgment. The receiver window is designed to let the receiver prevent the sender from sending data faster than the receiver can process the data. The CWND is designed to let the sender react to network congestion by slowing down its sending rate. It is the variation in the CWND, in reaction to lost packets, which RED relies upon.

To appreciate how RED works, you need to understand the processes by which a TCP sender lowers and increases the CWND. CWND is lowered in response to lost segments. CWND is raised based on the logic defined as the TCP slow start and TCP congestion-avoidance algorithms. In fact, most people use the term "slow start" to describe both features together, in part because they work closely together. The process works like this:

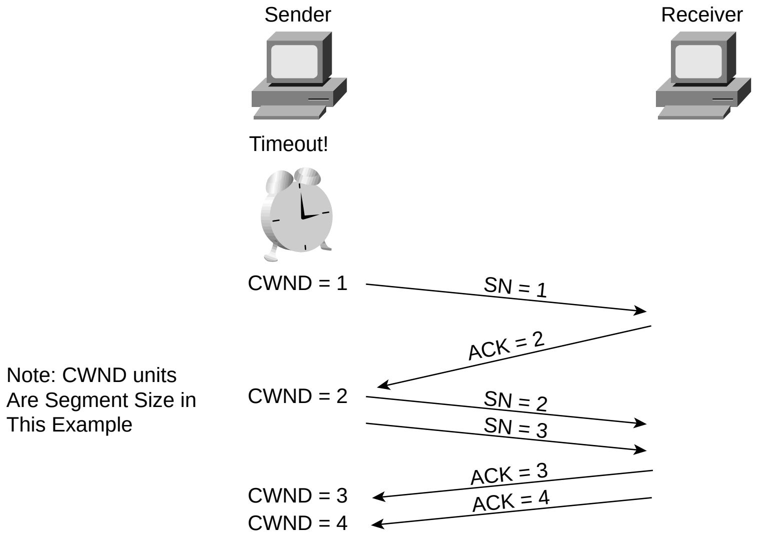
- A TCP sender fails to receive an acknowledgment in time, signifying a possible lost packet.
- The TCP sender sets CWND to the size of a single segment.
- Another variable, called slow start threshold (SSTHRESH) is set to 50 percent of the CWND value before the lost segment.
- After CWND has been lowered, slow start governs how fast the CWND grows up until the CWND has been increased to the value of SSTHRESH.
- After the slow start phase is complete, congestion avoidance governs how fast CWND grows after CWND > SSTHRESH.

Therefore, when a TCP sender fails to receive an acknowledgment, it reduces the CWND to a very low value (one segment size of window). This process is sometimes called *slamming the window* or *slamming the window shut*. The sender progressively increases CWND based first on slow start, and then on congestion avoidance. As you go through this text, remember that the TCP windows use a unit of bytes in reality. To make the discussion a little easier, I have listed the windows as a number of segments, which makes the actual numbers more obvious.

Slow start increases CWND by the maximum segment size for every packet for which it receives an acknowledgment. Because TCP receivers may, and typically do, acknowledge segments well before the full window has been sent by the sender, CWND grows at an exponential rate during slow start—a seemingly contradictory concept. Slow start gets its name from the fact that CWND has been set to a very low value at the beginning of the process, meaning it starts slowly, but slow start does cause CWND to grow quickly.

Figure 6-1 outlines the process of no acknowledgment received, slamming CWND to the size of a single segment, and slow start.

Figure 6-1 *Slamming the CWND Shut and the Slow Start Process After No Acknowledgment Received*

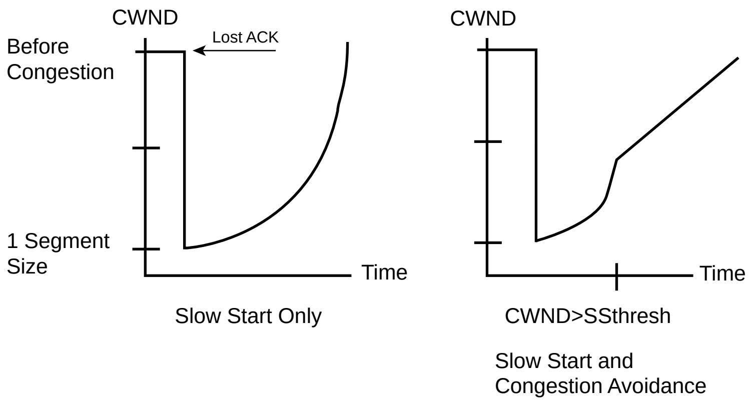


By increasing CWND when each acknowledgment is received, CWND actually increases at an exponential rate. So, Slow Start might be better called *slow start but fast recovery.*

Congestion avoidance is the second mechanism that dictates how quickly CWND increases after being lowered. As CWND grows, it begins to approach the original CWND value. If the original packet loss was a result of queue congestion, letting this TCP connection increase back to the original CWND may then induce the same congestion that caused the CWND to be lowered in the first place. Congestion avoidance just reduces the rate of increase for CWND as it approaches the previous CWND value. Once slow start has increased CWND to the value of SSTHRESH, which was set to 50 percent of the original CWND, congestion-avoidance logic replaces the slow start logic for increasing CWND. Congestion avoidance uses a formula that allows CWND to grow more slowly, essentially at a linear rate.

Figure 6-2 shows a graph of CWND with just slow start, and with slow start and congestion avoidance, after the sender times out waiting for an acknowledgment.

Figure 6-2 *Graphs of CWND with Slow Start and Congestion Avoidance*



Many people do not realize that the slow start process consists of a combination of the slow start algorithm and the congestion-avoidance algorithm. With slow start, CWND is lowered, but it grows quickly. With congestion avoidance, the CWND value grows more slowly as it approaches the previous CWND value.

In summary, UDP and TCP react to packet loss in the following ways:

- UDP senders do not reduce or increase sending rates as a result of lost packets.
- TCP senders do reduce their sending rates as a result of lost packets.
- TCP senders decide to use either the receiver window or the CWND, based on whichever is smaller at the time.
- TCP slow start and congestion avoidance dictate how fast the CWND rises after the window was lowered due to packet loss.

NOTE Depending on the circumstances, TCP sometimes halves CWND in reaction to lost packets, and in some cases it lowers CWND to one segment size, as was described in this first section. The more severe reaction of reducing the window to one segment size was shown in this section for a more complete description of slow start and congestion avoidance.

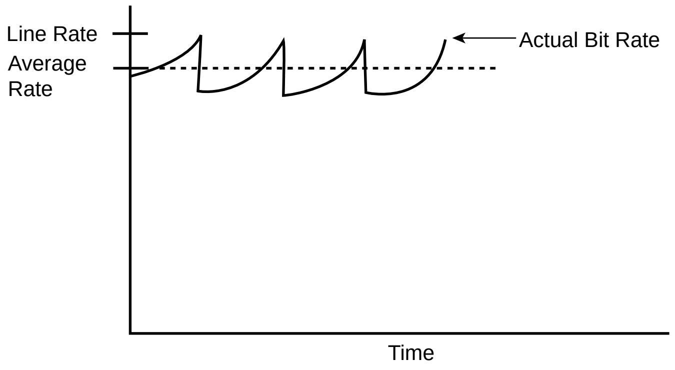
Tail Drop, Global Synchronization, and TCP Starvation

Tail drop occurs when a packet needs to be added to a queue, but the queue is full. Yes, tail drop is indeed that simple. However, tail drop results in some interesting behavior in real networks, particularly when most traffic is TCP based, but with some UDP traffic. Of course, the Internet today delivers mostly TCP traffic, because web traffic uses HTTP, and HTTP uses TCP.

The preceding section described the behavior of a single TCP connection after a single packet loss. Now imagine an Internet router, with 100,000 or more TCP connections running their traffic out of a high-speed interface. The amount of traffic in the combined TCP connections finally exceeds the output line rate, causing the output queue on the interface to fill, which in turn causes tail drop.

What happens to those 100,000 TCP connections after many of them have at least one packet dropped? The TCP connections reduce their CWND; the congestion in the queue abates; the various CWND values increase with slow start, and then with congestion avoidance. Eventually, however, as the CWND values of the collective TCP connections approach the previous CWND value, the congestion occurs again, and the process is repeated. When a large number of TCP connections experience near simultaneous packet loss, the lowering and growth of CWND at about the same time causes the TCP connections to *synchronize*. The result is called *global synchronization*. The graph in Figure 6-3 shows this behavior.

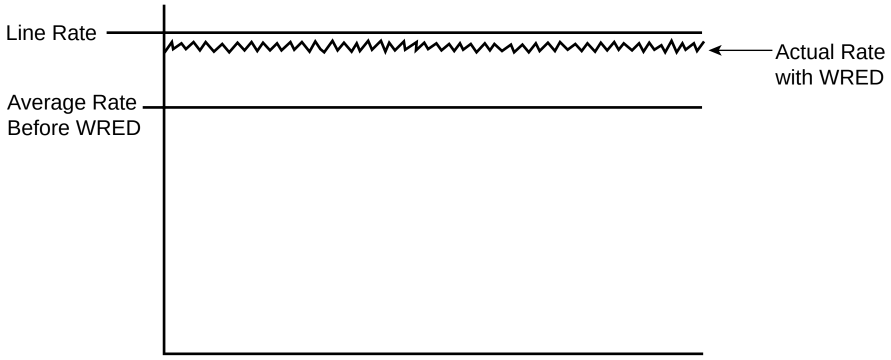
Figure 6-3 *Graph of Global Synchronization*



The graph shows the results of global synchronization. The router never fully utilizes the bandwidth on the link because the offered rate keeps dropping as a result of synchronization. Note that the overall rate does not drop to almost nothing because not all TCP connections happen to have packets drop when tail drop occurs, and some traffic uses UDP, which does not slow down in reaction to lost packets.

Weighted RED (WRED), when applied to the interface that was tail dropping packets, significantly reduces global synchronization. WRED allows the average output rates to approach line rate, with even more significant throughput improvements, because avoiding congestion and tail drops decreases the overall number of lost packets. Figure 6-4 shows an example graph of the same interface, after WRED was applied.

Figure 6-4 *Graph of Traffic Rates After the Application of WRED*



Another problem can occur if UDP traffic competes with TCP for bandwidth and queue space. Although UDP traffic consumes a much lower percentage of Internet bandwidth than TCP does, UDP can get a disproportionate amount of bandwidth as a result of TCP's reaction to packet loss. Imagine that on the same Internet router, 20 percent of the offered packets were UDP, and 80 percent TCP. Tail drop causes some TCP and UDP packets to be dropped; however, because the TCP senders slow down, and the UDP senders do not, additional UDP streams from the UDP senders can consume more and more bandwidth during congestion.

Taking the same concept a little deeper, imagine that several people crank up some UDP-based audio or video streaming applications, and that traffic also happens to need to exit this same congested interface. The interface output queue on this Internet router could fill with UDP packets. If a few high-bandwidth UDP applications fill the queue, a larger percentage of TCP packets might get tail dropped—resulting in further reduction of TCP windows, and less TCP traffic relative to the amount of UDP traffic.

The term "TCP starvation" describes the phenomena of the output queue being filled with larger volumes of UDP, causing TCP connections to have packets tail dropped. Tail drop does not distinguish between packets in any way, including whether they are TCP or UDP, or whether the flow uses a lot of bandwidth or just a little bandwidth. TCP connections can be starved for bandwidth because the UDP flows behave poorly in terms of congestion control. Flow-Based WRED (FRED), which is also based on RED, specifically addresses the issues related to TCP starvation, as discussed later in the chapter.

Random Early Detection (RED)

Random Early Detection (RED) reduces the congestion in queues by dropping packets so that some of the TCP connections temporarily send fewer packets into the network. Instead of waiting until a queue fills, causing a large number of tail drops, RED purposefully drops a percentage of packets before a queue fills. This action attempts to make the computers sending the traffic reduce the offered load that is sent into the network.

The name "Random Early Detection" itself describes the overall operation of the algorithm. RED randomly picks the packets that are dropped after the decision to drop some packets has been made. RED detects queue congestion early, before the queue actually fills, thereby avoiding tail drops and synchronization. In short, RED discards some randomly picked packets early, before congestion gets really bad and the queue fills.

NOTE IOS supports two RED-based tools: Weighted RED (WRED) and Flow-Based WRED (FRED). RED itself is not supported in IOS.

RED logic contains two main parts. RED must first detect when congestion occurs; in other words, RED must choose under what conditions it should discard packets. When RED decides to discard packets, it must decide how many to discard.

First, RED measures the average queue depth of the queue in question. RED calculates the average depth, and then decides whether congestion is occurring based on the average depth. RED uses the average depth, and not the actual queue depth, because the actual queue depth will most likely change much more quickly than the average depth. Because RED wants to avoid the effects of synchronization, it needs to act in a balanced fashion, not a jerky, sporadic fashion. Figure 6-5 shows a graph of the actual queue depth for a particular queue, compared with the average queue depth.

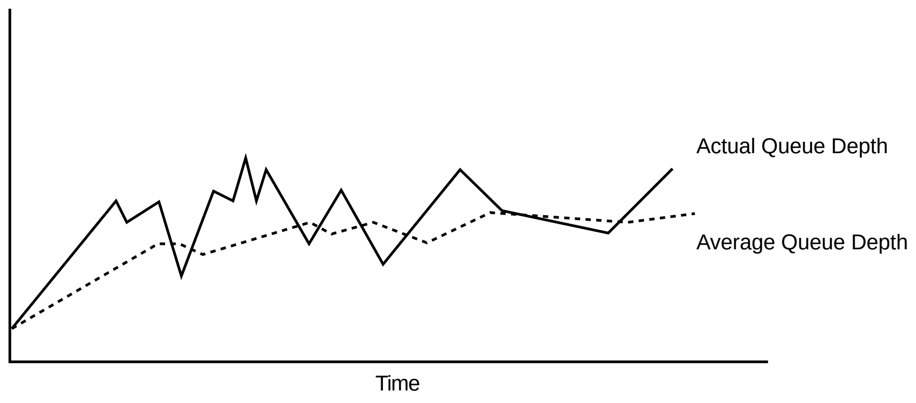
As seen in the graph, the calculated average queue depth changes more slowly than does the actual queue depth. RED uses the following algorithm when calculating the average queue depth:

$$\text{New average} = (\text{Old\_average} * (1 - 2^{-n})) + (\text{Current\_Q\_depth} * 2^{-n})$$

For you test takers out there, do not worry about memorizing the formula, but focus on the idea. WRED uses this algorithm, with a default for *n* of 9. This makes the equation read as follows:

$$\text{New average} = (\text{Old\_average} * .998) + (\text{Current\_Q\_depth} * .002)$$

Figure 6-5 *Graph of Actual Queue Depth Versus Average Queue Depth*



In other words, the current queue depth only accounts for .2 percent of the new average each time it is calculated. Therefore, the average changes slowly, which helps RED prevent overreaction to changes in the queue depth. When configuring WRED and FRED, you can change the value of *n* in this formula by setting the exponential weighting constant parameter. By making the exponential weighting constant smaller, you make the average change more quickly; by making it larger, the average changes more slowly.

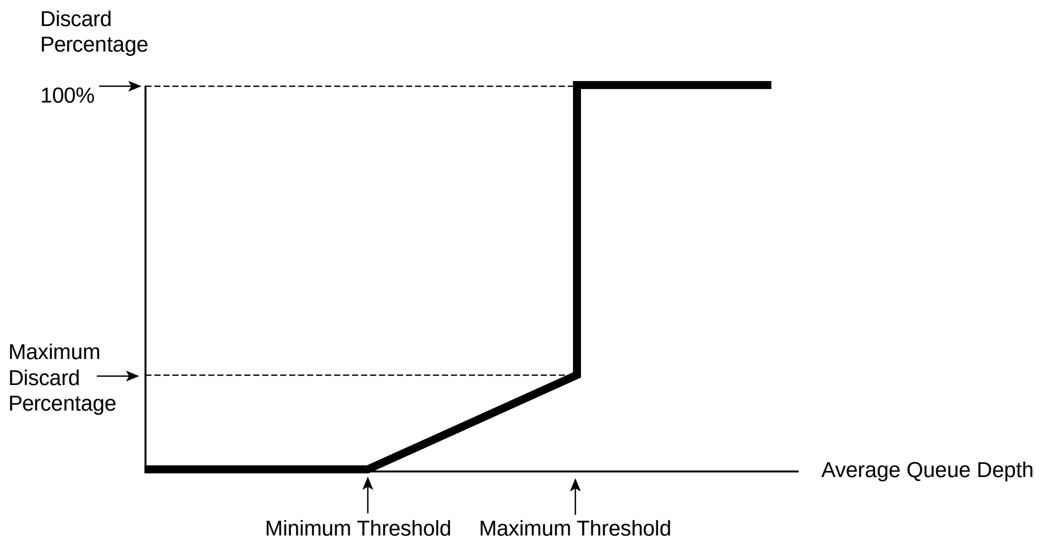
RED decides whether to discard packets by comparing the average queue depth to two thresholds, called the *minimum threshold* and *maximum threshold*. Table 6-2 describes the overall logic of when RED discards packets, as illustrated in Figure 6-6.

Table 6-2 *Three Categories of When RED Will Discard Packets and How Many*

Average Queue Depth Versus Thresholds	**Action**
Average < minimum threshold	No packets dropped.
Minimum threshold < average depth < maximum threshold	A percentage of packets dropped. Drop percentage increases from 0 to a maximum percent as the average depth moves from the minimum threshold to the maximum.
Average depth > maximum threshold	All new packets discarded similar to tail dropping.

As seen in Table 6-2 and Figure 6-6, RED does not discard packets when the average queue depth falls below the minimum threshold. When the average depth rises above the maximum threshold, RED discards all packets. In between the two thresholds, however, RED discards a percentage of packets, with the percentage growing linearly as the average queue depth grows. The core concept behind RED becomes more obvious if you notice that the maximum percentage of packets discarded is still much less than discarding all packets. Once again, RED wants to discard some packets, but not all packets. As congestion increases, RED discards a higher percentage of packets. Eventually, the congestion can increase to the point that RED discards all packets.

Figure 6-6 *RED Discarding Logic Using Average Depth, Minimum Threshold, and Maximum Threshold*



You can set the maximum percentage of packets discarded by WRED by setting the mark probability denominator (MPD) setting in IOS. IOS calculates the maximum percentage using the formula 1/MPD. For instance, an MPD of 10 yields a calculated value of 1/10, meaning the maximum discard rate is 10 percent.

Table 6-3 summarizes some of the key terms related to RED.

Table 6-3 *RED Terminology*

Term	Meaning
Actual queue depth	The actual number of packets in a queue at a particular point in time.
Average queue depth	Calculated measurement based on the actual queue depth and the previous average. Designed to adjust slowly to the rapid changes of the actual queue depth.
Minimum threshold	Compares this setting to the average queue depth to decide whether packets should be discarded. No packets are discarded if the average queue depth falls below this minimum threshold.
Maximum threshold	Compares this setting to the average queue depth to decide whether packets should be discarded. All packets are discarded if the average queue depth falls above this maximum threshold.

Table 6-3 *RED Terminology (Continued)*

Term	Meaning
Mark probability denominator	Used to calculate the maximum percentage of packets discarded when the average queue depth falls between the minimum and maximum thresholds.
Exponential weighting constant	Used to calculate the rate at which the average queue depth changes as compared with the current queue depth. The larger the number, the slower the change in the average queue depth.

The next two sections in this chapter cover WRED and FRED, including their respective configurations, as well as comparing RED, WRED, and FRED.

Weighted RED (WRED)

WRED behaves almost identically to RED, as described in the preceding section of this chapter. It calculates the average queue depth, and decides whether to discard packets, and what percentage of packets to discard, based on all the same variables as RED. The only real difference between the two is that WRED weights its behavior based on the IP precedence or IP differentiated services code point (DSCP) values of packets.

The other major concept that needs to be covered, before diving into WRED configuration, relates to where WRED can be enabled, and how it interoperates with queuing tools. Interestingly, although WRED can be enabled on an interface, it cannot be concurrently enabled along with any other queuing tool! When using Modular QoS command-line interface (MQC) to configure queuing, however, WRED can be used for individual class queues.

The following sections cover the following:

- How WRED weights packets
- When WRED can be enabled
- When WRED can be enabled to work with other queuing tools
- WRED configuration

How WRED Weights Packets

WRED bases its decisions about when to discard packets, and what percentage to discard, on the following four factors:

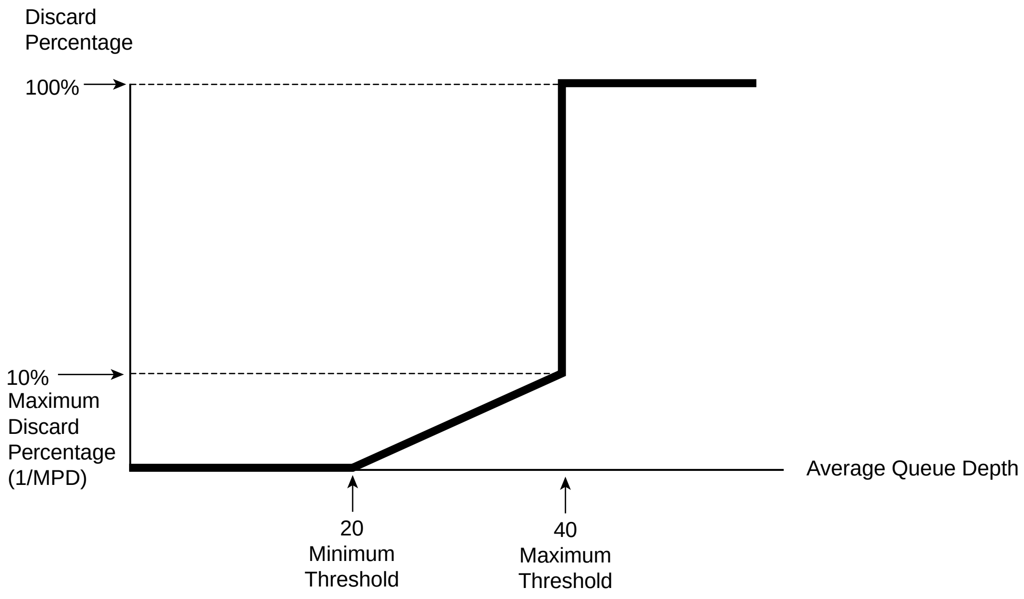
- The average queue depth
- The minimum threshold
- The maximum threshold
- The MPD

First, just like RED, WRED calculates the average queue depth. WRED then compares the average queue depth to the minimum and maximum thresholds to decide whether it should discard packets. If the average queue depth is between the two thresholds, WRED discards a percentage of the packets, with the percentage based on the MPD; if the average queue depth exceeds the maximum threshold, WRED discards all new packets.

To weight based on precedence or DSCP markings, WRED sets the minimum threshold, maximum threshold, and the MPD to different values per precedence or DSCP value. The average queue depth calculation, however, is not based on the precedence or DSCP value, but is instead calculated for all packets in the queue, regardless of the precedence or DSCP value.

An example of how WRED weights packets can help you make more sense out of how WRED behaves differently than RED. First, consider Figure 6-7, which happens to show the default settings for precedence 0.

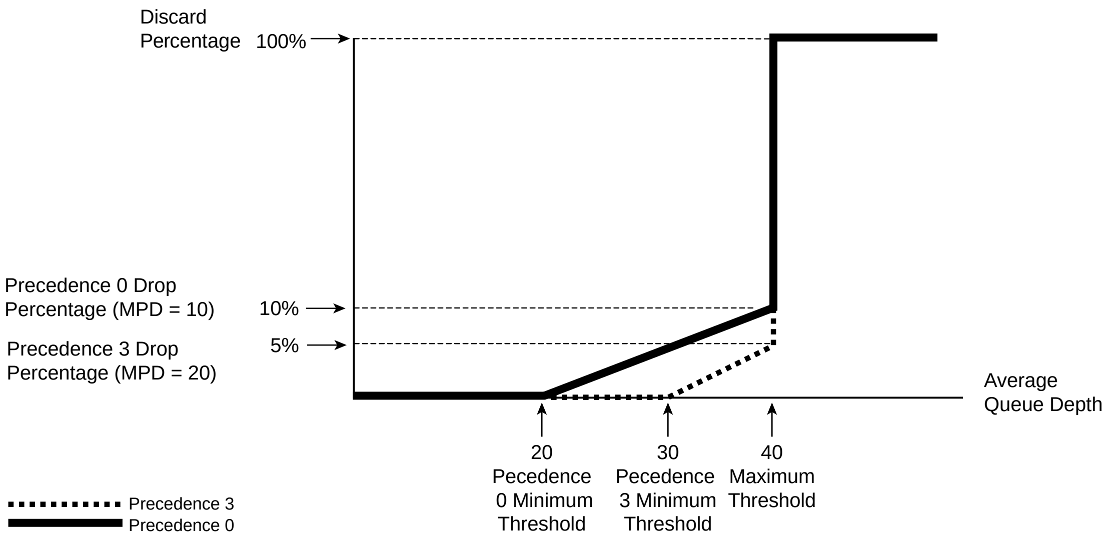
Figure 6-7 *Default WRED Settings for Precedence 0 for Thresholds and Percent to Discard*



WRED calculates the average queue depth just like RED, ignoring precedence, but it decides when to discard packets based on the precedence or DSCP value. Suppose, for instance, that the average queue depth just passed 20. For new precedence 0 packets that need to be placed into the queue, WRED begins discarding some packets. If the average queue depth continues to increase toward 40, WRED continues to discard precedence 0 packets, but more aggressively, up to a rate of 10 percent, when the average queue depth reaches 40. After the average queue depth passes 40, WRED discards all new precedence 0 packets. In fact, if all packets were precedence 0, RED and WRED would behave identically.

The real differences between RED and WRED can be seen with more than one IP precedence value. Figure 6-8 shows the default WRED settings for precedence 0, with some different settings for precedence 3 traffic. (The settings in the figure do not match WRED's precedence 3 defaults, which are listed later in this section.)

Figure 6-8 *Example WRED Settings for Precedences 0 and 3 for Thresholds and Discard Percent*



Suppose that the queue associated with the interface has a bunch of packets in it, marked with different precedence values, and the average queue depth just passed 20. For new precedence 0 packets that need to be placed into the queue, WRED begins discarding some precedence 0 packets, because the minimum threshold for precedence 0 is 20. WRED does not discard any precedence 3 packets, however, because the precedence 3 minimum threshold is 30. After the average queue depth reaches 30, WRED starts discarding precedence 3 packets as well. As the average queue depth reaches 40, precedence 0 packets are discarded at a rate approaching 10 percent, but precedence 3 packets are only discarded 5 percent of the time, because the MPD is set to 20, and 1/20 is 5 percent.

With the settings in this example, WRED discards precedence 0 packets earlier, and at a higher rate, as compared to precedence 3 packets. In short, the weighting feature of WRED just determines when WRED begins discarding a percentage of the packets (per-precedence minimum threshold), the maximum percentage discarded (based on per-precedence MPD), and the point at which WRED discards all packets of that precedence (based on the per-precedence maximum threshold).

IOS uses logical choices for the default settings for all WRED parameters. However, you can choose to override the parameters with configuration commands. Tables 6-4 and 6-5 list the IOS default values for minimum threshold, maximum threshold, and MPD with precedence-based WRED (Table 6-4) and DSCP-based WRED (Table 6-5).

Table 6-4 *Cisco IOS Software-Default Values for Precedence-Based WRED*

Precedence	Minimum Threshold	Maximum Threshold	Mark Probability Denominator	Calculated Maximum Percent Discarded
0	20	40	10	10%
1	22	40	10	10%
2	24	40	10	10%
3	26	40	10	10%
4	28	40	10	10%
5	31	40	10	10%
6	33	40	10	10%
7	35	40	10	10%
RSVP*	37	40	10	10%

* RSVP = Resource Reservation Protocol

Table 6-5 *Cisco IOS Software Default Values for DSCP-Based WRED**

DSCP**	Minimum Threshold	Maximum Threshold	Mark Probability Denominator	Calculated Maximum Percent Discarded
AF11, AF21, AF31, AF41	33	40	10	10%
AF12, AF22, AF32, AF42	28	40	10	10%
AF13, AF23, AF33, AF43	24	40	10	10%
EF	37	40	10	10%

* Stated values for IOS 12.2 Mainline software.

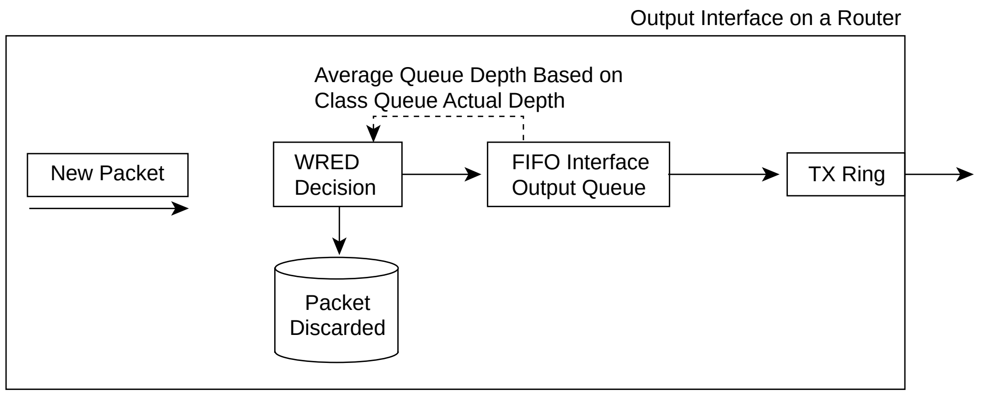
** Class selector DSCP values use the same values as their corresponding IP precedence values' settings.

Cisco IOS Software follows the suggested meaning of all DSCP values, including the fact that these four AF DSCP values should be given equal treatment. The last digit of the name of the AF DSCP value identifies the drop preference, with 3 being most likely to be dropped, and 1 being least likely to be dropped. Note, for instance, that the settings for assured forwarding (AF) DSCPs AF11, AF21, AF31, and AF41 are all identical. For the same reason, AF12, AF22, AF32, and AF42 have the same defaults, as do AF13, AF23, AF33, and AF43.

WRED and Queuing

WRED relies on the average queue depth concept, which calculates a rolling average of the queue depth of some queue. But which queue? Well, first consider a serial interface on a router, on which Weighted Fair Queuing (WFQ) is enabled by default. In this case, however, WFQ has been disabled, leaving a single first-in, first-out (FIFO) output queue on the interface. Figure 6-9 shows the basic idea.

Figure 6-9 *FIFO Output Queue and WRED Interaction*


Output Interface on a Router
Average Queue Depth Based on Class Queue Actual Depth
New Packet
WRED Decision
FIFO Interface Output Queue
TX Ring
Packet Discarded

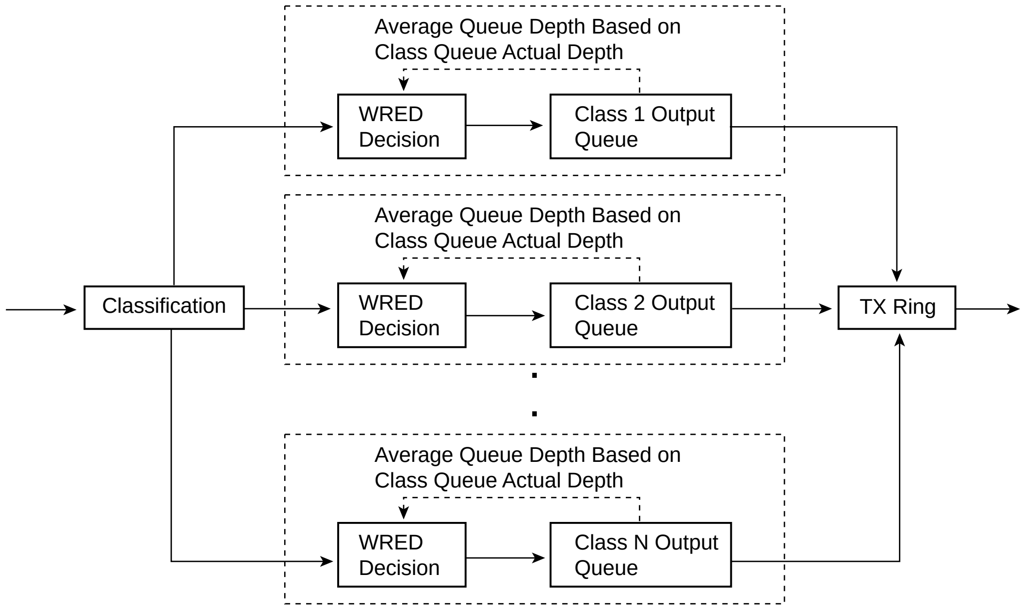
As was covered in depth in Chapter 4, “Congestion Management,” each interface has a TX Queue or TX Ring. If the TX Ring/TX Queue fills, IOS places new packets into the software queue(s) awaiting transmission. In this example, a single FIFO output queue is used, as shown. With WRED also enabled, WRED calculates the average queue depth of the single FIFO output queue. As new packets arrive, before being placed into the FIFO output queue, WRED logic decides whether the packet should be discarded, as described in detail earlier in this chapter.

With WRED enabled directly on a physical interface, IOS supports FIFO Queuing, and FIFO Queuing only! That fact certainly makes the explanation easier, because there is less to cover! So, WRED works just like Figure 6-9 when it is enabled directly on a physical interface, because WRED can only work with a single FIFO queue in that case.

You might recall that of all the queuing tools listed in Chapter 4, CBWFQ and Low Latency Queuing (LLQ, which is merely a variation of CBFWQ) are the only queuing tools that claim to be capable of using WRED. To use WRED with CBWFQ or LLQ, you need to configure

CBWFQ or LLQ as you normally would, and then enable WRED inside the individual classes as needed. Figure 6-10 illustrates an expanded diagram of CBWFQ, with the details that include WRED's part of the process.

Figure 6-10 *WRED with CBWFQ*


Average Queue Depth Based on Class Queue Actual Depth
WRED Decision
Class 1 Output Queue
Average Queue Depth Based on Class Queue Actual Depth
Classification
WRED Decision
Class 2 Output Queue
TX Ring
Average Queue Depth Based on Class Queue Actual Depth
WRED Decision
Class N Output Queue

As you recall, CBWFQ classifies traffic into various classes. Each class has a single FIFO queue inside the class, so WRED bases its average queue depth calculation on the actual depth of each per-class FIFO queue, respectively. In other words, a different instance of WRED operates on each of the FIFO queues in each class. WRED might be discarding packets aggressively in one congested class, without discarding any packets in a class that is not congested.

WRED can be enabled for some CBWFQ classes, and not for others. For instance, with LLQ, voice traffic is typically placed into the priority queue. Because voice is drop sensitive, and UDP based, it would be better not to just apply WRED to the voice class. Instead, you can apply WRED to the data classes that serve predominantly TCP flows. This way, WRED can be used to limit the queue congestion for the interface without performing drops on the voice traffic.

Now that you understand the basic operation of WRED, along with the meaning of the parameters that can be tuned, you can configure WRED.

WRED Configuration

WRED requires very little configuration if you want to take the IOS defaults for the various tunable settings, such as per-precedence and per-DSCP thresholds. If you want to change the defaults, the configuration details can become quite large.

This section begins with a table of configuration commands (Table 6-6) and **show** commands (Table 6-7), followed by three separate examples.

Table 6-6 *Command Reference for WRED*

Command	Mode and Function
random-detect [*dscp-based* | *prec-based*]	Interface or class configuration mode; enables WRED, specifying whether to react to precedence or DSCP.
random-detect [**attach** *group-name*]	Interface configuration mode; enables per-VC* WRED on ATM interfaces by referring to a **random-detect-group**.
random-detect-group *group-name* [*dscp-based* | *prec-based*]	Global configuration mode; creates a grouping of WRED parameters, which can be enabled on individual ATM VCs using the **random-detect attach** command.
random-detect precedence *precedence min-threshold max-threshold mark-prob-denominator*	Interface, class, or random-detect-group configuration modes; overrides default settings for the specified precedence, for minimum and maximum WRED thresholds, and for percentage of packets discarded.
random-detect dscp *dscpvalue min-threshold max-threshold* [*mark-probability-denominator*]	Interface, class, or random-detect-group configuration modes; overrides default settings for the specified DSCP, for minimum and maximum WRED thresholds, and for the percentage of packets discarded.
random-detect exponential-weighting-constant *exponent*	Interface, class, or random-detect-group configuration modes; overrides default settings for exponential weighting constant. Lower numbers make WRED react quickly to changes in queue depth; higher numbers make WRED react less quickly.

* VC = virtual circuit

Table 6-7 *Exec Command Reference for WRED*

Command	Function
show *queue* *interface-name interface-number* [**vc** [*vpi*/] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing random-detect [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]	Lists configuration and statistical information about the queuing tool on an interface.
show interfaces	Mentions whether WRED has been enabled on the interface
show interface random-detect	Lists information about WRED when distributed WRED is running on a VIP* interface
show policy-map [**interface** *interface-name interface-number*]	Lists WRED information when it is enabled inside an MQC policy map

* VIP = Versatile Interface Processor

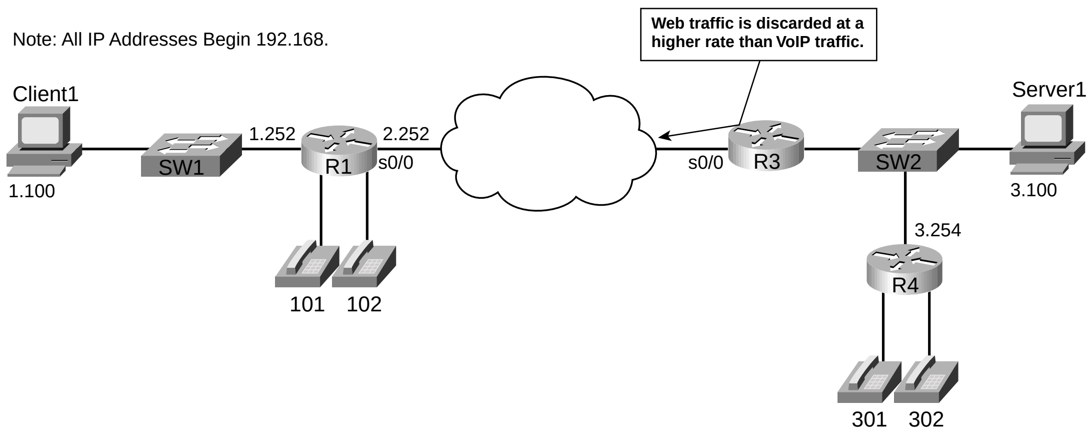
In the first example, R3 enables WRED on its S0/0 interface. WRED treats packets differently based on the IP precedence value, which has been marked with CB marking as the packets enter R3's E0/0 interface. The marking logic performed by CB marking is as follows:

- VoIP payload—DSCP EF
- HTTP traffic for web pages with "important" in the URL—DSCP AF21
- HTTP traffic for web pages with "not-so" in the URL—DSCP AF23
- All other—DSCP default

To generate traffic in this network, two voice calls will be made between the analog phones attached to R1 and R4. Multiple web browsers will load the standard page (this is the same page we have used in other chapters in this book) with two TCP connections created by each browser—one to get a file with the word "important" in it, and the other getting a file with "not-so" in it. An FTP download of a large file will also be initiated from the Server to Client1.

Example 6-1 shows the basic configuration and **show** commands output. Only the required commands and parameters have been used, with defaults for all other settings. The example uses the familiar network diagram, as repeated in Figure 6-11.

Figure 6-11 *Sample Network for All WRED Examples—Configuration on R3*


Note: All IP Addresses Begin 192.168.
Web traffic is discarded at a higher rate than VoIP traffic.
Client1
1.100
SW1
1.252
R1
2.252
s0/0
101
102
s0/0
R3
SW2
Server1
3.100
3.254
R4
301
302

Example 6-1 *WRED Default Configuration, R3, S0/0*

```
R3#show running-config
!
hostname R3
!
no ip domain-lookup
ip host r4 192.168.3.254
ip host r2 192.168.23.252
ip host r1 192.168.1.251
!
ip cef
!
class-map match-all voip-rtp
  match ip rtp 16384 16383
class-map match-all http-impo
  match protocol http url "*important*"
class-map match-all http-not
  match protocol http url "*not-so*"
class-map match-all class-default
  match any
!
!
policy-map laundry-list
  class voip-rtp
   set ip dscp ef
  class http-impo
   set ip dscp af21
  class http-not
   set ip dscp af23
```

continues

Example 6-1 *WRED Default Configuration, R3, S0/0 (Continued)*

```
  class class-default
   set ip dscp default
!
call rsvp-sync
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 ip nbar protocol-discovery
 half-duplex
 service-policy input laundry-list
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 random-detect
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
! Lines omitted for brevity.
!
R3#show queueing interface serial 0/0
Interface Serial0/0 queueing strategy: random early detection (WRED)
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 37

class    Random drop      Tail drop    Minimum Maximum  Mark
         pkts/bytes       pkts/bytes    thresh  thresh  prob
0         1776/315688      1012/179987      20      40  1/10
1            0/0              0/0           22      40  1/10
2            5/4725          16/17152       24      40  1/10
3            0/0              0/0           26      40  1/10
4            0/0              0/0           28      40  1/10
5            0/0              0/0           31      40  1/10
6            0/0              0/0           33      40  1/10
7            0/0              0/0           35      40  1/10
rsvp         0/0              0/0           37      40  1/10

R3#show queue s 0/0
Output queue for Serial0/0 is 57/0

Packet 1, linktype: ip, length: 64, flags: 0x88
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x053E, ttl: 253,
  TOS: 184 prot: 17, source port 18378, destination port 17260
```

Example 6-1 *WRED Default Configuration, R3, S0/0 (Continued)*

```
    data: 0x47CA 0x436C 0x0028 0x0000 0x8012 0x3F73 0x4C7E
          0x8D44 0x18D1 0x03FE 0xFC77 0xA2A7 0x35A2 0x54E7

Packet 2, linktype: ip, length: 64, flags: 0x88
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x0545, ttl: 253,
  TOS: 184 prot: 17, source port 16640, destination port 17178
    data: 0x4100 0x431A 0x0028 0x0000 0x8012 0x6330 0x21B4
          0x82AF 0x05C9 0x03FE 0x1448 0x8706 0xAFD9 0xD364
!
! Output omitted for brevity.
!

R3#show interfaces s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 20/255, rxload 4/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  591, LMI stat recvd 591, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 0/64, broadcasts sent/dropped 2726/0, interface broadcasts 252
2
  Last input 00:00:02, output 00:00:00, output hang never
  Last clearing of "show interface" counters 01:38:28
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 4391
  Queueing strategy: random early detection(RED)
  30 second input rate 29000 bits/sec, 58 packets/sec
  30 second output rate 122000 bits/sec, 91 packets/sec
     23863 packets input, 1535433 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     2 input errors, 0 CRC, 2 frame, 0 overrun, 0 ignored, 0 abort
     36688 packets output, 5638653 bytes, 0 underruns
     0 output errors, 0 collisions, 4 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

The WRED part of the configuration is quite short. The configuration shows the **random-detect** interface subcommand under serial 0/0. As you will see later, the command actually disables WFQ if configured. The rest of the highlighted configuration commands show the CB marking configuration, which implements the functions listed before the example. (For more information about CB marking, see Chapter 3, "Classification and Marking.")

After the configuration, the **show queueing interface serial 0/0** command output lists the WRED settings, and the statistics for each precedence value. The defaults for the exponential weighting constant, and the per-precedence defaults of minimum threshold, maximum threshold, and MPD are all listed. In addition, the command lists statistics for bytes/packets dropped by WRED, per precedence value. For those of you who did not memorize the DSCP values, you may not be able to correlate the DSCP values set by CB marking, and the precedence values interpreted by WRED. WRED just looks at the first 3 bits of the IP ToS byte when performing precedence-based WRED. So, DSCP best effort (BE) equates to precedence 0, DSCP AF21 and AF23 both equate to precedence 2, and DSCP expedited forwarding (EF) equates to precedence 5.

The **show queueing** command also lists a column of statistics for tail drop as well as random drops. In this example, WRED has dropped several packets, and the queue has filled, causing tail drops, as shown with the nonzero counters for random drops and tail drops in the **show queueing** command output.

The **show queue serial 0/0** command lists the same type of information seen in earlier chapters. However, this command lists one particularly interesting item relating to WRED in the first line of the command output. The actual queue depth for the single FIFO queue used with WRED is listed at 57 entries in this particular example. The earlier **show queueing** command lists an average queue depth of 37 just instants before. These two numbers just give us a small reminder that WRED decides to drop based on average queue depth, as opposed to actual queue depth.

Finally, the **show interfaces** command at the end of the example reminds us that WRED does not work with any other queuing method directly on the interface. The command uses the statement “Queueing strategy: random early detection (RED)” to remind us of that fact. WRED uses a single FIFO queue, and measures its average queue depth based on the queue depth of the FIFO queue.

The second WRED configuration example uses WRED on R3’s S0/0 interface again, but this time with DSCP WRED, and a few changes to the defaults. In fact, Example 6-2 just shows the changed configuration, with most of the configuration staying the same. For instance, the same CB marking configuration is used to mark the traffic, so the details are not repeated in the example. The example uses the familiar network diagram that was also used in the preceding example.

Example 6-2 *DSCP-Based WRED on R3 S0/0*

```
R3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface serial 0/0
R3(config-if)#random-detect dscp-based
R3(config-if)#random-detect dscp af21 50 60
R3(config-if)#random-detect dscp af23 20 30
R3(config-if)#random-detect ?
  dscp                            parameters for each dscp value
  dscp-based                      Enable dscp based WRED on an interface
  exponential-weighting-constant  weight for mean queue depth calculation
  flow                            enable flow based WRED
  prec-based                      Enable prec based WRED on an interface
```

Example 6-2 *DSCP-Based WRED on R3 S0/0 (Continued)*

```
  precedence                      parameters for each precedence value
  <cr>
R3(config-if)#random-detect exponential-weighting-constant 5
R3(config-if)#^Z

R3#show queue serial 0/0
Output queue for Serial0/0 is 37/0

Packet 1, linktype: ip, length: 64, flags: 0x88
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x0545, ttl: 253,
  TOS: 184 prot: 17, source port 16640, destination port 17178
    data: 0x4100 0x431A 0x0028 0x0000 0x8012 0xAB15 0x21E1
          0x71CF 0x05C9 0x03FE 0x7AA3 0x770B 0x2408 0x8264

Packet 2, linktype: ip, length: 64, flags: 0x88
  source: 192.168.3.254, destination: 192.168.2.251, id: 0x053E, ttl: 253,
  TOS: 184 prot: 17, source port 18378, destination port 17260
    data: 0x47CA 0x436C 0x0028 0x0000 0x8012 0x8759 0x4CAB
          0x7D04 0x18D1 0x03FE 0xDC15 0x3E4A 0x4E92 0x5447

R3#show queueing interface s 0/0
Interface Serial0/0 queueing strategy: random early detection (WRED)
    Exp-weight-constant: 5 (1/32)
    Mean queue depth: 38

dscp     Random drop      Tail drop    Minimum Maximum  Mark
         pkts/bytes       pkts/bytes    thresh  thresh  prob
af11        0/0              0/0          33      40  1/10
af12        0/0              0/0          28      40  1/10
af13        0/0              0/0          24      40  1/10
af21        8/9904          18/21288      50      60  1/10
af22        0/0              0/0          28      40  1/10
af23       13/18252         33/34083      20      30  1/10
af31        0/0              0/0          33      40  1/10
af32        0/0              0/0          28      40  1/10
af33        0/0              0/0          24      40  1/10
af41        0/0              0/0          33      40  1/10
af42        0/0              0/0          28      40  1/10
af43        0/0              0/0          24      40  1/10
cs1         0/0              0/0          22      40  1/10
cs2         0/0              0/0          24      40  1/10
cs3         0/0              0/0          26      40  1/10
cs4         0/0              0/0          28      40  1/10
cs5         0/0              0/0          31      40  1/10
cs6         0/0              0/0          33      40  1/10
cs7         0/0              0/0          35      40  1/10
ef          0/0              0/0          37      40  1/10
rsvp        0/0              0/0          37      40  1/10
default    16/16254         20/23216      20      40  1/10
```

The configuration begins with a change from precedence-based WRED to DSCP-based WRED using the **random-detect dscp-based** interface subcommand. The **random-detect dscp af21 50 60** changes the default minimum and maximum thresholds for AF21 to 50 and 60, respectively, with the **random-detect dscp af23 20 30** changing these same values for AF23. In addition, although Cisco does not recommend changing the exponential weighting constant, the configuration does offer an example of the syntax with the **random-detect exponential-weighting-constant 5** command. By setting it to a smaller number than the default (9), WRED will more quickly change the average queue depth calculation, more quickly reacting to changes in the queue depth.

The command output from the various **show** commands do not differ much compared to when DSCP-based WRED is enabled. The format now includes DSCP values rather than precedence values, as you may notice with the counters that point out drops for both AF21 and AF23, which were previously both treated as precedence 2.

WRED suffers from the lack of concurrent queuing tool support on an interface. However, WRED can be enabled inside a CBWFQ class, operating on the queue for the class, effectively enabling WRED concurrently with CBWFQ. The final WRED example shows a configuration for WRED using LLQ.

The last WRED configuration example repeats base configuration similar to one of the CBWFQ examples from Chapter 4. Voice, HTTP, and FTP traffic compete for the same bandwidth, with WRED applied per-class for the two HTTP classes and one FTP class. Note that because voice traffic is drop sensitive, WRED is not enabled for the low-latency queue. Because WRED can be enabled per class in conjunction with CBWFQ, WRED calculates average queue depth based on the per-class queue.

The criteria for each type of traffic is as follows:

- R3's S0/0 is clocked at 128 kbps.
- VoIP payload is marked with DSCP EF using CB marking on ingress to R3 E0/0. On egress of R3's S0/0 interface, DSCP EF traffic is placed in its own queue, *without WRED*. This class gets 58 kbps.
- Any HTTP traffic whose URL contains the string "important" anywhere in the URL is marked with AF21 using CB marking on ingress to R3 E0/0. On egress of R3's S0/0 interface, DSCP AF21 traffic is placed in its own queue, *with WRED*. This class gets 20 kbps.
- Any HTTP traffic whose URL contains the string "not-so" anywhere in the URL is marked with AF23 using CB marking on ingress to R3 E0/0. On egress of R3's S0/0 interface, DSCP AF23 traffic is placed in its own queue, *with WRED*. This class gets 8 kbps.
- All other traffic is marked with DSCP BE using CB marking on ingress to R3 E0/0. On egress of R3's S0/0 interface, DSCP BE traffic is placed in its own queue, *with WRED*. This class gets 20 kbps.

Example 6-3 lists the configuration and **show** commands used when WRED is enabled in LLQ classes dscp-af21, dscp-af23, and class-default.

Example 6-3 *WRED Used in LLQ Classes dscp-af21, dscp-af23, and class-default*

```
R3#show running-config
Building configuration...
!
!Portions omitted for brevity
!
ip cef
!
! The following classes are used in the LLQ configuration applied to S0/0
!
class-map match-all dscp-ef
  match ip dscp ef
class-map match-all dscp-af21
  match ip dscp af21
class-map match-all dscp-af23
  match ip dscp af23
!
! The following classes are used on ingress for CB marking
!
class-map match-all http-impo
  match protocol http url "*important*"
class-map match-all http-not
  match protocol http url "*not-so*"
class-map match-all class-default
  match any
class-map match-all voip-rtp
  match ip rtp 16384 16383
!
! Policy-map laundry-list creates CB marking configuration, used on
! ingress on E0/0
!
policy-map laundry-list
  class voip-rtp
   set ip dscp ef
  class http-impo
   set ip dscp af21
  class http-not
   set ip dscp af23
  class class-default
   set ip dscp default
!


! Policy-map queue-on-dscp creates LLQ configuration, with WRED
! inside three classes
!
```

continues

Example 6-3 *WRED Used in LLQ Classes dscp-af21, dscp-af23, and class-default (Continued)*

```
policy-map queue-on-dscp
  class dscp-ef
    priority 58
  class dscp-af21
   bandwidth 20
   random-detect dscp-based
  class dscp-af23
   bandwidth 8
   random-detect dscp-based
  class class-default
   fair-queue
   random-detect dscp-based
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 ip nbar protocol-discovery
 half-duplex
 service-policy input laundry-list
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 bandwidth 128
 no ip address
 encapsulation frame-relay
 load-interval 30
 max-reserved-bandwidth 85
 service-policy output queue-on-dscp
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
R3#show policy-map interface serial 0/0

 Serial0/0

  Service-policy output: queue-on-dscp

    Class-map: dscp-ef (match-all)
      46437 packets, 2971968 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: ip dscp ef
      Weighted Fair Queueing
        Strict Priority
        Output Queue: Conversation 264
        Bandwidth 58 (kbps) Burst 1450 (Bytes)
```

Example 6-3 *WRED Used in LLQ Classes dscp-af21, dscp-af23, and class-default (Continued)*

```
          (pkts matched/bytes matched) 42805/2739520
          (total drops/bytes drops) 0/0

      Class-map: dscp-af21 (match-all)
        2878 packets, 3478830 bytes
        30 second offered rate 76000 bps, drop rate 0 bps
        Match: ip dscp af21
        Weighted Fair Queueing
          Output Queue: Conversation 266
          Bandwidth 20 (kbps)
          (pkts matched/bytes matched) 2889/3494718
          (depth/total drops/no-buffer drops) 11/26/0
           exponential weight: 9
           mean queue depth: 5

   dscp     Transmitted       Random drop      Tail drop    Minimum Maximum  Mark
             pkts/bytes        pkts/bytes       pkts/bytes    thresh  thresh  prob
   af11         0/0              0/0              0/0           32      40  1/10
   af12         0/0              0/0              0/0           28      40  1/10
   af13         0/0              0/0              0/0           24      40  1/10
   af21      2889/3494718        8/9904         18/21288        32      40  1/10
   af22         0/0              0/0              0/0           28      40  1/10
   af23         0/0              0/0              0/0           24      40  1/10
   af31         0/0              0/0              0/0           32      40  1/10
   af32         0/0              0/0              0/0           28      40  1/10
   af33         0/0              0/0              0/0           24      40  1/10
   af41         0/0              0/0              0/0           32      40  1/10
   af42         0/0              0/0              0/0           28      40  1/10
   af43         0/0              0/0              0/0           24      40  1/10
   cs1          0/0              0/0              0/0           22      40  1/10
   cs2          0/0              0/0              0/0           24      40  1/10
   cs3          0/0              0/0              0/0           26      40  1/10
   cs4          0/0              0/0              0/0           28      40  1/10
   cs5          0/0              0/0              0/0           30      40  1/10
   cs6          0/0              0/0              0/0           32      40  1/10
   cs7          0/0              0/0              0/0           34      40  1/10
   ef           0/0              0/0              0/0           36      40  1/10
   rsvp         0/0              0/0              0/0           36      40  1/10
   default      0/0              0/0              0/0           20      40  1/10



      Class-map: dscp-af23 (match-all)
        1034 packets, 1250984 bytes
        30 second offered rate 32000 bps, drop rate 0 bps
        Match: ip dscp af23
        Weighted Fair Queueing
          Output Queue: Conversation 267
          Bandwidth 8 (kbps)
          (pkts matched/bytes matched) 1047/1266140
```

continues

Example 6-3 *WRED Used in LLQ Classes dscp-af21, dscp-af23, and class-default (Continued)*

```
        (depth/total drops/no-buffer drops) 11/46/0
         exponential weight: 9
         mean queue depth: 5

dscp     Transmitted       Random drop      Tail drop    Minimum Maximum  Mark
          pkts/bytes        pkts/bytes       pkts/bytes    thresh  thresh  prob
af11        0/0               0/0              0/0           32      40  1/10
af12        0/0               0/0              0/0           28      40  1/10
af13        0/0               0/0              0/0           24      40  1/10
af21        0/0               0/0              0/0           32      40  1/10
af22        0/0               0/0              0/0           28      40  1/10
af23      1047/1266140      13/18252         33/34083        24      40  1/10
af31        0/0               0/0              0/0           32      40  1/10
af32        0/0               0/0              0/0           28      40  1/10
af33        0/0               0/0              0/0           24      40  1/10
af41        0/0               0/0              0/0           32      40  1/10
af42        0/0               0/0              0/0           28      40  1/10
af43        0/0               0/0              0/0           24      40  1/10
cs1         0/0               0/0              0/0           22      40  1/10
cs2         0/0               0/0              0/0           24      40  1/10
cs3         0/0               0/0              0/0           26      40  1/10
cs4         0/0               0/0              0/0           28      40  1/10
cs5         0/0               0/0              0/0           30      40  1/10
cs6         0/0               0/0              0/0           32      40  1/10
cs7         0/0               0/0              0/0           34      40  1/10
ef          0/0               0/0              0/0           36      40  1/10
rsvp        0/0               0/0              0/0           36      40  1/10
default     0/0               0/0              0/0           20      40  1/10



    Class-map: class-default (match-any)
      847 packets, 348716 bytes
      30 second offered rate 2000 bps, drop rate 0 bps
      Match: any
      Weighted Fair Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 256
        (total queued/total drops/no-buffer drops) 0/0/0
         exponential weight: 9

dscp     Transmitted       Random drop      Tail drop    Minimum Maximum  Mark
          pkts/bytes        pkts/bytes       pkts/bytes    thresh  thresh  prob
af11        0/0               0/0              0/0           32      40  1/10
af12        0/0               0/0              0/0           28      40  1/10
af13        0/0               0/0              0/0           24      40  1/10
af21        0/0               0/0              0/0           32      40  1/10
af22        0/0               0/0              0/0           28      40  1/10
af23        0/0               0/0              0/0           24      40  1/10
af31        0/0               0/0              0/0           32      40  1/10
af32        0/0               0/0              0/0           28      40  1/10
af33        0/0               0/0              0/0           24      40  1/10
```

Example 6-3 *WRED Used in LLQ Classes dscp-af21, dscp-af23, and class-default (Continued)*

```
af41         0/0          0/0          0/0          32      40  1/10
af42         0/0          0/0          0/0          28      40  1/10
af43         0/0          0/0          0/0          24      40  1/10
cs1          0/0          0/0          0/0          22      40  1/10
cs2          0/0          0/0          0/0          24      40  1/10
cs3          0/0          0/0          0/0          26      40  1/10
cs4          0/0          0/0          0/0          28      40  1/10
cs5          0/0          0/0          0/0          30      40  1/10
cs6          0/0          0/0          0/0          32      40  1/10
cs7          0/0          0/0          0/0          34      40  1/10
ef           0/0          0/0          0/0          36      40  1/10
rsvp         0/0          0/0          0/0          36      40  1/10
default     59/767        0/0          0/0          20      40  1/10
```

The example lists a large configuration, but only a small amount pertains to WRED. Two sets of class maps have been configured—one set is used by the CB marking policy called laundry-list, and the other set is used by the LLQ policy map called queue-on-dscp. In **policy-map queue-on-dscp**, inside classes dscp-af21, dscp-af23, and class-default, the **random-detect dscp-based** command enables WRED. These three **random-detect** commands are highlighted in the **show running-config** output in the example.

Also note that WRED is not enabled on interface serial 0/0 in this configuration, because WRED applies to the output queues used by each class. Because WRED is not enabled on the main interface, to see statistics for WRED, you must use the **show policy-map interface** command. This command in the example lists WRED statistics inside each class in which WRED has been enabled. For the classes in which WRED is not enabled, such as the dscp-ef class, no additional WRED statistical information is listed. The default values for exponential weighting constant, and the per-DSCP defaults for minimum threshold, maximum threshold, and MPD are all listed in the command output.

WRED Summary

WRED provides a valuable tool for managing congestion in queues. Cisco IOS uses defaults that conform to the DiffServ Assured Forwarding conventions, which reduce the likelihood that you will need to configure thresholds for WRED. WRED can be particularly effective when used with MQC-based queuing tools, but when enabled directly on an interface, WRED has the unfortunate side effect of disallowing other queuing tools to be used. Table 6-8 lists some of WRED's key points.

Table 6-8 *WRED Feature Summary*

Feature	WRED
Discards packets to avoid congestion	Yes
Can be enabled on physical interface concurrently with a queuing tool	No
Can be combined with CBWFQ or LLQ policy map	Yes
Bases drop decision, at least in part, on different thresholds per precedence or DSCP value	Yes

Flow-Based WRED (FRED)

FRED uses flows to identify streams of traffic. A *flow* is a unidirectional sequence of packets between a source and destination that share the same protocol and transport layer information. The following seven fields are used to determine whether the received packet is a part of an existing flow:

- Source IP address
- Source port number
- Destination IP address
- Destination port number
- Layer 3 protocol
- IP precedence/DSCP marking
- Interface the packet is received on

FRED overcomes the problem of TCP starvation described in the first section of this chapter. When nonadaptive flows (UDP) compete with adaptive flows (TCP), WRED discards packets from all the flows. However, only the TCP flows adapt, sending less traffic into the network, but UDP flows do not slow down. Queues can become filled with packets from large-volume UDP flows, causing the TCP flows to get progressively less queue space. This phenomenon is called *TCP starvation*.

Both WRED and FRED discard packets before tail drop is required, hoping to get the senders of the traffic to slow down, thereby reducing congestion. FRED, however, adds the goal of stopping TCP starvation by watching for *hungry* UDP flows that try to use too much of a queue. FRED aggressively discards these hungry UDP flows, termed *nonadapative flows*, preventing these flows from consuming too much of a queue.

To recognize nonadaptive flows, FRED classifies packets into flows. Then FRED classifies each flow into one of three FRED flow types, as listed in Table 6-9.

Table 6-9 *FRED Flow Types*

Flow Type	Description	Transport Protocol	FRED Discard Policy
Robust	Adapts to lost packets by slowing down the rate of sending packets	TCP	Moderate discard rates
Fragile	Does not adapt to lost packets by slowing down, but the number of packets sent is not excessive	UDP	Low discard rates
Nonadaptive	Does not adapt to lost packets by slowing down, and the number of packets sent is excessive	UDP	High discard rates

Keep in mind that FRED specifically wants to defeat the UDP flows that consume too many queue entries. FRED still drops some packets from UDP flows when congestion occurs, based on normal WRED logic. For UDP flows, FRED simply decides which flows are taking too much of the queue, classifies these flows as nonadaptive, and discards packets in those flows more aggressively. For other UDP flows that FRED believes are not sending too many packets (*fragile flows*) FRED discards their packets less frequently.

NOTE UDP does not adapt to packet loss, so by definition, all UDP flows are *nonadaptive*. However, FRED uses the term "nonadaptive" to categorize those flows that are currently using too much of the queue.

The key to understanding how FRED works is to understand how FRED determines whether a particular UDP flow is fragile or nonadaptive. A couple of simple examples make the logic much more obvious. First, suppose FRED is enabled on a router's S0/0 interface, and the interface FIFO output queue has 40 queue entries maximum. At a particular point in time, 10 flows exist, called Flow 1, Flow 2, and so on. (Keep in mind that FRED only supports FIFO Queuing.) With a maximum queue depth of 40, and 10 flows, you can think of each flow's fair share of the queue space to be 40/10, or 4, in this case. FRED multiplies this fair share by a scaling factor, which defaults to 4. The scaling factor is used so that each flow has some capability to burst. This final number, 16 in this case, is the dividing line between fragile flows and nonadaptive flows.

Suppose, for instance, that Flow 1 and Flow 2 both use UDP. Flow 1 has 3 packets in the FIFO output queue, and Flow 2 has 20 packets in the queue. At first glance, it seems that Flow 2 should be considered a nonadaptive flow, because it has many packets in the queue. The calculated value of 4 * 40/10 (scaling factor * maximum queue length/number of flows), or 16, is greater than the number of packets in the queue that are part of Flow 1 (3 packets in the queue).

Therefore, FRED considers Flow 1 to be a fragile flow. Similarly, the calculated value of 16 is smaller than the number of packets in the queue that are part of Flow 2 (20 packets in the queue), so FRED considers Flow 2 to be nonadaptive.

The first example hides one subtlety in how FRED decides which UDP flows are fragile, and which are nonadaptive. Consider the same example, but suppose now that only five flows exist. The formula works out to be 4 * 40/5 = 32. (Notice that the only part of the formula that changes over time is the number of flows; the scaling factor remains static, as does the maximum queue length.) As the number of flows decreases, the threshold that determines whether a flow is nonadaptive increases. If Flow 2 still has 20 packets in the queue, Flow 2 would now be considered to be a fragile flow, with packets discarded less aggressively.

After FRED decides which flows are robust, fragile, and nonadaptive, FRED can apply WRED-like logic to decide whether to discard any packets at all, and if so, what percentage of the packets. Frankly, the published details on how FRED determines the flows are sketchy at best. For you QoS exam takers, the following details about how FRED discards packets for the three types of flows is unlikely to be covered, because the coverage in the corresponding courses is also sketchy, or not even mentioned.

For robust and fragile flows, FRED acts just like WRED in terms of packet discard. FRED still uses the per-precedence or per-DSCP minimum threshold, maximum threshold, and MPD to determine when to discard packets, and how many. You can override the default values for each precedence or DSCP value through configuration, just like with WRED.

For nonadaptive flows, FRED lowers the maximum threshold. By doing so, FRED increases the packet discard rate more quickly. More importantly, lowering the maximum threshold for packets in the flow causes FRED to discard all packets for these nonadaptive flows more quickly than for fragile and robust flows, essentially preventing these flows from consuming the entire queue.

To get a full sense of what happens to nonadaptive flows, consider the following example. For nonadaptive flows, FRED reduces the maximum threshold used for the flow by half of the difference between the maximum and minimum thresholds. FRED and WRED use defaults for precedence 0 as follows: minimum threshold 20, maximum threshold 40, drop percentage 10 percent (MPD=10). FRED reduces the maximum threshold for a precedence 0 nonadaptive flow to 30, because the difference between the maximum and minimum is 20, and half of that is 10. By reducing the maximum threshold by half of the difference between the minimum and maximum, FRED increases the packet discard rate more quickly as the average queue depth nears 30. Most importantly, FRED discards all packets for these nonadaptive precedence 0 flows when the average queue depth reaches 30, preventing these flows from taking over the entire queue.

Table 6-10 lists the terms used by FRED to describe these processes.

Table 6-10 *FRED Terminology*

Term	Definition
Average per-flow queue size	A calculated value, based on the formula maximum queue size/ number of active flows.
Active flow	A flow that currently has packets in the queue.
Maximum per-flow queue size	A calculated value, based on the formula (average queue size * scaling factor). (This same formula is used in the previous example that results in an answer of 16.) This value is used to determine which flows are fragile, and which are nonadaptive.
Scaling factor	Number used in the calculation of maximum per-flow queue size, which may be changed using configuration.
Average depth factor	Another name for scaling factor.

FRED Configuration

FRED configuration requires only slightly more effort than does WRED configuration, as long as default configuration values are acceptable. This section shows two FRED configuration examples that involve the same basic scenarios as the first two WRED examples in the previous section. FRED configuration and **show** commands are listed in Tables 6-11 and 6-12.

Table 6-11 *Command Reference for FRED*

Command	Mode and Function
random-detect flow	Interface configuration mode; enables FRED on the interface.
random-detect flow average-depth-factor *scaling-factor*	Interface configuration mode; changes scaling factor. The lower the number, the more aggressively FRED discards packets for flows using more than their fair share of the available queue space.
random-detect flow count *number*	Interface configuration mode; overrides default setting for the maximum number of concurrent flows tracked by FRED. (The default is 256.)
random-detect precedence *precedence min-threshold max-threshold mark-prob-denominator*	Interface, class, or random-detect-group configuration modes; overrides default settings for the specified precedence, for minimum and maximum WRED thresholds, and for percentage of packets discarded.

continues

Table 6-11 *Command Reference for FRED (Continued)*

Command	Mode and Function
random-detect dscp *dscpvalue min-threshold max-threshold* [*mark-probability-denominator*]	Interface, class, or random-detect-group configuration modes; overrides default settings for the specified DSCP, for minimum and maximum WRED thresholds, and for percentage of packets discarded.
random-detect exponential-weighting-constant *exponent*	Interface, class, or random-detect-group configuration modes; overrides default settings for exponential weighting constant. Lower numbers make WRED react quickly to changes in queue depth; higher numbers make WRED react less quickly.

Table 6-12 *Exec Command Reference for FRED*

Command	Function
show queue *interface-name interface-number* [**vc** [*vpi*/] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing random-detect [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]	Lists configuration and statistical information about the queuing tool on an interface
show interfaces	Mentions whether WRED has been enabled on the interface
show interface random-detect	Lists information about WRED when distributed WRED is running on a VIP interface

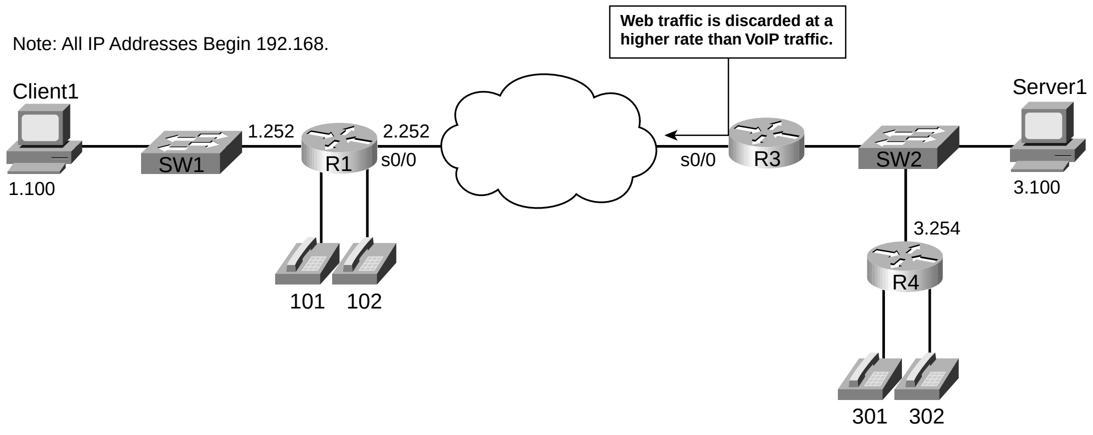
In the first example, R3 enables FRED on its S0/0 interface. FRED classifies the packets into flows and then decides which flows are currently considered to be robust, fragile, and nonadaptive. Based on the category, FRED decides whether new packets should be discarded.

This example continues the tradition of marking packets at ingress on R3's E0/0 interface. The marking logic performed by CB marking is as follows:

- VoIP payload—DSCP EF
- HTTP traffic for web pages with "important" in the URL—DSCP AF21
- HTTP traffic for web pages with "not-so" in the URL—DSCP AF23
- All other—DSCP default

To generate traffic in this network, two voice calls will be made between the analog phones attached to R1 and R4. Multiple web browsers will load the standard page used in this book, with two TCP connections created by each browser—one to get a file with the word "important" in it, and the other getting a file with "not-so" in it. An FTP download of a large file will also be initiated from the Server to Client1. Example 6-4 shows the basic configuration and **show** commands output. The example uses the familiar network diagram, with the configuration being added to R3.

Figure 6-12 *Sample Network for All FRED Examples—Configuration on R3*


Web traffic is discarded at a higher rate than VoIP traffic.
Note: All IP Addresses Begin 192.168.
Client1
1.100
SW1
1.252
R1
2.252
s0/0
101
102
s0/0
R3
SW2
Server1
3.100
3.254
R4
301
302

Example 6-4 *FRED Example Using All Default Values, R3 S0/0*

```
R3#show running-config
Building configuration...
!
! Some lines omitted for brevity
!
ip cef
!
! The following class-maps will be used for CB marking policy
! used at ingress to E0/0
!
class-map match-all http-impo
  match protocol http url "*important*"
class-map match-all http-not
  match protocol http url "*not-so*"
class-map match-all class-default
  match any
class-map match-all voip-rtp
  match ip rtp 16384 16383
!

! The following policy-map will be used for CB marking
!
```

continues

Example 6-4 *FRED Example Using All Default Values, R3 S0/0 (Continued)*

```
policy-map laundry-list
  class voip-rtp
   set ip dscp ef
  class http-impo
   set ip dscp af21
  class http-not
   set ip dscp af23
  class class-default
   set ip dscp default
!
call rsvp-sync
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 ip nbar protocol-discovery
 half-duplex
 service-policy input laundry-list
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 random-detect
 random-detect flow
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
!
! Some lines omitted for brevity
!

R3#show queueing random-detect
Current random-detect configuration:
  Serial0/0
    Queueing strategy: random early detection (WRED)
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 36
    Max flow count: 256       Average depth factor: 4
    Flows (active/max active/max): 2/8/256
class      Random drop      Tail drop    Minimum Maximum  Mark
           pkts/bytes       pkts/bytes    thresh  thresh  prob
0           25/12280         12/5651        20      40    1/10
1            0/0              0/0           22      40    1/10
2          103/125056       133/146549      24      40    1/10
3            0/0              0/0           26      40    1/10
```

Example 6-4 *FRED Example Using All Default Values, R3 S0/0 (Continued)*

```
4             0/0               0/0           28      40       1/10
5         641/41024         1475/94400        31      40       1/10
6             0/0               0/0           33      40       1/10
7             0/0               0/0           35      40       1/10
rsvp          0/0               0/0           37      40       1/10
```

The FRED part of the configuration is again quite short. The configuration lists the **random-detect** interface subcommand under serial 0/0, which enables precedence-based WRED on the interface. Because FRED is actually a superset of WRED, you also need to add the **random-detect flow** command to the interface. Interestingly, IOS adds the **random-detect** command to the interface if you only type the **random-detect flow** command. The rest of the detailed configuration in this example is just like the first WRED example configuration, repeating the CB marking configuration that marks the VoIP, FTP, and two different types of HTTP traffic. (For more information about CB marking, see Chapter 3.)

The only command that lists any new information, as compared with WRED, is the **show queueing random-detect interface serial 0/0** command. Most of the output looks familiar: It includes the various IP precedence values, with statistics. Just like with WRED, the command lists per-precedence default values for minimum threshold and maximum threshold. FRED still uses these values when determining the percentage of packets to discard. Specifically new for FRED, the command output lists two values used by the FRED algorithm when deciding to discard packets, namely the maximum flow count and average depth factor. The maximum flow count is the maximum number of unique active flows tracked by FRED. The average depth factor describes the scaling factor as described in the FRED concepts section, used when calculating the maximum per-flow queue size.

Although FRED does track and act on flow information, listing statistics per flow would be cumbersome, because the flows may well be short lived. The **show queueing** command still lists statistics, but it groups the statistics based on the precedence value (or DSCP value if DSCP-based FRED is used).

The second FRED configuration example uses FRED on the interface again, but this time with DSCP FRED, and a few defaults changed. In fact, Example 6-5 just shows the changed configuration, with most of the configuration staying the same. The same CB marking configuration is used to mark the traffic, for instance, so the details are not repeated in the example. The example uses the familiar network diagram used in Example 6-4.

Example 6-5 *DSCP-Based FRED on R3 S0/0*

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface serial 0/0
R3(config-if)#random-detect flow ?
  average-depth-factor  Average depth multiplier (1, 2, 4, 8 or 16)
```

continues

Example 6-5 *DSCP-Based FRED on R3 S0/0 (Continued)*

```
  count                 max number of dynamic flows
  <cr>

R3(config-if)#random-detect flow average-depth-factor 2
R3(config-if)#random-detect flow count 19
Number of WRED flows must be a power of 2 (16, 32, 64, 128, 256, 512, 1024
 2048, 4096, 8192, 16384 or 32768)
R3(config-if)#random-detect flow count 64
R3(config-if)#^Z
R3#
R3#show queueing random-detect
Current random-detect configuration:
  Serial0/0
    Queueing strategy: random early detection (WRED)
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 39
    Max flow count: 64       Average depth factor: 2
    Flows (active/max active/max): 13/13/64

dscp    Random drop      Tail drop    Minimum Maximum  Mark
        pkts/bytes       pkts/bytes    thresh  thresh  prob
af11        0/0              0/0          33      40  1/10
af12        0/0              0/0          28      40  1/10
af13        0/0              0/0          24      40  1/10
af21       19/23334          1/1404       33      40  1/10
af22        0/0              0/0          28      40  1/10
af23       11/14116          0/0          24      40  1/10
af31        0/0              0/0          33      40  1/10
af32        0/0              0/0          28      40  1/10
af33        0/0              0/0          24      40  1/10
af41        0/0              0/0          33      40  1/10
af42        0/0              0/0          28      40  1/10
af43        0/0              0/0          24      40  1/10
cs1         0/0              0/0          22      40  1/10
cs2         0/0              0/0          24      40  1/10
cs3         0/0              0/0          26      40  1/10
cs4         0/0              0/0          28      40  1/10
cs5         0/0              0/0          31      40  1/10
cs6         0/0              0/0          33      40  1/10
cs7         0/0              0/0          35      40  1/10
ef         11/704         2658/170112     37      40  1/10
rsvp        0/0              0/0          37      40  1/10
default     7/7396          16/14275      20      40  1/10
```

The configuration begins with a change from precedence-based FRED to DSCP-based FRED using the **random-detect dscp-based** interface subcommand. (The configuration already contained the **random-detect flow** command to enable flow-based WRED.) Two FRED options can be set with the **random-detect flow** command, as seen in the example. The average depth factor defines the multiple of the average queue space per flow that can be allocated to a single flow before FRED decides to start discarding packets. (Formula: average depth factor * maximum queue depth / number of active flows) The flow count, set to 64 in this example, just sets the maximum number of unique flows tracked by FRED. Just like with WFQ, the setting of the number of flows must be set to a power of 2.

Just like with DSCP-based WRED, the command output for DSCP-based FRED does not differ from the earlier precedence-based FRED example in too many ways. The changes to the default values have been highlighted in the example. The **show queueing** command contains the only notable difference between the command outputs in the first two examples, now listing information about all the DSCP values recommended in the DSCP RFCs. Notice that the counters point out drops for both AF21 and AF23, which were previously both treated as precedence 2 by precedence-based FRED. Also note that FRED has discarded some voice traffic (DSCP EF) in this example. Because FRED operates on all traffic in the interface FIFO queue, it cannot avoid the possibility of discarding voice traffic.

Table 6-13 summarizes many of the key concepts when comparing WRED and FRED.

Table 6-13 *WRED Versus FRED*

Feature	**WRED**	**FRED**
Discards packets to avoid congestion	Yes	Yes
Can be enabled on the physical interface concurrently with a queuing tool	No	No
Can be combined with CBWFQ or LLQ policy map	Yes	No
Bases drop decision, at least in part, on different thresholds per precedence or DSCP value	Yes	Yes
Bases drop decision, at least in part, on per-flow queue depth	No	Yes

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures are a convenient way to review the day before the exam.

Figure 6-13 outlines the process of packet loss, halving CWND, and slow start.

Figure 6-13 *Slamming the CWND Shut and the Slow Start Process After No Acknowledgment Received*

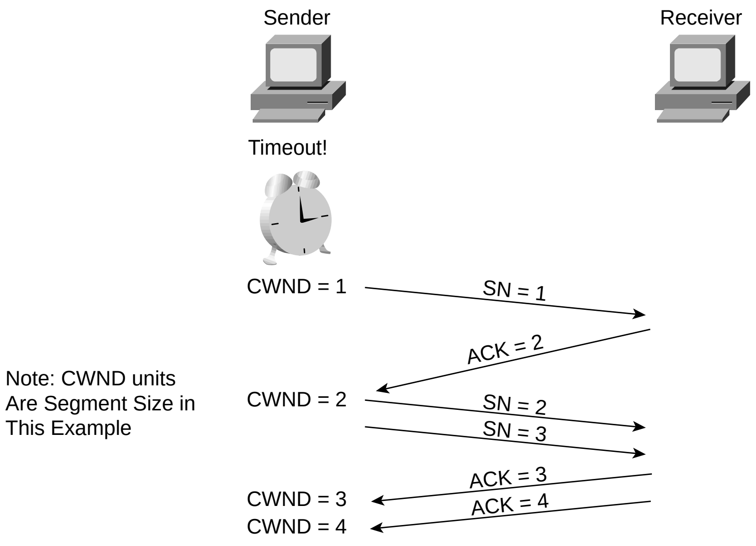
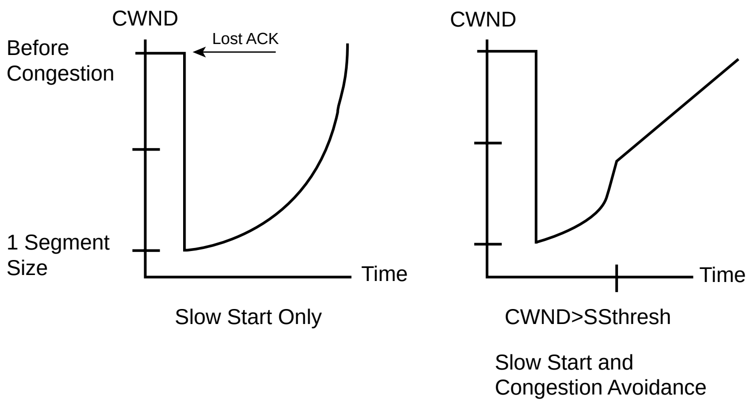


Figure 6-14 shows a graph of CWND after packet loss just using slow start, and another with slow start plus congestion avoidance.

Figure 6-14 *Graphs of CWND with Slow Start and Congestion Avoidance*



The key information about TCP and UDP operation when packets are dropped is summarized in the following list:

- UDP senders do not reduce or increase sending rates as a result of lost packets.
- TCP senders do reduce their sending rates as a result of lost packets.
- TCP senders decide to use either the receiver window or the CWND, based on whichever is lower at the time.
- TCP slow start and congestion avoidance dictate how fast the CWND rises after the window was lowered due to packet loss.

The graph in Figure 6-15 shows the proven behavior in the Internet with many TCP connections.

Figure 6-15 *Graph of Global Synchronization*

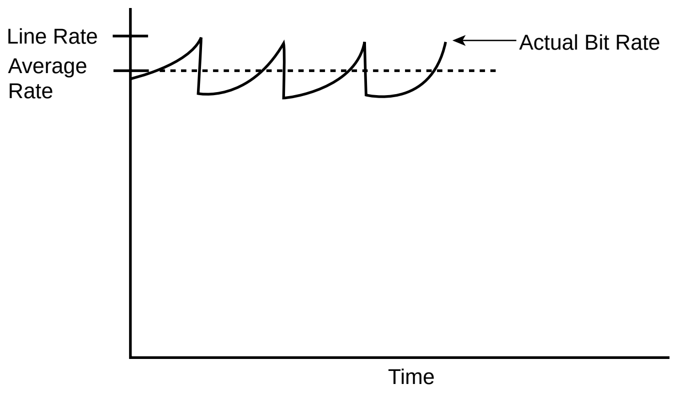


Table 6-14 describes the overall logic of when RED discards packets, with the same ideas outlined in Figure 6-16.

Table 6-14 *Three Categories of When RED Will Discard Packets, and How Many*

Average Queue Depth Versus Thresholds	Action
Average < minimum threshold	No packets dropped.
Minimum threshold < average depth < maximum threshold	A percentage of packets dropped. Drop percentage increases from 0 to a maximum percent as the average depth moves from the minimum threshold to the maximum.
Average depth > maximum threshold	All new packets discarded similar to tail dropping.

Figure 6-16 *RED Discarding Logic Using Average Depth, Minimum Threshold, and Maximum Threshold*

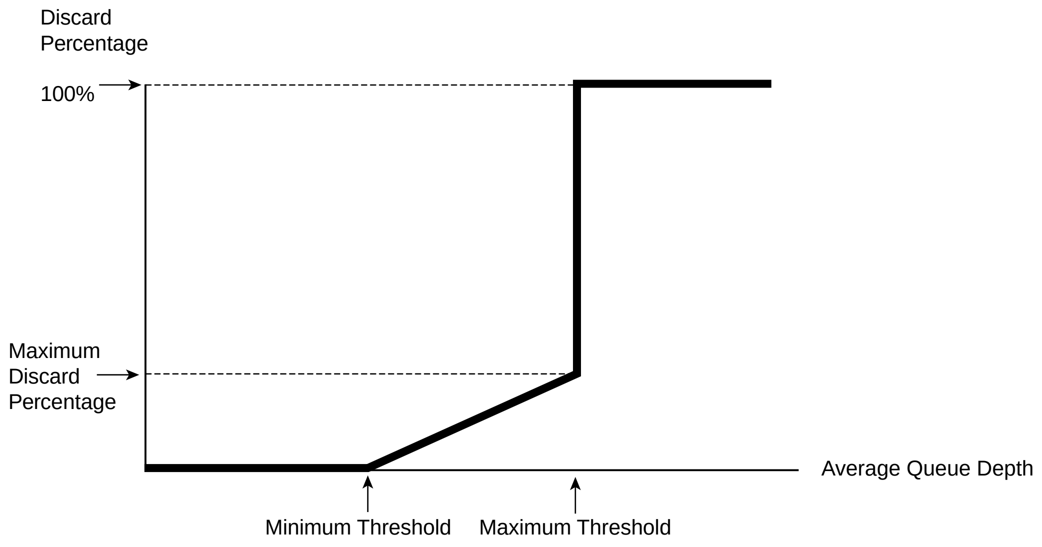


Table 6-15 summarizes some of the key terms related to RED.

Table 6-15 *RED Terminology*

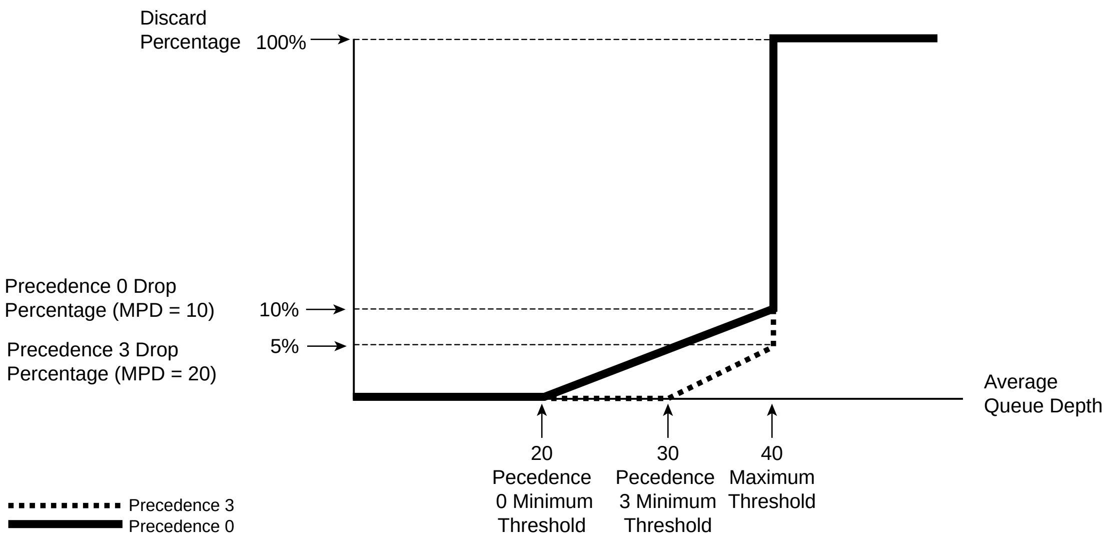
Term	Meaning
Actual queue depth	The actual number of packets in a queue at a particular point in time.
Average queue depth	Calculated measurement based on the actual queue depth and the previous average. Designed to adjust slowly to the rapid changes of the actual queue depth.

Table 6-15 *RED Terminology (Continued)*

Term	Meaning
Minimum threshold	Compares this setting to the average queue depth to decide whether packets should be discarded. No packets are discarded if the average queue depth falls below this minimum threshold.
Maximum threshold	Compares this setting to the average queue depth to decide whether packets should be discarded. All packets are discarded if the average queue depth falls above this maximum threshold.
Mark probability denominator	Used to calculate the maximum percentage of packets discarded when the average queue depth falls between the minimum and maximum thresholds.
Exponential weighting constant	Used to calculate the rate at which the average queue depth changes as compared with the current queue depth. The larger the number, the slower the change in the average queue depth.

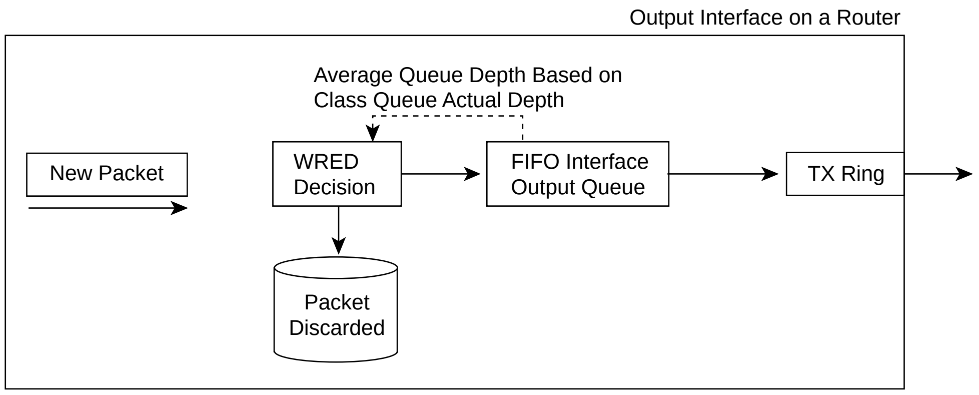
Figure 6-17 shows the default WRED settings for precedence 0, with some nondefault settings for precedence 5 traffic.

Figure 6-17 *Example WRED Settings for Precedences 0 and 5 for Thresholds and Discard Percent*



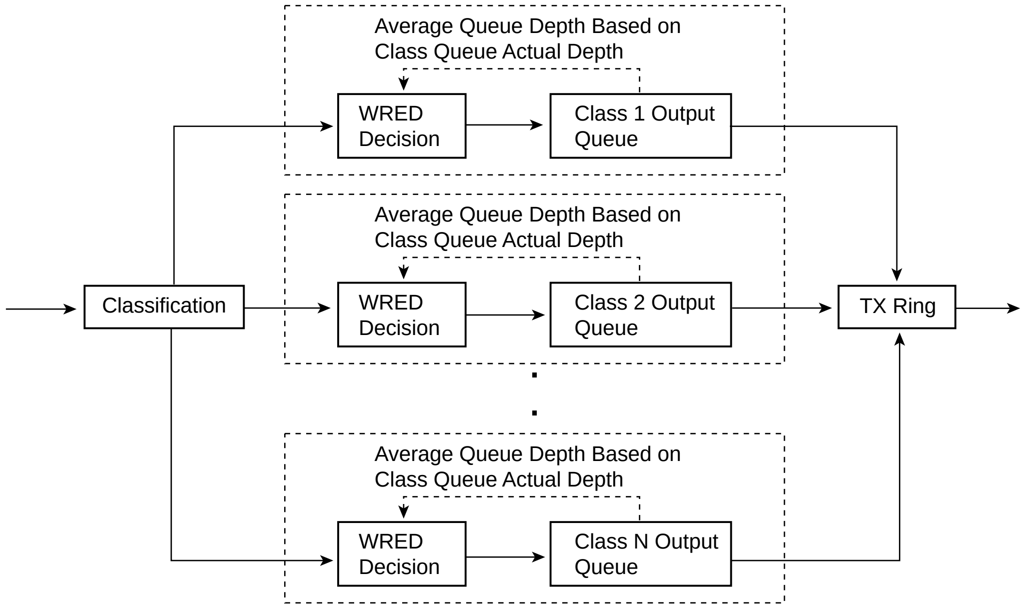
WRED measures the average queue depth of the FIFO queue on an interface, as shown in Figure 6-18.

Figure 6-18 *FIFO Output Queue and WRED Interaction*



WRED can be applied to each class queue with CBWFQ, as shown in Figure 6-19.

Figure 6-19 *WRED with CBWFQ*



Tables 6-16 and 6-17 list the WRED configuration and **show** commands, respectively.

Table 6-16 *Command Reference for WRED*

Command	Mode and Function
random-detect [*dscp-based* | *prec-based*]	Interface or class configuration mode; enables WRED, specifying whether to react to precedence or DSCP.
random-detect [**attach** *group-name*]	Interface configuration mode; enables per-VC WRED on ATM interfaces by referring to a **random-detect-group**.
random-detect-group *group-name* [*dscp-based* | *prec-based*]	Global configuration mode; creates a grouping of WRED parameters, which can be enabled on individual ATM VCs using the **random-detect attach** command.
random-detect precedence *precedence min-threshold max-threshold mark-prob-denominator*	Interface, class, or random-detect-group configuration modes; overrides default settings for the specified precedence, for minimum and maximum WRED thresholds, and for percentage of packets discarded.
random-detect dscp *dscpvalue min-threshold max-threshold* [*mark-probability-denominator*]	Interface, class, or random-detect-group configuration modes; overrides default settings for the specified DSCP, for minimum and maximum WRED thresholds, and for the percentage of packets discarded.
random-detect exponential-weighting-constant *exponent*	Interface, class, or random-detect-group configuration modes; overrides default settings for exponential weighting constant. Lower numbers make WRED react quickly to changes in queue depth; higher numbers make WRED react less quickly.

Table 6-17 *Exec Command Reference for WRED*

Command	Function
show *queue* *interface-name interface-number* [**vc** [*vpi*/] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing random-detect [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]	Lists configuration and statistical information about the queuing tool on an interface.
show interfaces	Mentions whether WRED has been enabled on the interface
show interface random-detect	Lists information about WRED when distributed WRED is running on a VIP interface
show policy-map [**interface** *interface-name interface-number*]	Lists WRED information when it is enabled inside an MQC policy map

Fred classifies each flow into one of three FRED flow types, as listed in Table 6-18.

Table 6-18 *FRED Flow Types*

Flow Type	Description	Transport Protocol	FRED Discard Policy
Robust	Adapts to lost packets by slowing down the rate of sending packets	TCP	Moderate discard rates
Fragile	Does not adapt to lost packets by slowing down, but the number of packets sent is not excessive	UDP	Low discard rates
Nonadaptive	Does not adapt to lost packets by slowing down, and the number of packets sent is excessive	UDP	High discard rates

The terms used by FRED to describe the processes covered so far are listed in Table 6-19.

Table 6-19 *FRED Terminology*

Term	Definition
Average per-flow queue size	A calculated value, based on the formula maximum queue size/ number of active flows.
Active flow	A flow that currently has packets in the queue.
Maximum per-flow queue size	A calculated value, based on the formula (average queue size * scaling factor). (This same formula is used in the previous example that results in an answer of 16.) This value is used to determine which flows are fragile, and which are nonadaptive.
Scaling factor	Number used in the calculation of maximum per-flow queue size, which may be changed using configuration.
Average depth factor	Another name for scaling factor.

FRED configuration and **show** commands are listed in Tables 6-20 and 6-21.

Table 6-20 *Command Reference for FRED*

Command	Mode and Function
random-detect flow	Interface configuration mode; enables FRED on the interface.
random-detect flow average-depth-factor *scaling-factor*	Interface configuration mode; changes scaling factor. The lower the number, the more aggressively FRED discards packets for flows using more than their fair share of the available queue space.

Table 6-20 *Command Reference for FRED (Continued)*

Command	Mode and Function
random-detect flow count *number*	Interface configuration mode; overrides default setting for the maximum number of concurrent flows tracked by FRED. (The default is 256.)
random-detect precedence *precedence min-threshold max-threshold mark-prob-denominator*	Interface, class, or random-detect-group configuration modes; overrides default settings for the specified precedence, for minimum and maximum WRED thresholds, and for percentage of packets discarded.
random-detect dscp *dscpvalue min-threshold max-threshold* [*mark-probability-denominator*]	Interface, class, or random-detect-group configuration modes; overrides default settings for the specified DSCP, for minimum and maximum WRED thresholds, and for percentage of packets discarded.
random-detect exponential-weighting-constant *exponent*	Interface, class, or random-detect-group configuration modes; overrides default settings for exponential weighting constant. Lower numbers make WRED react quickly to changes in queue depth; higher numbers make WRED react less quickly.

Table 6-21 *Exec Command Reference for FRED*

Command	Function
show queue *interface-name interface-number* [**vc** [*vpi*/] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing random-detect [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]	Lists configuration and statistical information about the queuing tool on an interface
show interfaces	Mentions whether WRED has been enabled on the interface
show interface random-detect	Lists information about WRED when distributed WRED is running on a VIP interface

Table 6-22 summarizes many of the key concepts when comparing WRED and FRED.

Table 6-22 *WRED Versus FRED*

Feature	WRED	FRED
Discards packets to avoid congestion	Yes	Yes
Can be enabled on the physical interface concurrently with a queuing tool	No	No
Can be combined with CBWFQ or LLQ policy map	Yes	No
Bases drop decision, at least in part, on different thresholds per precedence or DSCP value	Yes	Yes
Bases drop decision, at least in part, on per-flow queue depth	No	Yes

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD-ROM included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD-ROM nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD-ROM exam to include, or not include, the topics that are only on the CCIP QoS.

Congestion-Avoidance Concepts and Random Early Detection (RED)

1. Describe the function of the congestion window in TCP, and how it is changed as a result of packet loss.
2. Identify the two TCP windowing mechanisms, and describe when each is used.
3. Describe the process of TCP slow start, and when it occurs.
4. Describe the process of TCP congestion avoidance, and when it occurs.
5. Describe the meaning of the term “global synchronization,” and discuss what causes it.
6. Define the meaning of the term “tail drop.”
7. Define the meaning of the term “TCP starvation.”
8. Does RED compare the actual queue depth or the average queue depth to queue thresholds when deciding whether it should discard a packet? Why this one, and not the other?
9. Describe how RED uses actual queue depth to calculate average queue depth. Do not list the formula, but just describe the general idea.
10. Assume the RED minimum threshold is 20, the maximum threshold is 40, and the mark probability denominator is 10. What must be true for RED to discard all new packets?
11. Assume the RED minimum threshold is 20, the maximum threshold is 40, and the mark probability denominator is 10. What must be true for RED to discard 5 percent of all new packets?

12 Define how RED uses the mark probability denominator. Give one example.

13 Define the term “exponential weighting constant.” If the value is lowered compared to the default setting of 9, how does RED behave differently?

Weighted RED (WRED)

14 Spell out the words represented by the initials RED, WRED, and FRED.

15 List the queuing tools that can be concurrently supported on an interface when WRED has been enabled directly on a serial interface.

16 Identify the most important difference between RED operation and WRED operation.

17 Describe how WRED “weights” packets.

18 List the queuing tools that can enable WRED for use with some or all of their queues, effectively enabling WRED concurrently with the queuing tool.

19 What command enables you to look at WRED drop statistics when WRED is configured inside an MQC class?

20 Taking as many defaults as possible, list the configuration commands needed to configure precedence-based WRED on interface S1/1.

21 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based WRED on interface S1/1.

22 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based WRED inside **class class1**, inside **policy map my-policy**. (You can assume that the CBWFQ configuration has already been completed, and you just entered global configuration mode. Assume that you need just to enable WRED in **class class1**.)

23 List the command needed to set the minimum threshold to 25, the maximum threshold to 50, and the mark probability denominator to 4, for precedence 2.

24 What **show** command lists detailed statistics about random drops on interface S1/1?

Flow-Based WRED (FRED)

25 List the queuing tools that can be concurrently supported on an interface when FRED has been enabled directly on a serial interface.

26 Identify the most significant difference between FRED operation and WRED operation.

27 List the three categories of flows defined by FRED, and identify which category has its packets discarded most aggressively.

28 Describe how FRED prevents TCP starvation.

29 List the queuing tools that can enable FRED for use with some or all queues, effectively enabling FRED concurrently with the queuing tool.

30 Suppose that an interface has five active flows, with Flow 1 consuming 20 queue entries, and a maximum queue size of 40. Describe the terms “maximum per-flow queue depth,” and give an example of how it is calculated with this example. Use default values for any information not stated in the question.

31 Taking as many defaults as possible, list the configuration commands needed to configure precedence-based FRED on interface S1/1.

32 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based FRED on interface S1/1.

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Topics

- Explain the need for link-efficiency tools.
- Explain available LFI techniques including MLP interleaving and FR fragmentation using FRF.11 Annex-C or FRF.12.
- Explain Real Time Protocol header compression (cRTP) as a tool for improving link efficiency.
- Configure and monitor various LFI methods and cRTP.

QoS Exam Topics

- Describe payload compression and payload compression algorithms available on Cisco routers.
- Describe header compression and header compression algorithms available on Cisco routers.
- Configure Cisco IOS header compression mechanisms.
- Describe link fragmentation and interleaving (LFI).
- Configure link fragmentation and interleaving (LFI) mechanisms on Cisco IOS routers.

CHAPTER 7

Link-Efficiency Tools

Most WAN links are leased from a service provider, with one of the variables affecting the pricing being the bandwidth on the link. For instance, the distance and the bandwidth, or clock rate, on the link affect leased lines. Frame Relay service providers base their prices in part based on the access rate of the access links into the Frame Relay network, and the committed information rate (CIR) of the various virtual circuits (VCs).

If the offered load on the network consistently exceeds the bandwidth or clock rate of the link the traffic must flow across, unpredictable behavior can result. For example, queues consistently fill, causing more delay, jitter, and drops. If the offered load far exceeds the clock rate for a period of time, most data applications slow down significantly, with voice and video streams possibly even becoming unusable. Depending on how you configure quality of service (QoS) in the network, some traffic types may perform as expected, but with a likely result of allowing some other traffic types to degrade even more quickly, because most QoS tools by design end up favoring one type of traffic over another.

This chapter covers two classes of QoS tools that directly impact the usage of the bandwidth in a network—compression tools and link fragmentation and interleaving (LFI) tools. Compression tools compress the number of bytes in a packet so that fewer bytes need to be sent over a link.

LFI tools directly impact serialization delays—and serialization delay is impacted by actual link bandwidth. The slower the link, the longer it takes to serialize a packet. If a small packet must wait on a large packet to be serialized onto a link, the small packet may experience too much delay, particularly on slow-speed links. LFI tools reduce the delay experienced by a short packet by breaking larger packets into smaller pieces, and by transmitting the original small packets in between the fragments of the original larger packets. Smaller packets get better service, and in many cases, smaller packets are part of a delay-sensitive application, such as Voice over IP (VoIP).

“Do I Know This Already?” Quiz

The purpose of the “Do I Know This Already?” quiz is to help you decide whether you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 12-question quiz, derived from the major sections in “Foundation Topics” section of the chapter, helps you determine how to spend your limited study time.

Table 7-1 outlines the major topics discussed in this chapter and the “Do I Know This Already?” quiz questions that correspond to those topics.

Table 7-1 *“Do I Know This Already?” Foundation Topics Section-to-Question Mapping*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions	Score
1	Compression	1 to 6	
2	Link Fragmentation and Interleave	7 to 12	
All questions		1 to 12	

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the “Do I Know This Already?” quiz in Appendix A, “Answers to the ‘Do I Know This Already?’ Quizzes and Q&A Sections.” The suggested choices for your next step are as follows:

- **10 or less overall score**—Read the entire chapter. This includes the “Foundation Topics,” the “Foundation Summary,” and the “Q&A” sections.
- **11 or 12 overall score**—If you want more review on these topics, skip to the “Foundation Summary” section and then go to the “Q&A” section. Otherwise, move to the next chapter.

Compression Questions

1. Describe what is compressed, and what is not compressed, when using payload compression. Be as specific as possible regarding headers and data.
2. Describe what is compressed, and what is not compressed, when using TCP header compression. Be as specific as possible regarding headers and data.

3 Describe what is compressed, and what is not compressed, when using RTP header compression. Be as specific as possible regarding headers and data.

4 List the three point-to-point payload compression options available in IOS.

5 When TCP header compression is used, what is the range of sizes of the part of the frame that can be compressed, and what is the range of sizes for this field of the frame after compression?

6 When RTP header compression is used, what is the range of sizes of the part of the frame that can be compressed, and what is the range of sizes for this field of the frame after compression?

Link Fragmentation and Interleave Questions

7 List the words represented by the abbreviation LFI.

8 To achieve a 20-ms serialization delay on a 128-kbps link, how long can the fragments be?

9 What queuing tools can you enable directly on a serial interface when using multilink Point-to-Point Protocol with link fragmentation and interleaving (MLP LFI), as compared to when you are just using PPP?

10 What command can you use to determine the fragment size used for MLP LFI? What is the only parameter of the command?

11 What command enables FRF and sets the fragment size?

12 What other QoS feature for Frame Relay must you enable when you configure FRF.12 as well?

Foundation Topics

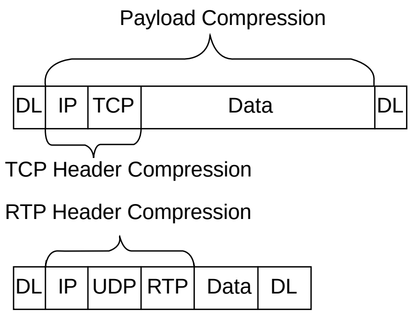
Payload and Header Compression

Compression involves mathematical algorithms that encode the original packet into a smaller string of bytes. After sending the smaller encoded string to the other end of a link, the compression algorithm on the other end of the link reverses the process, reverting the packet back to its original state.

Over the years, many mathematicians and computer scientists have developed new compression algorithms that behave better or worse under particular conditions. For instance, each algorithm takes some amount of computation, some memory, and they result in saving some number of bytes of data. In fact, you can compare compression algorithms by calculating the ratio of original number of bytes, divided by the compressed number of bytes—a value called the *compression ratio*. Depending on what is being compressed, the different compression algorithms have varying success with their compression ratios, and each uses different amounts of CPU and memory.

In Cisco routers, compression tools can be divided into two main categories: payload compression and header compression. *Payload compression* compresses headers and the user data, whereas header compression only compresses headers. As you might guess, payload compression can provide a larger compression ratio for larger packets with lots of user data in them, as compared to header compression tools. Conversely, *header compression* tools work well when the packets tend to be small, because headers comprise a large percentage of the packet. Figure 7-1 shows the fields compressed by payload compression, and by both types of header compression. (Note that the abbreviation DL stands for data link, representing the data-link header and trailer.)

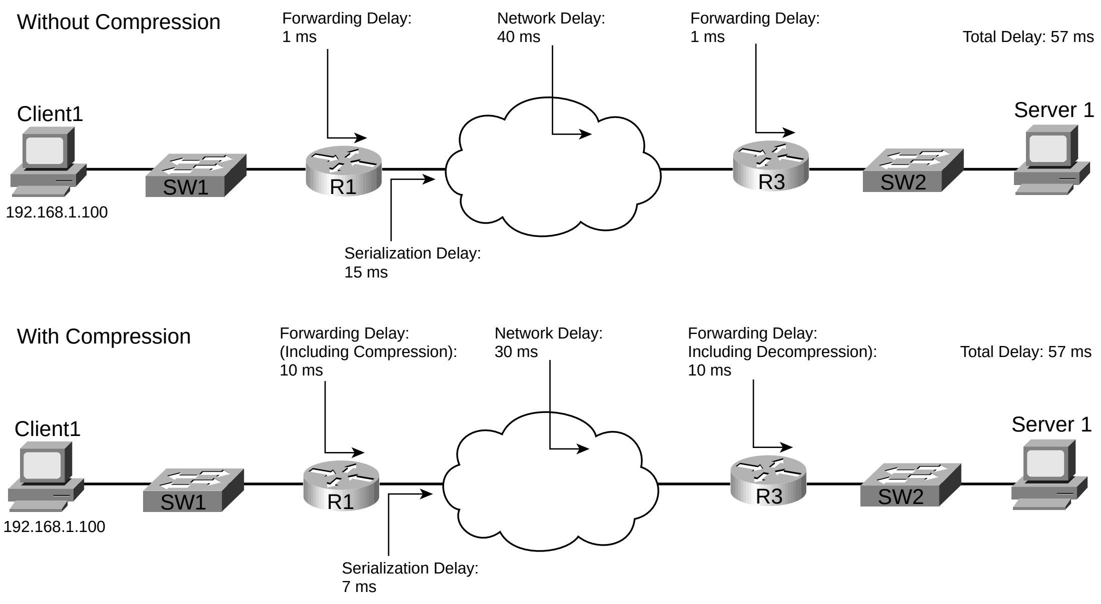
Figure 7-1 *Payload and Header Compression*



Both types of compression tools require CPU cycles and memory. Payload compression algorithms tend to take a little more computation and memory, just based on the fact that they have more bytes to process, as seen in Figure 7-1. With any of the compression tools, however, the computation time required to perform the compression algorithm certainly adds delay to the packet. The bandwidth gained by compression must be more important to you than the delay added by compression processing, otherwise you should choose not to use compression at all!

Figure 7-2 outlines an example showing the delays affected by compression. The compression and decompression algorithms take time, of course. However, serialization delay decreases, because the packets are now smaller. Queuing delay also decreases. In addition, on Frame Relay and ATM networks, the network delay decreases, because the network delay is affected by the amount of bits sent into the network.

Figure 7-2 *Delay Versus Bandwidth with Compression*



This example uses contrived numbers to make the relevant points. The example assumes a 2:1 compression ratio, with a 1500-byte packet, and 768-kbps access links on both ends. The various delay components in the top half of the figure, without compression, add up to 57 ms. In the lower figure, with compression, the delay is still 57 ms. Because a 2:1 compression ratio was achieved, twice as much traffic can be sent without adding delay—it seems like an obvious choice to use compression! However, compression uses CPU and memory, and it is difficult to predict how compression will work on a particular router in a particular network. Compression requires that you try it, monitor CPU utilization, look at the compression ratios, and experiment before just deciding to add it to all routers over an entire network. In addition, although this example shows no increase in delay, in most cases, just turning on software compression will probably increase delay slightly, but still with the benefit of increasing the amount of traffic you can send over the link.

Compression hardware minimizes the delay added by compression algorithms. Cisco offers several types of hardware compression cards that reduce the delay taken to compress the packets. Cisco offers compression service adapters on 7200, 7300, 7400, 7500 routers, compression advanced integration modules (AIMs) on 3660 and 2600 routers, and compression network modules for 3620s and 3640s. On 7500s with Versatile Interface Processors (VIPs), the compression work can be distributed to the VIP cards, even if no compression adapters are installed. Thankfully, when the compression cards are installed, IOS assumes that you want the compression to occur on the card by default, requiring you to specifically configure software compression if you do not want to use the compression hardware for some reason.

Payload Compression

Cisco IOS Software supplies three different options for point-to-point payload compression tools on serial links, namely Stacker, Microsoft Point-to-Point Compression (MPPC), and Predictor. Consider the following criteria when choosing between the payload compression options:

- The types of data-link protocols supported
- The efficiency of the compression algorithm used
- Whether the device on the other end of the link supports the tool

Of course, when two Cisco routers are used on each end, compatibility is not an issue, although Cisco does recommend that you use the same IOS revision on each end of the link when using compression.

Stacker and MPPC both use the same Lempel-Ziv compression algorithm. Predictor compression is named after the compression algorithm it uses. Predictor tends to take slightly less CPU and memory than does Lempel-Ziv, but Lempel-Ziv typically produces a better compression ratio.

The final factor on which tools to use is whether the tool supports the data-link protocol you are using. Of the three options, Stacker supports more data-link protocols than the other two tools. Table 7-2 outlines the main points of comparison for the three payload compression tools.

Table 7-2 *Point-to-Point Payload Compression Tools: Feature Comparison*

Feature	Stacker	MPPC	Predictor
Uses Lempel-Ziv (LZ) compression algorithm	Yes	Yes	No
Uses Predictor public domain compression algorithm	No	No	Yes
Supported on High-Level Data Link Control (HDLC)	Yes	No	No

Table 7-2 *Point-to-Point Payload Compression Tools: Feature Comparison (Continued)*

Feature	Stacker	MPPC	Predictor
Supported on X.25	Yes	No	No
Supported on Link Access Procedure, Balanced (LAPB)	Yes	No	Yes
Supported on Frame Relay	Yes	No	No
Supported on Point-to-Point Protocol (PPP)	Yes	Yes	Yes
Supported on ATM (using multilink PPP)	Yes	Yes	Yes

Cisco IOS Software supplies three different options for payload compression on Frame Relay VCs as well. Cisco IOS Software includes two proprietary options, called *packet-by-packet* and *data stream,* and one compression technique based on *Frame Relay Forum Implementation Agreement 9* (FRF.9). FRF.9 compression and data-stream compression function basically the same, with the only real difference being that FRF.9 implies compatibility with non-Cisco devices. Both data stream and FRF.9 compression use the Stacker algorithm (which is based on the Lempel-Ziv algorithm), with the dictionary created by Stacker being built and expanded for all packets on a VC. (Lempel-Ziv works by defining small strings of bytes inside the original packet, assigning these strings a short binary code, sending the short binary code rather than the longer original. The table of short binary codes, and their longer associated string of bytes, is called a *dictionary.*) The packet-by-packet compression method also uses Stacker, but the compression dictionary built for each packet is discarded—hence the name packet-by-packet compression. Table 7-3 lists the three tools and their most important distinguishing features.

Table 7-3 *Frame Relay Payload Compression Tools: Feature Comparison*

Feature	Packet-by-Packet	FRF.9	Data Stream
Uses Stacker compression algorithm	Yes	Yes	Yes
Builds compression dictionary over time for all packets on a VC, not just per packet	No	Yes	Yes
Cisco proprietary	Yes	No	Yes

Header Compression

Header compression algorithms take advantage of the fact that the headers are predictable. If you capture the frames sent across a link with a network analyzer, for instance, and look at IP packets from the same flow, you see that the IP headers do not change a lot, nor do the TCP headers, or UDP and RTP headers if RTP is used. Therefore, header compression can significantly reduce the size of the headers with a relatively small amount of computation. In fact, TCP header compression compresses the IP and TCP header (originally 40 bytes combined) down to between 3 and 5 bytes. Similarly, RTP header compression compresses the IP, UDP, and

RTP headers (originally 40 bytes combined) to 2 to 4 bytes. The variation in byte size for RTP headers results from the presence of a UDP checksum. Without the checksum, the RTP header is 2 bytes; with the checksum, the RTP header is 4 bytes.

TCP header compression results in large compression ratios if the TCP packets are relatively small. If the packets have a minimum length (64 bytes), with 40 of those being the IP and TCP headers, the compressed packet is between 27 and 29 bytes! That gives a compression ratio of 64/27, or about 2.37, which is pretty good for a compression algorithm that uses relatively little CPU. However, a 1500-byte packet with TCP header compression saves 35 to 37 bytes of the original 1500-byte packet, providing a compression ratio of 1500/1497, or about 1.002, a relatively insignificant savings in this case.

RTP header compression typically provides a good compression result for voice traffic, because VoIP tends always to use small packets. For instance, G.729 codecs in Cisco routers uses 20 bytes of data, preceded by 40 bytes of IP, UDP, and RTP headers. After compression, the headers are down to 4 bytes, and the packet size falls from 60 bytes to 24 bytes! Table 7-4 lists some of the overall VoIP bandwidth requirements, and the results of RTP header compression.

Table 7-4 *Bandwidth Requirements for Various Types of Voice Calls With and Without cRTP*

Codec	Payload Bandwidth	IP/UDP/RTP Header size	Layer 2 header Type	Layer 2 header Size	Total Bandwidth
G.711	64 kbps	40 bytes	Ethernet	14	85.6
G.711	64 kbps	40 bytes	MLPPP/FR	6	82.4
G.711	64 kbps	2 bytes (cRTP)	MLPPP/FR	6	67.2
G.729	8 kbps	40 bytes	Ethernet	14	29.6
G.729	8 kbps	40 bytes	MLPPP/FR	6	26.4
G.729	8 kbps	2 bytes (cRTP)	MLPPP/FR	6	11.2

* For DQOS test takers: These numbers are extracted from the DQOS course, so you can study these numbers. Note, however, that the numbers in the table do not include the Layer 2 trailer overhead.

Payload Compression Configuration

Payload compression requires little configuration. You must enable compression on both ends of a point-to-point serial link, or on both ends of a Frame Relay VC for Frame Relay. The **compress** command enables compression on point-to-point links, with the **frame-relay payload-compression** command enabling compression over Frame Relay.

Table 7-5 lists the various configuration and **show** commands used with payload compression, followed by example configurations.

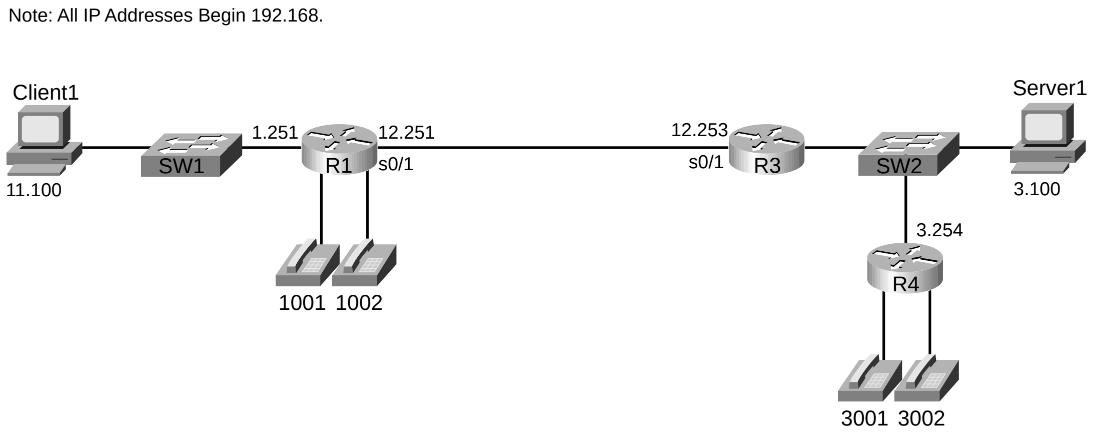
Table 7-5 *Configuration Command Reference for Payload Compression*

Command	Mode and Function
compress predictor	Interface configuration mode; enables Predictor compression on one end of the link.
compress stac	Interface configuration mode; enables Stacker compression on one end of the link.
compress mppc [**ignore-pfc**]	Interface configuration mode; enables MPPC compression on one end of the link.
compress stac [**distributed** | **software**]	Interface configuration mode; on 7500s with VIPs, allows specification of whether the compression algorithm is executed in software on the VIP.
compress {**predictor** | **stac** [**csa** *slot* | **software**]}	Interface configuration mode; On 7200s, allows specification of Predictor or Stacker compression on a compression service adapter (CSA).
compress stac caim *element-number*	Interface configuration mode; enables Stacker compression using the specified compression AIM.
frame-relay payload-compress {**packet-by-packet** | **frf9 stac** [*hardware-options*] | **data-stream stac** [*hardware-options*]}	Interface configuration mode; enables FRF.9 or data-stream style compression on one end of a Frame Relay link. Hardware-options field includes the following options: software, distributed (for use w/VIPs), and CSA (7200s only).

You can use the **show compress** command to verify that compression has been enabled on the interface and to display statistics about the compression behavior.

The first example uses the network described in Figure 7-3, with a PPP link between R1 and R3. The example uses the same familiar web browsing sessions, each of which downloads two JPGs. An FTP get transfers a file from the server to the client, and two voice calls between R1 and R4 are used.

Figure 7-3 *The Network Used in PPP Payload Compression Examples*



Example 7-1 shows the Stacker compression between R1 and R3.

Example 7-1 *Stacker Payload Compression Between R1 and R3 (Output from R3)*

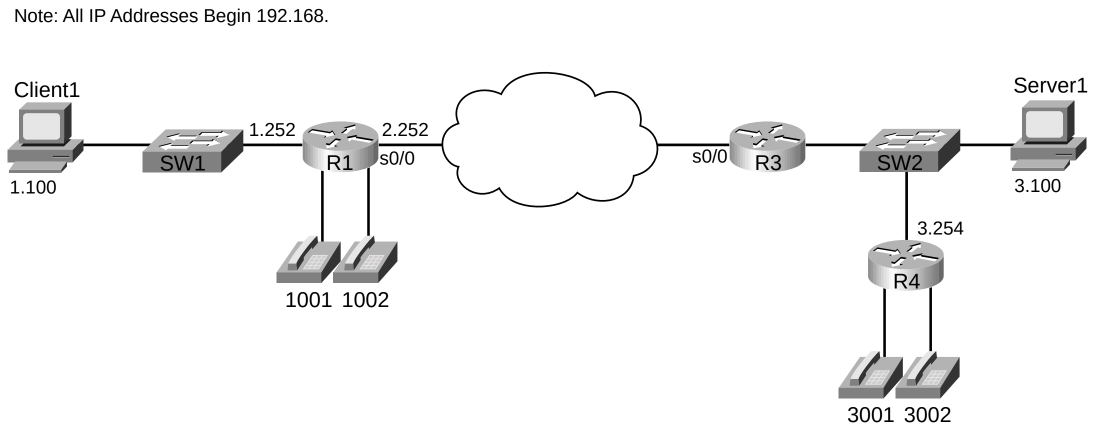
```
R3#show running-config
Building configuration...

!
! Lines omitted for brevity
!
interface Serial0/1
 bandwidth 128
 ip address 192.168.12.253 255.255.255.0
 encapsulation ppp
 compress stacker
 clockrate 128000
!
! Portions omitted for brevity
!
r3#show compress
 Serial0/1
     Software compression enabled
     uncompressed bytes xmt/rcv 323994/5494
     compressed   bytes xmt/rcv 0/0
     1  min avg ratio xmt/rcv 1.023/1.422
     5  min avg ratio xmt/rcv 1.023/1.422
     10 min avg ratio xmt/rcv 1.023/1.422
     no bufs xmt 0 no bufs rcv 0
     resyncs 2
```

The configuration requires only one interface subcommand, **compress stacker**. You must enter this command on both ends of the serial link before compression will work. The **show compress** command lists statistics about how well compression is working. For instance, the 1-, 5-, and 10-minute compression ratios for both transmitted and received traffic are listed, which gives you a good idea of how much less bandwidth is being used because of compression.

You can easily configure the other two payload compression tools. Instead of the **compress stacker** command as in Example 7-1, just use the **compress mppc** or **compress predictor** command.

Example 7-2 shows FRF.9 payload compression. The configuration uses a point-to-point subinterface and the familiar network used on most of the other configuration examples in the book, as shown in Figure 7-4.

Figure 7-4 *The Network Used in FRF.9 Payload Compression Example*



Example 7-2 *FRF.9 Payload Compression Between R1 and R3 (Output from R3)*

```
R3#show running-config
Building configuration...

!
! Lines omitted for brevity
!
interface Serial0/0
 description Single PVC to R1.
 no ip address
 encapsulation frame-relay IETF
 no ip mroute-cache
 load-interval 30
 clockrate 128000
```

continues

Example 7-2 *FRF.9 Payload Compression Between R1 and R3 (Output from R3) (Continued)*

```
 !
 interface Serial0/0.1 point-to-point
  description point-point subint global DLCI 103, connected via PVC to DLCI 101 (R1)
  ip address 192.168.2.253 255.255.255.0
  no ip mroute-cache
  frame-relay interface-dlci 101 IETF
  frame-relay payload-compression FRF9 stac
 !
 ! Portions omitted for brevity
 !
 R3#show compress
  Serial0/0 - DLCI: 101
      Software compression enabled
      uncompressed bytes xmt/rcv 6480/1892637
      compressed   bytes xmt/rcv 1537/1384881
      1  min avg ratio xmt/rcv 0.021/1.352
      5  min avg ratio xmt/rcv 0.097/1.543
      10 min avg ratio xmt/rcv 0.097/1.543
      no bufs xmt 0 no bufs rcv 0
      resyncs 1
        Additional Stacker Stats:
        Transmit bytes:  Uncompressed =        0 Compressed =        584
        Received bytes:  Compressed =     959636 Uncompressed =        0
```

Frame Relay payload compression takes a little more thought, although it may not be apparent from the example. On point-to-point subinterfaces, the **frame-relay payload-compression FRF9 stac** command enables FRF.9 compression on the VC associated with the subinterface. If a multipoint subinterface is used, or if no subinterfaces are used, however, you must enable compression as parameters on the **frame-relay map** command.

TCP and RTP Header Compression Configuration

Unlike payload compression, Cisco IOS Software does not have different variations on the compression algorithms for TCP and RTP header compression. To enable TCP or RTP compression, you just enable it on both sides of a point-to-point link, or on both sides of a Frame Relay VC.

Note that when enabling compression, it is best practice to enable the remote side of the WAN link before enabling the local side of the WAN link. This enables the administrator to retain control of WAN connectivity. If the local side of the WAN link is configured first, an out-of-band access must exist to access the remote side.

When configuring Frame Relay TCP or RTP header compression, the style of configuration differs based on whether you use point-to-point subinterfaces. On point-to-point subinterfaces, the **frame-relay ip tcp** or **frame-relay ip rtp** commands are used. If you use multipoint subinterfaces, or use the physical interface, you must configure the same parameters on **frame-relay map** commands.

Regardless of the type of data-link protocol in use, TCP and RTP compression commands allow the use of the **passive** keyword. The **passive** keyword means that the router attempts to perform compression only if the router on the other side of the link or VC compresses the traffic. The **passive** keyword enables you to deploy configurations on remote routers with the **passive** keyword, and then later add the compression configuration on the local router, at which time compression begins.

The TCP and RTP header compression configuration process, as mentioned, is very simple. Table 7-6 and Table 7-7 list the configuration and **show** commands, which are followed by a few example configurations.

Table 7-6 *Configuration Command Reference for TCP and RTP Header Compression*

Command	Mode and Function
ip tcp header-compression [**passive**]	Interface configuration mode; enables TCP header compression on point-to-point links
ip rtp header-compression [**passive**]	Interface configuration mode; enables RTP header compression on point-to-point links
frame-relay ip tcp header-compression [**passive**]	Interface/subinterface configuration mode; enables TCP header compression on point-to-point links
frame-relay ip rtp header-compression [**passive**]	Interface or subinterface configuration mode; enables RTP header compression on point-to-point links
frame-relay map ip *ip-address dlci* [**broadcast**] **tcp header-compression** [**active** | **passive**] [**connections** *number*]	Interface or subinterface configuration mode; enables TCP header compression on the specific VC identified in the **map** command
frame-relay map ip *ip-address dlci* [**broadcast**] **rtp header-compression** [**active** | **passive**] [**connections** *number*]	Interface or subinterface configuration mode; enables RTP header compression on the specific VC identified in the **map** command

Table 7-7 *Exec Command Reference for TCP and RTP Header Compression*

Command	Function
show frame-relay ip rtp header-compression [**interface** *type number*]	Lists statistical information about RTP header compression over Frame Relay; can list information per interface
show frame-relay ip tcp header-compression	Lists statistical information about TCP header compression over Frame Relay

continues

Table 7-7 *Exec Command Reference for TCP and RTP Header Compression (Continued)*

Command	Function
show ip rtp header-compression [*type number*] [**detail**]	Lists statistical information about RTP header compression over point-to-point links; can list information per interface
show ip tcp header-compression	Lists statistical information about TCP header compression over point-to-point links

The first example uses the same point-to-point link used in the payload compression section, as shown earlier in Figure 7-3. In each example, the same familiar web browsing sessions are used, each downloading two JPGs. An FTP get transfers a file from the server to the client, and two voice calls between R1 and R4 are used.

Example 7-3 example shows TCP header compression on R3.

Example 7-3 *TCP Header Compression on R3*

```
R3#show running-config
Building configuration...

!
! Lines omitted for brevity
!
interface Serial0/1
 bandwidth 128
 ip address 192.168.12.253 255.255.255.0
 encapsulation ppp
 ip tcp header-compression
 clockrate 128000
!
! Portions omitted for brevity
!
R3#show ip tcp header-compression
TCP/IP header compression statistics:
  Interface Serial0/1:
    Rcvd:    252 total, 246 compressed, 0 errors
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    371 total, 365 compressed,
             12995 bytes saved, 425880 bytes sent
             1.3 efficiency improvement factor
    Connect: 16 rx slots, 16 tx slots,
             218 long searches, 6 misses 0 collisions, 0 negative cache hits
             98% hit ratio, five minute miss rate 0 misses/sec, 1 max
```

To enable TCP header compression, the **ip tcp header-compression** command was added to both serial interfaces on R1 and R3. The **show ip tcp header-compression** command lists statistics about how well TCP compression is working. For instance, 365 out of 371 packets sent were compressed, with a savings of 12,995 bytes. Interestingly, to find the average number of bytes saved for each of the compressed packets, divide the number of bytes saved (12,995) by the number of packets compressed (365), which tells you the average number of bytes saved per packet was 35.6. For comparison, remember that TCP header compression reduces the 40 bytes of IP and TCP header down to between 3 and 5 bytes, meaning that TCP header compression should save between 35 and 37 bytes per packet, as is reflected by the output of the **show ip tcp header-compression** command.

To configure RTP header compression on point-to-point links, you perform a similar exercise as you did for TCP in Example 7-3, except you use the **rtp** keyword rather than the **tcp** keyword to enable RTP header compression. For a little variety, however, the next example shows RTP header compression, as enabled on a Frame Relay link between R3 and R1. The network used in this example matches Figure 7-4, shown in the Frame Relay payload compression example. Example 7-4 shows the configuration and statistics.

Example 7-4 *Frame Relay RTP Header Compression on R3*

```
R3#show running-config
Building configuration...

!
! Lines omitted for brevity
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
 frame-relay ip rtp header-compression
!
! Portions omitted for brevity
!
R3#show frame-relay ip rtp header-compression
 DLCI 101        Link/Destination info: point-to-point dlci
  Interface Serial0/0:
    Rcvd:    18733 total, 18731 compressed, 2 errors
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    16994 total, 16992 compressed,
             645645 bytes saved, 373875 bytes sent
             2.72 efficiency improvement factor
    Connect: 256 rx slots, 256 tx slots,
             0 long searches, 2 misses 0 collisions, 0 negative cache hits
             99% hit ratio, five minute miss rate 0 misses/sec, 0 max
```

To enable RTP header compression, the **frame-relay ip rtp header-compression** command was added to both serial subinterfaces on R1 and R3. (If a multipoint subinterface had been used instead, the same parameters could have been configured on a **frame-relay map** command.)

The **show frame-relay ip rtp header-compression** command lists statistics about how well cRTP is working. You might recall that cRTP reduces the 40 bytes of IP, UDP, and RTP header to between 2 and 4 bytes, saving between 36 and 38 bytes per packet. In the output in the example, with 16,992 packets compressed, a savings of 645,645 bytes is realized, which is an average per-packet savings of 37.99 bytes.

The **show frame-relay ip rtp header-compression** command output also lists an *efficiency improvement factor*, which is the compression ratio. It is calculated as the number of bytes that would have been sent without compression, divided by the actual number of bytes sent. From the shaded lines at the end of the preceding example, 645,645 + 373,875 bytes would have been sent (the number of bytes saved, plus actual number sent), for a total of 1,019,520 bytes that would have been sent. Dividing that total by the actual number sent (373,875) gives you the improvement factor of 2.72. For perspective, if you divide the packet size of a G.729 VoIP packet (60 bytes), by the compressed size of 22 bytes (which implies you saved 38 of the 40 bytes in the header), the ratio also calculates to 2.72. Therefore, the empirical ratio from the **show** command matches the theoretical ratio between bytes that would have been sent, and bytes that are actually sent, with cRTP.

Link Fragmentation and Interleaving

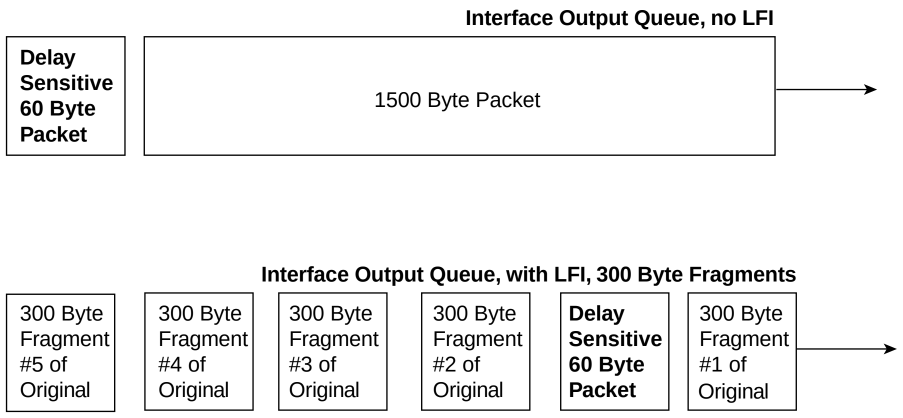
Both types of QoS tools covered in this chapter address bandwidth constraints to some degree. Compression tools directly attack bandwidth constraints by lowering the bandwidth required to forward packets. Link fragmentation and interleaving (LFI) tools directly lower delay by defeating a side effect of a small transmit clock speed, namely serialization delay.

A quick review of serialization delay should help you make more sense out of LFI tools. Serialization is the time required to send a frame over a physical link. If a link has a physical clock rate of *x* bps, it takes 1/*x* seconds to send a single bit. If a frame has *y* bits in it, it takes *y*/*x* seconds to serialize the frame. The faster the link, the lower the serialization delay. On a 56-kbps link, for example, it takes 1/56,000 of a second to send 1 bit. A 1500-byte frame (12,000 bits) takes 12,000/56,000 seconds to serialize, or roughly 214 ms.

When a router starts to send a frame out of an interface, it sends the complete frame. If a small, delay-sensitive frame needs to exit an interface, and the router has just begun to send a large frame, the small frame must wait until the whole large frame has been sent before the router will send the small, delay–sensitive frame. As seen in the preceding example, a 1500-byte frame takes 214 ms to serialize at 56 kbps, which is far too long for the small frame to wait if it is part of a VoIP stream.

LFI tools attack the serialization delay problem by ensuring that large packets do not delay smaller packets. It accomplishes this by dividing larger packets (fragmentation) and interleaving later-arriving smaller packets with the fragments from the larger packet. The smaller, delay-sensitive interleaved packets, typically VoIP, are defined in your QoS policy. Figure 7-5 outlines the basic process.

Figure 7-5 *Basic Concept Behind LFI Tools*


Interface Output Queue, no LFI
Delay Sensitive 60 Byte Packet
1500 Byte Packet
Interface Output Queue, with LFI, 300 Byte Fragments
300 Byte Fragment #5 of Original
300 Byte Fragment #4 of Original
300 Byte Fragment #3 of Original
300 Byte Fragment #2 of Original
Delay Sensitive 60 Byte Packet
300 Byte Fragment #1 of Original

As shown in the upper queue in the figure, without LFI, the small 60-byte packet must wait for the full 1500-byte packet to be forwarded. In the lower queue, with LFI enabled, IOS can choose to let the smaller packet exit the interface ahead of some of the fragments of the larger packet.

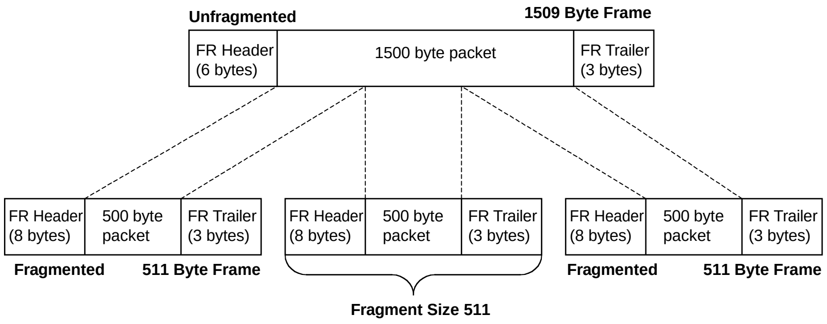
Before examining LFI in more detail, you need to take a closer look at the terms “packet” and “frame.” In most cases in this book, these terms have been used interchangeably. However, it is important to realize what really gets placed into the queues, and what really gets fragmented, when discussing LFI tools.

First, we need a shared definition of what each of the two terms mean. *Packet* refers to the entity that flows through the network, including the Layer 3 header, all headers from layers above Layer 3, and the end-user data. Packets do not include the data-link (Layer 2) headers and trailers. *Frames* include the packet, as well as the data-link (Layer 2) header and trailer.

Queuing tools actually place frames into the queues. For instance, Weighted Fair Queuing (WFQ) on a PPP serial interface places PPP frames into the queues. Concerning queuing tools, the distinction does not really have much bearing on the choices you make. In addition, because most people tend to use the term “packet” more often, this book just uses packet when it does not matter whether you care about the packet or the frame.

LFI tools require you to think about what happens to the packet, *and* what happens to the frame. Consider Figure 7-6, which shows some of the details of an unfragmented frame, and a fragmented frame, using Frame Relay.

Figure 7-6 *LFI Application to Packets and Frames, 1500-Byte Packet*



In the upper part of the figure, a 1500-byte packet has an extra 9 bytes of Frame Relay header and trailer added to it, to form a 1509-byte frame. In the lower part of the figure, the 1500-byte packet has been fragmented into three 500-byte fragments, and then placed into Frame Relay frames. It turns out that with FRF.12 LFI, an additional 2 bytes of header are needed to manage the fragments, so each of the three frames totals 511 bytes in length.

Technically, the fragment size used in the figure is 511 bytes, not 500. Most people would tend to think something like "the router fragmented the 1500-byte packet into three 500-byte fragments." In reality, the router performs logic like in the following list:

- The router fragments the *packet* into smaller pieces.
- The router adds the appropriate data-link headers and trailers, including any headers specifically needed for fragmentation support.
- The length of the resulting *frames* (including data-link headers/trailers) does not exceed the fragmentation size configured.
- The router adds these frames to the appropriate queue.

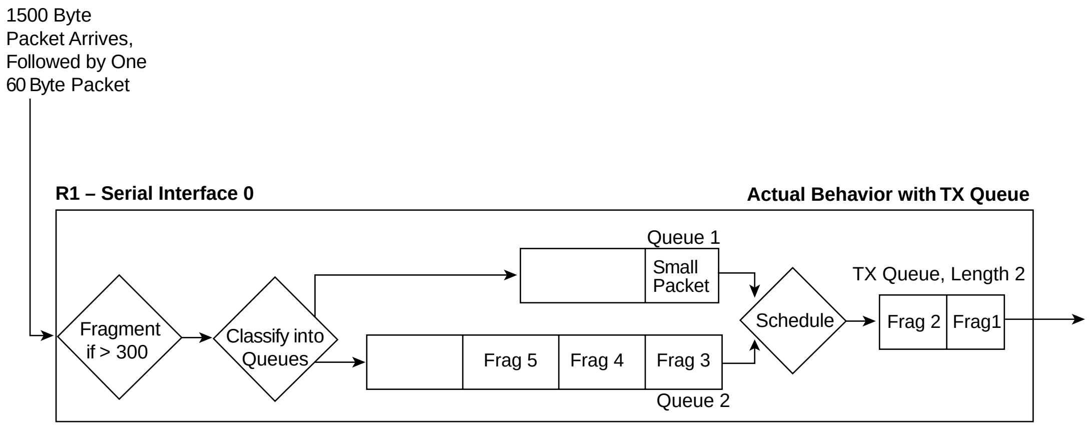
So, the router fragments packets into smaller pieces, but the size of the pieces is determined by the fragment size, which is based on the frame size. Therefore, does LFI really fragment packets, or frames? Frankly, either term works. When you are choosing the size of the fragments, however, always remember that the fragment size determines the size of the frames, *not* the packets. Therefore, you *should* consider the length of the data-link headers and trailers when choosing the size of the fragments.

Multilink PPP LFI

The core concept behind LFI, and its benefits, is very straightforward. The details, however, can be a little confusing, mainly because IOS LFI tools interact directly with IOS queuing tools. In addition, the two LFI tools covered on the Cisco QoS exams happen to behave differently as to how they interact with queuing tools. So to understand where LFI functions take place, you need to examine each tool specifically. This section covers multilink PPP LFI (MLP LFI), with Frame Relay fragmentation (FRF) covered in the next section of this chapter.

Figure 7-7 depicts how MLP LFI works with a queuing tool on an interface.

Figure 7-7 *MLP LFI Interaction with Queuing*



The figure outlines a lot of the detailed concepts behind LFI. In this example, a 1500-byte packet first arrives at R1, followed by a 60-byte packet. The fragmentation logic has been configured to fragment the frames down to a little more than 300 bytes, to make room for 300 bytes from the packet, and a little more for the data-link headers and trailers. After fragmentation, the queuing tool on the interface classifies the frames into their respective queues, which in this example happens to be two different queues. (The queuing tool's classification step works exactly as described in chapter 4, "Congestion Management.")

Now look to the far right side of the figure. The TX Queue is shown, with a queue length of 2. In this example, an assumption has been made that the small packet arrived after IOS had placed the first two fragments of the large packet into the two available slots in the TX Queue, with the last three fragments being placed into Queue 2. The TX Queue is always *absolutely* a single FIFO queue, as described in Chapter 4. In other words, the small packet does not interrupt the router while it is in the middle of sending fragment 1, nor does the small packet have a chance to be sent before fragment 2, because fragment 2 is already in the TX Queue. The best behavior the small packet can hope for is to be the next packet placed onto the end of the TX Queue. Therefore, for now, the small packet has been placed into Queue 1.

Now look just to the left of the TX Queue, between the two interface output queues and the TX Queue. The term "schedule" reminds us that the queuing scheduler chooses the next packet to be moved from the output queues to the TX Queue (as described in Chapter 4). The queuing tool's scheduler may decide to take the next packet from Queue 1 or Queue 2—a decision totally based on the logic of the queuing tool.

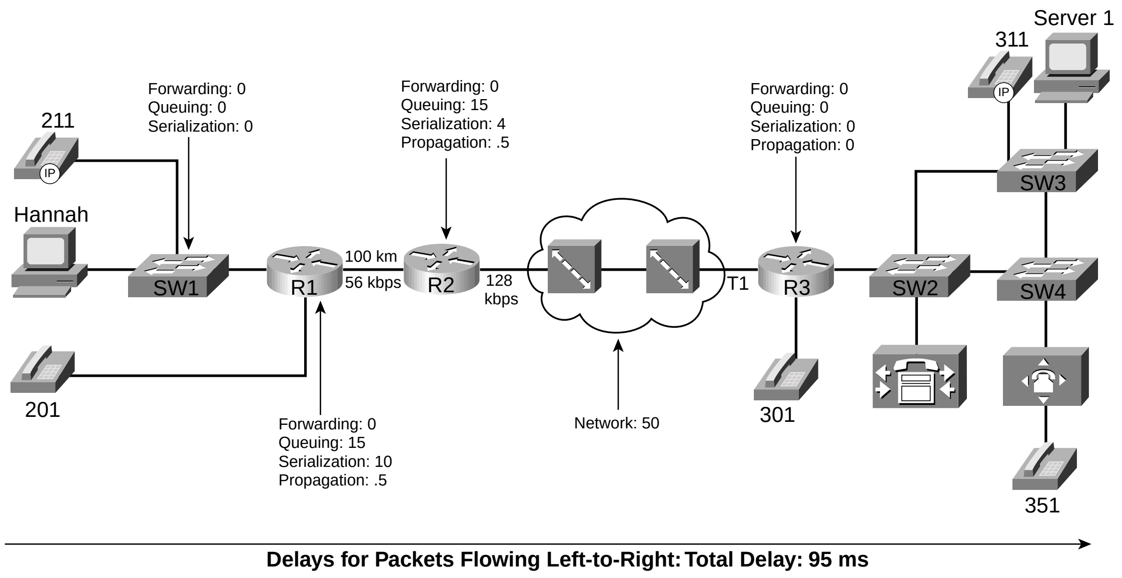
Interleaving occurs when the queuing scheduler decides to service the queue that holds the small packet next, rather than the queue holding the next fragment of the large packet. If Low Latency Queuing (LLQ) has been configured, and Queue 1 is the low-latency queue, the scheduler takes the small packet next, meaning that the small packet would be interleaved between fragments of the larger packet. If the queuing tool was Custom Queuing (CQ), and the queuing scheduler were able to send more bytes from Queue 2 in this cycle, fragment 3 would be sent next.

Maximum Serialization Delay and Optimum Fragment Sizes

How large should the fragments be to reduce serialization delay to an acceptable level? Well, the real answer lies in an analysis of the delay budgets for your network. From that analysis, you determine the maximum serialization delay you can have on each link.

The delay budget includes many delay components, such as queuing delay, propagation delay, shaping delay, network delay, and serialization delay. Based on that delay budget, you determine how much serialization delay you can afford on a particular link. Figure 7-8 depicts example delay values for various delay components.

Figure 7-8 *Review of Delay Components, Including Serialization Delay*



Now imagine that you need to configure R1 in the figure to use MLP LFI. You already know that you want a maximum serialization delay of 10 ms, and conveniently, MLP LFI enables you to configure a max-delay parameter. MLP LFI then calculates the fragment size, based on the following formula:

Max-delay * bandwidth

In this formula, bandwidth is the value configured on the **bandwidth** interface subcommand, and max-delay is the serialization delay configured on the **ppp multilink fragment-delay** command. For instance, R1 in Figure 7-8 shows a budget for 10 ms of serialization delay. On a 56-kbps link, a 10-ms max-delay would make the fragment size 56,000 * .01, or 560 bits, which is 70 bytes.

Cisco generally suggests a maximum serialization delay per link of 10 ms in multiservice networks. Because serialization delay becomes less than 10 ms for 1500-byte packets at link speeds greater than 768 kbps, Cisco recommends that LFI be considered on links with a 768-kbps clock rate and below.

The math used to find the fragment size, based on the serialization delay and bandwidth, is pretty easy. For perspective, Table 7-8 summarizes the calculated fragment sizes based on the bandwidth and maximum delay.

Table 7-8 *Fragment Sizes Based on Bandwidth and Serialization Delay*

Bandwidth/Link Speed	10-ms Delay	20-ms Delay	30-ms Delay	40-ms Delay
56 kbps	70	140	210	280
64 kbps	80	160	240	320
128 kbps	160	320	480	560
256 kbps	320	640	960	1280
512 kbps	640	1280	1920*	2560*
768 kbps	1000	2000*	3000*	4000*
1536 kbps	1600*	3200*	4800*	6400*

* Values over 1500 exceed the typical maximum transmit unit (MTU) size of an interface. Fragmentation of sizes larger than MTU does not result in any fragmentation.

Frame Relay LFI Using FRF.12

Cisco IOS Software supports two flavors of Frame Relay LFI. The more popular option, FRF.12, is based on Frame Relay Forum Implementation Agreement 12, with the other option, FRF.11-C, being based on Frame Relay Forum Implementation Agreement 11, Annex C. FRF.12 applies to data VCs, and FRF.11-C applies to voice VCs. Because most Frame Relay VCs are data VCs, and because most service providers do not offer FRF.11 (VoFR) VCs, this section focuses on FRF.12. The final part of the FRF configuration section at the end of this chapter covers the differences between FRF.12 and FRF.11-C.

NOTE Another LFI feature, called *multilink PPP over Frame Relay and ATM*, also provides an option for LFI. This option is suited for environments that use Frame Relay/ATM internetworking and desire to run delay-sensitive applications such as VoIP on slow-speed WAN links.

FRF.12 varies greatly from MLP LFI in terms of how it works with queuing tools. IOS requires that Frame Relay traffic shaping (FRTS) be used to also use FRF.12. Remember all the trivia from Chapter 5, “Traffic Policing and Shaping”? You may recall that FRTS can apply queuing tools to shaping queues associated with each VC, but FRTS only allows a single FIFO queue on the physical interface. When you add FRF.12 to FRTS, however, two interface FIFO output queues are created rather than the single FIFO queue. Figure 7-9 shows the two FIFO interface output queues, called *Dual FIFO* queues, with FTRS and FRF.12.

Figure 7-9 *Interface Dual FIFO Queues with FRTS and FRF.12*

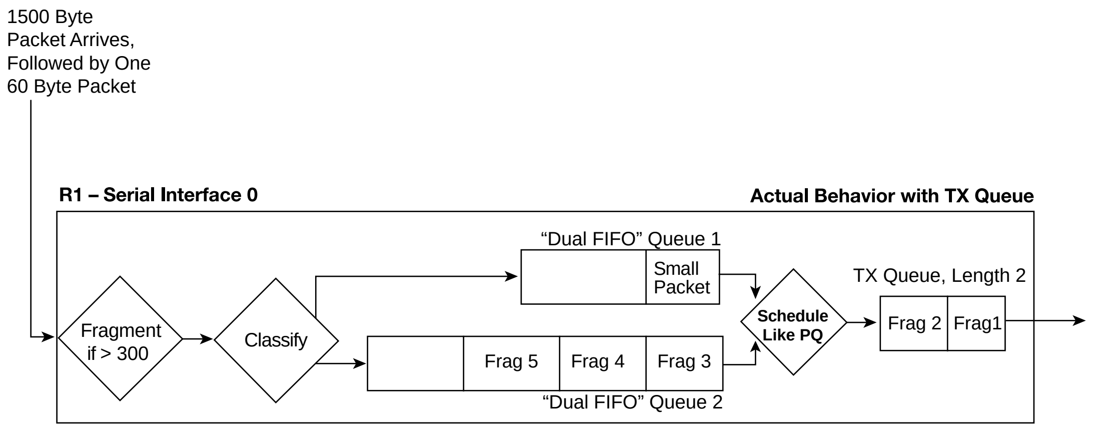


Figure 7-9 focuses on the interface output queues, ignoring the shaping queues. Just like in the figure that depicted MLP LFI, a 1500-byte packet arrives, followed by a 60-byte packet. The large packet is fragmented into five 300-byte packets, with the first two being placed into the TX Queue, and the last three ending up in one of the interface output queues. The small packet arrives next, and it is not fragmented, because it is less than 300 bytes in length. It is placed into the other Dual FIFO queue.

This two-queue Dual FIFO structure acts like the queuing tools described in Chapter 4 in many ways. It has classification logic that places packets into one queue or the other (more on that in a few paragraphs). It has a number of queues (always two), and it has particular behavior inside each queue (FIFO). It also performs scheduling between the two queues using an algorithm such as Priority Queuing's (PQ) scheduling algorithm. Therefore, to understand what happens, you need to take a closer look at the classification logic and the scheduling algorithm applied to the Dual FIFO queues.

First, when a packet passes through the fragmentation step in Figure 7-9, if there are no packets in either Dual FIFO queue, and there is room in the TX Queue/TX Ring, the fragments get placed into the TX Ring/TX Queue until it is full. That's why in Figure 7-9 the first two fragments of the large packet got placed into the 2-entry TX Queue. Then, when the TX Queue is full, packets are placed into one of the two Dual FIFO queues.

IOS schedules packets from the Dual FIFO interface queues into the interface TX Queue in a PQ-like fashion. The logic treats one of the two Dual FIFO queues like the PQ High queue, and the other like the PQ Normal queue. The scheduler always takes packets from the High queue first if one is available; otherwise, the scheduler takes a packet from the Normal queue. Just like PQ, the scheduler always checks the High queue for a packet before checking the Normal queue. Although IOS does not give a lot of information about the two Dual FIFO queues in **show** commands, one command (**show queueing interface**) does list counters for the High and Normal queues. (This book refers to these two queues as the High and Normal Dual FIFO queues, even though most other IOS documents and courses do not even name the two queues.)

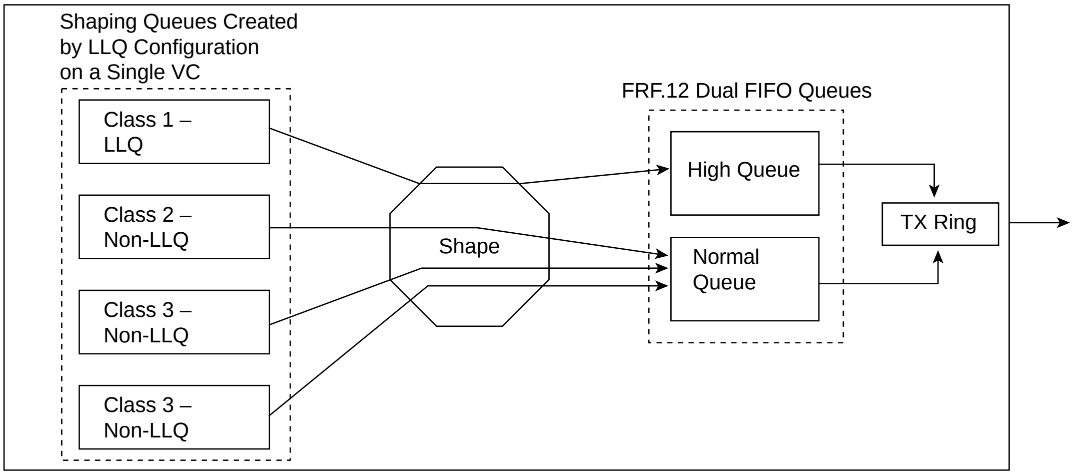
The classification logic with the FRF.12 Dual FIFO queues appears to be very straightforward. One popular school of thought concerns how FRF.12 classifies packets into the Dual FIFO queues, however, and another less-popular school of thought describes how classification really works. Most of the available references, *including the courses on which the exams are based*, state that the classification boils down to this:

- Fragmented packets are placed in the Normal Dual FIFO queue.
- Unfragmented packets are placed in the High Dual FIFO queue.

Putting the classification logic together with the queue service logic makes one neat package. LFI wants to interleave the small packets between fragments of the larger packets. By classifying the unfragmented packets into the High queue, and the fragments into the Normal queue, the PQ-like queue service algorithm interleaves unfragmented packets in front of fragmented packets.

In spite of what the courses actually say, FRF.12 actually classifies packets into one of the Dual FIFO interface output queues based on the queuing configuration for the shaping queues on each VC. FRTS allows a large variety of queuing tools to be configured for the shaping queues. Two of these queuing tools, if enabled on the shaping Queue of a VC, cause packets to be placed in the High Dual FIFO queue on the physical interface. Figure 7-10 outlines the main concept.

Figure 7-10 *Classification Between FRTS LLQ Shaping Queues and Interface Dual FIFO Queues with FRF.12*



The figure depicts LLQ for the shaping queue on a single VC feeding into the interface Dual FIFO queues. The shaping logic remains unchanged, as does the LLQ logic for the shaping queues—in other words, with or without FRF.12 configured, the behavior of shaping acts the same. The only difference created by adding FRF.12 to FRTS comes when FRTS must decide which of the two interface Dual FIFO queues to place the packet into after the shaper allows it to pass. (Without FRF.12, a single FIFO interface queue exists, in which case classification logic is not needed.)

As shown in the figure, the only way a packet makes it to the High Dual FIFO queue is to have first been in the low-latency queue. In other words, FRF.12 determines which packets are interleaved based on which packets were placed into the low-latency queue in the shaping queue.

The classification logic and the scheduling logic make perfect sense if you consider the packets that need the minimal latency. When you purposefully configure LLQ for shaping queues, the class of packets you place into the low-latency queue must be the ones for which you want to minimize latency. FRF.12 should interleave those same packets to further reduce latency; therefore, FRF.12 just places those same packets into the Dual FIFO High queue.

NOTE Because the Dual FIFO queues created by FRF.12 essentially creates a high-priority queue appropriate for VoIP traffic, when you are using FRTS, Cisco also recommends configuring LFI on links that run at speeds greater than 768 kbps. However, you should configure the fragment size to something larger than the MTU—for instance, 1500 bytes. By doing so, no packets are actually fragmented, but VoIP packets can be placed in the high-priority queue in the Dual FIFO queuing system on the physical interface.

FRTS interaction and usage of queuing tools can be difficult to understand, let along trying to keep track of all the options. Interestingly, FRTS supports one set of queuing tools without FRF.12 enabled, and a subset with FRF.12 enabled, and with yet another subset of those (LLQ and IP RTP Priority) that actually interleave the packets. Table 7-9 summarizes the queuing tools and identifies when you can use them with FRTS and FRF.12.

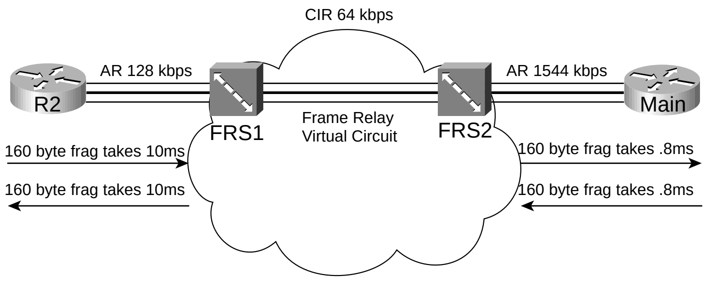
Table 7-9 *Queuing Tool Support with FRTS and FRF.12 (Cisco IOS Software Release 12.2 Mainline)*

Desired Features	**Queuing Tools Supported on Each VC (Shaping Queues)**
FRTS only	FIFO, PQ, Custom Queuing (CQ), Weighted Fair Queuing (WFQ), Class-Based Weighted Fair Queuing (CBWFQ), LLQ, IP RTP Priority
FRTS with FRF.12 enabled	WFQ, CBWFQ, LLQ, IP RTP Priority
FRTS, FRF.12, with actual interleaving of packets	LLQ, IP RTP Priority

Choosing Fragment Sizes for Frame Relay

FRF.12 uses the same basic math as does MLP LFI to determine the fragment size—max-delay * bandwidth. But what do you use for bandwidth in the formula? CIR? Do you use the shaping or policing rate? Or the access rate, which is the clock rate used on the access link? Figure 7-11 shows a typical Frame Relay network that provides a backdrop from which to discuss which values to use.

Figure 7-11 *Example Frame Relay Network Used to Explain How to Choose Fragment Sizes*



In most cases, you should choose to fragment based on the *slower access rate on either end of a VC*, which in this case is the 128-kbps access rate on R1. The reason you should use the lower of the two access rates becomes apparent only when you think of serialization delay inside the cloud, and in both directions. First, consider left-to-right flow in the network. To reduce serialization delay to 10 ms on a 128-kbps link, the fragment size should be set to 160 bytes (128,000 * .01 = 1280 bits). When R2 sends a packet, the serialization delay takes 10 ms.

Moving from left to right, a full-sized fragment sent from FRS1 to the Main router takes only 0.8 seconds to serialize! Reversing the direction, a 160-byte fragment leaving router Main, going into the cloud, only takes 0.8-ms serialization delay, so you might be tempted to make the fragment size much larger for packets sent by router Main. When the packets get to FRS2, however, and need to cross the access link to R1, a small fragment size of 160 bytes gives you an advantage of low serialization delay. If you were to make the fragment size on the Main router a much larger size, the frame would experience a much larger serialization delay on the link from FRS1 to R1.

One common misconception is that fragmentation size should be based on the CIR of the VC, rather than on the access rate. Fragmentation attacks the problem of serialization delay, and serialization delay is based on how long it takes to encode the bits onto the physical interface, which in turn is determined by the physical clock rate on the interface. So, you should always *base FRF.12 fragmentation sizes on the clock rate (access rate) of the slower of the two access links, not* on CIR.

FRF.12 configuration does not let you set the maximum delay, as does MLP LFI. Instead, you configure the fragment size directly. When planning, you normally pick a maximum serialization delay for each link first. So before you can configure FRF.12, you need to calculate the fragmentation size. When you know the lower of the two access rates, and the maximum serialization delay desired, you can calculate the corresponding fragment size with the following formula:

Max-delay * bandwidth

Table 7-8, shown earlier in this section, lists some of the more common combinations of maximum delay and bandwidth, with the resulting fragment sizes.

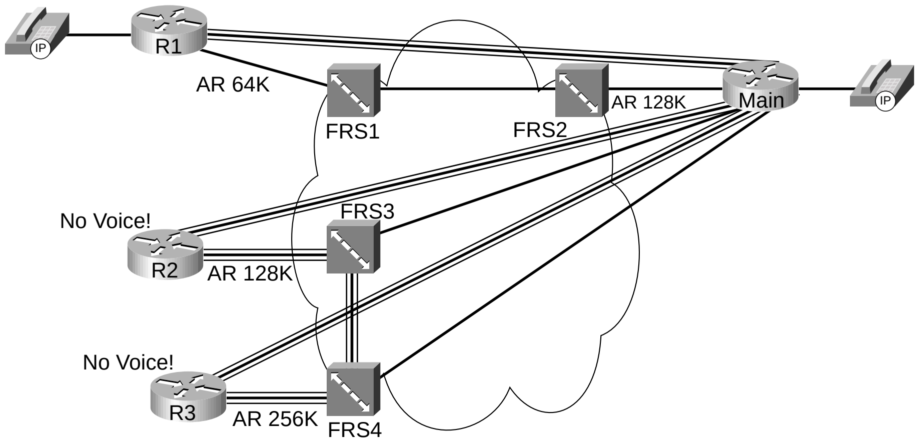
Fragmentation with More Than One VC on a Single Access Link

Cisco IOS Software enables FRF.12 per VC—in other words, you can perform LFI for the traffic on some VCs, but not others. Therefore, before configuring FRF.12, you should decide which VCs really need to use it.

You first need to determine whether any of the Frame Relay VCs need to use LFI. If the traffic on all the VCs can tolerate the delays without performing LFI, you are better off not fragmenting the frames. Fragmentation does increase the bandwidth required in the network slightly, because the fragmentation headers add a few bytes of overhead. Therefore, if serialization delay is not an issue, do not use FRF.12 at all.

Most installations consider LFI when voice is added to the network. If voice is added to all sites in a Frame Relay network, of course you would consider LFI on all the VCs. The interesting choice comes when you have compelling reasons for LFI on some VCs, and not-so-obvious compelling reasons for others. The network in Figure 7-12 shows just such a network, with voice between the Main site and R1, and no voice to the other two remote sites.

Figure 7-12 *Choosing Fragment Sizes*



In the figure, Main terminates three VCs that each share the same physical access link. Because voice traffic traverses the VC from Main to R1, LFI should be enabled on that VC, with the fragment size based on the slower of the two access links on either end of the VC. However, you should also enable LFI on all three VCs that terminate at Main.

The reason for LFI on all three VCs relates to an effect called *egress blocking*. Suppose that R1 sends a small voice packet, with R2 and R3 sending several large, unfragmented data packets. FRS2 receives the large packets first, and queues those packets to exit the access link to the Main router. The small voice packet arrives immediately after the large data packets, so the small voice packet must wait on the large packets to be forwarded.

Depending on the behavior of queuing in the Frame Relay switch, LFI may or may not help with the egress-blocking problem. With LFI on all the VCs, with the same general sequence of events, the small voice packet might just be behind a large number of small fragmented packets, rather than behind a small number of large packets. The Frame Relay switch might use a queuing algorithm that sends some frames from each VC intermittently, however, in which case LFI decreases the delay for the small voice packet. The key here is to understand how the VCs are configured to handle LFI.

For you exam takers out there, the general recommendation in the courses is to fragment on all VCs using a particular interface if at least one VC needs to use FRF. Therefore, in Figure 7-12, the only VC on which Cisco recommends not to bother with FRF.12 is the VC between R2 and R3.

MLP LFI and FRF both accomplish the same general task of reducing serialization delay for some packets. As seen in this chapter, the methods used by each differ in how each tool takes advantage of queuing tools. Table 7-10 summarizes the core functions of MLP LFI versus FRF.12, particularly how they each interact with the available queuing tools.

Table 7-10 *Comparisons Between MLP LFI and FRF.12*

Step in the Process	**MLP LFI**	**FRF.12**
Configures maximum delay, or actual fragment size	Maximum delay	Fragment size
Classification into the interface output queues	Based on the queuing tool enabled on the interface	All packets coming from LLQ or RTP Priority shaping queues placed in higher-priority queue
Number of interface output queues	Based on the queuing tool enabled on the interface	2 queues, called *Dual FIFO*
How queue service algorithm causes interleaving to occur	Based on queuing tool's inherent queue service algorithm; PQ, LLQ, and RTP Priority most aggressively interleave packets	PQ-like algorithm, always servicing High queue over Normal queue

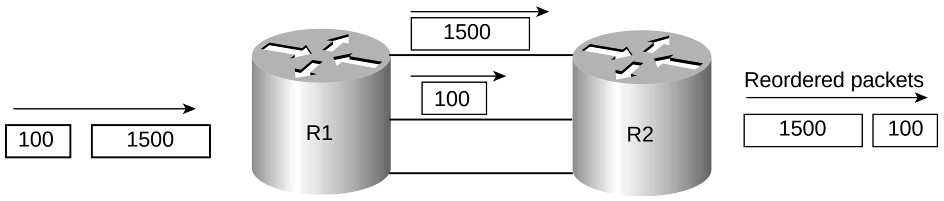
Multilink PPP Interleaving Configuration

Before you configure MLP LFI, think about why you would use MLP at all. If you have a point-to-point link, and need to perform LFI, you must migrate from your current Layer 2 protocol to MLP, to use MLP LFI. However, MLP itself has many benefits, and even a few brief thoughts about what MLP does will help you through some of the configuration tasks.

MLP enables you to have multiple parallel point-to-point links between a pair of devices, such as routers. The main motivation for MLP was to allow dial applications to continue adding additional switched WAN connections between the endpoints when more bandwidth was needed. For instance, maybe one dialed line was brought up, but when the utilization exceeded 60 percent, another line dialed, and then another, and so on.

MLP includes the inherent capability to load balance the traffic across the currently active lines, without causing reordering problems. To understand those concepts, consider Figure 7-13, with three parallel point-to-point links controlled by MLP.

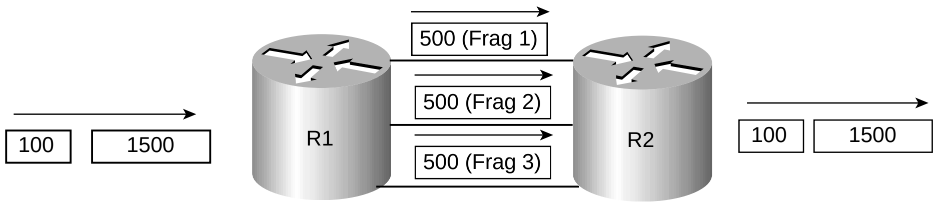
Figure 7-13 *MLP Bundled with 3 Active Links—What Could Happen*



With three active links, MLP could reorder packets. If a 1500-byte packet arrives, for instance, immediately followed by a 100-byte packet, MLP might send the first packet over one link, and the next packet over the second link. Because the second packet is much smaller, its serialization delay will be much smaller. Assuming that both links speeds are equal, the 100-byte packet will arrive before the larger packet. Consequently, the 100-byte packet is forwarded first, before the 1500-byte packet. If both packets are part of the same flow, the endpoint computer may have to do more work to reorder the packets, which TCP could do if it is being used. However, some UDP-based applications could require that the out-of-order packets be re-sent, depending on the application. Over time, the three links will experience different utilization averages, depending on the random occurrence of traffic in the network.

MLP does not behave as shown in Figure 7-13. Instead, MLP, by its very nature, fragments packets. Figure 7-14 shows what really happens.

Figure 7-14 *MLP Bundle with 3 Active Links—What Does Happen*



MLP always fragments PPP frames to load balance traffic equitably and to avoid out-of-order packets. Notice that the 1500-byte packet was fragmented into three 500-byte fragments, one for each link. By default, MLP fragments each packet into equal-sized fragments, one for each link. Suppose, for instance, that two links were active; the fragments would have been 750 bytes long. If four were active, each fragment would have been 375 bytes long. And yes, even the 100-byte packet would be fragmented, with one fragment being sent over each link.

The other point you should consider about basic MLP, before looking at MLP LFI configuration, is that the multiple links appear as one link from a Layer 3 perspective. In the figures, R1 and R2 each have one IP address that applies to all three links. To configure these details, most of the interface subcommands normally entered on the physical interface are configured somewhere else, and then applied to each physical interface that will comprise part of the same MLP bundle.

With those two basic MLP concepts in mind, you can now make more sense of the MLP LFI configuration. Tables 7-11 and 7-12 list the pertinent configuration and **show** commands, respectively, and are followed by some example configurations.

Table 7-11 *Configuration Command Reference for MLP Interleaving*

Command	Mode and Function
ppp multilink [**bap**]	Interface configuration mode; enables multilink PPP on the interface, dialer group, or virtual template
ppp multilink interleave	Interface configuration mode; enables interleaving of unfragmented frames with fragments of larger frames
ppp multilink fragment delay *time*	Interface configuration mode; Enables MLP fragmentation, and defines fragment size, with formula bandwidth/time
ppp multilink fragment disable	Interface configuration mode; Disables MLP fragmentation
ppp multilink group *group-number*	Interface configuration mode; links a physical interface to a dialer group or virtual template

Table 7-12 *Exec Command Reference for MLP Interleaving*

Command	Function
show ppp multilink	Lists information about the active links currently in the same MLP bundle
show interfaces	Lists statistics and status about each interface, including multilink virtual interfaces
show queueing [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]	Lists configuration and statistical information about the queuing tool on an interface

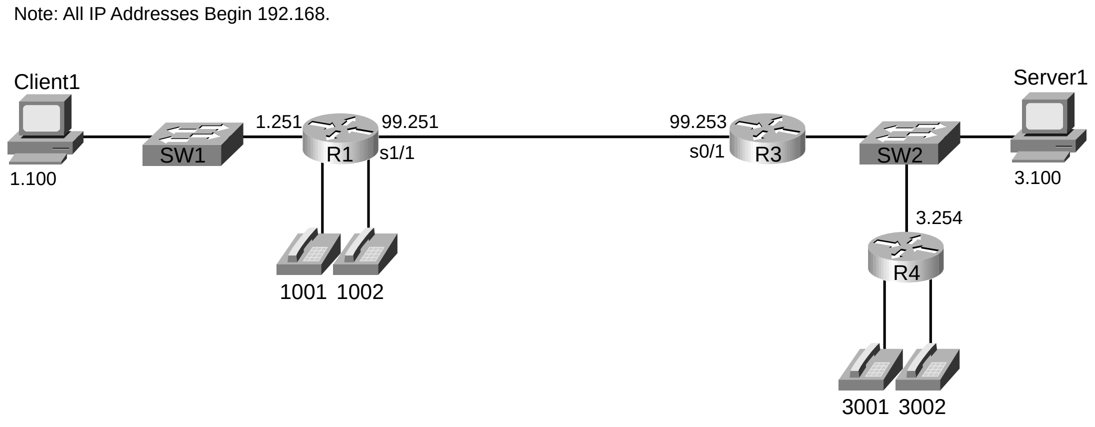
The first MLP LFI example shows a baseline configuration for MLP, without LFI. After this, the additional configuration commands for fragmentation, and then interleave, are added. Finally, the example ends with the addition of the **ip rtp priority** command, which enables one of the forms of queuing with a low-latency feature. The explanation after the example explains how some traffic was interleaved (or not) at each step along the way.

The criteria for the configuration is as follows:

- Clock rate is 128 kbps on the point-to-point link.
- Fragment to 10-ms fragments.
- Use RTP Priority.

In the example, Client 1 downloads two to three web pages, each of which has two frames inside the page. Each web page uses two separate TCP connections to download two separate large JPG files. Client 1 also downloads a file using FTP get. In addition, a VoIP call is placed between extensions 3002 and 1002. Figure 7-15 shows the network used for the example, and Example 7-5 shows the configuration and some sample **show** commands.

Figure 7-15 *Sample Network for MLP LFI Examples*



Example 7-5 *MLP LFI Configuration*

```
!
! STEP 1 STEP 1 STEP 1 STEP 1 STEP 1 STEP 1 STEP 1 STEP 1 STEP 1 STEP 1 STEP 1 STEP 1
!
R3#show running-config
!
! Many lines  omitted for brevity
!
username R1 password 0 me
!
interface Multilink9
 bandwidth 128
 ip address 192.168.99.253 255.255.255.0
 no ip route-cache cef
 load-interval 30
 fair-queue
 no cdp enable
 ppp multilink
 multilink-group 9
!
!
interface Serial0/1
 bandwidth 128
 no ip address
 encapsulation ppp
 no ip mroute-cache
 load-interval 30
 clockrate 128000
 ppp multilink
 multilink-group 9
```

continues

Example 7-5 *MLP LFI Configuration (Continued)*

```
!
!
! STEP 2 STEP 2 STEP 2 STEP 2 STEP 2 STEP 2 STEP 2 STEP 2 STEP 2 STEP 2
!
! Adding Fragmentation for 10ms fragments. Added same command on R1 as well,
! No shown here for brevity.
!
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface multilink 9
R3(config-if)#ppp multilink fragment-delay 10
R3(config-if)#^Z

R3#show interfaces multilink 9
Multilink9 is up, line protocol is up
  Hardware is multilink group interface
  Internet address is 192.168.99.253/24
  MTU 1500 bytes, BW 128 Kbit, DLY 100000 usec,
     reliability 255/255, txload 233/255, rxload 43/255
  Encapsulation PPP, loopback not set
  Keepalive set (10 sec)
  DTR is pulsed for 2 seconds on reset
  LCP Open, multilink Open
  Open: IPCP
  Last input 00:00:02, output never, output hang never
  Last clearing of "show interface" counters 00:20:41
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1056
  Queueing strategy: weighted fair
  Output queue: 64/1000/64/1055 (size/max total/threshold/drops)
     Conversations  5/11/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 96 kilobits/sec
  30 second input rate 22000 bits/sec, 44 packets/sec
  30 second output rate 117000 bits/sec, 48 packets/sec
     4459 packets input, 273353 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     62241 packets output, 5141980 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
!
! STEP 3 STEP 3 STEP 3 STEP 3 STEP 3 STEP 3 STEP 3 STEP 3 STEP 3 STEP 3
!
! Added Interleaving feature. Did same on R1, not shown here.
!
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface multilink 9
R3(config-if)#ppp multilink interleave
R3(config-if)#^Z
```

Example 7-5 *MLP LFI Configuration (Continued)*

```
R3#show interfaces multilink 9
Multilink9 is up, line protocol is up
  Hardware is multilink group interface
  Internet address is 192.168.99.253/24
  MTU 1500 bytes, BW 128 Kbit, DLY 100000 usec,
     reliability 255/255, txload 227/255, rxload 29/255
  Encapsulation PPP, loopback not set
  Keepalive set (10 sec)
  DTR is pulsed for 2 seconds on reset
  LCP Open, multilink Open
  Open: IPCP
  Last input 00:00:00, output never, output hang never
  Last clearing of "show interface" counters 00:22:00
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1406
  Queueing strategy: weighted fair
  Output queue: 27/1000/64/1405/164 (size/max total/threshold/drops/interleaves)
      Conversations  5/11/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 96 kilobits/sec
  30 second input rate 15000 bits/sec, 30 packets/sec
  30 second output rate 114000 bits/sec, 54 packets/sec
     6386 packets input, 386857 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     66702 packets output, 6267446 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
R3#

R3#show queue multilink 9
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1906
  Queueing strategy: weighted fair
  Output queue: 64/1000/64/1905/16500 (size/max total/threshold/drops/interleaves)
     Conversations  5/11/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 96 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/4/0/982
  Conversation 4, linktype: ip, length: 74
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xF513, ttl: 127,
  TOS: 0 prot: 6, source port 80, destination port 1517

  (depth/weight/total drops/no-buffer drops/interleaves) 52/32384/1700/0/0
  Conversation 17, linktype: ip, length: 62
  source: 192.168.3.254, destination: 192.168.99.251, id: 0x08E5, ttl: 253,
  TOS: 0 prot: 17, source port 18490, destination port 17228
!
! Lines omitted for brevity
```

continues

Example 7-5 *MLP LFI Configuration (Continued)*

```
!
!
! STEP 4 STEP 4 STEP 4 STEP 4 STEP 4 STEP 4 STEP 4 STEP 4 STEP 4 STEP 4
!
! Adding RTP Priority Queuing configuration next. Did same on R1.
!
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface multilink 9
R3(config-if)#ip rtp priority 16384 16383 65
R3(config-if)#^Z
R3#

R3#show interfaces multilink 9
Multilink9 is up, line protocol is up
  Hardware is multilink group interface
  Internet address is 192.168.99.253/24
  MTU 1500 bytes, BW 128 Kbit, DLY 100000 usec,
     reliability 255/255, txload 231/255, rxload 41/255
  Encapsulation PPP, loopback not set
  Keepalive set (10 sec)
  DTR is pulsed for 2 seconds on reset
  LCP Open, multilink Open
  Open: IPCP
  Last input 00:00:03, output never, output hang never
  Last clearing of "show interface" counters 00:23:36
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1784
  Queueing strategy: weighted fair
  Output queue: 17/1000/64/1783/4441 (size/max total/threshold/drops/interleaves)
     Conversations  5/11/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 31 kilobits/sec
  30 second input rate 21000 bits/sec, 43 packets/sec
  30 second output rate 116000 bits/sec, 59 packets/sec
     10217 packets input, 618117 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     72195 packets output, 7661717 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions

R3#show queue multilink 9
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1784
  Queueing strategy: weighted fair
  Output queue: 18/1000/64/1783/6643 (size/max total/threshold/drops/interleaves)
     Conversations  6/11/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 31 kilobits/sec
```

Example 7-5 *MLP LFI Configuration (Continued)*

```
   (depth/weight/total drops/no-buffer drops/interleaves) 1/0/0/0/0
   Conversation 40, linktype: ip, length: 62
   source: 192.168.3.254, destination: 192.168.99.251, id: 0x08E5, ttl: 253,
   TOS: 0 prot: 17, source port 18490, destination port 17228

   (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/11/0/1282
   Conversation 0, linktype: ip, length: 74
   source: 192.168.3.100, destination: 192.168.1.100, id: 0xED88, ttl: 127,
   TOS: 0 prot: 6, source port 80, destination port 1513
!
! Lines omitted for brevity
!
!
! STEP 5 STEP 5 STEP 5 STEP 5 STEP 5 STEP 5 STEP 5 STEP 5 STEP 5 STEP 5
!
R3#show running-config
!
! Lines omitted for brevity


!
username R1 password 0 me
!
interface Multilink9
 bandwidth 128
 ip address 192.168.99.253 255.255.255.0
 no ip route-cache cef
 load-interval 30
 fair-queue
 no cdp enable
 ppp multilink
 ppp multilink fragment-delay 10
 ppp multilink interleave
 multilink-group 9
 ip rtp priority 16384 16383 65
!
interface Serial0/1
 bandwidth 128
 no ip address
 encapsulation ppp
 no ip mroute-cache
 load-interval 30
 clockrate 128000
 ppp multilink
 multilink-group 9
!
! Lines omitted for brevity
!
```

Cisco IOS Software enables you to use a couple of styles of configuration to configure MLP. This example shows the use of a virtual interface called a *multilink interface*. MLP needs to make more than one physical point-to-point link behave like a single link. To make that happen, IOS uses multilink interfaces to group together commands that normally would be applied to a physical interface. Each physical interface that is part of the same multilink group takes on the shared characteristics of the multilink interface. For instance, three parallel serial links in the same multilink group share a single IP address on one end of the link.

Example 7-5 is rather long. To help you find your way, the example includes comments lines with Step 1, Step 2, and so on. The following list explains these comments:

Step 1 In the example, multilink group 9 defines the interesting parameters in this example. Under the **interface multilink 9** command, the IP address configuration (192.168.99.253), along with the **bandwidth** command, is listed. The **multilink ppp** command implies that this multilink group indeed uses multilink PPP. The **multilink group 9** command tells IOS that this interface (multilink 9) is part of multilink group 9; this same command links each physical interface to the multilink group configuration. Now the interface configuration details that will be shared by all links in the same MLP bundle have been configured.

To add R3's serial 0/1 interface to the MLP bundle, the **multilink group 9** interface subcommand is added under serial 0/1. After the equivalent configuration has been added to R1, a single leased point-to-point serial link was up between R3 and R1, and each router was able to ping the other across the link.

Step 2 The example next shows the addition of the **ppp multilink fragment-delay 10** command. Interestingly, this command does not enable fragmentation, but rather defines the maximum fragment size. Because the bandwidth was already set to 128, IOS calculates the fragment size as bandwidth * max-delay (in seconds), or 128,000 * .01, which results in a 1280 bit (160 byte) fragment size.

Before the **ppp multilink fragment-delay 10** command was added, in this example, MLP did not fragment. If the **ppp multilink fragment-delay** command had not been added, and four links were active in the MLP bundle, MLP would fragment frames into four equal-sized fragments. If three links were active, each frame would be fragmented into three equal-sized fragments, and so on. With one active link, MLP does not actually fragment the frames until the **ppp multilink fragment-delay** command is added.

After adding the **ppp multilink fragment-delay 10** command in the example, the voice-call quality did not improve. So far, the configuration asked for fragmentation, but not for interleaving. Without interleaving, the unfragmented packets still must wait on all the fragments of larger packets to be serialized. In the context of QoS, it seems rather silly not to automatically interleave the shorter, unfragmented frames. Remember, however, that MLP fragments frames to load balance across multiple links without running into the problems relating to out-of-order packets.

Step 3 To gain the QoS advantage of reducing serialization delay by interleaving packets, the **ppp multilink interleave** command is added to the configuration next. The **show interfaces** command that follows lists a (highlighted) line that now shows a counter for interleaved packets, with the counter indeed showing that some packets have been interleaved.

Now the example has added all the requirements for MLP LFI, and all seems well—but it is not! The voice quality is still barely tolerable, with long delay and many breaks in the speech. The voice quality still suffers because of the queuing tool, WFQ. Notice that the next command in the example, **show queue multilink 9**, lists a voice flow with some statistics highlighted for the voice flow. The command lists drops for the highlighted voice flow, which is making a large impact of voice quality. Although the voice packets that get serviced do get interleaved, causing the interleave counter in the **show interfaces** command to increment, the quality still suffers because of the queuing delay and drops.

Step 4 The best ways to prevent the drops is to enable Low Latency Queuing (LLQ) for the voice traffic. Just to have another example of IP RTP Priority for practice, I chose to use the IP RTP Priority feature, enabling it with the **ip rtp priority 65** command. However, LLQ is still recommended today for voice traffic. After adding the **ip rtp priority** command, the **show interfaces** command still shows interleaves, which is good, but the **show queue** command does not show any drops for the voice flow. In fact, the voice-call quality improved significantly to the point that all New Jersey housewives would give the call a Mean Opinion Score of 5!

Step 5 The final configuration on R3 is listed at the end of the example for completeness.

Frame Relay Fragmentation Configuration

The configuration of FRF.12 requires very little effort in itself. However, FRF.12 requires FRTS, and for the FRF.12 interleaving function to actually work, you need to enable IP RTP Priority or LLQ for the shaping queues on one or more VCs. Therefore, although the FRF.12 new configuration details are brief, the related tools make the configuration a little longer.

The **show** commands related to FRF.12 give a fairly detailed view into what is actually happening with fragmentation and are covered as part of a couple of examples. Tables 7-13 and 7-14 list the configuration and **show** commands, respectively, and are followed by two FRF.12 examples.

Table 7-13 *Command Reference for Frame Relay Fragmentation*

Command	Mode and Function
frame-relay traffic-shaping	Interface subcommand; enables FRTS on the interface
class *name*	Interface DLCI subcommand; enables a specific FRTS map class for the DLCI
frame-relay class *name*	Interface or subinterface command; enables a specific FRTS map class for the interface or subinterface
map-class frame-relay *map-class-name*	Global configuration mode; Names a map class, and places user in map-class configuration mode.
frame-relay fragment *fragment_size*	Map-class configuration mode; enables FRF.12 for VCs using this class

Table 7-14 *Exec Command Reference for Frame Relay Fragmentation*

Command	Function
show frame-relay fragment [**interface** *interface*] [*dlci*]	Shows fragmentation statistics
show frame-relay pvc [*interface-type interface-number*] [*dlci*]	Shows statistics about overall performance of a VC
show queueing [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]	Lists configuration and statistical information about the queuing tool on an interface

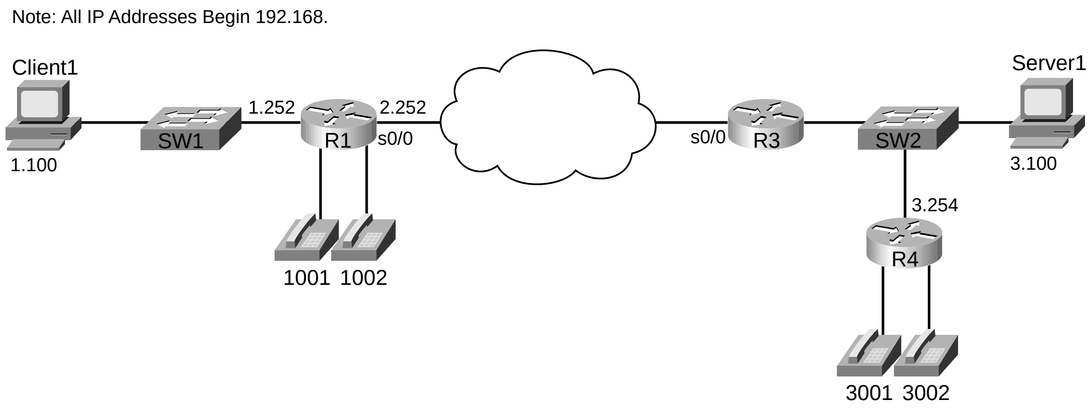
The FRF.12 sample configuration that follows uses the same FRTS configuration details as one of the FRTS examples from Chapter 5, but with the FRF.12 configuration added. The criteria for the configuration is as follows:

- Clock rate is 128 kbps on the access links on each end of the VC between R3 and R1.
- Fragment to 10-ms fragments.
- Shape all traffic at a 64-kbps rate.
- Configure Tc for 10ms.

- Do not use a Be.
- Enable the configuration on the subinterface.
- Do not specify a particular queuing method for the shaping queue.

In the example, Client 1 downloads two to three web pages, each of which has two frames inside the page. Each web page uses two separate TCP connections to download two separate large JPG files. Client 1 also downloads a file using FTP get. In addition, a VoIP call is placed between extensions 3002 and 1002. Figure 7-16 shows the network used for the example, and Example 7-6 shows the configuration and some sample **show** commands.

Figure 7-16 *Sample Network for FRF.12 Configuration*



Example 7-6 *FRF.12 Configuration Sample*

```
R3#show running-config
!
! Many lines  omitted for brevity
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 clockrate 128000
 bandwidth 128
 frame-relay traffic-shaping
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
 ip address 192.168.2.253 255.255.255.0
 frame-relay class shape-all-64
```

continues

Example 7-6 *FRF.12 Configuration Sample (Continued)*

```
 frame-relay interface-dlci 101 IETF
!
interface Serial0/0.2 point-to-point
 description point-to-point subint connected to DLCI 102 (R2)
 ip address 192.168.23.253 255.255.255.0
 frame-relay interface-dlci 102
!
!
! Many lines omitted for brevity
!
map-class frame-relay shape-all-64
 frame-relay traffic-rate 64000 640
 no frame-relay adaptive-shaping
 frame-relay fair-queue
 frame-relay fragment 160
!

R3#show frame-relay fragment interface s 0/0.1 101

 fragment size 160                      fragment type end-to-end
 in fragmented pkts 52                  out fragmented pkts 9367
 in fragmented bytes 5268               out fragmented bytes 1511866
 in un-fragmented pkts 5552             out un-fragmented pkts 6320
 in un-fragmented bytes 341743          out un-fragmented bytes 405268
 in assembled pkts 5577                 out pre-fragmented pkts 7387
 in assembled bytes 346749              out pre-fragmented bytes 1862784
 in dropped reassembling pkts 0         out dropped fragmenting pkts 0
 in timeouts 0
 in out-of-sequence fragments 0
 in fragments with unexpected B bit set 0
 in fragments with skipped sequence number 0
 out interleaved packets 0

R3#show frame-relay fragment
interface         dlci  frag-type    frag-size  in-frag    out-frag   dropped-fr
ag
Serial0/0.1       101   end-to-end   160        54         9700       0

R3#show frame-relay pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

              Active     Inactive      Deleted       Static
  Local          2            0            0            0
  Switched       0            0            0            0
  Unused         0            0            0            0

DLCI = 101, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0.1

  input pkts 6094          output pkts 7926         in bytes 379217
  out bytes 1998660        dropped pkts 0           in FECN pkts 0
  in BECN pkts 0           out FECN pkts 0          out BECN pkts 0
```

Example 7-6 *FRF.12 Configuration Sample (Continued)*

```
  in DE pkts 0             out DE pkts 0
  out bcast pkts 31        out bcast bytes 2416
  pvc create time 00:25:19, last time pvc status changed 00:15:50
R3#show interface s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 20/255, rxload 4/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  14, LMI stat recvd 14, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 65/0, interface broadcasts 61
  Last input 00:00:03, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:02:20
  Queueing strategy: dual fifo
  Output queue: high size/max/dropped 0/256/0
  Output queue 77/128, 176 drops; input queue 0/75, 0 drops
  30 second input rate 26000 bits/sec, 51 packets/sec
  30 second output rate 122000 bits/sec, 126 packets/sec
     6599 packets input, 409250 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     17824 packets output, 2169926 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

R3#show queueing interface s 0/0
Interface Serial0/1 queueing strategy: priority

Output queue utilization (queue/count)
        high/0 medium/0 normal/16513 low/0
!
! Many lines  omitted for brevity
!
```

Take a look at the highlighted configuration at the beginning of the example. FRF is configured with the **frame-relay fragment** command inside a **map-class** command. In this case, the fragment size was set to 160 bytes, which at a clock rate of 128 kbps implies 10 ms of serialization delay for the longest frame. The rest of the configuration enables FRTS on serial 0/0.1, which enables fragmentation, because the **frame-relay fragment 160** command sits inside **map-class frame-relay shape-all-64**. To review, FRTS is enabled for all VCs on interface s0/0 by the **frame-relay traffic-shaping** command on the physical interface. To use the specific parameters in the shape-all-64 map class, the **frame-relay class shape-all-64** command is used under **interface s0/0.1**, causing FRTS to use the shaping parameters in the map class, rather than defaults, for the VC to R1.

The next command in the example, **show frame-relay fragment int s 0/0.1 101,** lists most of the detailed statistics about FRF behavior. It lists counters for input and output packets and bytes as you might expect. In the example, 405,268 bytes have been sent in unfragmented frames, and 1,511,866 bytes in the fragmented frames, for a total of 1,917,134 bytes. It also lists a measurement of the number of output bytes that would have been sent had fragmentation not been used, as shown in the counters labeled "pre-fragmented." The number of prefragmented bytes, listed as 1,862,784 in the command, implies that R3 sent about 55,000 more bytes than it otherwise would have had to send had fragmentation not been used.

The shorter version of the same command, **show frame-relay fragment**, lists the basic configuration settings and just the counter of input and output fragmented packets.

The **show frame-relay pvc** command lists statistics for each VC, but the counters represent values before fragmentation occurs. To see how the counters in this command and the **show frame-relay fragment** command compare, examine the highlighted value for packets sent, and the value of prefragmented output packets in the **show frame-relay fragment** command. The **show frame-relay pvc** command lists 7926 packets sent. The **show frame-relay fragment** command lists 7387 prefragmented packets sent, which is only a few less than what was seen in the **show frame-relay pvc** command that was typed just a few seconds later. Therefore, the **show frame-relay pvc** command counters are based on the packets before fragmentation.

Next, the **show interfaces serial 0/0** command lists one last tidbit of insight into FRF operation. The highlighted portion of the command output lists the queuing method as Dual FIFO. As explained earlier, FRF causes two physical interface output queues to be used, as shown in Figure 7-9. The High queue gets PQ-like treatment, which is how FRF interleaves frames between the fragments in the other FRF output queue.

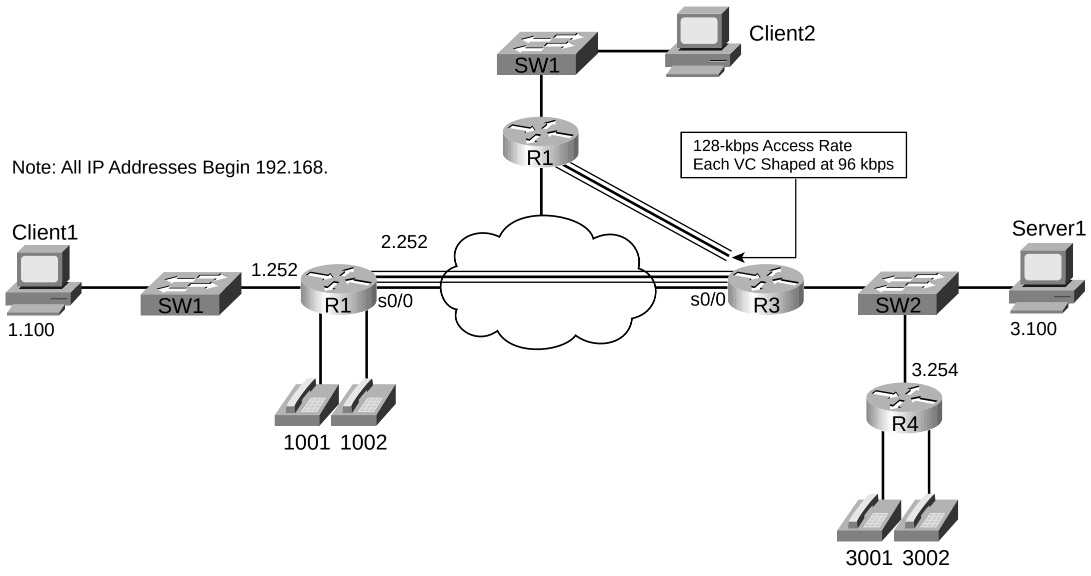
The final command in the example actually drives home two essential points. The **show queueing interface serial 0/0** command lists information about queuing on interface serial 0/0, just as was seen many times in Chapter 4. In this case, it describes the queuing on the physical interface, which we call Dual FIFO. Notice, however, that the command lists the queuing method as "priority," and it lists counters for four queues, named High, Medium, Normal, and Low. So although the queuing method is called Dual FIFO, it behaves like PQ, with only two queues in use.

You may have noticed that the only queue with nonzero counters in the **show queueing** command is the Normal queue. With FRF.12, the only way to get a packet into the High interface queue is to use IP RTP Priority or LLQ on a shaping queue. In the previous example, the default queuing tool, WFQ, is used on the shaping queues. The next example shows a more typical configuration, with LLQ in use. Networks that need FRF.12 most often are those supporting voice traffic. These same networks need to use LLQ in the shaping queues to help reduce delay and jitter, and use FRF to reduce serialization delay. Shaping should also be tuned with a low

Tc value, to reduce the delay waiting for the next shaping interval. The next example uses a class map called shape-all-96-shortTC, which includes FRF.12, LLQ, with shaping using a 10-ms Tc.

This next example also oversubscribes the access link from R3 to the FR cloud. When supporting voice, Cisco recommends to not oversubscribe an access link. However, many data-oriented Frame Relay designs purposefully oversubscribe the access links. Therefore, when the time comes to add VoIP traffic, increasing the CIRs on all the VCs may not be financially possible. This example uses two VCs, with each shaped at 96 kbps, with a 128-kbps access rate. Figure 7-17 outlines the network.

Figure 7-17 *Frame Relay Network with the R3 Access Link Oversubscribed*



The criteria for the example is as follows:

- Clock rate is 128 kbps on all access links.
- Fragment to 10-ms fragments.
- Shape all VCs at a 96-kbps rate.
- Set Tc to 10 ms.
- Do not use a Be.
- Configure LLQ, with VoIP in the low-latency queue, and all other traffic in another queue.

In the example, Client 1 and Client 2 each download one web page, which has two frames inside the page. Each page download uses two separate TCP connections to download two separate large JPG files. Both PCs also download a file using FTP get. In addition, a VoIP call is placed between extensions 3002 and 1002. Figure 7-17 depicts the network used in the example. Example 7-7 shows the configuration and some sample **show** commands.

Example 7-7 *Oversubscribed Access Link, with FRF.12, LLQ, and Tc = 10 ms*

```
R3#show running-config
!
! Portions omitted for brevity
!
class-map match-all voip-rtp
  match ip rtp 16384 16383
!
!
policy-map voip-and-allelse
  class voip-rtp
    priority 30
  class class-default
   fair-queue
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 clockrate 128000
 bandwidth 128
 frame-relay traffic-shaping
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (
 ip address 192.168.2.253 255.255.255.0
 frame-relay class shape-all-96-shortTC
 frame-relay interface-dlci 101 IETF
!
interface Serial0/0.2 point-to-point
 description point-to-point subint connected to DLCI 102 (R2)
 ip address 192.168.23.253 255.255.255.0
 frame-relay class shape-all-96-shortTC
 frame-relay interface-dlci 102
!
map-class frame-relay shape-all-96-shortTC
 no frame-relay adaptive-shaping
 frame-relay cir 96000
 frame-relay bc 960
 service-policy output voip-and-allelse
 frame-relay fragment 160
!
```

Example 7-7 *Oversubscribed Access Link, with FRF.12, LLQ, and Tc = 10 ms (Continued)*

```
R3#show traffic-shape serial 0/0.1

Interface   Se0/0.1
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
101           96000     120    960       0         10        120       -

R3#show traffic-shape serial 0/0.2

Interface   Se0/0.2
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
102           96000     120    960       0         10        120       -

R3#show frame-relay fragment interface serial 0/0.1 101

 fragment size 160                       fragment type end-to-end
 in fragmented pkts 14                   out fragmented pkts 843
 in fragmented bytes 1386                out fragmented bytes 135638
 in un-fragmented pkts 596               out un-fragmented pkts 1607
 in un-fragmented bytes 35967            out un-fragmented bytes 103064
 in assembled pkts 603                   out pre-fragmented pkts 1706
 in assembled bytes 37283                out pre-fragmented bytes 234764
 in dropped reassembling pkts 0          out dropped fragmenting pkts 0
 in timeouts 0
 in out-of-sequence fragments 0
 in fragments with unexpected B bit set 0
 in fragments with skipped sequence number 0
 out interleaved packets 662

R3#show frame-relay fragment interface serial 0/0.2 102
 fragment size 160                       fragment type end-to-end
 in fragmented pkts 0                    out fragmented pkts 1541
 in fragmented bytes 0                   out fragmented bytes 235482
 in un-fragmented pkts 285               out un-fragmented pkts 27
 in un-fragmented bytes 12764            out un-fragmented bytes 1555
 in assembled pkts 285                   out pre-fragmented pkts 296
 in assembled bytes 12764                out pre-fragmented bytes 227911
 in dropped reassembling pkts 0          out dropped fragmenting pkts 0
 in timeouts 0
 in out-of-sequence fragments 0
 in fragments with unexpected B bit set 0
 in fragments with skipped sequence number 0
 out interleaved packets 0

R3#show policy-map interface serial 0/0.2
 Serial0/0.2: DLCI 102 -

  Service-policy output: voip-and-allelse
```

continues

Example 7-7 *Oversubscribed Access Link, with FRF.12, LLQ, and Tc = 10 ms (Continued)*

```
    Class-map: voip-rtp (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: ip rtp 16384 16383
      Weighted Fair Queueing
        Strict Priority
        Output Queue: Conversation 24
        Bandwidth 30 (kbps) Burst 750 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: class-default (match-any)
      1440 packets, 1275806 bytes
      30 second offered rate 55000 bps, drop rate 0 bps
      Match: any

R3#show queueing interface serial 0/0
Interface Serial0/0 queueing strategy: priority

Output queue utilization (queue/count)
        high/9514 medium/0 normal/34038 low/0
```

The FRF.12 configuration portion of the example is comprised again of a single command, **frame-relay fragment 160**, configured inside **map-class frame-relay shape-all-96-shortTC**. The rest of the configuration meets the other requirements stated before the example. The map class includes a setting of CIR to 96,000 bps, and a Bc of 960 bits, yielding a Tc value of 960/96,000, or 10 ms. The **policy-map voip-and-allelse** command defines LLQ for VoIP, with all other traffic being placed in the class-default queue. The **service-policy output voip-and-allelse** command under **class-map frame-relay shape-all-96-shortTC** enables LLQ for all VCs that use the class. FRTS is of course enabled on the physical interface, with the **frame-relay class shape-all-96-shortTC** subinterface command causing each of the two VCs to use the FRTS parameters in the map class, rather than the FRTS defaults.

Immediately following the configuration in the example, two **show frame-relay traffic-shaping** commands verify the settings made in the configuration. Both show a calculated Tc value of 10 ms, and a Bc of 960.

Next comes a pair of **show frame-relay fragment interface** commands, one for subinterface s 0/0.1, and one for serial interface 0/0.2. On serial 0/0.1, the last line of output points out that interleaves did occur. For interleaves to occur, at least a small amount of congestion must occur on the physical interface. With two VCs shaped at 96 kbps, and a 128-kbps access link, and plenty of generated traffic, some congestion did occur. Notice however that serial 0/0.2 does not show any interleaved packets. All the traffic generated going from R3 to R2 was FTP and HTTP traffic, neither of which gets classified into the low-latency queue. In the **show policy-map interface serial 0/0.2** command that ends the example, for instance, notice that the counters show no packets have been in the low-latency queue. This example shows a perfect case where you have a VC (R3 to R2) that does not really need FRF for itself; FRF should be enabled, however, so that the small packets from other VCs can be interleaved.

The **show queueing interface serial 0/0** command shows incrementing counters for both the High queue and the Normal queue. Because one voice call is using the VC from R3 to R1, the voice packets get placed into the R3-to-R1 VC's low-latency queue, and then placed into the Dual FIFO High queue. (Interestingly, I did a few repetitive **show queueing** commands, once every 5 seconds, and saw the High queue counter increment about 250 packets per 5-second interval. With 20 ms of payload, each G.729 call sends 50 packets per second, so the counters reflected the fact that the voice packets from the low-latency queue were being placed into the Dual FIFO High queue.)

FRF.11-C and FRF.12 Comparison

Most of the coverage of LFI over Frame Relay has focused on FRF.12. However, IOS offers another Frame Relay LFI service called FRF.11-C. You can use each of these two LFI options only when you use particular types of Frame Relay VCs. To appreciate how the two options differ, you first need to understand these two types of VCs.

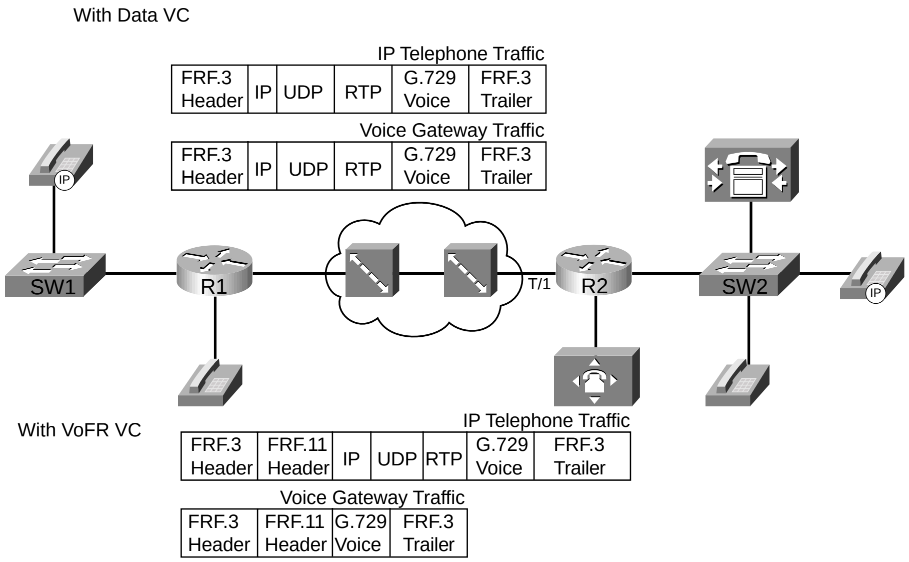
The Frame Relay Forum (FRF) created data VCs originally to carry multiprotocol data traffic, as defined in the FRF.3 Implementation Agreements. Service providers around the world offer Frame Relay FRF.3 data VCs.

Later, with the advent of packetized voice, the Frame Relay Forum decided to create a new type of VC that would allow for better treatment of voice traffic. The FRF created the FRF.11 Implementation Agreement, which defines Voice over Frame Relay (VoFR) VCs. These VCs still pass data traffic, but they also pass voice traffic. Therefore, if a Frame Relay switch knows that one frame is data, and another is voice, for example, the switch can implement some form of queuing to give the voice frame low latency. FRF.11 headers include a field that identifies the frame as voice or data, making it possible for the cloud to perform QoS for the voice traffic.

The key to understanding the difference between the two basic types of VCs is to look at the headers used when the frames cross the Frame Relay network. It helps to understand when VoFR VCs can be useful, and when they cannot. Figure 7-18 shows the framing when FRF.3 and FRF.11 are used, both for IP telephony traffic and for local voice gateway traffic.

In all cases in the figure, G.729 codecs are used. With FRF.3 VCs, the IP Phone and voice gateway traffic sits inside IP/UDP/RTP headers. In other words, the IP Phones encapsulate the G.729 payload using VoIP, and the routers do the same for the analog and digital voice trunks. Although the packets traverse the Frame Relay network, all the voice traffic is considered to be VoIP traffic when you use FRF.3 data VCs.

Figure 7-18 *Framing of Voice Traffic with FRF.3 and FRF.11 VCs*


With Data VC
IP Telephone Traffic
FRF.3 Header | IP | UDP | RTP | G.729 Voice | FRF.3 Trailer
Voice Gateway Traffic
FRF.3 Header | IP | UDP | RTP | G.729 Voice | FRF.3 Trailer
IP
SW1
R1
T/1
R2
SW2
IP
With VoFR VC
IP Telephone Traffic
FRF.3 Header | FRF.11 Header | IP | UDP | RTP | G.729 Voice | FRF.3 Trailer
Voice Gateway Traffic
FRF.3 Header | FRF.11 Header | G.729 Voice | FRF.3 Trailer

With FRF.11 VCs, some voice traffic can be VoIP, and some can be VoFR traffic. The traffic to and from the directly attached analog and digital voice links can be encapsulated using VoFR, as shown in the lowest of the four example frames. The IP telephony traffic, however, still must be encapsulated first in IP, because the traffic must pass across other links besides this single Frame Relay cloud. The VoFR encapsulation requires far less overhead, because the IP, RTP, and UDP headers are not needed. However, VoFR you can use encapsulation only when the Frame Relay-attached routers are the endpoints for the packets holding the voice payload. Because the larger percentage of packetized voice over time will be from IP Phones and the like, VoFR services are not typically offered by Frame Relay providers.

FRF.11-C provides for LFI over VoFR VCs, similarly to how FRF.12 provides LFI services for FRF.3 data VCs. Just like FRF.12, FRF.11-C uses Dual FIFO interface output queues, with PQ logic applied to the two queues. However, FRF.11-C uses a different classification logic, as follows:

- VoFR frames are placed into the High queue on the interface.
- All data frames are placed into the Normal queue.

In the preceding figure, the VoFR frames created for the voice gateway would be placed in the High queue, but the IP Phone packets would be placed into the Normal queue, because they would be considered to be data.

The other main difference between FRF.11-C and FRF.12 has to do with how the tools decide what to fragment, and what not to fragment. FRF.12 fragments all packets over a certain length. FRF.11-C, however, never fragments VoFR frames, even if they are larger than the fragment size. FRF.11-C fragments data frames only, and only if they are larger than the fragment size.

Table 7-15 summarizes some of the key comparison points about FRF.12 and FRF.11-C.

Table 7-15 *FRF.11-C and FRF.12 Comparison*

Function	FRF.12 Behavior	FRF.11-C Behavior
Queuing option on the interface output queues	Dual FIFO	Dual FIFO
Classification into the interface output queues	Based on queuing tool used for shaping, with LLQ and IP RTP Priority putting packets into the high-priority queue	Voice frames placed in High queue, all others in Normal queue, regardless of shaping queue configuration
Fragmentation based on size or type of packet	Based only on size; must be careful not to fragment voice packets	Non-VoFR (VoIP and data) frames fragmented if they exceed the fragmentation size; voice frames are not, regardless of size
Frame Relay network aware of voice vs. nonvoice frames, and acts accordingly	No	Yes
Underlying type of VC, and general public availability.	FRF.3, available from most if not all public Frame Relay services	FRF.11, not generally available from public Frame Relay services

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures are a convenient way to review the day before the exam.

Figure 7-19 shows the fields compressed by payload compression, and by both types of header compression. (The abbreviation "DL" stands for data link, representing the data-link header and trailer.)

Figure 7-19 *Payload and Header Compression*

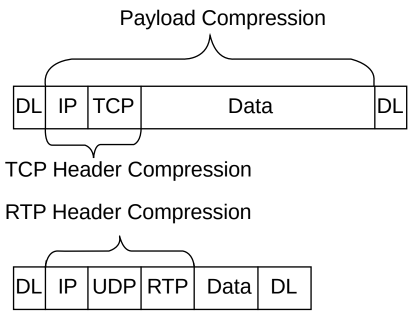


Table 7-16 outlines the main points of comparison for the three payload compression tools.

Table 7-16 *Point-to-Point Payload Compression tools—Feature Comparison*

Feature	**Stacker**	**MPPC**	**Predictor**
Uses Lempel-Ziv (LZ) compression algorithm	Yes	Yes	No
Uses Predictor public domain compression algorithm	No	No	Yes
Supported on High-Level Data Link Control (HDLC)	Yes	No	No
Supported on X.25	Yes	No	No
Supported on Link Access Procedure, Balanced (LAPB)	Yes	No	Yes
Supported on Frame Relay	Yes	No	No
Supported on Point-to-Point Protocol (PPP)	Yes	Yes	Yes
Supported on ATM (using multilink PPP)	Yes	Yes	Yes

Table 7-17 list the various configuration and **show** commands used with payload compression.

Table 7-17 *Configuration Command Reference for Payload Compression*

Command	Mode and Function
compress predictor	Interface configuration mode; enables Predictor compression on one end of the link.
compress stac	Interface configuration mode; enables Stacker compression on one end of the link.
compress mppc [**ignore-pfc**]	Interface configuration mode; enables MPPC compression on one end of the link.
compress stac [**distributed** | **software**]	Interface configuration mode; on 7500s with VIPs, allows specification of whether the compression algorithm is executed in software on the VIP.
compress {**predictor** | **stac** [**csa** *slot* | **software**]}	Interface configuration mode; On 7200s, allows specification of Predictor or Stacker compression on a compression service adapter (CSA).
compress stac caim *element-number*	Interface configuration mode; enables Stacker compression using the specified compression AIM.
frame-relay payload-compress {**packet-by-packet** | **frf9 stac** [*hardware-options*] | **data-stream stac** [*hardware-options*]}	Interface configuration mode; enables FRF.9 or data-stream style compression on one end of a Frame Relay link. Hardware-options field includes the following options: software, distributed (for use w/VIPs), and CSA (7200s only).

The TCP and RTP header compression configuration process, as mentioned, is very simple. Tables 7-18 and 7-19, respectively, list the configuration and **show** commands.

Table 7-18 *Configuration Command Reference for TCP and RTP Header Compression*

Command	Mode and Function
ip tcp header-compression [**passive**]	Interface configuration mode; enables TCP header compression on point-to-point links.
ip rtp header-compression [**passive**]	Interface configuration mode; enables RTP header compression on point-to-point links.
frame-relay ip tcp header-compression [**passive**]	Interface/subinterface configuration mode; enables TCP header compression on point-to-point links.
frame-relay ip rtp header-compression [**passive**]	Interface or subinterface configuration mode; enables RTP header compression on point-to-point links.

continues

Table 7-18 *Configuration Command Reference for TCP and RTP Header Compression (Continued)*

Command	Mode and Function
frame-relay map ip *ip-address dlci* [**broadcast**] **tcp header-compression** [**active** | **passive**] [**connections** *number*]	Interface or subinterface configuration mode; enables TCP header compression on the specific VC identified in the **map** command.
frame-relay map ip *ip-address dlci* [**broadcast**] **rtp header-compression** [**active** | **passive**] [**connections** *number*]	Interface or subinterface configuration mode; enables RTP header compression on the specific VC identified in the **map** command.

Table 7-19 *Exec Command Reference for TCP and RTP Header Compression*

Command	Function
show frame-relay ip rtp header-compression [**interface** *type number*]	Lists statistical information about RTP header compression over Frame Relay; can list information per interface
show frame-relay ip tcp header-compression	Lists statistical information about TCP header compression over Frame Relay
show ip rtp header-compression [*type number*] [**detail**]	Lists statistical information about RTP header compression over point-to-point links; can list information per interface
show ip tcp header-compression	Lists statistical information about TCP header compression over point-to-point links

LFI tools attack the serialization delay problem by breaking the large packets into smaller pieces (fragmentation), and then sending the smaller frames ahead of most of the new fragments of the original large frame (interleaving). Figure 7-20 outlines the basic process.

Figure 7-20 *Basic Concept Behind LFI Tools*

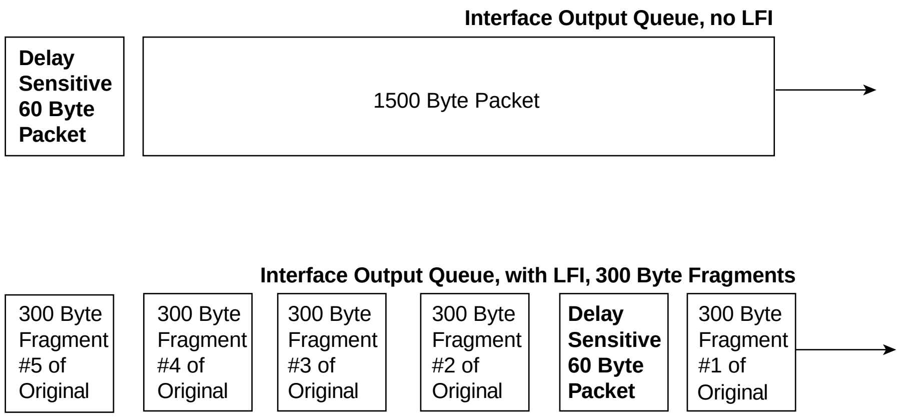
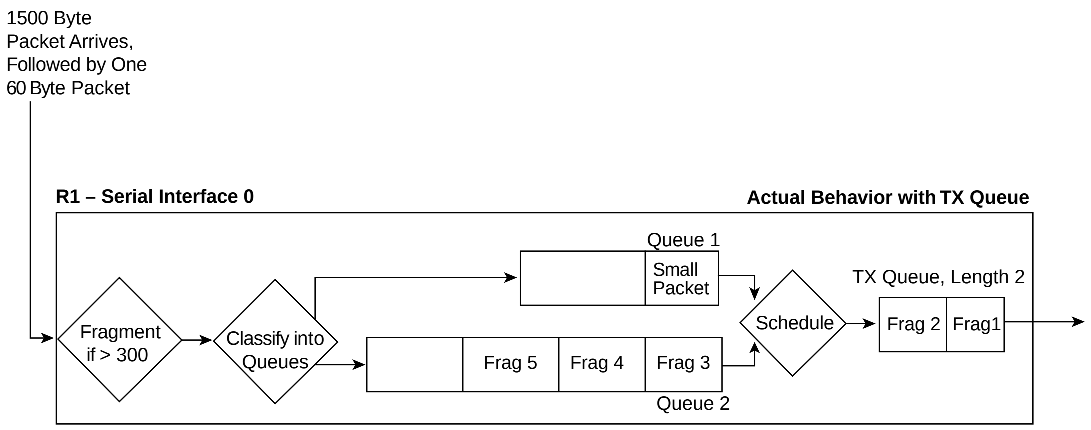

Interface Output Queue, no LFI
Delay Sensitive 60 Byte Packet
1500 Byte Packet
Interface Output Queue, with LFI, 300 Byte Fragments
300 Byte Fragment #5 of Original
300 Byte Fragment #4 of Original
300 Byte Fragment #3 of Original
300 Byte Fragment #2 of Original
Delay Sensitive 60 Byte Packet
300 Byte Fragment #1 of Original

Figure 7-21 depicts how MLP LFI works with a queuing tool on an interface.

Figure 7-21 *MLP LFI Interaction with Queuing*



For perspective, Table 7-20 summarizes the calculated fragment sizes based on the bandwidth and maximum delay.

Table 7-20 *Fragment Sizes Based on Bandwidth and Serialization Delay*

Bandwidth/Link Speed	10-ms Delay	20-ms Delay	30-ms Delay	40-ms Delay
56 kbps	70	140	210	280
64 kbps	80	160	240	320
128 kbps	160	320	480	560
256 kbps	320	640	960	1280
512 kbps	640	1280	1920*	2560*
768 kbps	1000	2000*	3000*	4000*
1536 kbps	1600*	3200*	4800*	6400*

* Values over 1500 exceed the typical maximum transmit unit (MTU) size of an interface. Fragmentation of sizes larger than MTU does not result in any fragmentation.

Two of these queuing tools, if enabled on the shaping queue of a VC, cause packets to be placed in the High Dual FIFO queue on the physical interface. Figure 7-22 outlines the main concept.

Figure 7-22 *Classification Between FRTS LLQ Shaping Queues and Interface Dual FIFO Queues with FRF.12*

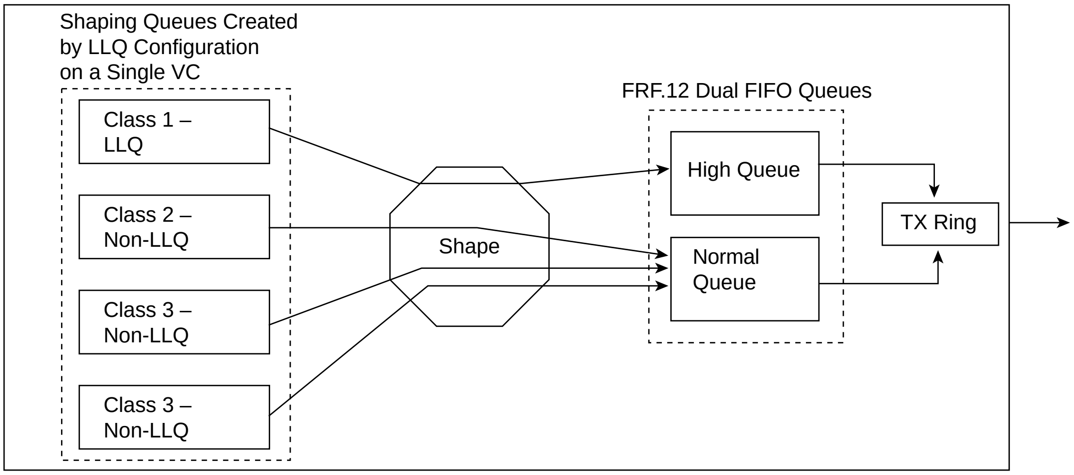


Table 7-21 summarizes the queuing tools and identifies when you can use them with FRTS and FRF.12:

Table 7-21 *Queuing Tool Support with FRTS and FRF.12 (IOS 12.2 Mainline)*

Desired Features	**Queuing Tools Supported on Each VC (Shaping Queues)**
FRTS only	FIFO, PQ, Custom Queuing (CQ), Weighted Fair Queuing (WFQ), Class Based Weighted Fair Queuing (CBWFQ), LLQ, IP RTP Priority
FRTS with FRF.12 enabled	WFQ, CBWFQ, LLQ, IP RTP Priority
FRTS, FRF.12, with actual interleaving of packets	LLQ, IP RTP Priority

Table 7-22 summarizes the core functions of MLP LFI versus FRF.12, particularly how they each interact with the available queuing tools.

Table 7-22 *Comparisons Between MLP LFI and FRF.12*

Step in the Process	**MLP LFI**	**FRF.12**
Configures maximum delay, or actual fragment size	Maximum delay	Fragment size
Classification into the interface output queues	Based on the queuing tool enabled on the interface	All packets coming from LLQ or RTP Priority shaping queues placed in higher-priority queue

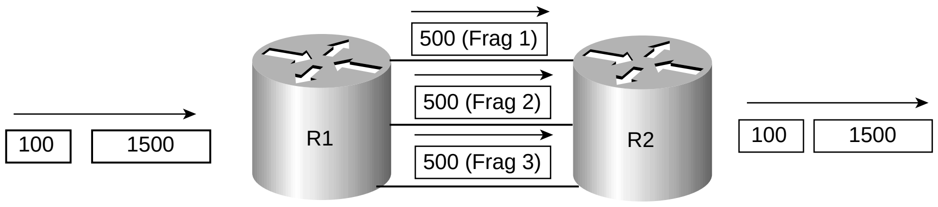
Table 7-22 *Comparisons Between MLP LFI and FRF.12 (Continued)*

Step in the Process	**MLP LFI**	**FRF.12**
Number of interface output queues	Based on the queuing tool enabled on the interface	2 queues, called *Dual FIFO*
How queue service algorithm causes interleaving to occur	Based on queuing tool's inherent queue service algorithm; PQ, LLQ, and RTP Priority most aggressively interleave packets	PQ-like algorithm, always servicing High queue over Normal queue

* The popular theory disagrees with this table. The popular theory states that all unfragmented packets end up in the high-priority queue, and all fragments end up in the Normal queue.

MLP, by its very nature, fragments packets. Figure 7-23 shows what really happens.

Figure 7-23 *MLP Bundle with 3 Active Links—What Does Happen*



Tables 7-23 and 7-24 list the pertinent configuration and **show** commands for MLP interleaving, respectively.

Table 7-23 *Configuration Command Reference for MLP Interleaving*

Command	**Mode and Function**
ppp multilink [**bap**]	Interface configuration mode; enables multilink PPP on the interface, dialer group, or virtual template
ppp multilink interleave	Interface configuration mode; enables interleaving of unfragmented frames with fragments of larger frames
ppp multilink fragment delay *time*	Interface configuration mode; enables MLP fragmentation, and defines fragment size, with formula bandwidth/time
ppp multilink fragment disable	Interface configuration mode; disables MLP fragmentation
ppp multilink group *group-number*	Interface configuration mode; links a physical interface to a dialer group or virtual template

Table 7-24 *Exec Command Reference for MLP Interleaving*

Command	Function
show ppp multilink	Lists information about the active links currently in the same MLP bundle
show interfaces	Lists statistics and status about each interface, including multilink virtual interfaces
show queueing [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]	Lists configuration and statistical information about the queuing tool on an interface

Tables 7-25 and 7-26 list the configuration and **show** commands for Frame Relay fragmentation, respectively.

Table 7-25 *Command Reference for Frame Relay Fragmentation*

Command	Mode and Function
frame-relay traffic-shaping	Interface subcommand; enables FRTS on the interface.
class *name*	Interface DLCI subcommand; enables a specific FRTS map-class for the DLCI.
frame-relay class *name*	Interface or subinterface command; enables a specific FRTS map-class for the interface or subinterface.
map-class frame-relay *map-class-name*	Global Configuration mode; Names a map-class, and places user in map-class configuration mode
frame-relay fragment *fragment_size*	Map-class configuration mode; enables FRF.12 for VCs using this class

Table 7-26 *Exec Command Reference for Frame Relay Fragmentation*

Command	Function
show frame-relay fragment [**interface** *interface*] [*dlci*]	Shows fragmentation statistics
show frame-relay pvc [*interface-type interface-number*] [*dlci*]	Shows statistics about overall performance of a VC
show queueing [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]	Lists configuration and statistical information about the queuing tool on an interface

Figure 7-24 shows the framing when FRF.3 and FRF.11 are used, both for IP telephony traffic and for local voice gateway traffic.

Figure 7-24 *Framing of Voice Traffic with FRF.3 and FRF.11 VCs*

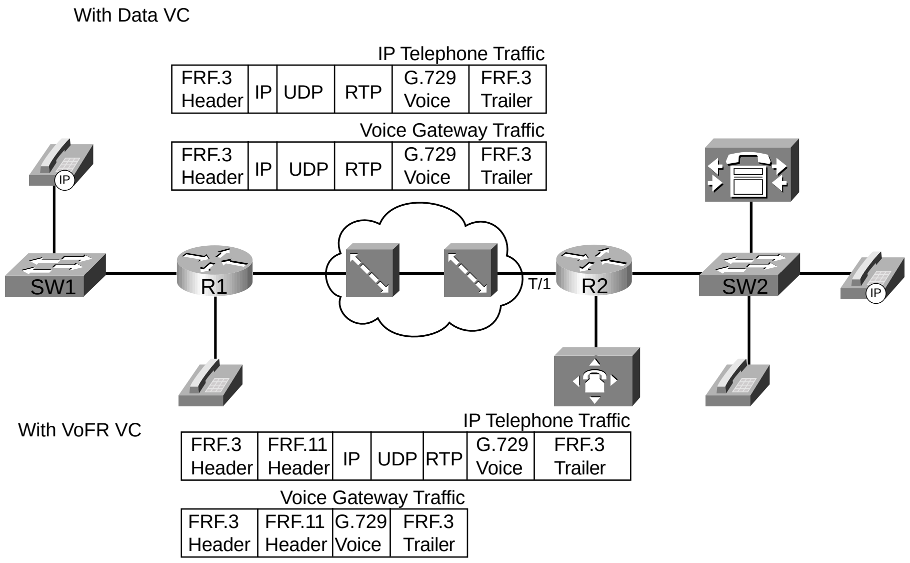

With Data VC
IP Telephone Traffic
FRF.3 Header | IP | UDP | RTP | G.729 Voice | FRF.3 Trailer
Voice Gateway Traffic
FRF.3 Header | IP | UDP | RTP | G.729 Voice | FRF.3 Trailer
IP
SW1
R1
T/1
R2
SW2
IP
With VoFR VC
IP Telephone Traffic
FRF.3 Header | FRF.11 Header | IP | UDP | RTP | G.729 Voice | FRF.3 Trailer
Voice Gateway Traffic
FRF.3 Header | FRF.11 Header | G.729 Voice | FRF.3 Trailer

Table 7-27 summarizes some of the key comparison points about FRF.12 and FRF.11-C.

Table 7-27 *FRF.11-C and FRF.12 Comparison*

Function	**FRF.12 Behavior**	**FRF.11-C Behavior**
Queuing option on the interface output queues	Dual FIFO	Dual FIFO
Classification into the interface output queues	Based on queuing tool used for shaping, with LLQ and IP RTP Priority putting packets into the high-priority queue	Voice frames placed in High queue, all others in Normal queue, regardless of shaping queue configuration
Fragmentation based on size, or type of packet	Based only on size; must be careful not to fragment voice packets	Nonvoice frames fragmented, and voice frames are not, regardless of size
Frame Relay network can be aware of voice vs. nonvoice frames, and acts accordingly	No	Yes
Underlying type of VC, and general public availability	FRF.3, available from most if not all public Frame Relay services	FRF.11, not generally available from public Frame Relay services

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD-ROM included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD-ROM nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD-ROM exam to include, or not include, the topics that are only on the CCIP QoS.

Compression Tools

1. Describe what is compressed, and what is not compressed, when using payload compression. Be as specific as possible regarding headers and data.
2. Describe what is compressed, and what is not compressed, when using TCP header compression. Be as specific as possible regarding headers and data.
3. Describe what is compressed, and what is not compressed, when using RTP header compression. Be as specific as possible regarding headers and data.
4. List the three point-to-point payload compression options available in IOS.
5. Suppose a packet is sent across a network with no compression. Later, a packet of the exact same size and contents crosses the network, but payload compression is used on the one serial link in the network. Describe the difference in bandwidth and delay in the network between these two packets.
6. List the three options for Frame Relay payload compression in IOS. Which of the three is not Cisco proprietary?
7. Which payload compression tool in IOS supports Link Access Procedure, Balanced (LAPB), High-Level Data Link Control (HDLC) and Point-to-Point Protocol (PPP) encapsulations?
8. How much bandwidth should a G.729 call require over Frame Relay, and how much should be required with cRTP?
9. When TCP header compression is used, what is the range of sizes of the part of the frame that can be compressed, and what is the range of sizes for this field of the frame after compression?

10 When RTP header compression is used, what is the range of sizes of the part of the frame that can be compressed, and what is the range of sizes for this field of the frame after compression?

11 To configure Stacker payload compression on a point-to-point link, what command(s) is used, and in what configuration modes?

12 To configure Stacker payload compression on a point-to-point Frame Relay subinterface, what command(s) is used, and in what configuration modes?

13 To configure FRF.9 payload compression on a point-to-point Frame Relay subinterface, what command(s) is used, and in what configuration modes?

14 What command lists compression statistics for payload compression on a point-to-point link?

15 What command lists compression statistics for payload compression on a Frame Relay point-to-point subinterface?

16 To configure TCP header compression on a point-to-point Frame Relay subinterface, what command(s) is used, and in what configuration modes?

17 To configure RTP header compression on a point-to-point link, what command(s) is used, and in what configuration modes?

LFI Tools

18 List the words represented by the abbreviation LFI.

19 Describe the main motivation for LFI tools in relation to the support of data, voice, and video traffic.

20 To achieve a 20-ms serialization delay on a 128-kbps link, how long can the fragments be?

21 To achieve a 10-ms serialization delay on a 64-kbps link, how long can the fragments be?

22 Suppose that a 1500-byte packet exits a 56-kbps serial interface, and LFI is not used. How long is the serialization delay?

23 Which queuing tools can you enable directly on a serial interface when using multilink Point-to-Point Protocol with link fragmentation and interleaving (MLP LFI), as compared to when you are just using PPP?

24 Which queuing tools can you enable for shaping queues when using FRF.12? Which ones actually interleave the traffic?

25 Explain the popularly stated scheduling logic, which is consistent with the Cisco QoS courses, that defines how FRF.12 determines which packets can be interleaved in front of fragments of other packets.

26 Explain the scheduling logic used by MLP LFI to determine which packets can be interleaved in front of fragments of other packets.

27 Suppose a 1500-byte packet arrives and needs to be sent over an MLP bundle that has two active links. LFI has not been configured. Which link does the packet flow across to achieve MLP load balancing?

28 What command can you use to determine the fragment size used for MLP LFI? What is the only parameter of the command?

29 What command enables the interleaving feature of MLP LFI?

30 What commands list counters for the number of interleaved packets using MLP LFI?

31 What other QoS feature for Frame Relay must you enable also when you configure FRF.12?

32 What command enables FRF and sets the fragment size?

33 What command lists counters for the numbers of packets and bytes that were fragmented and unfragmented by FRF.12?

34 What command lists counters for the numbers of packets and bytes that would have been sent if FRF.12 fragmentation had not been performed?

35 How do FRF.12 and FRF.11-C differ in terms of deciding which packets can be interleaved, and which cannot?

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Topics

- Describe what call admission control (CAC) does.
- Correctly list five local CAC methods and their primary function.
- Correctly list two measurement-based CAC methods and their primary function.
- Correctly describe IntServ/RSVP and its main function.
- Given an enterprise network scenario, correctly determine which method(s) of achieving call admission control best meets the customer requirements.

QoS Exam Objectives

- Describe Resource Reservation Protocol (RSVP).
- Configure RSVP on Cisco IOS routers.

CHAPTER 8

Call Admission Control and QoS Signaling

Suppose that you and a passenger are inside a vehicle traveling on a northbound highway. The highway consists of four regular lanes and one high-occupancy-vehicle (HOV) lane. To gain entry into the HOV lane, your vehicle must contain two or more people. In this instance, the HOV lane represents a priority queue, whereas the remaining four lanes represent lower-priority queues. As traffic begins to build in the four regular lanes, you decide to merge into the HOV lane. Over the next few miles/kilometers, the HOV lane maintains a constant rate of speed while the four regular lanes begin to stall. You are now enjoying the benefits of a priority queue. As you continue traveling northbound, you notice more vehicles, meeting the requirements of two or more passengers, begin to merge into the HOV lane. Over the next few miles/kilometers, your speed decreases until the HOV is now stalled.

In this instance each vehicle in the HOV lane has met the criteria to be placed in the high-priority queue. The expectation is that the HOV lane will move quickly. Contrary to these expectations, as more vehicles enter the HOV lane the slower the lane becomes. How do you prevent this issue from arising?

This scenario illustrates the need to have some mechanism in place to limit the amount of traffic that can gain access into the priority queue, ensuring that a consistent flow of traffic can be maintained across a network link. This concept is called *call admission control* (CAC), which is the subject of this chapter.

NOTE For those of you studying for the CCIP QoS 642-641 exam, most of the contents of this chapter are not covered on that exam. The RSVP topics in this chapter are. So, you may want to choose to skip sections of this chapter, and just read the sections covering RSVP. As always, recheck Cisco's posted exam topics to make sure nothing has changed!

NOTE This chapter is based on the VoIP Call Admission Control white paper, which can be found at www.cisco.com/en/US/tech/tk652/tk701/technologies_white_paper09186a00800da46 7.shtml.

"Do I Know This Already?" Quiz

The purpose of the "Do I Know This Already?" quiz is to help you decide whether you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 14-question quiz, derived from the major sections in "Foundation Topics" section of the chapter, helps you determine how to spend your limited study time.

Table 8-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 8-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions	Score
1	Call Admission Control Concepts	1 to 3	
2	Local-Based CAC	4 to 6	
3	Measurement-Based CAC	7 to 9	
4	Resource-Based CAC	10 to 14	
All questions		1 to 14	

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **12 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," "Foundation Summary," and "Q&A" sections.
- **13 or 14 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section, and then go to the "Q&A" section. Otherwise, move to the next chapter.

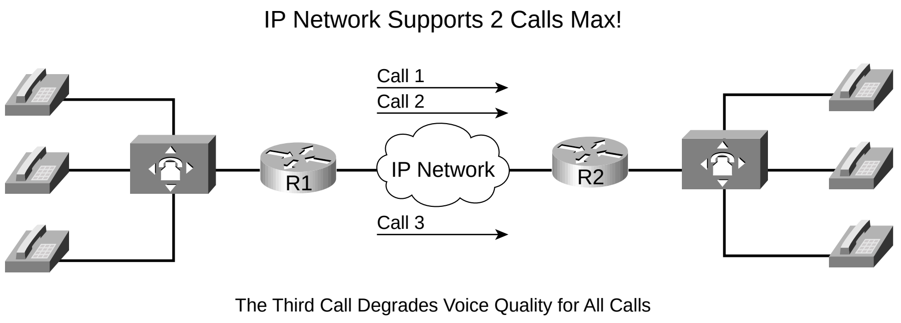
1. Why is call admission control needed in an environment where LLQ has been properly implemented?
2. How does a channel-associated signaling circuit, such as E&M or T1 CAS, react to call admission control?
3. Name four possible measures that a CAC mechanism can take in the event that the resources are not available to proceed with the call.
4. What is the definition of local-based CAC?
5. What Cisco IOS command is used to enable CAC on a VoFR network?
6. What Cisco IOS command is used to enable physical DS0 limitation?
7. What is the definition of measurement-based CAC?
8. What is the difference between an SAA packet and a ping packet?
9. What Cisco IOS command is used to allow the destination node to participate in measurement-based CAC?
10. What is the definition of resource-based CAC?
11. For a gatekeeper to provide CAC, what must be configured for each link that requires protection?
12. What is IntServ and how does it work?
13. Name the messages used by RSVP to provide resource reservation and CAC.
14. What level of service must be in place to provide CAC?

Foundation Topics

Call Admission Control Overview

Call admission control (CAC) mechanisms extend the capabilities of other quality of service (QoS) methods to ensure that voice traffic across the network link does not suffer latency, jitter, or packet loss that can be introduced by the addition of other voice traffic. CAC achieves this task by determining whether the required network resources are available to provide suitable QoS for a new call, before the new call is placed. Simply put, CAC protects voice conversations from other voice conversations. Figure 8-1 demonstrates the need for CAC. In this example, if the WAN link between the two private branch exchanges (PBXs) has sufficient bandwidth to carry only two Voice over IP (VoIP) calls, admitting the third call will impair the voice quality of all three calls.

Figure 8-1 *VoIP Network Without CAC*


IP Network Supports 2 Calls Max!
Call 1
Call 2
R1
IP Network
R2
Call 3
The Third Call Degrades Voice Quality for All Calls

Similar to the earlier HOV example, the reason for this impairment is that the employed queuing mechanisms do not provide CAC. If packets exceeding the configured or budgeted rate are received in the priority queue, in this case more than two calls, these packets are just tail dropped from the priority queue. There is no capability in the queuing mechanisms to distinguish which IP packet belongs to which voice call. As mentioned in Chapter 4, “Congestion Management,” both Low Latency Queuing (LLQ) and IP RTP Priority police traffic inside the low-latency queue when the interface is congested, so any packet exceeding the configured rate within a certain period of time is dropped. In this event, all three calls will experience packet loss and jitter, which can be perceived as clipped speech or dropped syllables in each of the voice conversations.

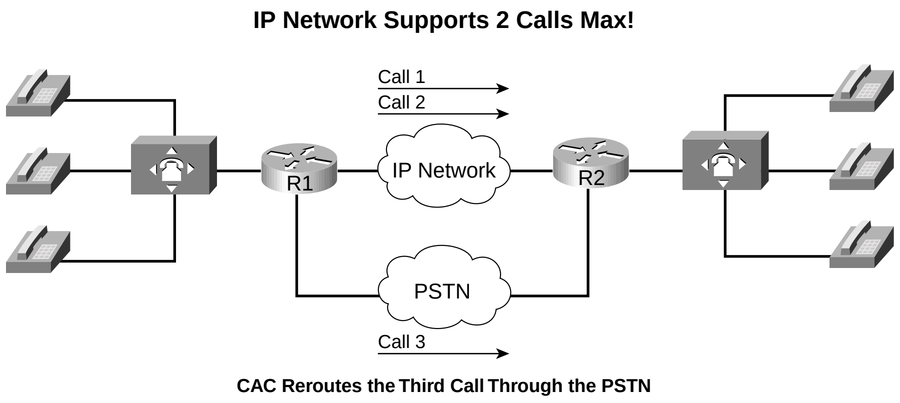
The addition of CAC preserves the quality of the voice conversations in progress by rejecting a new call when insufficient network resources are available to allow the new call to proceed.

Call Rerouting Alternatives

If a call has been rejected by a CAC mechanism due to insufficient network resources, there needs to be some alternate route in place to establish the call. In the absence of an alternate route, the caller will hear a reorder tone. The reorder tone is called a *fast-busy* tone in North America, and is known as *overflow tone* or *equipment busy* in other parts of the world. This tone is often intercepted by Public Switched Telephone Network (PSTN) switches or PBXs with an announcement such as “All circuits are busy, please try your call again later.”

Figure 8-2 illustrates an originating gateway, router R1, with CAC configured to reroute a call to the PSTN when insufficient network resources are available to route the call over the WAN link.

Figure 8-2 *Legacy VoIP Network with CAC*


IP Network Supports 2 Calls Max!
Call 1
Call 2
R1
IP Network
R2
PSTN
Call 3
CAC Reroutes the Third Call Through the PSTN

In a legacy VoIP environment, also known as a *toll-bypass environment*, the configuration of the originating gateway determines where the call is rerouted. The following scenarios can be configured:

- **Alternate WAN path**—The call can be rerouted to take advantage of an alternate WAN link if such a path exists. This is accomplished by configuring a second VoIP dial peer with a higher preference than the primary VoIP dial peer. When the primary VoIP dial peer rejects the call, the second VoIP dial peer is matched, causing the call to use the alternate WAN link.
- **Alternate PSTN path**—The call can be rerouted to take advantage of an alternate time-division multiplexing (TDM) network path if such a path exists. This is accomplished by configuring a plain old telephone service (POTS) dial peer and a physical TDM interface connected to a PSTN circuit or a PBX interface. When the primary VoIP dial peer rejects the call, the POTS dial peer is matched, causing the call to use the alternate PSTN link.

- **Return to originating switch**—The call can be returned to the originating TDM switch to leverage any existing rerouting capabilities within the originating switch. How this is accomplished depends on the interface type providing the connectivity between the originating switch and originating gateway:
  - Common channel signaling (CCS): CCS trunks, such as Primary Rate ISDN (PRI) and Basic Rate ISDN (BRI), separate the signaling and voice conversations into two distinct channels. The signaling channel is referred to as the *D channel*, and the voice conversation is known as the *bearer channel*. This separation of channels gives the originating gateway the capability to alert the originating switch in the event that insufficient network resources are available to place the call. This allows the originating switch to tear down the connection and resume handling of the call with an alternate path.
  - Channel-associated signaling (CAS): CAS trunks, such as E&M and T1 CAS, combine the signaling and voice conversations in a single channel. The originating gateway has no means of alerting the originating switch if insufficient network resources are available to place the call. For the originating gateway to return the initial call to the originating switch, a second channel must be used to reroute the voice conversation back to the originating switch. This process, known as *hairpinning*, causes the initial call channel and the second rerouted channel to remain active during the life of the voice conversation.

An IP telephony environment uses much of the same concepts as a legacy VoIP environment to handle CAC. However, an additional layer of control is added by the introduction of the CallManager cluster, which keeps the state of voice gateways and the availability of network resources in a central location. In an IP telephony environment, the configuration of the CallManager cluster in conjunction with the voice gateways determines whether, and where, a call is rerouted in the event of a reject due to insufficient network resources.

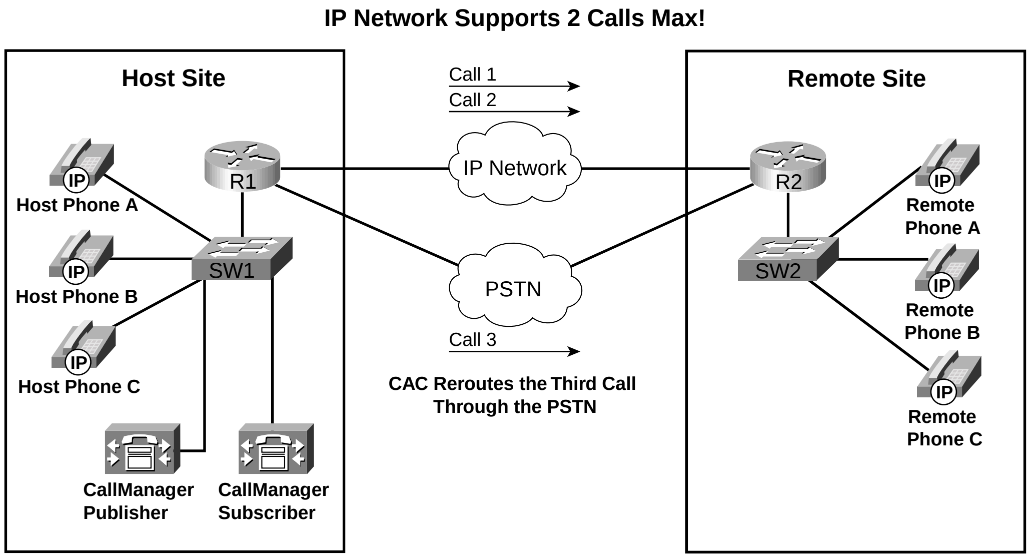
Figure 8-3 illustrates an IP telephony solution with CAC configured to reroute a call to the PSTN when there is insufficient network resources to route the call over the WAN link.

Bandwidth Engineering

To successfully implement CAC mechanisms in your packet network, you must begin with a clear understanding of the bandwidth required by each possible call that can be placed. In Chapter 7, "Link-Efficiency Tools," you learned about bandwidth requirements for two of the most popular codecs deployed in converged networks, G.711 and G.729.

The G.711 codec specification carries an uncompressed 64-kbps payload stream, known in the traditional telephony world as *pulse code modulation* (PCM). G.711 offers toll-quality voice conversations at the cost of bandwidth consumption. The G.711 codec is ideally suited for the situation in which bandwidth is abundant and call quality is the primary driver, such as in LAN environments.

Figure 8-3 *IP Telephony Network with CAC*



The G.729 codec specification carries a compressed 8-kbps payload stream, known in the traditional telephony world as *conjugate-structure algebraic-code-excited linear-prediction* (CS-ACELP). G.729 offers a tradeoff: reduced overall bandwidth consumption with a slight reduction in voice quality. G.729 is ideally suited for the situation in which bandwidth is limited, such as in a WAN environment.

As you learned in previous chapters, several other features play a role in determining the bandwidth requirement of a voice call, including header compression, Layer 2 headers, and voice samples per packet. Voice Activation Detection (VAD) can also play a role in the bandwidth required by each call. VAD can be used to reduce the packet payload size by transmitting 2 bytes of payload during silent times rather than the full payload size. For example, the payload on a single G.711 packet using Cisco defaults is 160 bytes. VAD can reduce the size of the payload to 2 bytes during silent times in the conversations. Although VAD can offer bandwidth savings, Cisco recommends that VAD be disabled due to the possible voice-quality issues that it may induce. For the purposes of bandwidth engineering, VAD should not be taken into account.

Table 8-2 illustrates a few of the possible G.711 and G.729 bandwidth requirements.

Table 8-2 *Bandwidth Requirements*

Codec	Payload per Packet	IP/UDP/RTP Header Size	L2 Header Type	L2 Header Size	Packets per Second	Bandwidth per Call
G.711	160 bytes	40 bytes	Ethernet	14 bytes	50 pps	85.6 kbps
G.711	240 bytes	40 bytes	Ethernet	14 bytes	33 pps	77.6 kbps
G.711	160 bytes	40 bytes	MLPPP/FR	6 bytes	50 pps	82.4 kbps
G.711	160 bytes	2 bytes (cRTP)	MLPPP/FR	6 bytes	50 pps	67.2 kbps
G.729	20 bytes	40 bytes	Ethernet	14 bytes	50 pps	29.6 kbps
G.729	20 bytes	40 bytes	MLPPP/FR	6 bytes	50 pps	26.4 kbps
G.729	30 bytes	40 bytes	MLPPP/FR	6 bytes	33 pps	20 kbps
G.729	20 bytes	2 bytes (cRTP)	MLPPP/FR	6 bytes	50 pps	11.2 kbps

* For DQOS test takers: These numbers are extracted from the DQOS course, so you can study those numbers. Note, however, that the numbers in the table and following examples do not include the L2 trailer overhead. Go to www.cisco.com, and search for "QoS SRND" for a document that provides some great background on QoS, and the bandwidth numbers that include data-link overhead.

The formula used to calculate the bandwidth for this combination of factors is as follows:

Bandwidth per call = (Payload + IP/UDP/RTP + L2) * 8 * pps

For example, using G.729 @ 50 pps over Frame Relay without header compression results in the following calculation:

Bandwidth per call = (20 + 40 + 6) * 8 * 50 = 26.4 kbps

For example, using G.711 @ 50 pps over Ethernet without header compression results in the following calculation:

Bandwidth per call = (160 + 40 + 14) * 8 * 50 = 85.6 kbps

The elements in the bandwidth per call formula correspond to the following values:

- **Payload**—Payload size per packet depends on the codec selected and the number of voice samples in each packet. One voice sample represents 10 ms of speech. By default, Cisco includes two of these samples in each packet, transmitting 20 ms of speech in each packet. This means that there must be 50 packets per second to maintain a full second of voice conversation, as shown in the following:

  20 ms * 50 pps = 1 second of voice conversation

After the number of samples per packet and packets per second has been determined, the payload size per packet is easily calculated by using the following formula:

Codec @ pps = (Codec payload bandwidth) / (Number of bits in a byte) / (Packets per second)

For example, the following shows a G.711 voice conversation using 50 pps:

G.711 @ 50 pps = 64 kbps / 8 bits / 50 pps = 160 bytes

For example, the following shows a G.711 voice conversation using 33 pps:

G.711 @ 33 pps = 64 kbps / 8 bits / 33 pps = 240 bytes

For example, the following shows a G.729 voice conversation using 50 pps:

G.729 @ 50 pps = 8 kbps / 8 bits / 50 pps = 20 bytes

For example, the following shows a G.729 voice conversation using 33 pps:

G.729 @ 33.334 pps = 8 kbps / 8 bits / 33.334 pps = 30 bytes

- **IP/UDP/RTP headers**—This is the combination of the IP header, UDP header, and RTP header overhead expressed in bytes. Without compression, this combination equals 40 bytes.
- **Layer 2 header type**—The Layer 2 transport technologies have the following header overheads:
  - — Ethernet: 14 bytes
  - — PPP and MLP: 6 bytes
  - — Frame Relay: 6 bytes
  - — ATM (AAL5): 5 bytes (plus cell fill waste)
  - — MLP over Frame Relay: 14 bytes
  - — MLP over ATM (AAL5): 5 bytes for every ATM cell + 20 bytes for the MLP and AAL5 encapsulation of the IP packet

Figure 8-4 illustrates the packet structure of the Layer 2 and IP/UDP/RTP headers and the payload for a voice packet**.**

Figure 8-4 *Voice Packet Structure*

Layer 2	IP	UDP	RTP	Payload of Speech Samples
Variable Size Based on Layer 2 Protocol	20 Bytes	8 Bytes	12 Bytes	Variable Size Based on Codec Selection and Number of Speech Samples Included

- **8**—Each byte has 8 bits.
- **pps**—The number of packets per second required to deliver a full second of a voice conversation. This value depends on the number of 10-ms samples within each packet. By default Cisco includes two 10-ms samples in each packet, transmitting 20 ms of sampled speech in each packet. If the number of samples per packet changes, the packets per second required to deliver a full second of voice conversation changes as well. If the packets per second increase, the overhead associated with the voice conversation increases, which requires additional bandwidth to deliver the same payload. Likewise, if the packets per second decrease, the overhead associated with the voice conversation decreases, which requires less bandwidth to deliver the same payload. The following calculations demonstrate the relationship between the packets per second and the samples included in each packet:
  - 10 ms * 100 pps = 1 second of voice conversation
  - 20 ms * 50 pps = 1 second of voice conversation
  - 30 ms * 33 pps = 1 second of voice conversation

Armed with this information you can begin to build out bandwidth requirements based on the network infrastructure, codec, packet payload, and the number of simultaneous calls that need to be supported.

Figure 8-5 illustrates a small IP telephony network configured to use the G.711 codec @ 50 pps for all calls placed over the LAN; the G.729 codec @ 50 pps is used for all calls placed over the WAN.

In this example, RTP header compression and VAD are not in use and the Cisco default of 50 packets per second is assumed. A call from Host B phone to Host C phone across the switched LAN infrastructure consumes 85.6 kbps of bandwidth, as shown in the following equation:

(160 + 40 + 14) * 8 * 50 = 85.6 kbps

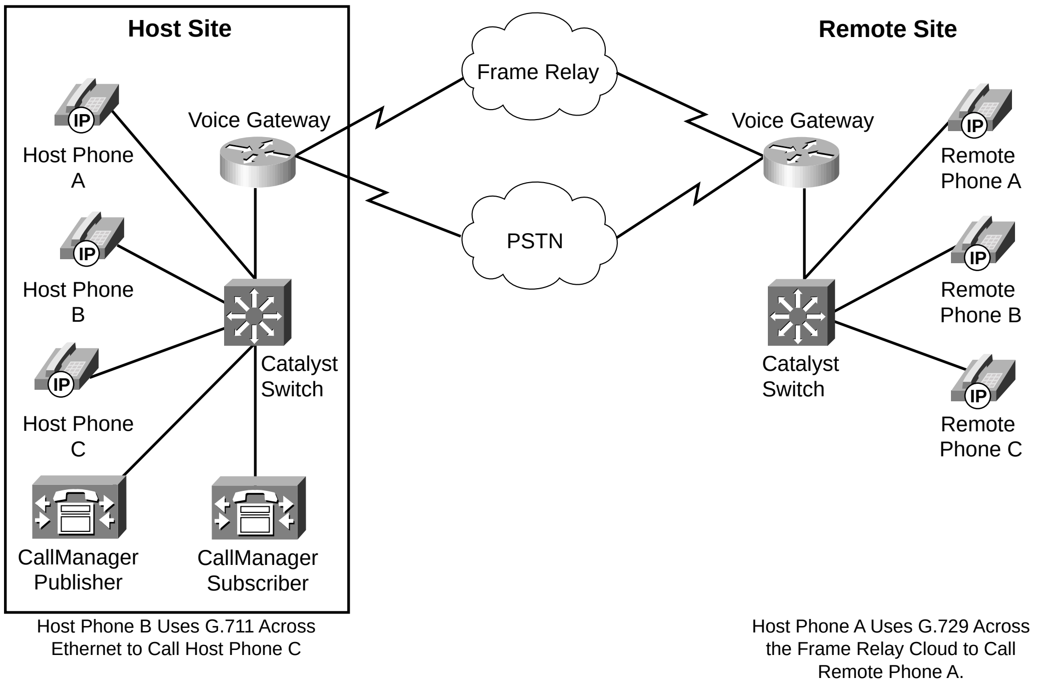
A call placed from Host A phone across the WAN infrastructure to Remote A phone in this scenario requires 26.4 kbps, as shown in the following equation:

(20 + 40 + 6) * 8 * 50 = 26.4 kbps

Assuming that you must allow 6 simultaneous calls across this WAN link at any given time, 158.4 kbps of WAN bandwidth is required to support the voice conversations, as shown in the following equation:

6 * 26.4 kbps = 158.4 kbps

Figure 8-5 *Bandwidth Considerations*



Assuming that you must provide for a guaranteed minimum of 256 kbps for data traffic, the total circuit bandwidth requirements can be derived from the following formula:

(Number of calls desired * bandwidth per call) + (Total data requirements)

or

(6 calls * 26.4 kbps) + 256 kbps = 414.4 kbps

Examining circuit speeds available today, a 512-kbps link can be used for this IP telephony network to meet the assumed voice and data requirements for 414.4 kbps. The remaining 97.6 kbps can be used for additional overhead, such as routing protocols.

Table 8-3 illustrates the relationship between codec, header compression, number of simultaneous calls, and the minimum bandwidth required for data traffic. Although the number of simultaneous calls, packet payload, and data requirements remained constant in this example, the codec selection and header compression varied the total circuit bandwidth requirements significantly.

Table 8-3 *Impacting the Total Bandwidth Requirements*

Codec	Compression	Bandwidth per Call	Maximum Number of Calls Required	Voice Bandwidth Required	Data Minimum Bandwidth Required	Total Bandwidth Required	Minimum Circuit Bandwidth
G.729	No	26.4 kbps	6	158.4 kbps	256 kbps	414.4 kbps	512 kbps
G.729	RTP header compression	11.2 kbps	6	66.6 kbps	256 kbps	322.6 kbps	512 kbps
G.711	No	82.4 kbps	6	494.4 kbps	256 kbps	750.4 kbps	768 kbps
G.711	RTP header compression	67.2 kbps	6	403.2 kbps	256 kbps	659.2 kbps	768 kbps

When you have a clear understanding of the bandwidth required for supporting the addition of voice on your packet network, you can begin to design the proper CAC mechanisms for your converged network.

CAC Mechanisms

When a call is placed in a circuit-switched network, a single 64-kbps circuit (DS0) is reserved on each PSTN trunk that the call must traverse to reach the called party and establish the voice conversation. This 64-kbps circuit remains established, without interruption from other voice channels, for the life of the conversation. As voice traffic converges on packet-switched networks, the uninterrupted channel of the circuit-switched network is no longer available. Due to the bursty nature of data traffic, it is difficult to determine whether a packet network has the available resources to carry a voice call at any given moment in time. However, several methods of CAC have been introduced into packet-switched networks in an attempt to provide the same level of call protection enjoyed by a circuit-switched network.

CAC mechanisms in a packet-switched network fall into the following three categories.

- **Local CAC mechanisms**—Local CAC mechanisms base the availability of network resources on local nodal information, such as the state of the outgoing LAN or WAN link. If the interface to the LAN or WAN is inaccessible, there is no need to execute complex decision logic based on the network state, because that network is unreachable and cannot be used to route calls. Local CAC mechanisms also have the capability, through configuration of the local device, to limit the number of voice calls that are allowed to traverse the packet network. If a WAN has enough bandwidth to allow three simultaneous calls without degradation, for instance, local CAC can be used to allow admittance to only the three calls.

- **Measurement-based CAC mechanisms**—Measurement-based CAC techniques look into the packet network to gauge the current state of the network. Unlike local CAC, measurement-based CAC uses a measurement of the packet network's current state to determine whether a new call should be allowed. Probes sent to the destination IP address and examination of the response for measurement data, such as loss and delay, are used to determine the measurement of the network's current state.
- **Resource-based CAC mechanisms**—Resource-based CAC approaches the issue of protecting voice conversations by first calculating the resources required to establish and protect the call on each leg the call traverses toward the destination. After the required resources have been identified, resource-based CAC attempts to reserve these resources for use by the voice conversation.

CAC Mechanism Evaluation Criteria

As each CAC method in this chapter is described, it is evaluated against various factors and criteria that will help determine which CAC mechanism is the most appropriate for the network design under consideration. As seen in the wording of the DQOS exam topics, an important part of the DQOS exam includes identifying these CAC tools and their basic features. Table 8-4 describes the criteria that is used to evaluate the different CAC tools.

Table 8-4 *CAC Feature Evaluation Criteria*

Evaluation Criteria	Description
Voice over *X* (Vo*X*) supported	The voice technologies to which the CAC method applies, such as VoIP and VoFR. Some methods apply to a single technology, whereas other methods apply to multiple technologies.
Toll bypass or IP telephony	Whether the method is suitable for use only between voice gateways connected to the PSTN or a PBX (toll bypass), or will the method function with IP Phone endpoints (IP telephony).
Platforms and releases	The Cisco IOS platforms this feature is available on, and the software release in which it was introduced.
PBX trunk types supported	Some CAC features have a dependency on the PSTN or PBX trunk type used in the connection, or act differently with CCS trunks versus CAS trunks.
End-to-end, local, or IP cloud	The scope of visibility of the CAC feature. Some mechanisms work locally on the originating gateway only, others consider the cloud between the source and destination nodes, some consider the destination POTS interface, and some work end to end.
Per call, interface, or endpoint	Different mechanisms involve different elements of the network. Several CAC methods work per call, but some work per interface and some work per endpoint or IP destination.

continues

Table 8-4 *CAC Feature Evaluation Criteria (Continued)*

Evaluation Criteria	Description
Topology awareness	Whether the CAC mechanism takes into account the topology of the network, and therefore provides protection for the links and nodes in the topology.
Guarantees QoS for duration of call	Whether the mechanism make a one-time decision before allowing the call, or whether it also protects the QoS of the call for the duration of the call by reserving the required resources.
Postdial delay	Whether the mechanism imposes an additional postdial delay because it requires extra messaging or processing during call setup.
Messaging network overhead	Whether the method uses additional messaging that must be provisioned in the network to gather the information necessary for the CAC decision.

Local Voice CAC

Local CAC mechanisms are the simplest CAC mechanisms to understand and implement. They operate on the originating gateway and consider only the local conditions of that gateway.

Physical DS0 Limitation

Physical DS0 limitation is a design methodology that limits the number of physical DS0 connections into the gateway. This limitation, in conjunction with other queuing methods, ensures that the gateway can successfully provide IP bandwidth across the WAN for each voice conversation originating from the individual DS0 trunks.

If you have determined that there is 158.4 kbps of WAN bandwidth available to handle 6 simultaneous G.729 calls, for example, DS0 limitations can be implemented by allowing only 6 DS0 connections between the PBX and the originating gateway. These 6 DS0 connections can be in the form of time slots on a digital T1/E1 trunk or individual analog connections, such as FXS, FXO, and E&M trunks.

Figure 8-6 shows a network using physical DS0 limitation to provide CAC.

This CAC design method works well in a situation where there is a TDM-to-IP gateway; however, it not effective in an IP telephony environment where a TDM-to-IP conversion does not exist on the WAN router. Calls originated from IP Phones are presented to the WAN router as an IP stream, without a physical TDM interface to provide a DS0 limitation. Another CAC mechanism must be used to solve this issue. Figure 8-7 demonstrates this concept.

Figure 8-6 *VoIP Physical DS0 Limitation*

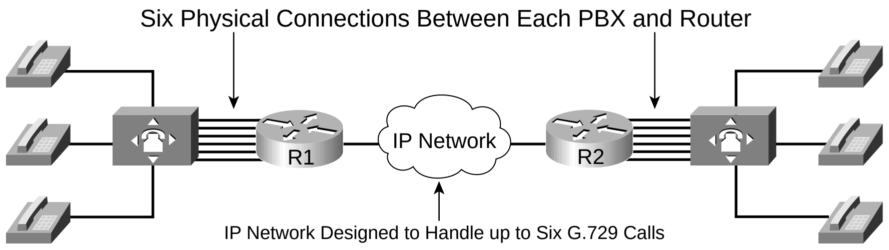
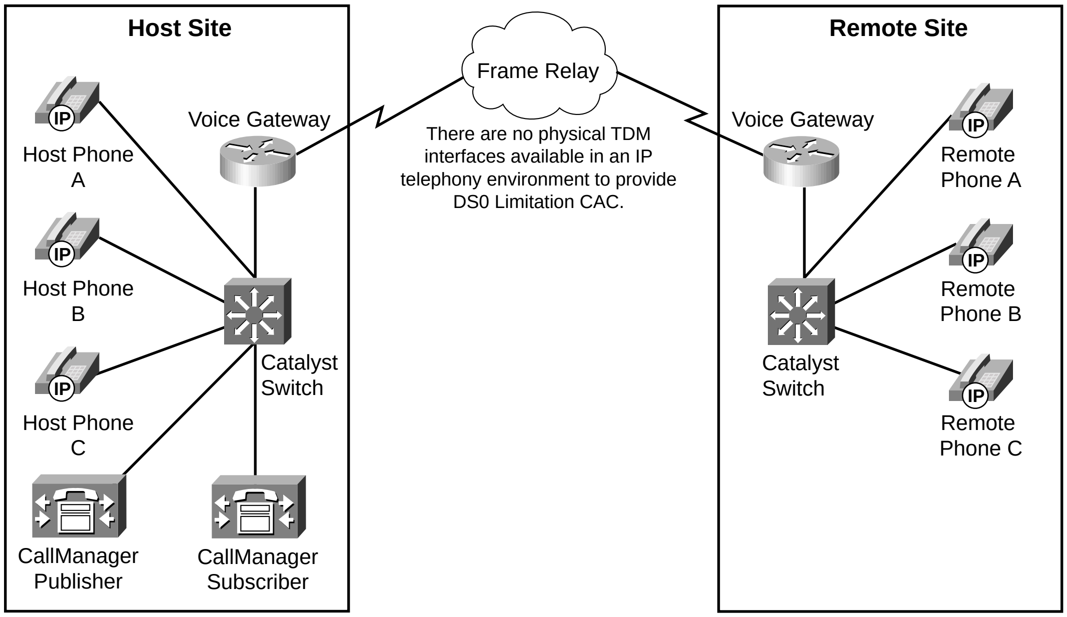

Six Physical Connections Between Each PBX and Router
R1
IP Network
R2
IP Network Designed to Handle up to Six G.729 Calls

Figure 8-7 *IP Telephony Physical DS0 Limitation*



Restricting physical DS0s on the gateway offers the following advantages:

- Adds no extra CPU usage on the gateway or bandwidth overhead to the network
- Predominant CAC mechanism deployed in toll-bypass networks today

- Protects the quality of voice conversations on the WAN link by limiting the number of voice conversations that are allowed
- Offers a known maximum bandwidth consumption rate based on the total number of possible simultaneous calls

The physical DS0 CAC method has the following limitations:

- Not effective for IP telephony applications
- Limited to relatively simple topologies
- Does not react to link failures or changing network conditions

Table 8-5 evaluates the physical DS0 limitation mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-5 *DS0 Limitation CAC Evaluation Criteria*

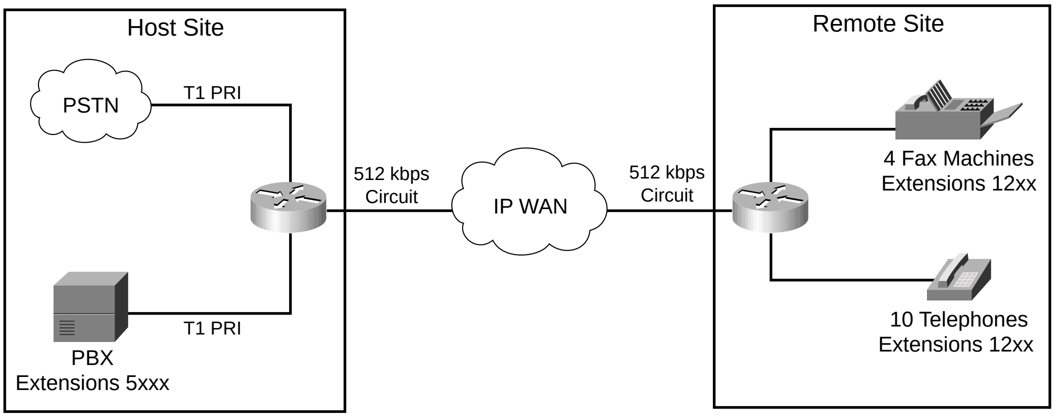
Evaluation Criteria	**Value**
Vo*X* supported	Independent of the Vo*X* technology used
Toll bypass or IP Telephony	Toll bypass only
Platforms and releases	All voice gateways and all Cisco IOS releases
PBX trunk types supported	All
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per DS0/trunk (per call)
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Max-Connections

Similar to physical DS0 limitation, Max-Connections uses the concept of limiting the number of simultaneous calls to help protect the quality of voice conversations. Unlike physical DS0 limitation, the **max-conn** command is a physical gateway configuration applied on a per–dial peer basis.

The first advantage that Max-Connections offers over DS0 limitation is the capability to provision for the oversubscription of TDM interfaces on an originating gateway without compromising the quality of the voice conversations being carried over the WAN. Figure 8-8 illustrates a T1 PRI connection from the PSTN and a T1 PRI connection from a PBX. Because a T1 PRI connection has the capability of supporting 23 channels, it is theoretically possible to have 46 simultaneous voice conversations on the host site router that are attempting to traverse the WAN to reach endpoints in the remote site.

Figure 8-8 *Max-Connections*



In this example, nine concurrent calls can be supported. Assuming that the data requirement for the IP WAN circuit is 256 kbps, and the codec in use is G.729 at 50 pps without the use of VAD or header compression, the maximum number of calls that can be successfully supported and protected across the 512-kbps WAN link is 9, as shown in the following:

$$\frac{\text{((Total circuit bandwidth) – (Total data requirements))}}{\text{(Bandwidth per call)}} = \text{Number of protected calls}$$

or

$$\frac{\text{(512 kbps – 256 kbps)}}{\text{26.4 kbps}} = \text{9 calls}$$

NOTE This calculation does not take into account the bandwidth required for routing updates. Instead, this calculation shows the theoretical maximum number of calls that can traverse this link assuming no packets other than the listed data requirements are present on the link.

If the theoretical maximum of 46 calls over the 2 PRIs is attempted, voice quality for each conversation will suffer greatly, because the current bandwidth of the WAN can provide for a total of 9 simultaneous calls without voice-quality degradation. Example 8-1 shows how you can configure the **max-conn** command on the host site router to limit the number of simultaneous calls on the VoIP dial peer to 9. For this example, assume that all endpoints in the remote site have a directory number consisting of 4 digits, and each directory number begins with the number 12 followed by 2 additional digits.

Example 8-1 *Max-Connections Host Site Router*

```
dial-peer voice 100 voip
!Sets the maximum number of connections (active admission control).
 max-conn 9
 destination-pattern 12..
 ip precedence 5
 session target ipv4:10.1.1.2
```

Assume that all endpoints in the host site have a directory number consisting of 4 digits and each directory number begins with the number 5 followed by 3 additional digits. Example 8-2 shows how you can configure the **max-conn** command on the remote site router to limit the number of simultaneous calls on the VoIP dial peer to 9.

Example 8-2 *Max-Connections Remote Site*

```
dial-peer voice 100 voip
!Sets the maximum number of connections (active admission control).
 max-conn 9
 destination-pattern 5...
 ip precedence 5
 session target ipv4:10.1.1.1
```

The second advantage that Max-Connections offers over DS0 limitation is the capability to limit the number of calls that will be allowed to traverse an IP WAN between multiple sites. Because the **max-conn** command is applied on a per–dial peer basis, a clear understanding of the call volume desired and the available bandwidth between sites must be achieved. To limit the total number of aggregate voice conversations allowed to traverse the WAN link, the **max-conn** command must exist on each VoIP dial peer. The aggregate of these configurations must not exceed the bandwidth provisioned for the call volume.

Suppose, for example, that the two additional remote sites are added to the preceding example, each with the same data requirements of 256 kbps. In this scenario, 9 simultaneous calls can be protected from each remote site, as shown in the following:

$$\frac{((\text{Total circuit bandwidth}) - (\text{Total data requirements}))}{(\text{Bandwidth per call})} = \text{Number of protected calls}$$

or

$$\frac{(512\text{ kbps} - 256\text{ kbps})}{26.4\text{ kbps}} = 9\text{ calls}$$

The total circuit bandwidth for the host site is increased to a full T1 to handle the two additional sites, as shown in the following:

> ((Total circuit bandwidth) – (Total data requirements)) / (Bandwidth per call) = Number of protected calls
>
> (1536 kbps – (3 * 256 kbps)) / 26.4 kbps = 29 calls

NOTE These calculations do not take into account the bandwidth required for routing updates. Instead, this calculation shows the theoretical maximum number of calls that can traverse these links assuming no packets other than the listed data requirements are present on the link.

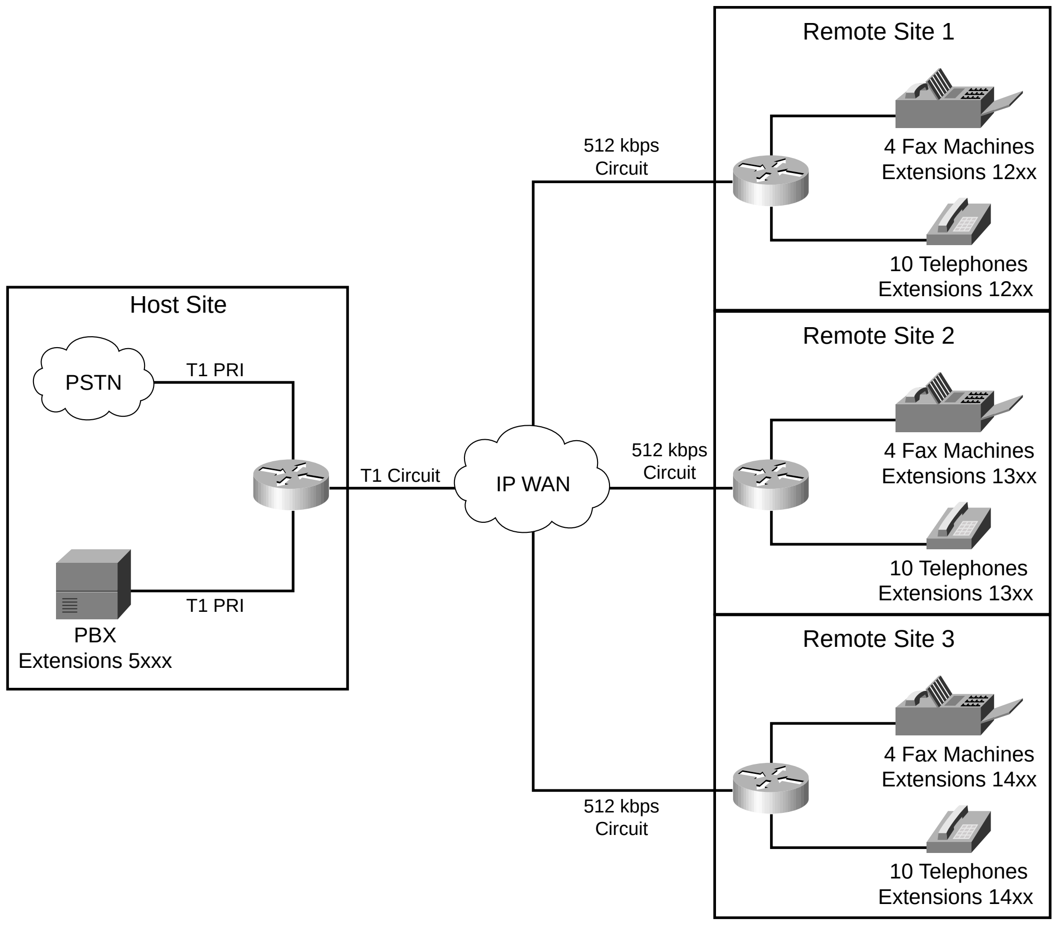
Figure 8-9 illustrates this multiple-site converged network.

Suppose that endpoints in all remote sites have a directory number consisting of 4 digits, and each directory number at Remote Site 1 begins with the number 12 followed by 2 additional digits. Directory numbers at Remote Site 2 begin with the number 13 followed by 2 additional digits. Directory numbers at Remote Site 3 begin with the number 14 followed by 2 additional digits. Example 8-3 shows how you can configure the **max-conn** command on the host site router to limit the number of simultaneous calls per location on the VoIP dial peer to 9 with an aggregate maximum of 27 calls allowed across the IP WAN.

Example 8-3 *Max-Connections Host Site Router per Site*

```
dial-peer voice 1 voip
! 9 calls allowed on VoIP Dial Peer to Remote Site 1
 max-conn 9
 destination-pattern 12..
 ip precedence 5
 session target ipv4:10.1.1.2
!
dial-peer voice 2 voip
! 9 calls allowed on VoIP Dial Peer to Remote Site 2
 max-conn 9
 destination-pattern 13..
 ip precedence 5
 session target ipv4:10.1.2.2
!
dial-peer voice 3 voip
! 9 calls allowed on VoIP Dial Peer to Remote Site 1
 max-conn 9
 destination-pattern 14..
 ip precedence 5
 session target ipv4:10.1.3.2
```

Figure 8-9 *Max-Connections Multi-Site*


Remote Site 1
512 kbps Circuit
4 Fax Machines Extensions 12xx
10 Telephones Extensions 12xx
Host Site
PSTN
T1 PRI
Remote Site 2
512 kbps Circuit
4 Fax Machines Extensions 13xx
T1 Circuit
IP WAN
10 Telephones Extensions 13xx
T1 PRI
PBX Extensions 5xxx
Remote Site 3
4 Fax Machines Extensions 14xx
512 kbps Circuit
10 Telephones Extensions 14xx

Assume that all endpoints in the host site have a directory number consisting of 4 digits, and each directory number begins with the number 5 followed by 3 additional digits. Example 8-4 shows how you can configure the **max-conn** command each remote site router to limit the number of simultaneous calls on the VoIP dial peer to 9, with an aggregate maximum of 27 calls allowed across the IP WAN.

Example 8-4 *Max-Connections Remote Site 1*

```
dial-peer voice 1 voip
!VoIP Dial Peer from Remote Site 1 to Host Site
 max-conn 9
 destination-pattern 5...
 ip precedence 5
 session target ipv4:10.1.1.1
```

Example 8-5 shows the configuration of the **max-conn** command at Remote Site 2.

Example 8-5 *Max-Connections Remote Site 2*

```
dial-peer voice 1 voip
!VoIP Dial Peer from Remote Site 2 to Host Site
 max-conn 9
 destination-pattern 5...
 ip precedence 5
 session target ipv4:10.1.2.1
```

Example 8-6 shows the configuration of the **max-conn** command at Remote Site 3.

Example 8-6 *Max-Connections Remote Site 3*

```
dial-peer voice 1 voip
!VoIP Dial Peer from Remote Site 3 to Host Site
 max-conn 9
 destination-pattern 5...
 ip precedence 5
 session target ipv4:10.1.3.1
!
```

After the maximum number of calls specified by the **max-conn** command has been reached, another mechanism must be used to connect the call via an alternate route. This is achieved by configuring a second dial peer with the same destination pattern, but with a higher preference. Remember that the dial peer with the lowest preference, that can route the call, will be matched.

Example 8-7 shows the configuration of an alternate path using the **max-conn** command. In this example, **dial-peer voice 1 voip** is defined with a preference of 1 and **dial-peer voice 100 pots** is defined with a preference of 2. This indicates that dial peer 1 is preferred over dial peer 100. The two dial peers share the same **destination-pattern**, meaning that they will both match any dialed digits beginning with 12; however, they will attempt to connect the call using different paths. Dial peer 1 will attempt to connect the call over the IP network sending the dialed digits, whereas dial peer 100 will prefix the digits 91404555 to the dialed digits and attempt to connect the call using the PSTN. Dial peer 1 will connect calls until the number of active calls reaches the configured **max-conn** of 9. When this maximum has been reached, dial peer 1 can no longer connect calls. At this point, dial peer 100 will begin to connect calls using the alternate path to the PSTN.

Example 8-7 *Max-Connections Alternate Path*

```
dial-peer voice 1 voip
!Defines first priority for call routing.
 preference 1
!Sets the maximum number of connections (active admission control).
 max-conn 9
 destination-pattern 12..
 ip precedence 5
 session target ipv4:10.1.1.2
!
dial-peer voice 100 pots
!Defines second priority for call routing.
 preference 2
 destination-pattern 12..
 direct-inward-dial
  port 0:D
!Adds prefix 91404555 in front of the called number before sending the digits to the PSTN
 prefix 91404555
```

Max-Connections also offer the capability to limit the number of calls allowed on a POTS dial peer by making the value of the **max-conn** command for that POTS dial peer lower than the physical number of time slots that are available on a T1/E1 connection between the PSTN or a PBX and an originating gateway.

Although the Max-Connections feature is useful in many scenarios, it has the following drawbacks:

- Although it provides some protection for the voice gateway egress WAN link, it provides little or no protection for links in the network backbone.
- It does not work for IP telephony applications that do not use dial peers.
- It is limited to simple topologies.
- It does not react to link failures or changing network conditions.

Table 8-6 evaluates the Max-Connections mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-6 *Max-Connections CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	All Vo*X* that use dial peers
Toll bypass or IP telephony	Toll bypass only
Platforms and releases	All voice gateways and all Cisco IOS releases
PBX trunk types supported	All
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per dial peer
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Voice over Frame Relay—Voice Bandwidth

In a Voice over Frame Relay (VoFR) network, the **frame-relay voice-bandwidth** command is used in a Frame Relay map class to set aside the bandwidth required to successfully transport the desired number of calls. This method of bandwidth provisioning operates in much the same way as IP RTP Priority and Low Latency Queuing features that reserve bandwidth for traffic flows. Unlike LLQ or RTP Priority, the **frame-relay voice-bandwidth** command has the capability to provide CAC. Because VoFR operates at Layer 2, Frame Relay headers can be examined to determine whether a frame is carrying voice payload or data payload. The channel identification (CID) in the voice frames is used to identify which individual frames belong with the current voice conversations in progress. Because the **frame-relay voice-bandwidth** command sets aside a maximum bandwidth for voice conversation, and tracks the number of conversations in progress, the **frame-relay voice-bandwidth** command has the capability to deny the admission of an additional conversation if the maximum bandwidth allocated to voice will be exceeded.

This CAC feature only applies when VoFR is used, as defined in Frame Relay Forum Implementation Agreement FRF 11. VoFR does not use IP, UDP, and RTP to encapsulate the voice traffic. By eliminating the need for IP and RTP/UDP headers, VoFR reduces the amount of overhead needed to transport the voice payload, as show in the following formula:

Bandwidth per call = (Payload + L2) * 8 * pps

For example, a G.729 call using 50 pps requires 10.4 kbps, as shown in the following calculation:

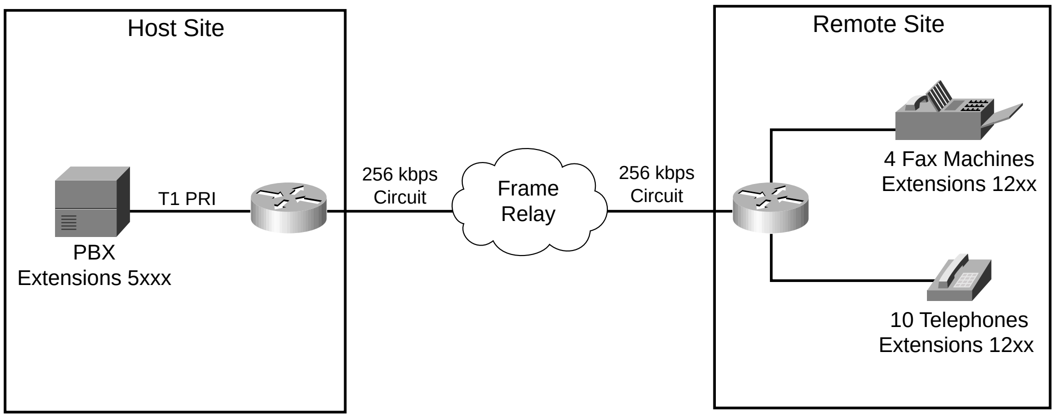
Bandwidth per call = (20 + 6) * 8 * 50 = 10.4 kbps

For example, a G.711 call using 50 pps requires 69.6 kbps, as shown in the following calculation:

Bandwidth per call = (160 + 6) * 8 * 50 = 69.6 kbps

Figure 8-10 shows a host site connected to a remote site via a Frame Relay network. Assume that VoFR was selected to carry the voice payload and 6 simultaneous calls, using G.729 codec with 50 pps, are required to be successfully transported and protected.

Figure 8-10 *Voice over Frame Relay (VoFR)*



The bandwidth required to successfully support and protect six simultaneous calls is determined by the following formula:

Bandwidth required = Number of desired calls * Bandwidth per call

In the case of the network in Figure 8-10, the following bandwidth is required:

62.4 kbps = 6 calls * 10.4 kbps

After the bandwidth requirements have been determined, the requirement can be applied to the VoFR map class to establish voice conversations. If the bandwidth requirements are not applied to the VoFR map class, the Voice-Bandwidth size defaults to 0, resulting in CAC rejects for all call attempts because of insufficient bandwidth.

Example 8-8 demonstrates how CAC for VoFR is configured by provisioning 64 kbps to transport and protect voice conversations across the Frame Relay network.

Example 8-8 *Frame Relay Voice Bandwidth*

```
interface Serial0/0
 encapsulation frame-relay
 no fair-queue
 frame-relay traffic-shaping
!
interface Serial0/0.1 point-to-point
  frame-relay interface-dlci 100
  class vofr
!
map-class frame vofr
  frame cir 256000
  frame bc 2560
  frame fragment 320
  frame fair-queue
!64 kbps is enough for six G.729 calls at 10.4 kbps each.
  frame-relay voice-bandwidth 64000
```

You can implement this CAC method only if VoFR is a viable technology in your network.

Table 8-7 evaluates the VoFR Voice-Bandwidth mechanism against the CAC evaluation criteria described earlier in this chapter.

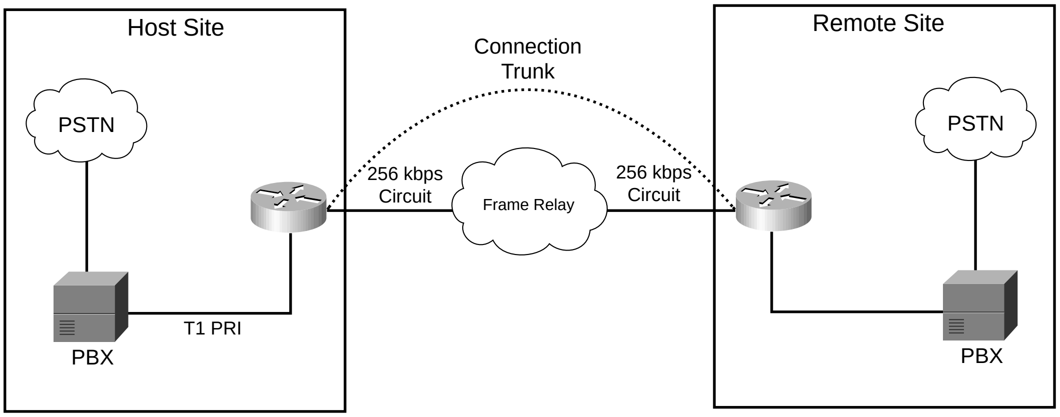
Table 8-7 *VoFR Voice-Bandwidth CAC Evaluation Criteria*

Evaluation Criteria	**Value**
Vo*X* supported	VoFR
Toll bypass or IP telephony	Toll bypass only
Platforms and releases	Cisco 2600s, 3600s, 3810, and 7200 router; Cisco IOS Release 12.0(4)T
PBX trunk types supported	All
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per call, per PVC
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Trunk Conditioning

Cisco IOS supports a function called a *permanent trunk connection*, sometimes called a *connection trunk*. A connection trunk creates a permanent trunk connection across the VoIP part of the network. To accomplish this, the **connection trunk** command is configured on a voice port to emulate a permanent connection across a packet network. The bandwidth required by the connection trunk is allocated at the creation of the trunk and remains reserved until the trunk is torn down. Figure 8-11 illustrates this concept.

Figure 8-11 *Trunk Conditioning*



Trunk conditioning is used to monitor the connection trunk state. If the connection trunk becomes unavailable, the originating gateway has the capability to signal the origination PBX and indicate that an alternate route must be found.

A unique attribute of trunk conditioning, compared to other CAC features, is that trunk conditioning has visibility into the condition of the POTS connection on the terminating side of the network and the condition of the WAN. In Figure 8-11, if there is a failure in either the WAN or the remote-side TDM connection, the originating gateway can detect this and signal the origination PBX, indicating that an alternate route must be found. This information is carried as part of the keepalive messages that are generated on connection trunk configurations.

You can tune the precise bit pattern that will be generated to the originating PBX. The ABCD bits can be configured to specific busy or out-of-service (OOS) indications that the originating PBX will recognize and act upon.

Trunk conditioning is therefore not a call-by-call feature, as are those discussed so far. It is a PBX trunk busy-back (or OOS) feature. If there is a failure in the WAN, the trunk to the originating PBX is taken out of service so that no calls can be made across that trunk until the WAN connectivity is recovered.

The following example demonstrates how a connection trunk is configured between the host and remote sites. Example 8-9 shows the configuration of the master for the trunk.

Example 8-9 *Connection Trunk Host Site*

```
controller T1 1/0
framing esf
linecode b8zs
ds0-group 1 timeslots 1 type e & m-wink-start
ds0-group 2 timeslots 2 type e & m-wink-start
clock source line
!--- The ds0-group command creates the logical voice-ports:
!--- voice-port 1/0:1 and voice-port 1/0:2.
!
voice-port 1/0:1
connection trunk 2000
!"master side"
!This starts the Trunk connection using digits 2000 to match
!a VoIP dial-peer. The digits are generated internally by the
!router and are not received from the voice-port.
!
voice-port 1/0:2
connection trunk 2001
!
dial-peer voice 100 voip
destination-pattern 200.
!matches connection trunk string 2000 and 2001
dtmf-relay h245-alphanumeric
session target ipv4:10.1.1.2
ip qos dscp cs5 media
!
dial-peer voice 1 pots
destination-pattern 1000
port 1/0:1
!This dial-peer maps to the remote site's voice-port 1/0:1.
!
dial-peer voice 2 pots
destination-pattern 1001
port 1/0:2
!This dial-peer maps to the remote site's voice-port 1/0:2.
!
interface Serial0/1
ip address 10.1.1.1 255.255.255.0
```

Example 8-10 shows the configuration of the slave for the trunk.

Example 8-10 *Connection Trunk Host Site*

```
controller T1 1/0
framing esf
linecode b8zs
ds0-group 1 timeslots 1 type e & m-wink-start
ds0-group 2 timeslots 2 type e & m-wink-start
clock source line
!
voice-port 1/0:1
connection trunk 1000 answer-mode
!"slave side"
!The answer-mode specifies that the router should not attempt to
!initiate a trunk connection, but should wait for an incoming call
!before establishing the trunk.
!
voice-port 1/0:2
connection trunk 1001 answer-mode
!
dial-peer voice 1 voip
destination-pattern 100.
dtmf-relay h245-alphanumeric
session target ipv4:10.1.1.1
ip qos dscp cs5 media
!
dial-peer voice 2 pots
destination-pattern 2000
port 1/0:1
!This dial-peer terminates the connection from the host site's voice-port 1/0:1.
!
dial-peer voice 3 pots
destination-pattern 2001
port 1/0:2
!This dial-peer terminates the connection from the host site's voice-port 1/0:2.
!
interface Serial0/1
ip address 10.1.1.2 255.255.255.0
clockrate 128000
```

Trunk conditioning is limited in scope because it applies to connection trunk networks only.

Table 8-8 evaluates the trunk conditioning mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-8 *Trunk Conditioning CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	VoIP/H.323, VoFR, VoATM (connection trunk configurations only)
Toll bypass or IP telephony	Toll bypass applications only
Platforms and Releases	Cisco 2600 and 3600 series routers, and Cisco MC3810 multiaccess concentrators; Cisco IOS Release 12.1(3)T
PBX trunk types supported	Analog and CAS
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per telephony interface
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None; uses preexisting connection trunk keepalives

Local Voice Busyout

Several CAC mechanisms generate a busy signal to the originating PBX to indicate that an alternate route must be found to successfully place a call. The preceding section discussed trunk conditioning, which operates on connection trunk networks only. Similar functionality is needed for switched networks. Local Voice Busy-Out (LVBO) is the first of two features that achieve this.

LVBO enables you to take a PBX trunk connection to the attached gateway completely out of service when WAN conditions are considered unsuitable to carry voice traffic. This technique has the following advantages:

- With the trunk out of service, each call will not be rejected individually and incur a postdial delay.
- Prevents the need for hairpinning-rejected calls back to the originating PBX, using up multiple DS0 slots for a single call.
- Works well to redirect rejected calls with PBXs that either do not have the intelligence or are not configured appropriately.
- Prevents a third DS0 on the same T1/E1 circuit from accepting the original call if the call is hairpinned back to the gateway from the originating PBX. This condition is referred to as *tromboning*.

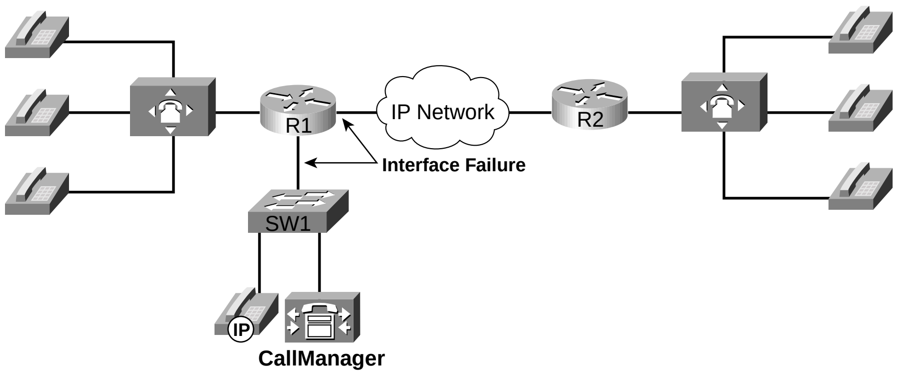
LVBO provides the originating gateway with the capability to monitor the state of various network interfaces, both LAN and WAN, and signal the originating PBX to use an alternate route should any of the monitored links fail. If any or all of the interfaces change state, the gateway can be configured to busy-back the trunk to the PBX. The reason this feature is called *Local* VBO is because only local links can be monitored. This feature has no visibility into the network beyond the link of the local gateway.

LVBO in current software works on CAS and analog PBX/PSTN trunks only. On CCS trunks, the cause code functionality can be used to inform the PBX switch to redirect a rejected call. LVBO can be configured in one of two ways:

- To force individual voice ports into the busyout state
- To force an entire T1/E1 trunk into the busyout state

Figure 8-12 illustrates the operation of the LVBO feature, and Example 8-11 shows the configuration necessary. In the example, the originating gateway is monitoring two interfaces, Ethernet interface e0/1 and WAN interface s0/1, on behalf of voice port 2/0:1, which is a T1 CAS trunk connected to a PBX. As shown in Figure 8-12, this feature is only applicable if the origination device is a PBX/PSTN interface, although the destination device can be anything, including an IP-capable voice device.

Figure 8-12 *Local Voice Busyout*



Example 8-11 *Local Voice Busyout*

```
Controller T1 2/0
  ds0-group 1 timeslot 1-4 type e&m-wink-start

Voice-port 2/0:1
  busyout monitor serial0/1
  busyout monitor ethernet0/1
```

The following limitations apply to the LVBO feature:

- It has local visibility only in current software (Cisco IOS Release 12.2); it monitors only Ethernet LAN interfaces (not Fast Ethernet), serial interfaces, and ATM interfaces.
- It applies only to analog and CAS trunk types.

Table 8-9 evaluates the LVBO mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-9 *LVBO CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	All
Toll bypass or IP telephony	Toll bypass (calls originating from PBX and terminating to IP telephony destinations)
Platforms and releases	Cisco 2600 and 3600 series routers, MC3810 multiaccess concentrators; Cisco IOS Release 12.1(2)T
PBX trunk types supported	Analog and CAS
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per WAN, LAN, and telephony interface
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Measurement-Based Voice CAC

This section focuses on the following measurement-based CAC techniques:

- Advanced Voice Busyout (AVBO)
- PSTN fallback

These are the first of two types of CAC mechanisms that add visibility into the network itself, in addition to providing local information on the originating gateway as discussed in the preceding sections.

Before we discuss the actual features within this category, some background information on service assurance agent (SAA) probes is necessary, because this is the underlying technique used by the measurement-based CAC methods. SAA probes traverse the network to a given IP destination and measure the loss and delay characteristics of the network along the path traveled. These values are returned to the originating gateway to use in making a decision on the condition of the network and its capability to carry a voice call.

Note the following attributes of measurement-based CAC mechanisms that are derived from their use of SAA probes:

- Because an SAA probe is an IP packet traveling to an IP destination, all measurement-based CAC techniques apply to VoIP only (including VoIP over Frame Relay and VoIP over ATM networks).
- As probes are sent into the network, a certain amount of overhead traffic is produced in gathering the information needed for CAC.
- If the CAC decision for a call must await a probe to be dispatched and returned, some small additional postdial delay occurs for the call. This should be insignificant in a properly designed network.

Service Assurance Agents

SAA is a generic network management feature that provides a mechanism for network congestion analysis, providing analysis for a multitude of other Cisco IOS features. It was not implemented for the purpose of accomplishing CAC, nor is it a part of the CAC suite. But its capabilities to measure network delay and packet loss are useful as building blocks on which to base CAC features.

NOTE The SAA feature was called *response time responder* (RTR) in earlier releases of Cisco IOS Software.

SAA Probes Versus Pings

SAA probes are similar in concept to the popular *ping* IP connectivity mechanism, but are far more sophisticated. SAA packets can be built and customized to mimic the type of traffic for which they are measuring the network, in this case a voice packet. A ping packet is almost by definition a best-effort packet, and even if the IP precedence is set, it does not resemble a voice packet in size or protocol. Nor will the QoS mechanisms deployed in the network classify and treat a ping packet as a voice packet. The delay and loss experienced by a ping is therefore a very crude worst-case measure of the treatment a voice packet might be subject to while traversing the very same network. With the penetration of sophisticated QoS mechanisms in network backbones, a ping becomes unusable as a practical indication of the capability of the network to carry voice.

SAA Service

The SAA service is a client/server service defined on TCP or UDP. The client builds and sends the probe, and the server (previously the RTR responder) returns the probe to the sender. The

SAA probes used for CAC go out randomly on ports selected from within the top end of the audio UDP-defined port range (16384 to 32767); they use a packet size based on the codec the call will use. IP precedence can be set if desired, and a full RTP/UDP/IP header is used like the header a real voice packet would carry. By default the SAA probe uses the RTCP port (the odd RTP port number), but it can also be configured to use the RTP media port (the even RTP port number) if desired.

SAA was introduced on selected platforms in Cisco IOS Release 12.0(7)T. With the release of 12.2 Mainline IOS, all router platforms support SAA; however, the IP Phones do not currently support SAA probes or respond to SAA probes.

Calculated Planning Impairment Factor

The ITU standardizes network transmission impairments in ITU G.113. This standard defines the term *calculated planning impairment factor* (ICPIF), which is a calculation based on network delay and packet loss figures. ICPIF yields a single value that can be used as a gauge of network impairment.

ITU G.113 provides the following interpretations of specific ICPIF values:

- 5: Very good
- 10: Good
- 20: Adequate
- 30: Limiting case
- 45: Exceptional limiting case
- 55: Customers likely to react strongly

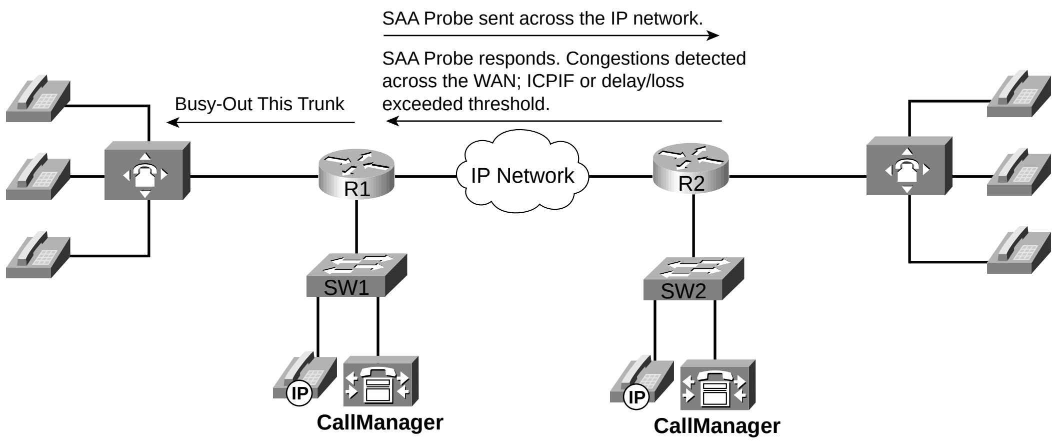
SAA probe delay and loss information is used in calculating an ICPIF value that is then used as a threshold for CAC decisions, based either on the ITU interpretation described or on the requirements of an individual customer network.

Advanced Voice Busyout

AVBO is an enhancement to LVBO. Whereas LVBO provides for busyout based on local conditions of the originating gateway, AVBO adds the capability to trigger an SAA probe to one or more configured IP destinations. The information returned by the probe, which can be either the explicit loss and delay values, or the ICPIF congestion threshold, is used to trigger a busyout of the TDM trunk connection to the PBX.

AVBO therefore introduces the capability to busy out a PBX trunk, or individual voice ports, based on the current conditions of the IP network. Figure 8-13 illustrates this capability.

Figure 8-13 *Advanced Voice Busyout*


SAA Probe sent across the IP network.
SAA Probe responds. Congestions detected across the WAN; ICPIF or delay/loss exceeded threshold.
Busy-Out This Trunk
R1
IP Network
R2
SW1
SW2
IP
CallManager
IP
CallManager

Example 8-12 shows a sample configuration of AVBO on a T1 CAS trunk connected to a PBX.

Example 8-12 *Advanced Voice Busyout*

```
controller T1 2/0
 ds0-group 1 timeslots 1-4 type e&m-immediate-start
!
voice-port 2/0:1
  voice-class busyout 4
!
voice class busyout 4
 busyout monitor Serial0/1
 busyout monitor Ethernet0/1
 busyout monitor probe 1.6.6.48 codec g729r8 icpif 10
```

When using AVBO, remember the following restrictions and limitations:

- Busyout results based on probes (measurement based) are not absolute. Some conditions, such as fleeting spikes in traffic, can cause a *false positive* to happen.
- The IP addresses monitored by the probes are statically configured (as shown in the configuration example). It is necessary to ensure, manually, that these IP addresses are indeed the destinations to which calls are being made. There is no automatic coordination between the probe configuration and the actual IP destinations to which VoIP dial peers or a gatekeeper may direct calls.
- The destination node (the device that owns the IP address to which the probe is sent) *must* support an SAA responder and have the **rtr responder** command enabled.

- This feature cannot busy back the local PBX trunk based on the state of the telephony trunk on the remote node; it monitors IP network only.
- SAA probe-based features do not work well in networks where traffic load fluctuates dramatically in a short period of time.
- As with LVBO, this feature can be applied only to analog and CAS trunks; CCS trunks are not yet supported.

Table 8-10 evaluates the AVBO mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-10 *AVBO CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	VoIP only
Toll bypass or IP telephony	Toll bypass (calls originating from PBX and terminating to IP telephony destinations)
Platforms and Releases	2600s, 3600s, and MC3810 with Release 12.1(3)T All router platforms with Release 12.2 Mainline
PBX trunk types supported	Analog and CAS
End to end, local, or IP cloud	IP cloud
Per call, interface, or endpoint	Per IP destination
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	Periodic SAA probes

PSTN Fallback

PSTN fallback allows the originating gateway to redirect a call request based on the measurement of an SAA probe. The name PSTN fallback is to some extent a misnomer because a call can be redirected to any of the rerouting options discussed earlier in this chapter, not only to the PSTN. In the event that a call is redirected to the PSTN, redirection can be handled by the outgoing gateway itself, or redirection can be performed by the PBX that is attached to the outgoing gateway. For this reason, this feature is sometimes referred to as *VoIP fallback*.

Unlike AVBO, PSTN fallback is a per-call CAC mechanism. PSTN fallback does not busy out the TDM trunks or provide any general indication to the attached PBX that the IP cloud cannot take calls. The CAC decision is triggered only when a call setup is attempted.

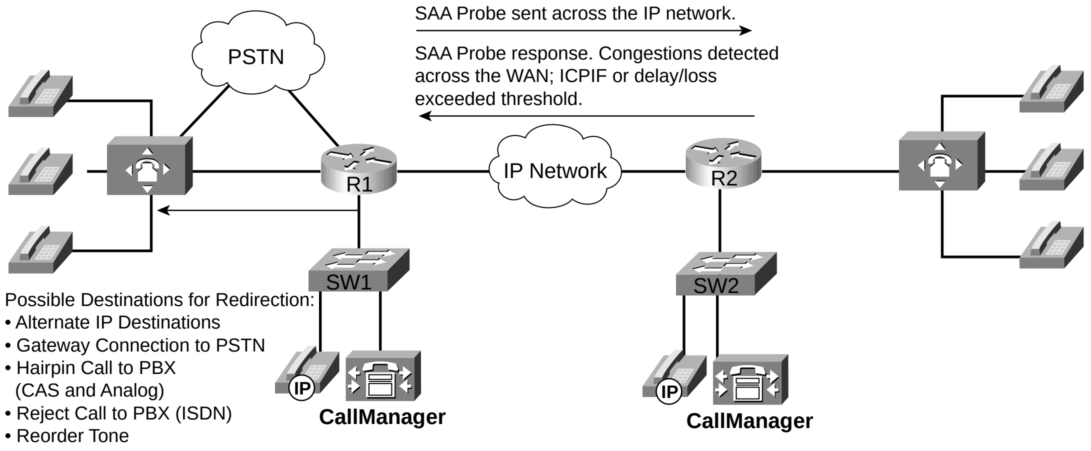
Because PSTN fallback is based on SAA probes, it has all the benefits and drawbacks of a measurement-based technique. It is unusually flexible in that it can make CAC decisions based on any type of IP network. All IP networks will transport the SAA probe packet as any other IP packet. Therefore it does not matter whether the customer backbone network comprises one or more service provider (SP) networks, the Internet, or any combination of these network types. The only requirement is that the destination device supports the SAA responder functionality.

Although PSTN fallback is not used directly by IP Phones and PC-based VoIP application destinations, it can be used indirectly if these destinations are behind a Cisco IOS router that supports the SAA responder.

SAA Probes Used for PSTN Fallback

When a call is attempted at the originating gateway, the network congestion values for the IP destination are used to allow or reject the call. The network congestion values for delay, loss, or ICPIF are obtained by sending an SAA probe to the IP destination the call is trying to reach. The threshold values for rejecting a call are configured at the originating gateway. Figure 8-14 illustrates this concept.

Figure 8-14 *PSTN Fallback*


SAA Probe sent across the IP network.
SAA Probe response. Congestions detected across the WAN; ICPIF or delay/loss exceeded threshold.
PSTN
R1
IP Network
R2
SW1
SW2
IP
CallManager
IP
CallManager
Possible Destinations for Redirection:
• Alternate IP Destinations
• Gateway Connection to PSTN
• Hairpin Call to PBX (CAS and Analog)
• Reject Call to PBX (ISDN)
• Reorder Tone

IP Destination Caching

Unlike AVBO, PSTN fallback does not require the static configuration of the IP destinations. The software keeps a cache of configurable size that tracks the most recently used IP destinations to which calls were attempted. If the IP destination of a new call attempt is found in the cache, the CAC decision for the call can be made immediately. If the entry does not appear in the cache, a new probe is started and the call setup is suspended until the probe response arrives. Therefore, an extra postdial delay is imposed only for the first call to a new IP destination. Figure 8-15 illustrates these possible scenarios.

Figure 8-15 *PSTN Fallback Call Setup*

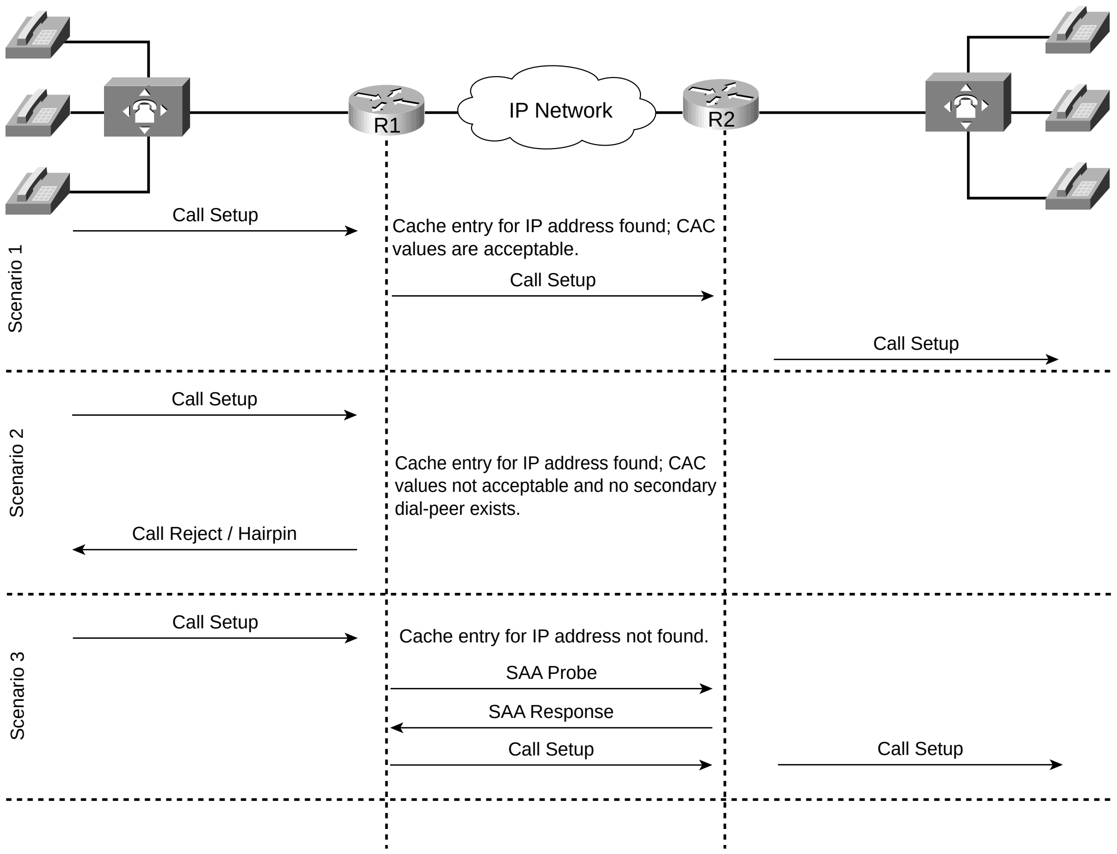

IP Network
R1
R2
Scenario 1
Call Setup
Cache entry for IP address found; CAC values are acceptable.
Call Setup
Call Setup
Scenario 2
Call Setup
Cache entry for IP address found; CAC values not acceptable and no secondary dial-peer exists.
Call Reject / Hairpin
Scenario 3
Call Setup
Cache entry for IP address not found.
SAA Probe
SAA Response
Call Setup
Call Setup

Figure 8-15 demonstrates three possible scenarios. In all scenarios, a call setup message is send to router 1 (R1). R1 consults its cache to determine whether a path exists and, if so, that the ICPIF or delay/loss thresholds have not been exceeded. In scenario one, both conditions are true and the call setup message is forwarded to router 2 (R2) to connect the call. In scenario two, a path to the IP destination is found in cache; however, the ICPIF or loss/delay exceed the threshold for that path and the call is either rejected or hairpinned back to the origination PBX, depending on the interface type connecting the PBX with R1. In scenario three, a path to the IP destination is not found in cache. An SAA probe is sent to the IP destination to determine the ICPIF or loss/delay values. If the response shows that the thresholds have not been exceeded, the call setup message is forwarded on to router 2 (R2) to connect the call.

After an IP destination has been entered into the cache, a periodic probe with a configurable timeout value is sent to that destination to refresh the information in the cache. If no further calls are made to this IP destination, the entry ages out of the cache and probe traffic to that destination is discontinued. In this way, PSTN fallback dynamically adjusts the probe traffic to the IP destinations that are actively seeing call activity.

SAA Probe Format

Each probe consists of a configurable number of packets. The delay, loss, and ICPIF values entered into the cache for the IP destination are averaged from all the responses.

If an endpoint is attempting to establish a voice conversation using a G.729 or G.711 codec, the probe's packet size emulates the codec of the requesting endpoint. Additional codecs use G.711-like probes.

The IP precedence of the probe packets can also be configured to emulate the priority of a voice packet. This parameter should be set equal to the IP precedence used for other voice media packets in the network. Typically the IP precedence value is set to 5 for voice traffic.

PSTN Fallback Configuration

PSTN fallback is configured on the originating gateway and applies only to calls initiated by the originating gateway. Inbound call attempts are not considered. PSTN fallback is configured at the global level and therefore applies to all outbound calls attempted by the gateway. You cannot selectively apply PSTN fallback to calls initiated by certain PSTN/PBX interfaces. The SAA responder feature is configured on the destination node, also referred to as the *terminating gateway.*

To apply PSTN fallback, enter the following global configuration commands:

- Originating gateway: the **call fallback** command
- Destination node: the **rtr responder** command

Table 8-11 lists the options and default values of the **call fallback** command.

Table 8-11 *Call Fallback Command*

Call Fallback Command Keyword	Description	Default Value
cache-size *x*	Specifies the call fallback cache size for network traffic probe entries.	128
cache-timeout *x*	Specifies the time after which the cache entries of network conditions are purged.	600s
instantaneous-value-weight *x*	Specifies that the call fallback subsystem take an average from the last two cache entries for call requests.	66
jitter-probe num-packets *x*	Specifies the number of packets in a jitter probe used to determine network conditions.	15

Table 8-11 *Call Fallback Command (Continued)*

Call Fallback Command Keyword	Description	Default Value
jitter-probe precedence *x*	Specifies the priority of the jitter-probe transmission.	2
jitter-probe priority-queue	Assigns a priority queue for jitter-probe transmissions.	Off
key-chain	Specifies MD5 authentication for sending and receiving SAA probes.	None
map subnet	Specifies that the call fallback router keep a cache table by subnet addresses of distances for several destination peers sitting behind the router.	None
probe-timeout *x*	Sets the timeout for an SAA probe for call fallback purposes.	30s
threshold delay *x* **loss** *y*	Specifies that the call fallback threshold use only packet delay and loss values.	None
Icpif *x*	Specifies that the call fallback use the ICPIF threshold.	10

Examples 8-13 and 8-14 demonstrate PSTN fallback between a host and remote site. SAA is configured on the remote site to answer the probes from the host site. When the number 1234 is dialed from the host site and congestion is observed on the link between the host site and the remote site, the call is redirected to port 3/0:23 and connected to the remote site over the PSTN.

Probes are sent every 20 seconds with 15 packets in each probe. The probes share the priority queue with the other voice packets. The delay and loss threshold command is configured with a delay threshold of 150 ms and a loss threshold of 5 percent and the cache aging timeout is configured for 10,000 seconds.

Example 8-13 shows the configuration for the host site router.

Example 8-13 *Call Fallback Host Site Configuration*

```
Hostname Host-Site
!
call fallback probe-timeout 20
call fallback threshold delay 150 loss 5
call fallback jitter-probe num-packets 15
call fallback jitter-probe priority-queue
call fallback cache-timeout 10000
call fallback active
!
interface Serial1/0
ip address 10.1.1.1 255.255.255.0
!
interface Serial3/0:23
 no ip address
```

continues

Example 8-13 *Call Fallback Host Site Configuration (Continued)*

```
 no logging event link-status
 isdn switch-type primary-ni
 isdn incoming-voice voice
 no cdp enable
!
voice-port 3/0:23
!
dial-peer voice 100 voip
destination-pattern 12..
preference 1
session target ipv4:10.1.1.2
!
dial-peer voice 10 pots
destination-pattern 12..
preference 2
port 3/0:23
!Adds prefix in front of the dialed number route over the PSTN
prefix 9140455512
!
dial-peer voice 20 pots
destination-pattern 9T
port 3/0:23
```

Example 8-14 shows the configuration for the remote site router.

Example 8-14 *Call Fallback Remote Site Configuration for* ***Host Name Remote Site***

```
!
interface Serial1/0
ip address 10.1.1.2 255.255.255.0
!
interface Serial3/0:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn incoming-voice voice
 no cdp enable
!
voice-port 3/0:23
!
dial-peer voice 100 voip
destination-pattern 5...
preference 1
session target ipv4:10.1.1.1
!
dial-peer voice 10 pots
destination-pattern 5...
preference 2
port 3/0:23
```

Example 8-14 *Call Fallback Remote Site Configuration for* ***Host Name Remote Site*** *(Continued)*

```
!Adds prefix in front of the dialed number route over the PSTN
prefix 914085555
!
dial-peer voice 20 pots
destination-pattern 9T
port 3/0:23
!
rtr responder
!
```

With the configuration of Examples 8-13 and 8-14, 15 probes are placed in the priority queue of the host site router every 20 seconds. The responses received from the remote site router must not exceed 150 ms of delay or 5 lost packets for the host site routers to determine that this path will support the QoS necessary for a voice conversation. If this path is not used in 10,000 seconds, it is removed from the cache and a new set of probes have to be launched at the next request. The **rtr responder** command is enabled on the remote site router to enable responses to the probes.

PSTN Fallback Scalability

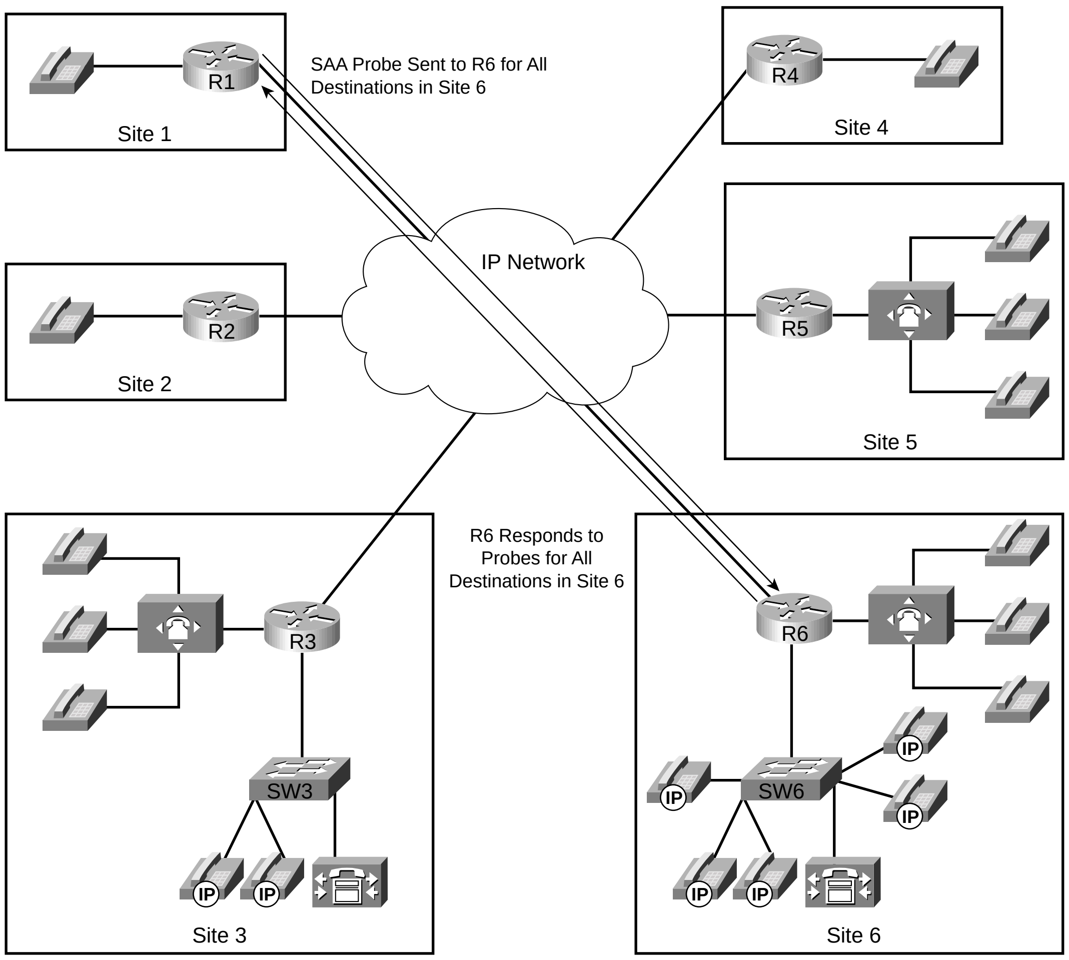
Examples 8-13 and 8-14 in the preceding section describe a simple network in which the remote site router acts as the terminating gateway for the voice conversation. The IP address of the remote site router and the network congestion values of the link between the remote and host router are held in the cache of the host site router. When the host site router receives the digits 12 followed by 2 additional digits, the host cache is consulted to determine whether the call can be successfully placed.

As the network becomes more complex, such as with the addition of IP telephony, it becomes necessary to designate a single terminating gateway to represent a number of IP destination devices. Consider the example illustrated in Figure 8-16. There are a large number of IP Phones at Site 6, each one having a unique IP address.

If Site 1 calls an IP Phone at Site 6, the cache at Site 1 does not need to contain an entry for each separate IP destination at Site 6. All IP call destinations at Site 6 can be mapped to the IP address of the WAN edge router at Site 6 so that a single probe from Site 1 to Site 6 can be used to probe CAC information for all calls destined to Site 6. The same principle applies if there were multiple terminating gateways at Site 6.

The probe traffic can therefore be reduced substantially by sending probes to IP destinations that represent the portion of the network that is most likely to be congested, such as the WAN backbone and WAN edge. This same scalability technique also provides a mechanism to support IP destinations that do not support SAA responder functionality.

Figure 8-16 *PSTN Fallback Scalability*


R1
Site 1
SAA Probe Sent to R6 for All Destinations in Site 6
R4
Site 4
R2
Site 2
IP Network
R5
Site 5
R6 Responds to Probes for All Destinations in Site 6
R3
SW3
IP
IP
Site 3
R6
SW6
IP
IP
IP
IP
IP
Site 6

PSTN Fallback Summary

PSTN fallback is a widely deployable, topology-independent CAC mechanism that can be used over any backbone. Consider the following attributes of PSTN fallback when designing a network:

- Because it is based on IP probes, PSTN fallback applies to VoIP networks only.
- PSTN fallback does not reroute calls in progress when network conditions change.

- A slight increase in postdial delay will apply to the first call to a destination not yet in the cache.
- No interaction occurs between the SAA probe timer and the H.225 timer setting: The SAA probe occurs before the H.323 call setup is sent to the destination, and the H.225 timer occurs after H.323 call setup is sent.
- PSTN fallback performs well in steady traffic that has a gradual ramp-up and ramp-down, but poorly in quickly fluctuating traffic with a bursty ramp-up and ramp-down.
- An erroneous CAC decision could be reached in a bursty environment based on noncurrent information due to the periodic nature of the probes.
- *Proxy* destinations for the probes can be used by mapping destination IP addresses to a smaller number of IP addresses of the nodes located between the originating gateway and the terminating gateways.
- No bandwidth measurements are taken by the probes, delay and loss measurements only.
- MD5 keychain authentication can be configured for security to ensure that probes are initiated only by trusted sources, which will circumvent "denial-of-service" type attacks by untrusted sources initiating large volumes of probes.

Table 8-12 evaluates the PSTN fallback mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-12 *PSTN Fallback CAC Evaluation Criteria*

Evaluation Criteria	**Value**
Vo*X* supported	VoIP only
Toll bypass or IP telephony	Toll bypass; however, calls originating from a PBX and terminating to IP telephony destinations can be protected
Platforms and releases	Cisco 2600/3600, MC3810: Release 12.1.(3)T AS5300: Release 12.2.(2)T 7200/7500 support SAA responder
PBX trunk types supported	All PBX/PSTN trunk signaling types (analog, digital CAS and CCS) For analog and digital CAS—alternate IP destination, hairpin For digital CCS, reject the call to the PBX or PSTN for rerouting
End to end, local, or IP cloud	IP cloud
Per call, interface, or endpoint	Per active/cached IP destination

continues

Table 8-12 *PSTN Fallback CAC Evaluation Criteria (Continued)*

Evaluation Criteria	Value
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	Only for first call that initiates probe
Messaging network overhead	Periodic SAA probes

Resource-Based CAC

There are two types of resource-based CAC mechanisms:

- Those that monitor the use of certain resources and calculate a value that affect the CAC decision
- Those that reserve resources for the call

The reservation mechanisms are the only ones that can attempt to guarantee QoS for the duration of the call. All other local, measurement-based and resource calculation-based CAC mechanisms just make a one-time decision prior to call setup, based on knowledge of network conditions at that time.

The following resources are of interest to voice calls:

- DS0 time slot on the originating and terminating TDM trunks
- Digital signal processor (DSP) resources on the originating and terminating gateways
- CPU use of the involved nodes, typically the gateways
- Memory use of the involved nodes, typically the gateways

The following sections focus on the following four resource-based CAC techniques:

- Resource Availability Indication
- CallManager Locations
- Gatekeeper Zone Bandwidth
- Resource Reservation Protocol

Like the measurement-based CAC techniques, these techniques add visibility into the network itself in addition to the local information discussed in the previous sections.

Resource Availability Indication

To allow gatekeepers to make intelligent call routing decisions, the terminating gateway uses resource availability indication (RAI) to report resource availability to the gatekeeper. Resources monitored by the terminating gateway include DS0 channels and DSP channels. When a monitored resource falls below a configurable threshold, the gateway sends an RAI message to the gatekeeper indicating that the gateway is almost out of resources. When the available resources then cross above another configurable threshold, the gateway sends an RAI message indicating that the resource depletion condition no longer exists. The gatekeeper never has knowledge of the individual resources or the type of resources that the gateway considers. The RAI message is a simple yes or no toggle indication sent by the terminating gateway to control whether the gatekeeper should allow subsequent voice calls to be routed to the terminating gateway. The gatekeeper responds with a resource availability confirmation (RAC) upon receiving an RAI message to acknowledge its reception.

As a CAC mechanism, RAI is unique in its capability to provide information on the terminating POTS connection. Other mechanisms discussed in this chapter enable CAC decisions based on local information at the originating gateway and on the condition of the IP cloud between the originating gateway and terminating gateways. No other CAC mechanism has the capability to consider the availability of resources to terminate the POTS call at the terminating gateway. Another difference is that with RAI the CAC decision is controlled by the terminating gateway. In all the other methods, the CAC decision is controlled by the originating gateway or by the gatekeeper.

RAI was included in Cisco IOS Software Release 12.0(5)T on the Cisco AS5300 Gateway, and Cisco IOS Software Release 12.1(1)T for other gateways in H323v2.

Gateway Calculation of Resources

The calculation to reach the CAC decision is performed by the terminating gateway. Different gateway platforms may use different algorithms. The H.323 standard does not prescribe the calculation or the resources to include in the calculation. It merely specifies the RAI message format and the need for the gatekeeper to discontinue routing calls to the terminating gateway in the event that the gateway has insufficient available resources for the additional call, and the gateway will inform the gatekeeper to resume routing calls when resources become free.

Calculating utilization first takes into account the number of accessible channels on the target device. Accessible channels are either active or idle voice channels on the device that are used to carry voice conversations. Disabled channels are not counted as accessible channels.

The following formula is used to determine accessible channels:

Accessible channels = Channels being used + Free channels

When the number of accessible channels is known, the utilization can be calculated from the following formula:

$$\text{Utilization} = \frac{\text{Channels being used}}{\text{Accessible channels}}$$

Suppose, for instance, that you have four T1 CAS circuits. Two of the T1 CAS circuits are used for incoming calls, and the remaining two T1 CAS circuits are used for outgoing calls. You have busied out 46 time slots of the outgoing time slots, and you have one call on one of the outgoing time slots. You will have the following:

- Total voice channels = 96
- Outgoing total voice channels = 48
- Disabled voice channels = 46
- Voice channels being used = 1
- Free voice channels = 1

The outgoing accessible channels in this situation are as follows:

1 (voice channels being used) + 1 (free voice channels)= 2

The DS0 utilization for this device is as follows:

$$\text{Utilization} = \frac{\text{Channels being used}}{\text{Accessible channels}}$$

or

$$50\% = \frac{1}{(1 + 1)}$$

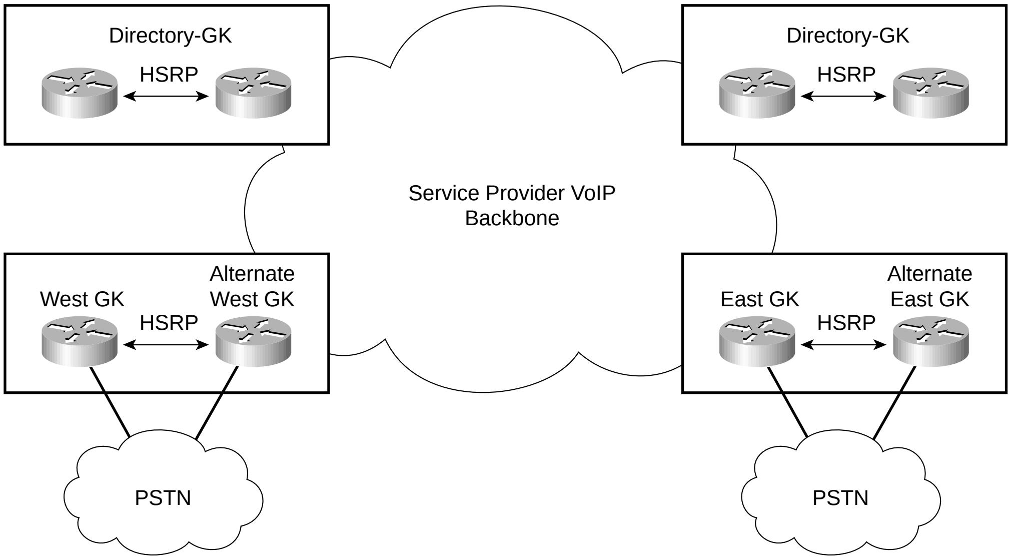
The utilization for the outgoing channels is equal to 50 percent. If the configured high threshold is 90 percent, the gateway will still accept calls. Only DS0s reachable through a VoIP dial peer are included in the calculation.

The preceding calculation took the DS0 resources into consideration. Remember that the DSP resources are monitored and calculated in the same manner. The terminating gateway sends an RAI message in the event that either the DS0 or DSP resources reach the low or high threshold.

RAI in Service Provider Networks

RAI is an indispensable feature in SP networks that provide VoIP calling services such as debit and credit card calling and VoIP long-distance phone service. Figure 8-17 shows the general structure of these networks.

Figure 8-17 *Service Provider VoIP Network Topology*



Around the world there are points of presence (POPs) where racks of gateways, such as Cisco AS5300 access servers, connect to the PSTN with T1/E1 trunks. The call routing is managed through several levels of gatekeepers as shown in Figure 8-17. Call volume is high, and these gateways handle voice traffic only (no data traffic other than minimal IP routing and network management traffic).

When a customer on the West Coast dials a number residing on the East Coast PSTN, the East Coast gatekeeper must select an East Coast gateway that has an available PSTN trunk to terminate the call. If an East Cost gateway cannot be found, the customer's call fails. In the event of a failed call, the originating gateway must retry the call or the customer must redial the call. In either case, there is no guarantee that the same out-of-capacity terminating gateway will not be selected again.

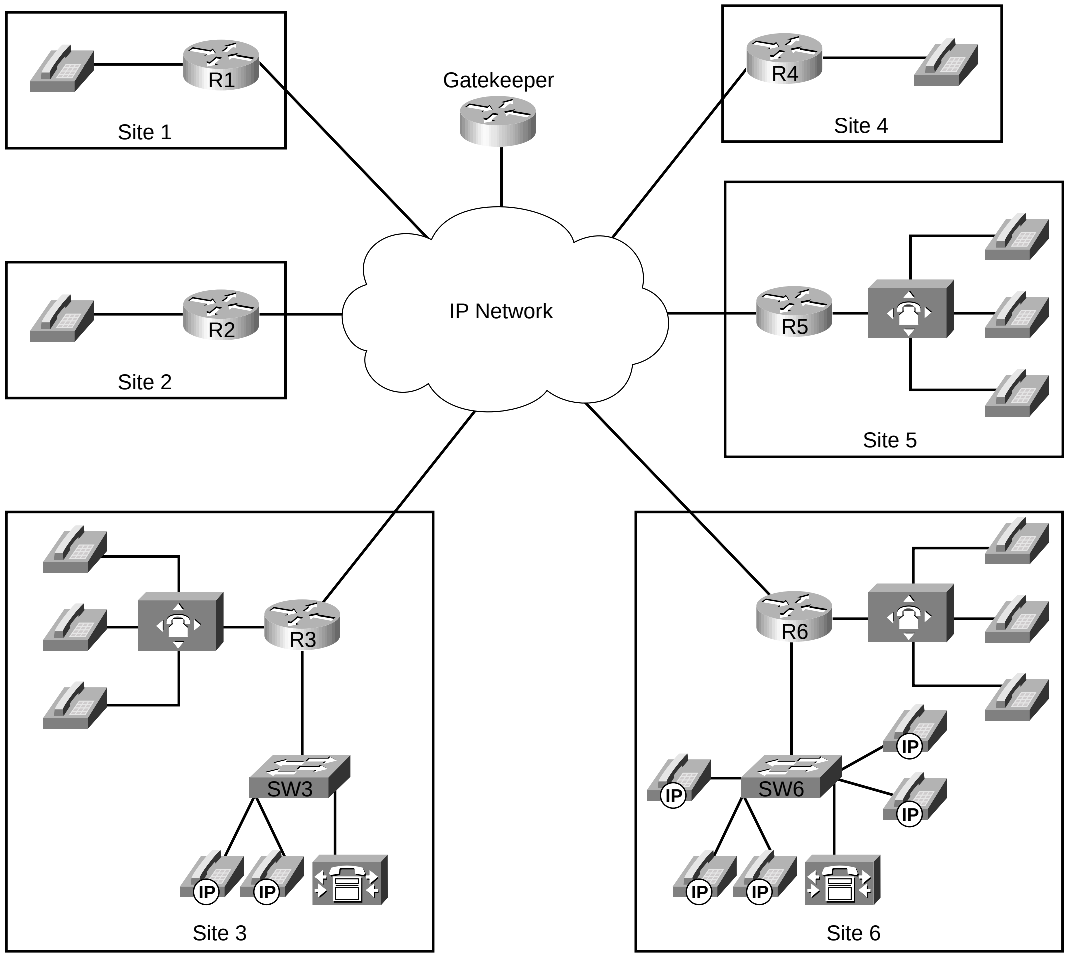
This scenario is inefficient and provides poor customer service. It is important that calls are not routed by the gatekeeper to a terminating gatekeeper that cannot terminate the call due to the lack of PSTN trunk capacity.

In general, calls are load balanced by the gatekeeper across the terminating gateways in its zone. But the gateways could have different levels of T1/E1 capacity and by load balancing across the gateways one gateway could become shorter on resources than another. It is in this situation that RAI is imperative. The overloaded terminating gateway has the capability to initiate an indication to the gatekeeper that it is too busy to take more calls.

RAI in Enterprise Networks

RAI is generally less applicable in enterprise networks than in SP networks because there is often only one gateway at each site, as shown in Figure 8-18. This is almost always true for the typical hub-and-spoke enterprise network. Even at the large sites, there may be multiple T1/E1 trunks to the attached PBX, but there are seldom multiple gateways.

Figure 8-18 *Enterprise VoIP Network Topology*



If a single gateway is used to terminate a call, where the called user resides on a specific PBX and is reachable only through a specific gateway in the network, RAI does not provide additional network intelligence. With no alternate gateway to handle excess calls, a call will always fail whenever the single terminating gateway is too busy. In addition, in enterprise networks the probability of congestion is typically higher in the IP cloud than in the number of terminating POTS trunks. In the SP networks discussed earlier, congestion is more common in the terminating POTS trunks than in the IP cloud.

In spite of these limitations, RAI can still be used for enterprise networks provided the gateway to PBX connections at the remote sites consist of T1/E1 trunks. If a terminating gateway is too busy, it triggers a PSTN reroute instead of selecting an alternate gateway as in the SP network situation.

RAI Operation

The discussion of where and how RAI is used in SP and enterprise networks clearly shows that RAI is most useful in situations where multiple terminating gateways can reach the same destination, or called, phone number. However, RAI has value in any situation where there is a desire to prevent a call from being routed to a gateway that does not have the available POTS capacity to terminate the call.

When a gatekeeper receives an RAI unavailable indication from a gateway, it removes that gateway from its gateway selection algorithm for the phone numbers that gateway would normally terminate. An RAI available indication received later returns the gateway to the selection algorithm of the gatekeeper.

RAI is an optional H.323 feature. When you implement a network, therefore, it is prudent to verify that both the gateways and gatekeepers support this feature. Cisco gatekeepers support RAI. Cisco gateway support for RAI is detailed in a later section of this chapter.

RAI Configuration

RAI on the gateway is configured with high-water- and low-water-mark thresholds, as shown in Figure 8-19. When resource usage rises above the high-water mark, an RAI unavailable indication is sent to the gatekeeper. When resource availability falls below the low-water mark, an RAI available indication is sent to the gatekeeper. To prevent hysteresis based on the arrival or disconnection of a single call, the high- and low-water marks should be configured 10 percentage points apart. (Hysteresis refers to the process of switching a feature on, then off, then on, and so on, over and over again, as would happen if the high- and low-water marks were configured to very similar values.)

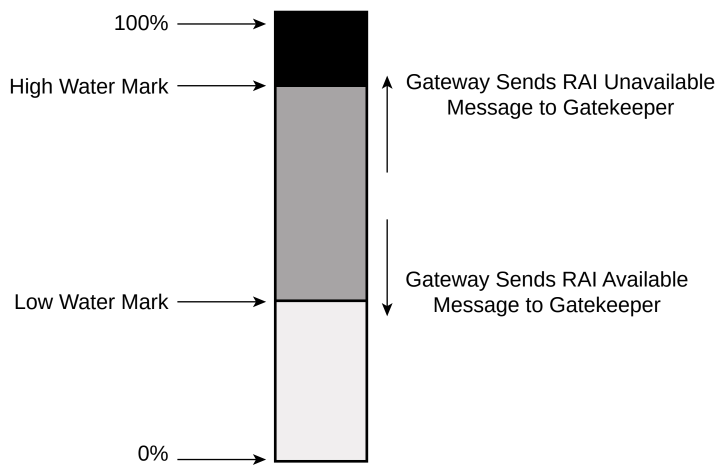
To configure RAI, use the following gateway configuration command:

```
resource threshold [all] [high %-value] [low %-value]
```

To configure an RAI unavailable resource message to be sent to the gatekeeper at a high-water mark of 90 percent and a RAI available resource message to be sent to the gatekeeper at a low-water mark of 80 percent, enter the following command:

```
resource threshold high 90 low 80
```

Figure 8-19 *RAI Configuration*


100%
High Water Mark
Low Water Mark
0%
Gateway Sends RAI Unavailable Message to Gatekeeper
Gateway Sends RAI Available Message to Gatekeeper

RAI Platform Support

The Cisco AS5300 access server has supported RAI since Cisco IOS Release 12.0(5)T. The Cisco 2600 and 3600 series routers have supported RAI for T1/E1 connections only, not for analog trunks, since Release 12.1.3T. The other Cisco IOS gateways do not yet support RAI as of Release 12.1(5)T or 12.2. The RAI calculation includes digital signal processors (DSPs) and DS0s, and may not be the same for all platforms. In current software, CPU and memory are not yet included in the RAI availability indication.

Table 8-13 evaluates the RAI mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-13 *RAI CAC Evaluation Criteria*

Evaluation Criteria	**Value**
Vo*X* supported	VoIP only
Toll bypass or IP telephony	Toll bypass Potentially IP telephony, but CM does not yet support RAI
Platforms and releases	Cisco AS5300 access server: Cisco IOS Release 12.0(5)T Cisco 2600 and 3600 series routers T1/E1: Cisco IOS Release 12.1(3)T
PBX trunk types supported	All
End to end, local, or IP cloud	Local at the terminating gateway (DSP and DS0 resources; algorithm platform dependent)

Table 8-13 *RAI CAC Evaluation Criteria (Continued)*

Evaluation Criteria	Value
Per call, interface, or endpoint	Per gateway
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	Occasional RAI toggle between gateway and gatekeeper

Cisco CallManager Resource-Based CAC

The term "locations" refers to a CAC feature in a Cisco CallManager centralized call-processing environment. Because a centralized call-processing model is deployed in a hub-and-spoke fashion without the use of a terminating gateway at each spoke, an IP telephony CAC mechanism must be used to protect the desired number of voice conversations. Locations are designed to work in this nondistributed environment.

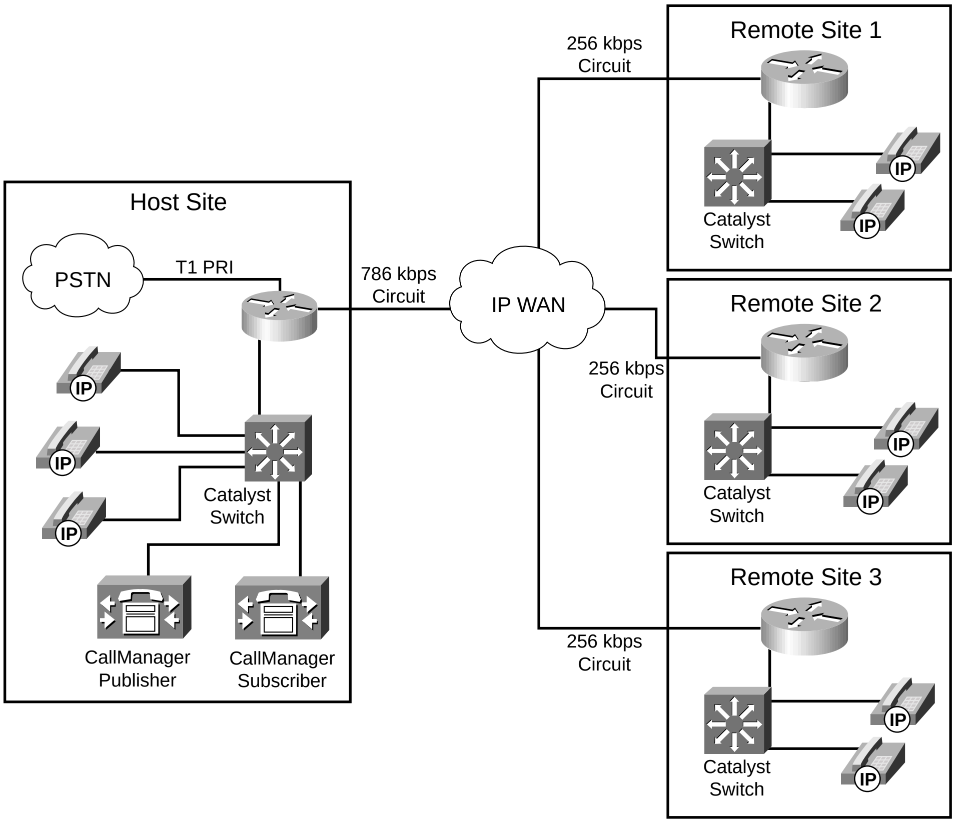
Figure 8-20 shows a typical CallManager centralized call-processing model.

Location-Based CAC Operation

The Cisco CallManager centralized call-processing model uses a hub-and-spoke topology. The host site, or hub, is the location of the primary Cisco CallManager controlling the network while the remote sites, containing IP endpoints registered to the primary CallManager, represent the spokes. The Cisco CallManager Administration web page can be used to create locations and assign IP endpoints, such as IP Phones, to these locations. After the locations have been created and devices have been assigned, you can allocate bandwidth for conversations between the hub and each spoke locations.

For calls between IP endpoints in the host site of Figure 8-20, the available bandwidth is assumed to be unlimited, and CAC is not considered. However, calls between the host site and remote sites travel over WAN links that have limited available bandwidth. As additional calls are placed between IP endpoints over the WAN links, the audio quality of each voice conversation can begin to degrade. To avoid this degradation in audio quality in a CallManager centralized call-processing deployment, you can use the locations feature to define the amount of available bandwidth allocated to CallManager controlled devices at each location and thereby decrease the number of allowed calls on the link.

Figure 8-20 *CallManager Centralized Call-Processing Model*



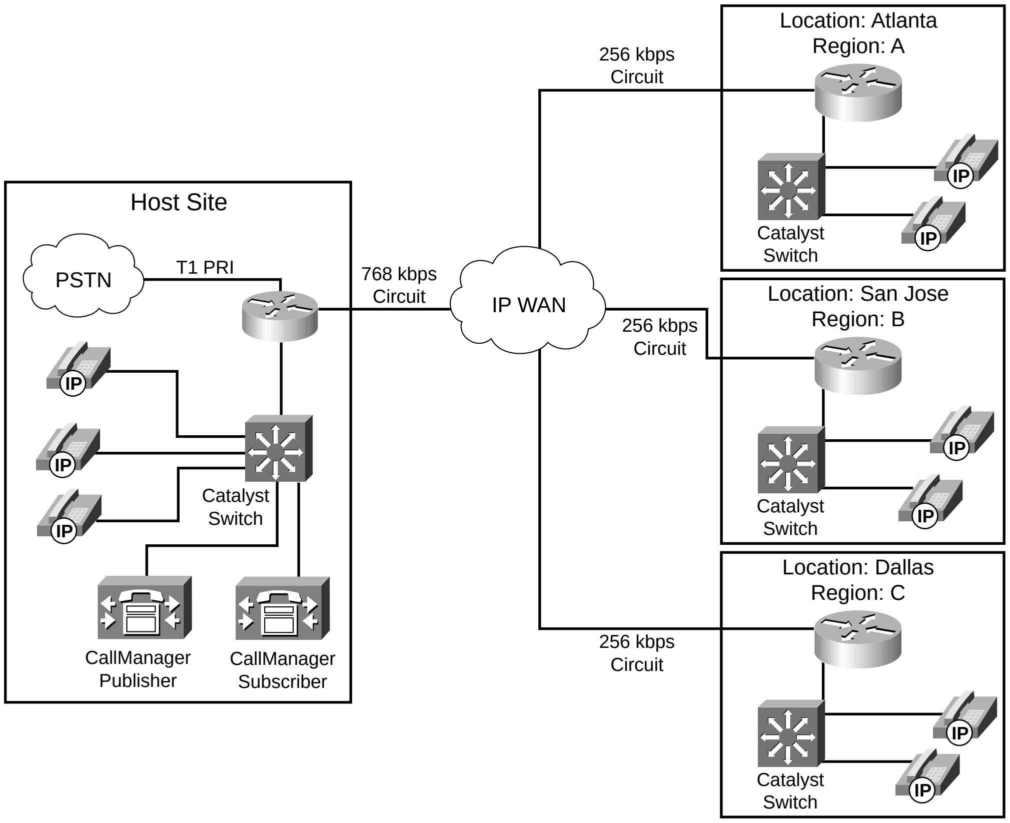
The amount of bandwidth allocated by CallManager locations is used to track the number of voice conversations in progress across an IP WAN link on a per-CallManager basis. CallManager servers in a cluster are not aware of calls that have been set up by other members in the cluster. To effectively use location-based CAC, all devices in the same location must register with the same centralized CallManager server.

Locations and Regions

In Cisco CallManager, locations work in conjunction with regions to define how voice conversations are carried over a WAN link. Regions define the type of compression, such as G.711 or G.729, used on the link, and locations define the amount of available bandwidth allocated for that link.

Figure 8-21 illustrates a CallManager centralized call-processing model with three remote sites. Each remote site has a location name that all IP endpoints in that location are associated with and a region that determines which codec is be used for voice conversations to IP endpoints in this location.

Figure 8-21 *CallManager Centralized Call-Processing Model with Regions and Locations Defined*



Each IP endpoint uses the G.711 codec for voice conversations with other endpoints in the same region. When an IP endpoint establishes a voice conversation with an IP endpoint in another region, the G.729 codec is used. The following examples illustrate this:

- **HQ IP Phone A calls HQ IP Phone B**—Because both IP Phones reside in the same region, the G.711 codec is used.
- **HQ IP Phone A calls Atlanta IP Phone A**—Because these IP Phones reside in different regions, the G.729 codec is used.

Calculation of Resources

After the regions have been defined and all devices have been configured to reside in the desired location, you can begin to allocate the desired bandwidth between the locations to use for CAC. This allocated bandwidth does not reflect the actual bandwidth required to establish voice conversations; instead, the allocated bandwidth is a means for CallManager to provide CAC. CAC is achieved by defining a maximum amount of bandwidth per location to use and then subtract a given amount, dependent upon the codec used, from that maximum for each established voice conversation. In this way, CallManager has the capability to deny or reroute call attempts that exceed the configured bandwidth capacity.

Table 8-14 shows the amount of bandwidth that is subtracted, per call, from the total allotted bandwidth for a configured region.

Table 8-14 *Location-Based CAC Resource Calculations*

Codec	Bandwidth Reserved
G.711	80 kbps
G.729	24 kbps
G.723	24 kbps
GSM	29 kbps
Wideband	272 kbps

In Figure 8-21, suppose that you need to protect six simultaneous calls between the HQ and Atlanta locations, but allow only four between the HQ and Dallas location and three between the HQ and San Jose location. Because each region has been configured to use the G.729 codec between regions, each of these voice conversations represent 24 kbps to the configured location. Remember that the location bandwidth does not represent the actual bandwidth in use.

For the Atlanta location to allow 6 simultaneous calls, the Atlanta Location Bandwidth needs to be configured for 144 kbps, as shown in the following calculation:

6 calls = 6 * 24 kbps = 144 kbps

For the Dallas location to allow 4 simultaneous calls, the Dallas location bandwidth needs to be configured for 48 kbps, as shown in the following calculation:

4 calls = 4 * 24 kbps = 48 kbps

Finally, for the San Jose location to allow 3 simultaneous calls, the San Jose location bandwidth needs to be configured for 72 kbps, as shown in the following calculation:

3 Calls = 3 * 24 kbps = 72 kbps

The link to the Atlanta location, configured for 144 kbps, could support 1 G.711 call at 80 kbps, 6 simultaneous G.729 calls at 24 kbps each, or 1 G.711 call and 2 G.729 calls simultaneously.

Any additional calls that try to exceed the bandwidth limit are rejected and the call is either rerouted or a reorder tone returns to the caller.

CallManager continues to admit new calls onto a WAN link until the available bandwidth for that link (bandwidth allocated to the location minus all active voice conversations) drops below zero.

Automatic Alternate Routing

Prior to CallManager version 3.3, if a call was blocked due to insufficient location bandwidth, the caller would receive a reorder tone and manually have to redial the called party through the PSTN. Automated alternate routing (AAR), introduced in CallManager version 3.3, provides a mechanism to reroute calls through the PSTN or another network WAN link by using an alternate number when Cisco CallManager blocks a call because of insufficient location bandwidth. With automated alternate routing, the caller does not need to hang up and redial the called party.

The fully qualified E.164 address, also known as the *PSTN number*, is obtained from the external phone number mask configured on the called device. The calling device is associated with an AAR group, which is configured with the digits that will be prepended to the phone number mask to access the PSTN gateway and reroute the call over the PSTN.

Location-Based CAC Summary

Table 8-15 evaluates location-based CAC against the CAC evaluation criteria described earlier in this chapter.

Table 8-15 *Location-Based CAC Evaluation Criteria*

Evaluation Criteria	**Value**
Vo*X* supported	VoIP only
Toll bypass or IP telephony	IP Telephony only
Platforms and releases	CallManager 3.0 (AAR was added in CallManager release 3.3.)
PBX trunk types supported	None
End to end, local, or IP cloud	End-to-end between originating and terminating location, although locations have no knowledge of the network topology in between
Per call, interface, or endpoint	Per call
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Gatekeeper Zone Bandwidth

The gatekeeper function is an IOS-based mechanism specific to H.323 networks. Different levels of Cisco IOS Software provide various, specific capabilities within this feature. In Cisco IOS Releases 12.1(5)T and 12.2, the gatekeeper function has the capability to limit the number of simultaneous calls across a WAN link similar to the CallManager location-based CAC discussed in the preceding section.

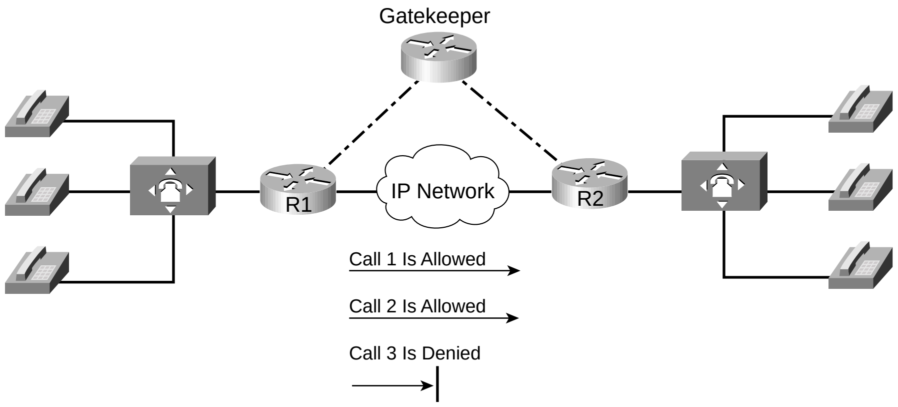
Gatekeeper Zone Bandwidth Operation

By dividing the network into zones, a gatekeeper can control the allocation of calls in its local zone and the allocation of calls between its own zone and any other remote zone in the network. For the purpose of understanding how this feature operates, assume a voice call is equal to 64 kbps of bandwidth. Actual bandwidth used by calls in a gatekeeper network are addressed in the "Zone Bandwidth Calculation" section.

Single-Zone Topology

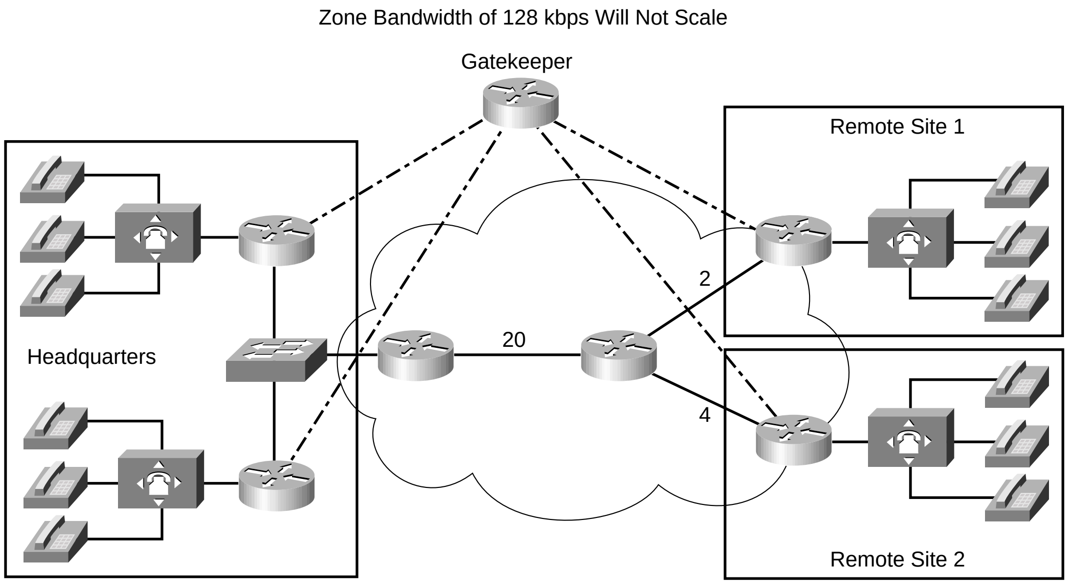
Figure 8-22 shows a single-zone gatekeeper network with two gateways that illustrates gatekeeper CAC in its simplest form. If the WAN bandwidth of the link between the two gateways can carry no more than two calls, the gatekeeper must be configured so that it denies the third call. Assuming every call is 64 kbps, the gatekeeper is configured with a zone bandwidth limitation of 128 kbps to achieve CAC in this simple topology.

Figure 8-22 *Simple Single-Zone Topology*



Most networks, however, are not as simple as the one shown in Figure 8-22. Figure 8-23 shows a more complex topology; however, it is still configured as a single-zone network. In this topology, the legs in the WAN cloud each have separate bandwidth provisioning and therefore separate capabilities of how many voice calls can be carried across each leg. The numbers on the WAN legs in Figure 8-23 show the maximum number of calls that can be carried across each leg.

Figure 8-23 *Complex Single-Zone Topology*



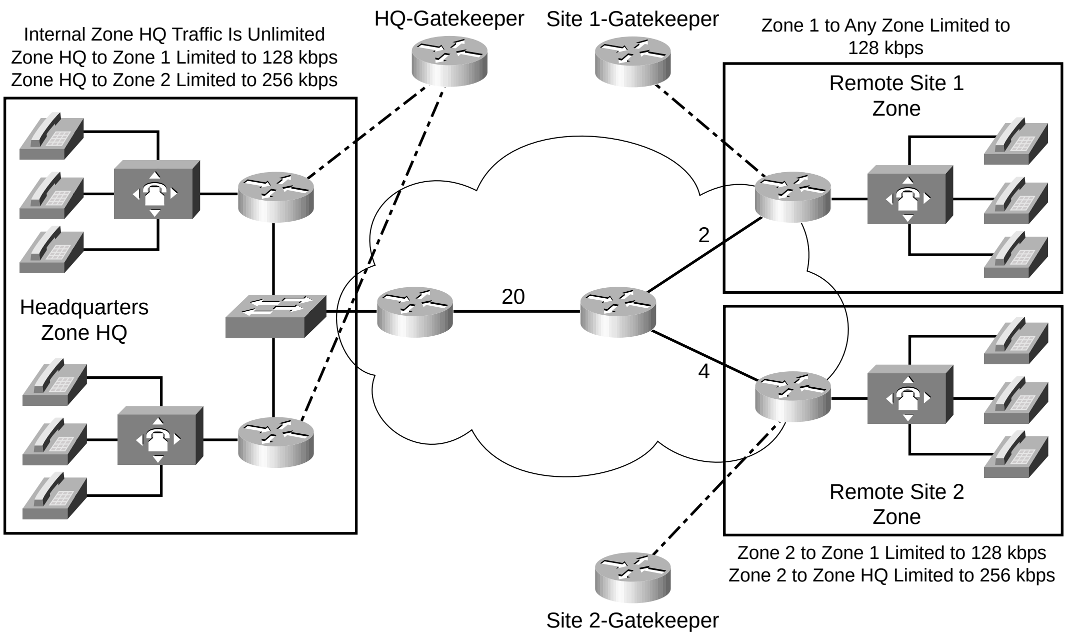
Assume that the gatekeeper zone bandwidth is still configured to allow a maximum of 128 kbps, limiting the number of simultaneous calls to two.

With a single zone configured, the gatekeeper protects the bandwidth of the WAN link from Site 1 to the WAN aggregation point by not allowing more than two calls across that link. When two calls are in progress, however, additional calls between the PBXs at Headquarters are blocked even though there is ample bandwidth in the campus backbone to handle the traffic.

Multizone Topology

To solve the single-zone problem of reducing the call volume of the network to the capabilities of the lowest-capacity WAN link, you can design multiple gatekeeper zones into the network. A good practice in designing multiple zones is to create one zone per site, as shown in Figure 8-24.

Figure 8-24 *Simple Enterprise Multizone Topology*


HQ-Gatekeeper
Site 1-Gatekeeper
Internal Zone HQ Traffic Is Unlimited
Zone HQ to Zone 1 Limited to 128 kbps
Zone HQ to Zone 2 Limited to 256 kbps
Zone 1 to Any Zone Limited to 128 kbps
Remote Site 1 Zone
Headquarters Zone HQ
20
2
4
Remote Site 2 Zone
Zone 2 to Zone 1 Limited to 128 kbps
Zone 2 to Zone HQ Limited to 256 kbps
Site 2-Gatekeeper

The Site 1 gatekeeper limits the number of calls active in Site 1, regardless of where those calls originate or terminate, to two simultaneous calls by limiting the available bandwidth to 128 kbps. Because there is only one gateway at Site 1, there is no need to configure a limit for the intrazone call traffic. All interzone traffic is limited to two calls to protect the WAN link connecting Site 1.

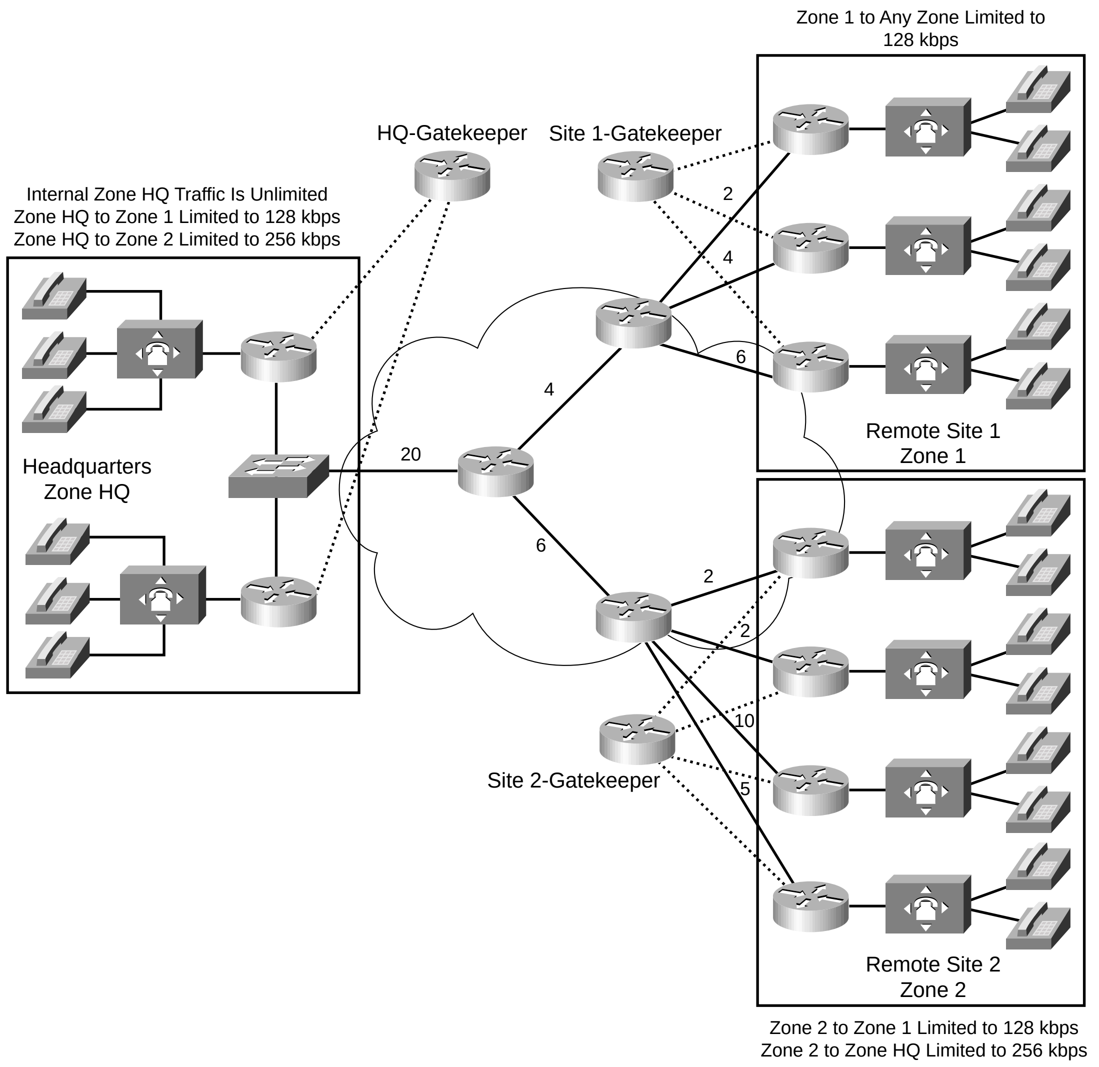
At Site 2 there is also a single gateway, and therefore no need to limit the intrazone call traffic. There are separate interzone limits for the following scenarios:

- Calls between Site 2 and the Headquarters site are limited to a maximum of four calls on the WAN link connecting Site 2.
- Calls between Site 2 and Site 1 are limited to a maximum of two calls on the WAN link connecting Site 1.

The Headquarters site has a similar configuration to Site 2 except that calls are unlimited within Headquarters; not because there is a single gateway, but because there is ample bandwidth between the gateways at this site.

In the network topology in Figure 8-24, gatekeeper CAC provides sufficient granularity needed to protect voice conversations across the low-speed WAN links. Consider another network topology in which there are multiple gateways per zone, however, with each gateway at the remote sites having a separate WAN connection to the aggregation point. Figure 8-25 shows such a network topology.

Figure 8-25 *Complex Enterprise Multizone Topology*

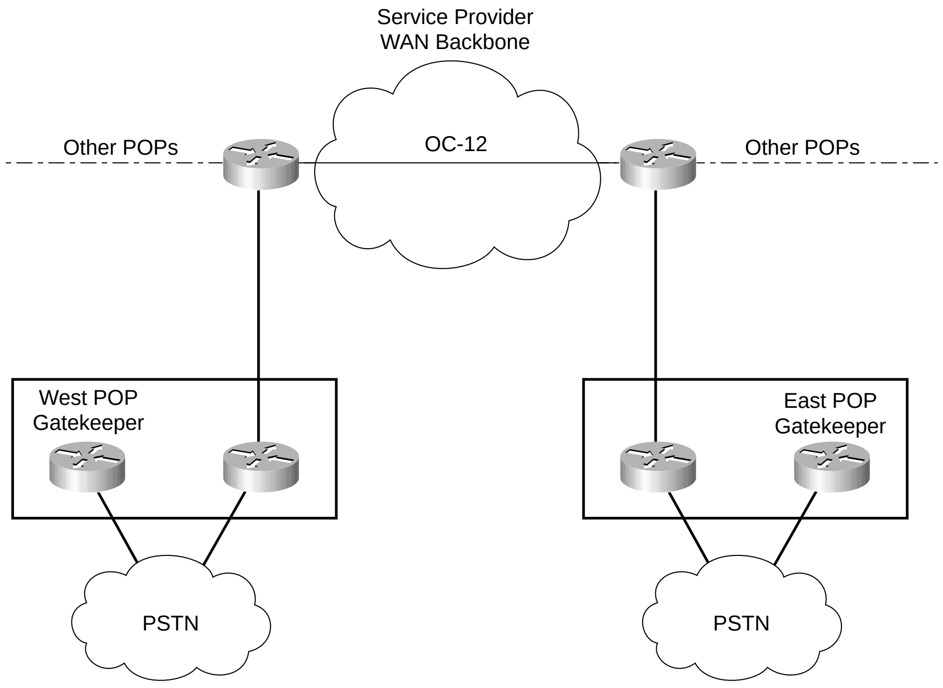

Zone 1 to Any Zone Limited to 128 kbps
HQ-Gatekeeper
Site 1-Gatekeeper
Internal Zone HQ Traffic Is Unlimited
Zone HQ to Zone 1 Limited to 128 kbps
Zone HQ to Zone 2 Limited to 256 kbps
2
4
6
4
20
Headquarters Zone HQ
Remote Site 1 Zone 1
6
2
2
10
5
Site 2-Gatekeeper
Remote Site 2 Zone 2
Zone 2 to Zone 1 Limited to 128 kbps
Zone 2 to Zone HQ Limited to 256 kbps

Of the three gateways in remote Site 1, the slowest WAN access link can carry a maximum of two simultaneous calls. Because the gatekeeper bandwidth limitation is configured per zone and not per gateway, there is no facility within gatekeeper CAC to limit the calls to specific gateways within the zone. To ensure protection of voice conversations, the zone bandwidth must be configured for the slowest link in the zone. For both remote Sites 1 and 2, the slowest link is 128 kbps bandwidth or two calls.

This configuration ensures proper voice quality at all times, but it is also wasteful of the gateways that could terminate more calls without oversubscribing their WAN bandwidth. In this network configuration, CAC is activated too soon and reroutes calls over to the PSTN when in fact they could have been carried by the WAN. In this type of topology, gatekeeper CAC is not sufficient to protect voice quality over the WAN link, and, at the same time, optimize the bandwidth use of all WAN links.

The last configuration to consider is an SP network where the gateways in the POPs are connected via Fast Ethernet to the WAN edge router, as shown in Figure 8-26.

Figure 8-26 *Service Provider Topology with Multiple Gateways per Zone*



In this network, gatekeeper CAC is sufficient, even though there are multiple gateways per zone, because the connections to specific gateways within the zone are not the links that need protection. The bandwidth that needs protection is the WAN access link going into the backbone that aggregates the call traffic from all gateways. A gatekeeper bandwidth limitation for the zone indeed limits the number of calls over that link. It is assumed that the OC-12 backbone link is overengineered and requires no protection.

In summary, a multizone gatekeeper network offers the following CAC attributes:

- The WAN bandwidth at each connecting site can be protected, provided each site is also a zone. For small remote sites in an enterprise network, this often translates into a zone per gateway, which might not be a practical design.
- The bandwidth within a site can be protected if necessary, but this is frequently of little value because there is only one gateway in the site (small remote offices, or a customer premises equipment [CPE] entry point to an SP-managed network service) or because a high-speed LAN is between the gateways (large sites and SP POPs).
- Gatekeeper CAC is a method well suited to limit the number of calls between sites.

Gatekeeper CAC cannot protect the bandwidth on WAN segments not directly associated with the zones. For example, gatekeeper CAC cannot protect the backbone link marked with 20 calls in the simple enterprise topology shown in Figure 8-24, unless the slowest link approach is followed.

Zone-per-Gateway Design

Because the zone-per-gateway design offers the finest granularity of gatekeeper CAC, it is worth exploring a little further. In enterprise networks, this often makes sense from the following points of view:

- Geographical considerations
- CAC to protect the WAN access link into a site containing a single gateway
- Dialing plans that coincide with sites, allowing a zone prefix to easily translate to the gateway serving a specific site

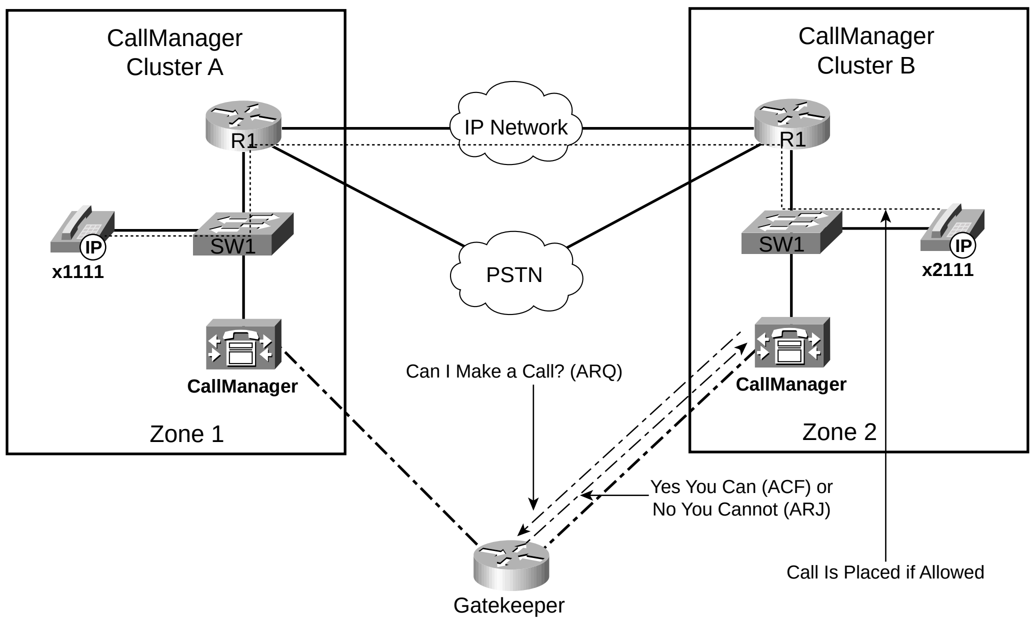
A gatekeeper is a logical concept, not a physical concept. Each gatekeeper therefore does not imply that there is a separate box in the network; it merely means that there is a separate *local zone* statement in the configuration.

With Cisco IOS Release 12.1(5)T and Release 12.2, a small remote site router has the capability to provide gateway, gatekeeper, and WAN edge functionality; however, a zone-per-gateway design complicates the scalability aspect that gatekeepers bring to H.323 networks. You should therefore carefully consider the advantages and limitations of such a design.

Gatekeeper in CallManager Networks

Of all the CAC mechanisms discussed in this chapter, gatekeeper zone bandwidth is the only method applicable to multi-site distributed CallManager clusters. In this scenario, the CallManager behaves like a VoIP gateway to the H.323 gatekeeper, as is shown in Figure 8-27.

Figure 8-27 *Gatekeeper in a CallManager Topology*



In this example, the CallManager in Zone 2 requests the WAN bandwidth to carry a voce conversation from the gatekeeper on behalf of the IP Phone x2111. The gatekeeper determines whether each zone has sufficient bandwidth to establish the call. If the gatekeeper allows the call to proceed, the CallManager in Zone 2 contacts the CallManager in Zone 1 to begin call setup. If the gatekeeper rejects the call, the CallManager in Zone 2 can reroute the call over the PSTN to reach the called party.

Zone Bandwidth Calculation

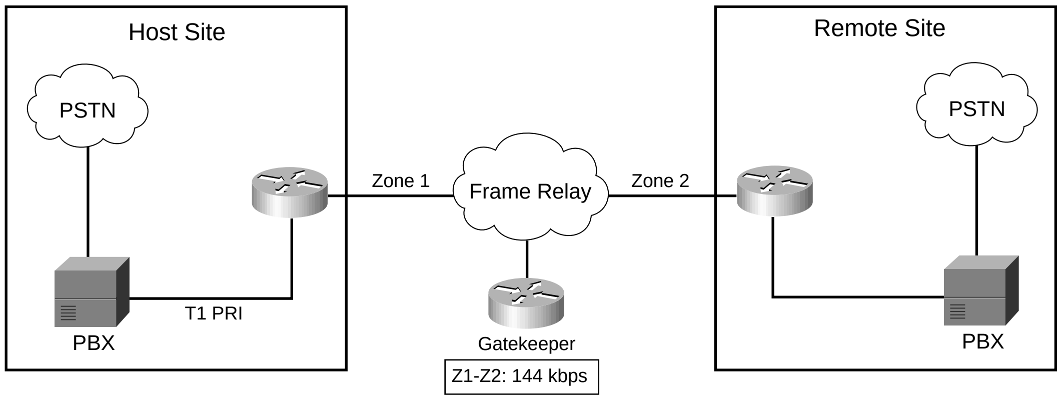
The gatekeeper does not have any knowledge of network topology and does not know how much bandwidth is available for calls. Nor does the gatekeeper know how much of the configured bandwidth on the links is currently used by other traffic. The gatekeeper takes a fixed amount of bandwidth, statically configured on the gatekeeper, and subtracts a certain amount of bandwidth for each call that is set up. Bandwidth is returned to the pool when a call is disconnected. If a request for a new call causes the remaining bandwidth to become less than zero, the call is denied. The gatekeeper does not use bandwidth reservation. Instead, the gatekeeper performs a static calculation to decide whether a new call should be allowed.

It is the responsibility of the gateways to inform the gatekeeper of how much bandwidth is required for a call. Video gateways therefore could request a different bandwidth for every call setup. One video session may require 256 kbps, whereas another requires 384 kbps. Voice gateways should consider codec, Layer 2 encapsulation, and compression features, such as compressed Real Time Protocol (cRTP), when requesting bandwidth from the gatekeeper. Sometimes these features are not known at the time of call setup, in which case a bandwidth change request can be issued to the gatekeeper after call setup to adjust the amount of bandwidth used by the call. As of Cisco IOS Software Release 12.2(2)XA, Cisco has implemented only the functionality of reporting any bandwidth changes when the reported codec changes.

Prior to Cisco IOS Software Release 12.2(2)XA on a Cisco H.323 Gateway, calls were always reported to require a bandwidth of 64 kbps. This is the unidirectional payload bandwidth for a Cisco G.711 codec. If the endpoints in the call chose to use a more efficient codec, this was not reported to the Cisco gatekeeper. In the Cisco IOS Software Release 12.2(2)XA version of the Cisco H.323 Gateway or a later version that conforms with H.323 version 3, the reported bandwidth is bidirectional. Initially, 128 kbps is specified. If the endpoints in the call select a more efficient codec, the Cisco gatekeeper is notified of the bandwidth change.

Figure 8-28 illustrates a network that uses a gatekeeper to limit bandwidth to 144 kbps between two zones. A connection is requested with an initial bandwidth of 128 kbps. After the call has been set up, the endpoints report the change in the bandwidth. Assuming the endpoints are using G.729, 16 kbps is subtracted from the configured maximum of 144 kbps. The 16 kbps represents a bidirectional G.729 payload stream to the gatekeeper.

Figure 8-28 *Gatekeeper Bandwidth Calculation*



In the event that the second call arrives before the endpoint requests the change in bandwidth for the first call, the Cisco gatekeeper rejects the second call, because the total requested bandwidth exceeds the 144 kpbs configured. As call 1 is being set up, for example, the gatekeeper records this call as a 128-kbps call and waits for a codec change before adjusting the recorded bandwidth. At this point, the gatekeeper has subtracted 128 kbps from the configured 144 kbps, leaving 16 kbps available. If call 2 requests admission before the codec change is reported in call 1, the gatekeeper does not have enough available bandwidth to allow this 128-kbps call to proceed.

Gatekeeper zone bandwidth remains an inexact science because the gateway may not have full knowledge of the bandwidth required by the call. The following examples describe common situations where the gateway will not have full knowledge of the bandwidth required per call:

- The gateway is attached to an Ethernet segment in a campus network where cRTP does not apply and where the Layer 2 headers are larger than they would be for Frame Relay or Multilink Point-to-Point Protocol (MLP) on the WAN legs.
- A different codec is used in the campus network from the WAN segments, leveraging codec transcoding functionality at the WAN edge.
- In the backbone of the network, ATM is used as the transport technology. In this case, cell padding should be taken into account for bandwidth calculations.
- cRTP may be used at the WAN edge router.

Zone Bandwidth Configuration

As of Cisco IOS Software Release 12.1(5)T and Release 12.2, the following types of zone bandwidth limitations can be configured on the gatekeeper:

- The maximum bandwidth for all H.323 traffic between the local zone and a specified remote zone.
- The maximum bandwidth allowed for a single session in the local zone. This configuration is typically used for video applications, not for voice.
- The maximum bandwidth for all H.323 traffic allowed collectively to all remote zones.

Tables 8-16 and 8-17 list the gatekeeper command and options used to configure gatekeeper zone bandwidth.

Table 8-16 *Gatekeeper Bandwidth Command*

Command	Mode and Function
bandwidth {**interzone** | **total** | **session**} {**default** | **zone** *zone-name*} *max-bandwidth*	Specifies the gatekeeper zone bandwidth restrictions
bandwidth remote *max-bandwidth*	Specifies the total bandwidth for H.323 traffic between this gatekeeper and any other gatekeeper

Table 8-17 *Gatekeeper Bandwidth Command Options*

Bandwidth Command Options	Function
interzone	Specifies the total amount of bandwidth for H.323 traffic from the zone to any other zone.
total	Specifies the total amount of bandwidth for H.323 traffic allowed in the zone.
session	Specifies the maximum bandwidth allowed for a session in the zone.
default	Specifies the default value for all zones.
zone *zone-name*	Names the particular zone.
max-bandwidth	Maximum bandwidth. For **interzone** and **total**, the range is from 1 to 10,000,000 kbps. For **session**, the range is from 1 to 5000 kbps.

Gatekeeper Zone Bandwidth Summary

Gatekeeper CAC works well in network designs where the desire is to limit the number of calls between sites. This may be required due to either bandwidth limitations or business policy. If bandwidth limitations are on the WAN links, manual calculations can be performed to translate the maximum number of calls allowed between sites into a bandwidth figure that will cause the gatekeeper to deny calls when the calculated number is exceeded.

Gatekeepers do not share database information. If the primary gatekeeper fails, a secondary gatekeeper can begin to perform CAC for the network; however, the secondary gatekeeper has no knowledge of the amount of bandwidth currently used in the zone or the true number of active voice conversations. Until the secondary gatekeeper has an accurate count of current calls and consumed bandwidth, the network may become oversubscribed with voice conversations. If alternate gatekeepers are used as the redundancy method, this problem is circumvented.

A major advantage of gatekeeper CAC is that it is the only CAC method that can incorporate mixed networks of Cisco IOS gateways and CallManagers with IP Phones.

Table 8-18 evaluates the gatekeeper zone bandwidth mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-18 *Gatekeeper Zone Bandwidth CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	VoIP/H.323 only
Toll bypass or IP telephony	Toll bypass and IP telephony (Some caveats exist if both the CallManager and Cisco IOS gateways are used in the same zone.)
Platforms and releases	Cisco IOS gateways since Release 11.3 (CM has recent changes in E.164 registration, and bandwidth requested per call.)
PBX trunk types supported	All
End to end, local, or IP cloud	End-to-end between originating gateway and terminating gateway, although not aware of the network topology in between
Per call, interface, or endpoint	Per call
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	Part of the gatekeeper RAS messaging

Integrated Services / Resource Reservation Protocol

In Chapter 2, "QoS Tools and Architectures," this book introduced the concepts of differentiated services (DiffServ) and integrated services (IntServ). Most of the rest of the chapters in this book cover QoS features that either use DiffServ, or behave similarly. This section covers some of the key concepts and configuration related to IntServ.

The IntServ model, defined in RFC 1633, includes provisions for best-effort traffic, real-time traffic, and controlled-link sharing. Controlled-link sharing is a management facility that enables administrators to divide traffic into classes and assign a minimum percentage of the link to each desired class during times of congestion, while sharing this assigned bandwidth with other classes when congestion is not present.

In the IntServ model, an application requests a specific level of service from the network before sending data. An application informs the network of its traffic profile by sending an explicit signaling message, requesting a particular level of service based on bandwidth and delay requirements. The application is expected to send data only after it gets a confirmation from the network. The network performs admission control, based on information from the application and available network resources. The network also commits to meeting the QoS requirements of the application as long as the traffic remains within the profile specifications. After a confirmation from the network has been received, the application can send data that conforms to the described traffic profile. The network fulfills its commitment by maintaining per-flow state and then performing packet classification, policing, and intelligent queuing based on that state.

The Resource Reservation Protocol (RSVP) defines the messages used to signal resource admission control and resource reservations conforming to IntServ. When the application signals the required level of service, if resources are available, the network devices accept the RSVP reservation.

To provide the committed service levels, the networking devices must identify packets that are part of the reserved flows, and provide appropriate queuing treatment. When committing service to a new flow, Cisco IOS installs a traffic classifier in the packet-forwarding path, allowing the network devices to identify packets for which a reservation has been made. To provide the required service level, Cisco IOS uses existing QoS tools.

Most networks that use RSVP and IntServ also use DiffServ features as well. For instance, the same queuing tool that provides minimum bandwidth commitments for DiffServ can also be used to dynamically reserve bandwidth for IntServ and RSVP. As noted in Chapter 2, DiffServ does scale much better than IntServ. Cisco has a relatively new feature called *RSVP aggregation*, which integrates resource-based admission control with DiffServ, which allows for better scalability, but with strict QoS guarantees for prioritized traffic, such as VoIP calls.

RSVP Levels of Service

To request a level of service from the network, the acceptable levels of service must be defined. Currently RSVP defines three distinct levels of service:

- Guaranteed QoS
- Controlled-load network element service
- Best-effort levels of service

The *guaranteed* QoS level of service is used to guarantee that the required bandwidth can be provided to a flow with a fixed delay. This service makes it possible to provide a service that guarantees both delay and bandwidth. This type of RSVP service is used by voice gateways when reserving bandwidth for voice flows.

The *controlled-load* level of service tightly approximates the behavior visible to applications receiving best-effort service under uncongested conditions from the same series of network elements. So, if the application can work well when the network is uncongested, the controlled load level of RSVP service can work well. If the network is functioning correctly with controlled-load service, applications may assume the following:

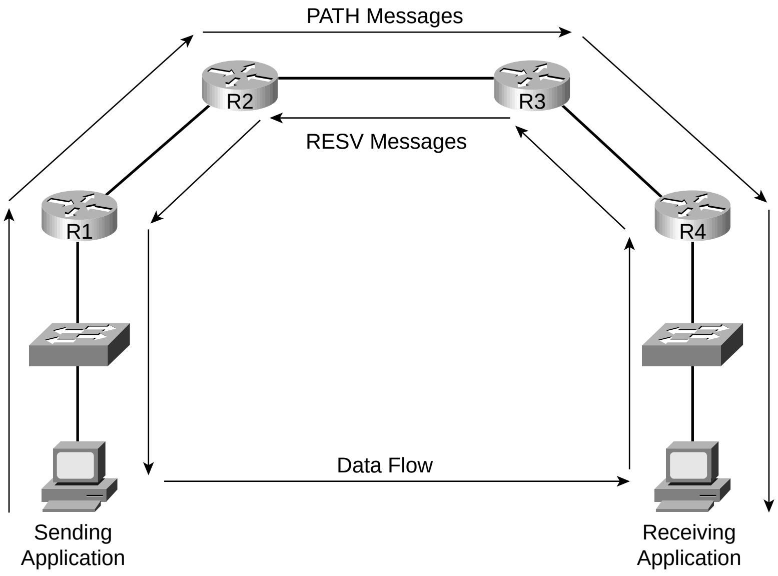
- **Very low loss**—If the network is not congested, queues will not fill, so no drops occur in queues. The only packet loss occurs due to transmission errors.
- **Very low delay and jitter**—If the network is not congested, queues will not fill, and queuing delay is the largest variable component of both delay and jitter.

The *best-effort* level of service does not grant priority to the flow. The flow receives the same treatment as any other flow within the router.

RSVP Operation

RSVP uses path and resv messages to establish a reservation for a requested data flow. A *path message* carries the traffic description for the desired flow from the sending application to the receiving application, and the *resv message* carries the reservation request for the flow from the receiving application to the sending application. Figure 8-29 shows a network that consists of several routers carrying the path and resv messages for a sending and receiving application.

Figure 8-29 *RSVP Path and Resv Messages*

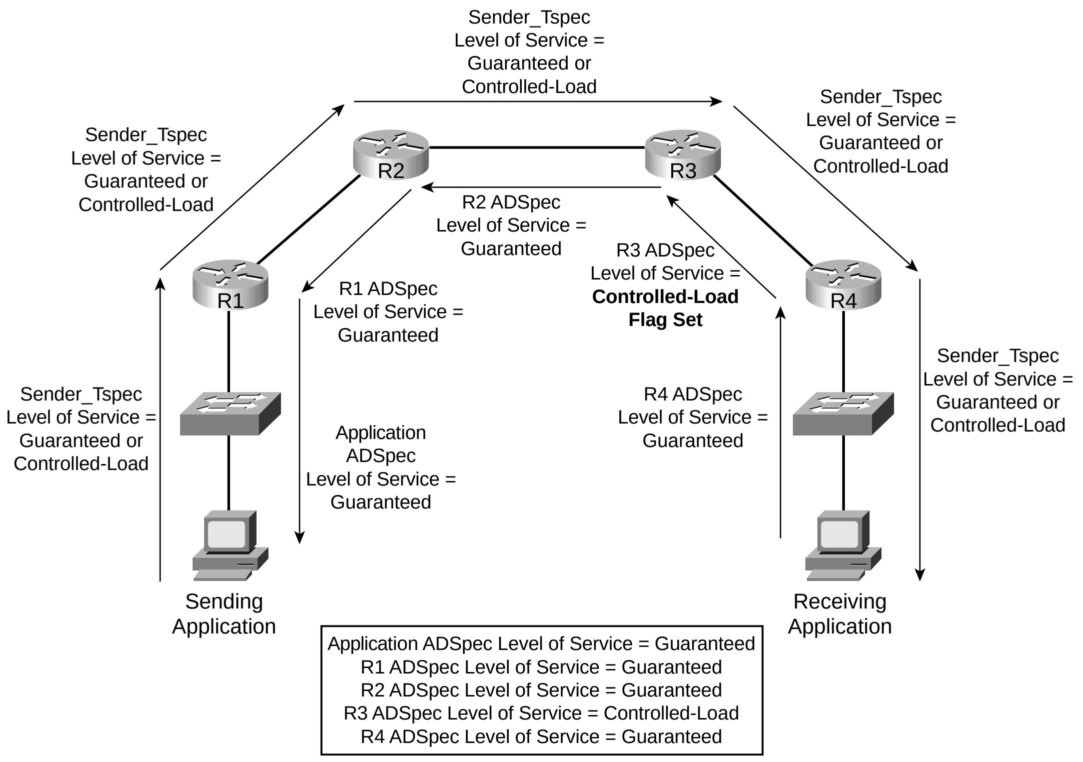


In this example, the sending application responds to an application request by sending a path message to the RSVP router R1 describing the level of service required to support the requested application. R1 forwards the path message to router R2, R2 forwards the message to router R3, R3 forwards the message to R4, and finally R4 forwards the message to the receiving application. If the level of service described in the path message can be met, the receiving application sends a resv message to RSVP router R4. The resv message causes each router in the data path to the sending application to place a reservation with the requested level of service for the flow. When the resv message reaches the sending application, the data flow begins.

Path messages describe the level of service required for a particular flow through the use of the *Sender_Tspec* object within the message. The Sender_TSpec object is defined by the sending application and remains in tact as the path message moves through the network to the receiving application. Also included in the path message is the ADSpec object. The ADSpec object is modified by each intermediary router as the RSVP path message travels from the sending application to the receiving application. The ADSpec object carries traffic information generated by the intermediary routers. The traffic information describes the properties of the data path, such as the availability of the specified level of service. If a level of service described in the ADSpec object is not implemented in the router, a flag is set to alert the receiving application.

Effectively, the Sender_Tspec states what the application needs, and the ADSpec allows each successive router to comment on the level of service it can and cannot support. The receiving application can look at both settings and decide whether to accept or reject the reservation request. Figure 8-30 shows the Sending_TSpec and ADSpec objects sent in a path message from a sending application.

Figure 8-30 *Path Messages Sending_TSpec and ADSpec Objects*

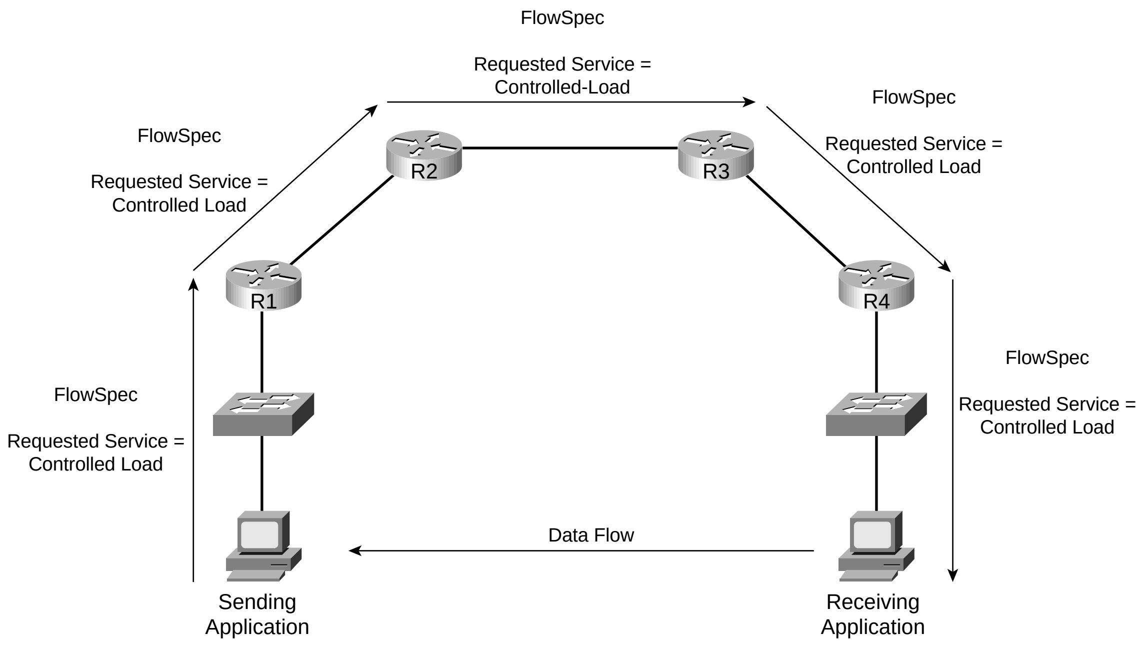


Sender_Tspec Level of Service = Guaranteed or Controlled-Load
Sender_Tspec Level of Service = Guaranteed or Controlled-Load
Sender_Tspec Level of Service = Guaranteed or Controlled-Load
R2
R3
R2 ADSpec Level of Service = Guaranteed
Sender_Tspec Level of Service = Guaranteed or Controlled-Load
R3 ADSpec Level of Service = **Controlled-Load Flag Set**
R1
R1 ADSpec Level of Service = Guaranteed
R4
Sender_Tspec Level of Service = Guaranteed or Controlled-Load
Sender_Tspec Level of Service = Guaranteed or Controlled-Load
R4 ADSpec Level of Service = Guaranteed
Application ADSpec Level of Service = Guaranteed
Sending Application
Receiving Application
Application ADSpec Level of Service = Guaranteed
R1 ADSpec Level of Service = Guaranteed
R2 ADSpec Level of Service = Guaranteed
R3 ADSpec Level of Service = Controlled-Load
R4 ADSpec Level of Service = Guaranteed

In Figure 8-30 the Sending_TSpec is advertising the required level of service for this application as guaranteed. Because intermediary routers do not modify the Sending_TSpec, this value is passed through the network to the receiving application. The ADSpec lists the level of service available at each intermediate router along the data path. As the path message reaches router R3, the ADSpec object is modified to indicate that router R3 can only provide a controlled-load level of service. Essentially, the sending application has asked for guaranteed RSVP service, and the ADSpec implies that at least one router can only support a controlled-load service for this request.

The receiver of the path message replies with an resv message. Resv messages request a reservation for a specific flow through the use of a FlowSpec object. The FlowSpec object provides a description of the desired flow and indicates the desired level of service for the flow to the intermediary routers, based on the information received from the path message in the ADSpec object. A single level of service must be used for all nodes in the network—in other words, it can be controlled-load or guaranteed service, but not both. Because the sending application requires either guaranteed or controlled-load level of service to begin the flow, and router R3 can only provide controlled-load level of service, the receiving application can only request a reservation specifying controlled-load level of service. Example 8-31 shows the FlowSpec object sent in a resv message from a receiving application.

Figure 8-31 *Resv Messages FlowSpec Objects*


FlowSpec
Requested Service = Controlled-Load
FlowSpec
Requested Service = Controlled Load
FlowSpec
Requested Service = Controlled Load
R2
R3
R1
R4
FlowSpec
Requested Service = Controlled Load
FlowSpec
Requested Service = Controlled Load
Data Flow
Sending Application
Receiving Application

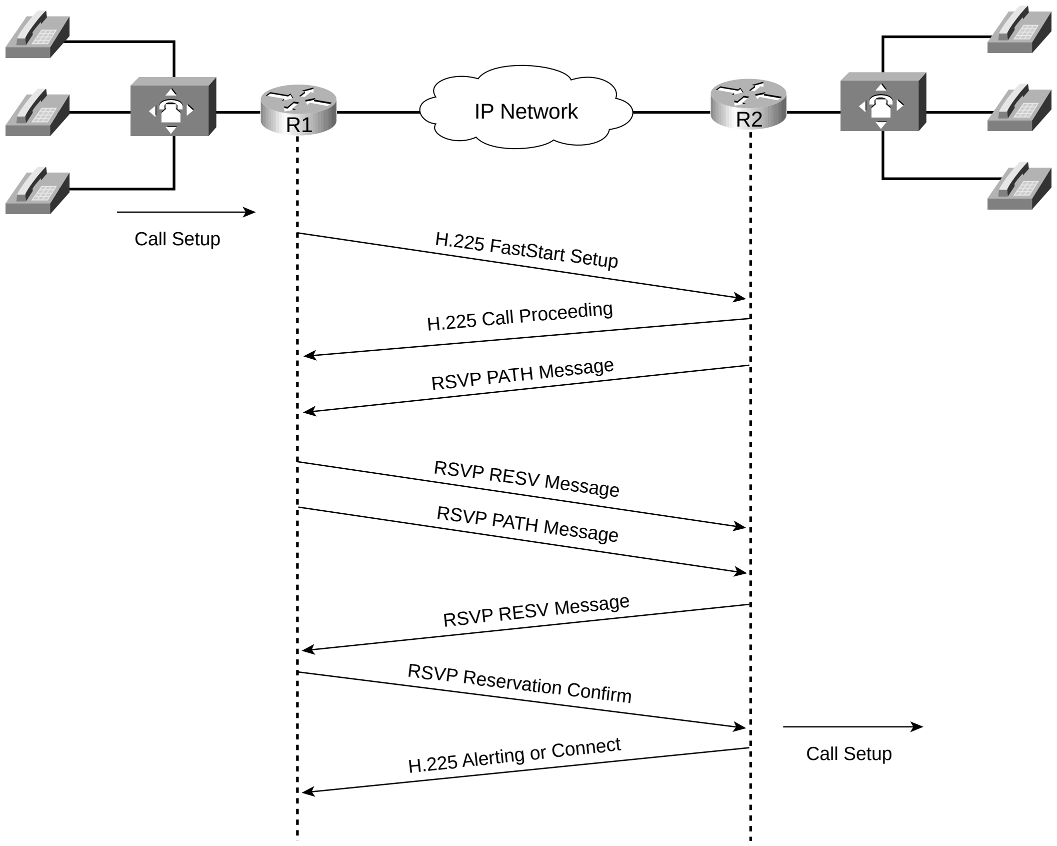
After the reservation for controlled-load service has been installed on the intermediary routers, the flow is established between the sending application and the receiving application.

RSVP/H.323 Synchronization

Voice gateways use H.323 signaling when setting up voice calls. To reserve network resources for voice calls with RSVP, RSVP in an H.323 environment operates by synchronizing its signaling with H.323 version 2 signaling. This synchronization ensures a bandwidth reservation is established in both directions before a call moves to the *alerting*, or *ringing*, H.323 state. RSVP and H.323 version 2 synchronization provides the voice gateways with the capability to modify H.323 responses based on current network conditions. These modifications allow a gateway to deny or reroute a call in the event that QoS cannot be guaranteed.

Figure 8-32 shows a call flow of the H.323 call setup messages and the RSVP reservation messages.

Figure 8-32 *RSVP Call Setup for an H.323 Voice Call*



In this example, the originating gateway initiates a call to the terminating gateway with an H.225 setup message. The terminating gateway responds with a call proceeding message. Because voice conversations require a bidirectional traffic stream, each gateway initiates a reservation request by sending an RSVP path message. An RSVP resv message is sent in response indicating that the requested reservation has been made. The originating gateway sends an RSVP resvconf message to the terminating gateway confirming the reservation. At this point, the terminating gateway proceeds with the H.323 call setup by sending an alerting message to the originating gateway.

RSVP Synchronization Configuration

To configure RSVP for use with H.323, you need to enable RSVP as normal and add a few commands to the H.323 dial peers. By default, RSVP is disabled on each interface to remain backward compatible with systems that do not implement RSVP. To enable RSVP for IP on an interface, use the **ip rsvp bandwidth** command. This command starts RSVP and sets the maximum bandwidth for all RSVP flows combined, and a single-flow bandwidth limit. The default maximum bandwidth is up to 75 percent of the bandwidth available on the interface. By default, a single flow has the capability to reserve the entire maximum bandwidth.

To configure a network to use RSVP, regardless of whether it is used for voice calls, you just configure the **ip rsvp bandwidth** command on every link in the network. You also need to configure a queuing tool that supports RSVP on each interface, namely WFQ, IP RTP Priority, or LLQ. Configuration details are listed later in this section.

To use RSVP to allow voice gateways to reserve bandwidth for voice calls, RSVP synchronization must be configured on the voice gateways. When doing so, each dial peer is configured with an accept QoS (**acc-qos**) value, and a request QoS (**req-qos**) value. Each of these two settings indicates the level of service that the dial peer will request, and the level that it will accept to make a reservation. Table 8-19 describes the available options for the **acc-qos** and **req-qos** commands.

Table 8-19 **acc-qos** *and* **req-qos** *Command Options*

acc-qos and req-qos Command Options	Function
best-effort	Indicates that RSVP makes no bandwidth reservation.
controlled-load	Indicates that RSVP guarantees a single level of preferential service, presumed to correlate to a delay boundary. The controlled-load service uses admission (or capacity) control to ensure that preferential service is received even when the bandwidth is overloaded.
guaranteed-delay	Indicates that RSVP reserves bandwidth and guarantees a minimum bit rate and preferential queuing if the bandwidth reserved is not exceeded.

This table was derived from the following:
www.cisco.com/en/US/partner/products/sw/iosswrel/ps1834/products_feature_guide09186a008008045c.html.

For instance, two voice gateways could be configured to request, and to only accept, guaranteed-delay service. That combinations make sense with voice CAC—voice calls need a particular amount of bandwidth, and they need low delay. Examples 8-15 and 8-16 demonstrate the requested QoS and acceptable QoS dial-peer configuration of the **req-qos** and **acc-qos** commands, respectively.

Example 8-15 *Originating Gateway* **req-qos** *and* **acc-qos** *Dial-Peer Configuration*

```
dial-peer voice 300 voip
 destination-pattern 3......
 session target ipv4:10.1.1.1
!Configures RSVP CAC for voice calls using the dial peer.
 req-qos guaranteed-delay
 acc-qos guaranteed-delay
```

Example 8-16 *Terminating Gateway* **req-qos** *and* **acc-qos** *Dial-Peer Configuration*

```
dial-peer voice 300 voip
 destination-pattern 3......
 session target ipv4:10.1.1.2
!Configures RSVP CAC for voice calls using the dial peer.
 req-qos guaranteed-delay
 acc-qos guaranteed-delay
```

RSVP and H.323 synchronization has an added oddity that is not obvious at first glance. H.323 call setup actually includes two sets of call setup messages: one for the voice that travels in one direction, and another set for the voice in another direction. Each of the two requires a separate RSVP reservation. In addition to that, depending on what has been configured on the originating and terminating gateways, the gateways may request a high level of service (for example, guaranteed load), and be willing to accept a lower level of service. So, Table 8-20 summarizes the results of nine call setup scenarios based on the QoS levels that can be configured in the VoIP dial peers at the originating and terminating gateways. This table does not include cases where the requested QoS is configured for best effort and the acceptable QoS is configured for a value other than best effort, because those configurations are considered invalid.

In Examples 8-15 and 8-16, the dial-peer configuration for both the originating and terminating gateway were configured for guaranteed-delay. Based on Table 8-20, the call will proceed only if both RSVP reservations succeed.

Table 8-20 **acc-qos** *and* **req-qos** *Call States*

Originating Gateway		Terminating Gateway		
Requested QoS	**Acceptable QoS**	**Requested QoS**	**Acceptable QoS**	**Results**
Controlled-load or guaranteed-delay	Controlled-load or guaranteed-delay	Controlled-load or guaranteed-delay	Controlled-load or guaranteed-delay	Call proceeds only if both RSVP reservations succeed.
Controlled-load or guaranteed-delay	Controlled-load or guaranteed-delay	Controlled-load or guaranteed-delay	Best-effort	Call proceeds only if both RSVP reservations succeed.
Controlled-load or guaranteed-delay	Controlled-load or guaranteed-delay	Best-effort	Best-effort	Call is released.
Controlled-load or guaranteed-delay	Best-effort	Controlled-load or guaranteed-delay	Controlled-load or guaranteed-delay	Call proceeds only if both RSVP reservations succeed.
Controlled-load or guaranteed-delay	Best-effort	Controlled-load or guaranteed-delay	Best-effort	Call proceeds regardless of RSVP results. If RSVP reservation fails, call receives best-effort service.
Controlled-load or guaranteed-delay	Best-effort	Best-effort	Best-effort	Call proceeds with best-effort service.
Best-effort	Best-effort	Controlled-load or guaranteed-delay	Controlled-load or guaranteed-delay	Call is released.
Best-effort	Best-effort	Controlled-load or guaranteed-delay	Best-effort	Call proceeds with best-effort service.
Best-effort	Best-effort	Best-effort	Best-effort	Call proceeds with best-effort service.

Classification for Voice Packets into LLQ

As you learned in previous chapters, LLQ is one of the most important Cisco QoS mechanisms to ensure quality for voice conversations, because it prioritizes voice packets over data packets at the router egress interface. For this to work, voice packets must be classified such that they are placed in the priority queue (PQ) portion of LLQ.

Cisco IOS provides the service levels that RSVP accepts by working in conjunction with IOS queuing tools. As a general Cisco IOS feature, RSVP has its own set of reserved queues within Class-Based Weighted Fair Queuing (CBWFQ) for traffic with RSVP reservations. So, RSVP can create hidden queues that compete with the explicitly defined CBWFQ queues, with the RSVP queues getting very low weights—and remember, with all WFQ-like features, lower weight means better service.

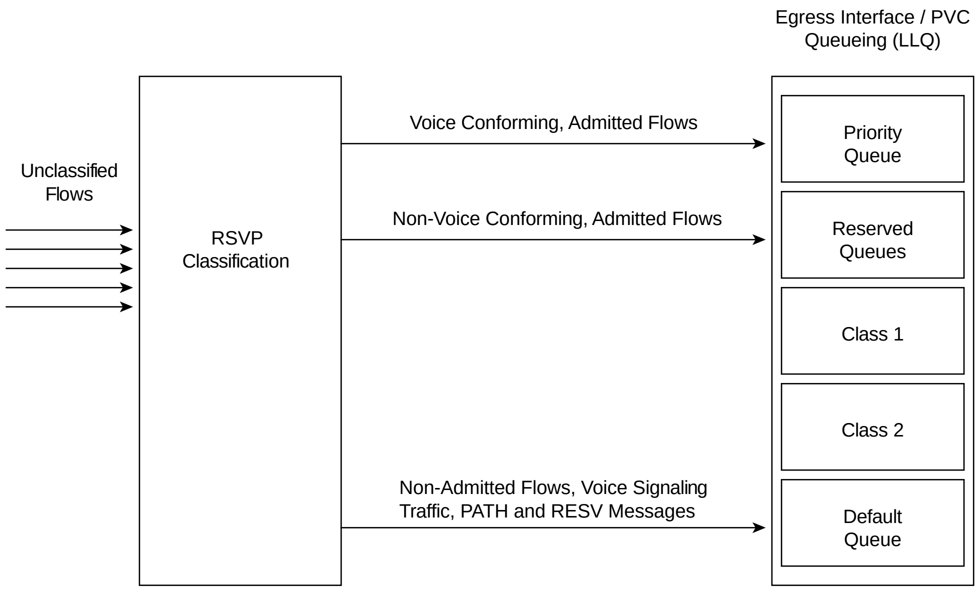
You would just configure CBWFQ, and then RSVP, and not consider the two features together. However, the performance of the voice flows with RSVP reservations would actually suffer. The reason is that the reserved hidden RSVP queues, although they have a low weight, are separate from the PQ feature of CBWFQ/LLQ. Packets in reserved RSVP queues do not get strict priority over packets from other queues. It has long been known that this treatment, a low-weight queue inside WFQ, is insufficient for voice quality over a congested interface. Therefore, when RSVP is configured for a voice call, the voice packets need to be classified into the PQ.

IOS solves the problem by having RSVP put voice-like traffic into the existing LLQ priority queue on the interface. RSVP uses a profile to classify a flow of packets as a voice flow. The profile considers packet sizes, arrival rates, and other parameters to determine whether a packet flow is considered a voice flow. The internal profile, named **voice-like**, is tuned to classify all voice traffic originating from a Cisco IOS gateway as a voice flow without the need for additional configuration. Therefore, while RSVP makes the reservations, it then in turn classifies voice-like traffic into the LLQ PQ, and other non-voice-like traffic with reservations into RSVP queues, as shown in Figure 8-33.

To perform the extra classification logic shown in the figure, RSVP is the first egress interface classifier to examine an arriving packet. If RSVP considers the packet a voice flow, the packet is put into the PQ portion of LLQ. If the flow does not conform to the voice profile **voice-like**, but is nevertheless an RSVP reserved flow, it is placed into the normal RSVP reserved queues. If the flow is neither a voice flow, nor a data RSVP flow, LLQ classifies the packet as it normally would.

Although RSVP voice traffic can be mixed with traffic specified in the priority class within a policy map, voice quality can suffer if both methods are implemented simultaneously. The two methods do not share bandwidth allocation and therefore will lead to an inefficient use of bandwidth on the interface. As bandwidth is defined in the configuration for the egress interfaces, all the bandwidth and priority classes will be allocated bandwidth at configuration time. No bandwidth is allocated to RSVP at configuration time because RSVP requests its bandwidth when the traffic flow begins. RSVP therefore gets allocated bandwidth from the pool that is left after the other features have already allocated their bandwidth.

Figure 8-33 *RSVP Packet-Classification Criteria*


Egress Interface / PVC
Queueing (LLQ)
Unclassified
Flows
RSVP
Classification
Voice Conforming, Admitted Flows
Non-Voice Conforming, Admitted Flows
Non-Admitted Flows, Voice Signaling
Traffic, PATH and RESV Messages
Priority
Queue
Reserved
Queues
Class 1
Class 2
Default
Queue

It is important to note that RSVP classifies only voice bearer traffic, not signaling traffic. Another classification mechanism such as an ACL or DiffServ code point (DSCP) / IP precedence must still be used to classify the voice-signaling traffic if any treatment better than best effort is desired for signaling traffic.

Bandwidth per Codec

Both LLQ and RSVP see the Layer 3 IP packet. Layer 2 encapsulations (FR, MLPPP, and so on) are added after queuing, so the bandwidth allocated by both LLQ and RSVP for a call is based on the Layer 3 bandwidth of the packets. This number is slightly different from the actual bandwidth used on the interface after Layer 2 headers and trailers have been incorporated. RSVP bandwidth reserved for a call also excludes both cRTP and VAD. Table 8-21 summarizes the bandwidth RSVP allocates for calls using different Cisco IOS gateway codecs.

Table 8-21 *RSVP Bandwidth Reservations for Voice Codecs*

Codec	Bandwidth Reserved per Call in LLQ
G.711 (A-law and µ-law)	80 kbps
G.723.1 and G.723.1A (5.3 kbps)	22 kbps

Table 8-21 *RSVP Bandwidth Reservations for Voice Codecs (Continued)*

Codec	Bandwidth Reserved per Call in LLQ
G.723.1 and G.723.1A (6.3 kbps)	23 kbps
G.726 (16 kbps)	32 kbps
G.726 (24 kbps)	40 kbps
G.726 (32 kbps)	48 kbps
G.728	32 kbps
G.729 (all versions)	24 kbps

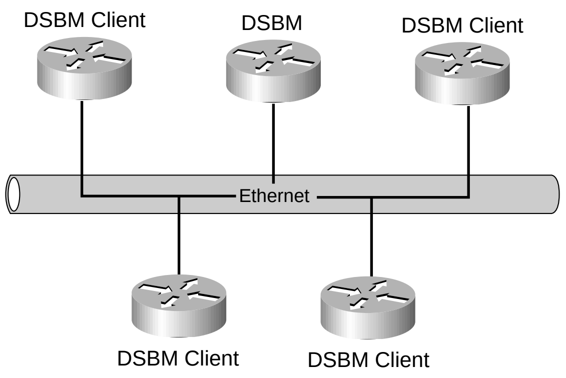
Subnet Bandwidth Management

Subnet bandwidth management (SBM) specifies a signaling method and protocol for LAN-based CAC for RSVP flows. SBM allows RSVP-enabled Layer 2 and Layer 3 devices to support reservation of LAN resources for RSVP-enabled data flows.

SBM uses the concept of a designated entity elected to control the reservations for devices on the managed LAN. The elected candidate is called the *designated subnetwork bandwidth manager* (DSBM). It is the DSBM's responsibility to exercise admission control over requests for resource reservations on a managed segment. A managed segment includes a Layer 2 physical segment shared by one or more senders, such as a shared Ethernet network. The presence of a DSBM makes the segment a managed one. One or more SBMs may exist on a managed segment, but there can be only one DSBM on each managed segment. A router interface can be configured to participate in the DSBM election process. The contender configured with the highest priority becomes the DSBM for the managed segment.

Figure 8-34 shows a managed segment in a Layer 2 domain that interconnects a group of routers.

Figure 8-34 *DSBM Managed Subnet*



When a DSBM client sends or forwards an RSVP path message over an interface attached to a managed segment, it sends the path message to the DSBM of the segment rather than to the RSVP session destination address, as is done in conventional RSVP processing. As part of its message-processing procedure, the DSBM builds and maintains a path state for the session and notes the previous Layer 2 or Layer 3 hop from which it received the path message. After processing the path message, the DSBM forwards it toward its destination address.

The DSBM receives the RSVP resv message and processes it in a manner similar to how RSVP handles reservation request processing, basing the outcome on available bandwidth. The procedure is as follows:

- If it cannot grant the request because of lack of resources, the DSBM returns a resverror message to the requester.
- If sufficient resources are available and the DSBM can grant the reservation request, it forwards the resv message toward the previous hops using the local path state for the session.

RSVP Configuration

The following three tasks are performed on a gateway to originate or terminate voice traffic using RSVP:

1. Turn on the synchronization feature between RSVP and H.323. This is a global command and is turned on by default when Cisco IOS Release 12.1(5)T or later is loaded.
2. Configure RSVP on both the originating and terminating sides of the VoIP dial peers. Configure both the requested QoS (**req-qos**) and the acceptable QoS (**acc-qos**) commands. The **guaranteed-delay** option must be chosen for RSVP to act as a CAC mechanism. Other combinations of parameters may lead to a reservation, but not offer CAC.
3. Enable RSVP and specify the maximum bandwidth on the interfaces that the call will traverse.

Table 8-22 lists the commands used to define and enable RSVP.

Table 8-22 *RSVP Profile,* **req-qos** *and* **acc-qos** *Commands*

Command	Mode and Function
ip rsvp bandwidth *total-bandwidth per-reservation-bandwidth*	Interface configuration mode; defines the cumulative total, and per-reservation, bandwidth limits
call rsvp-sync	Global configuration mode; enables H.323 RSVP synchronization
ip rsvp pq-profile	Global configuration mode; specifies the criteria for determining which flows go into the priority queue

Table 8-22 *RSVP Profile,* **req-qos** *and* **acc-qos** *Commands (Continued)*

Command	Mode and Function
req-qos {**best-effort** | **controlled-load** | **guaranteed-delay**}	Dial-peer configuration mode; specifies the desired quality of services requested to be used in reaching a specified VoIP dial peer
acc-qos {**best-effort** | **controlled-load** | **guaranteed-delay**}	Dial-peer configuration mode; specifies the acceptable quality of service for any inbound and outbound call on a VoIP dial peer
fair-queue [***congestive-discard-threshold*** [***dynamic-queues*** [***reservable-queues***]]]	Interface configuration mode; enables WFQ on an interface, with the last parameter defining the number of RSVP-created queues

Example 8-17 demonstrates the configuration required to enable RSVP with LLQ.

Example 8-17 *Enabling RSVP with LLQ*

```
!Global command enabling RSVP as CAC, turned on by default.
!
!RSVP classification default profile for Cisco VoIP packets
ip rsvp pq-profile voice-like
ip rsvp bandwidth 200 25
!
interface serial 0/0
 service-policy output LLQ-policy
!
voice-port 1/0:0
!
dial-peer voice 100 pots
 destination-pattern 2......
 port 1/0:0
!
dial-peer voice 300 voip
 destination-pattern 3......
 session target ipv4:10.10.2.2
!Configures RSVP CAC for voice calls using the dial peer.
 req-qos guaranteed-delay
 acc-qos guaranteed-delay
```

In the example, the **call rsvp-sync** command enables synchronization of H.323 and RSVP, which allows new call requests to reserve bandwidth using RSPV. The **ip rsvp pq-profile** command tells IOS to classify voice packets into a low-latency queue priority queue, assuming LLQ is configured on the same interface as RSVP. On interface serial 0/0, a **service-policy** command enables a policy map that defines LLQ, and the **ip rsvp bandwidth** command reserves a total of 200 kbps, with a per-reservation limit of 25 kbps. The **req-qos** and **acc-qos** commands tell IOS to make RSVP reservation requests when new call requests have been made.

Although RSVP does support reservations for voice traffic with H.323 sync, IOS does of course support RSVP independent of voice traffic. To support data traffic, LLQ may not even be needed. Example 8-18 demonstrates the configuration required to enable RSVP on a PPP interface, but with WFQ used rather than LLQ.

Example 8-18 *Enabling RSVP on a PPP Interface*

```
interface Serial0/1
 bandwidth 1536
 ip address 10.10.1.1 255.255.255.0
 encapsulation ppp
!Enables WFQ as the basic queueing method.
 fair-queue 64 256 36
!Enables RSVP on the interface.
 ip rsvp bandwidth 1152 24
```

You can also configure RSVP along with traffic shaping. RSVP can reserve bandwidth inside shaping queues. In Example 8-19, the configuration shows RSVP enabled with Frame Relay traffic shaping (FRTS). The **ip rsvp bandwidth** command in this case is a subinterface subcommand, essentially reserving bandwidth inside the FRTS queues for each virtual circuit on that subinterface.

Example 8-19 *Enabling RSVP on a Frame Relay Interface*

```
interface Serial0/0
 bandwidth 1536
 encapsulation frame-relay
 no fair-queue
 frame-relay traffic-shaping
!
interface Serial0/0.2 point-to-point
 ip address 10.10.2.2 255.255.255.0
 frame-relay interface-dlci 17
  class VoIPoFR
!Enables RSVP on the subinterface.
 ip rsvp bandwidth 64 24
map-class frame-relay VoIPoFR
 no frame-relay adaptive-shaping
 frame-relay cir 128000
 frame-relay bc 1280
 frame-relay mincir 128000
!Enables WFQ as the basic queueing method.
 frame-relay fair-queue
 frame-relay fragment 160
```

Finally, Example 8-20 shows RSVP and SBM enabled on Ethernet interface 2. After RSVP is enabled, the interface is configured as a DSBM and SBM candidate with a priority of 100. The configured SBM priority is higher than the default of 64, making the interface a good contender for DSBM status. The maximum configurable priority value is 128, however, so another interface configured with a higher priority could win the election and become the DSBM.

Table 8-23 lists the commands used to enable and define the DSBM in Example 8-20, which also shows the example SBM configuration.

Table 8-23 *SBM Commands*

Command	Mode and Function
ip rsvp bandwidth	Enables RSVP on an interface
ip rsvp dsbm candidate [*priority*]	Configures the interface to participate as a contender in the DSBM dynamic election process, whose winner is based on the highest priority
ip rsvp dsbm non-resv-send-limit rate *kbps*	Configures the average rate, in kbps, for the DSBM candidate
ip rsvp dsbm non-resv-send-limit burst *kilobytes*	Configures the maximum burst size, in KB, for the DSBM candidate
ip rsvp dsbm non-resv-send-limit peak *kbps*	Configures the peak rate, in kbps, for the DSBM candidate

Example 8-20 *Enabling DSBM on an Ethernet Interface*

```
interface Ethernet2
ip address 145.2.2.150 255.255.255.0
no ip directed-broadcast
ip pim sparse-dense-mode
no ip mroute-cache
media-type 10BaseT
ip rsvp bandwidth 7500 7500
ip rsvp dsbm candidate 100
ip rsvp dsbm non-resv-send-limit rate 500
ip rsvp dsbm non-resv-send-limit burst 1000
ip rsvp dsbm non-resv-send-limit peak 500
```

Monitoring and Troubleshooting RSVP

This section covers a few details about the commands used to troubleshoot RSVP installations. Table 8-24 lists other RSVP commands that can be useful in monitoring and troubleshooting RSVP.

Table 8-24 *RSVP Monitoring and Troubleshooting Commands*

Command	Mode and Function
show ip rsvp neighbor [*interface-type interface-number*]	Displays current RSVP neighbors
show ip rsvp request [*interface-type interface-number*]	Displays RSVP-related request information being requested upstream
show ip rsvp reservation [*interface-type interface-number*]	Displays RSVP-related receiver information currently in the database
show ip rsvp sbm [*detail*] [*interface-name*]	Displays information about a SBM configured for a specific RSVP-enabled interface or for all RSVP-enabled interfaces on the router
show ip rsvp sender [*interface-type interface-number*]	Displays Resource Reservation Protocol (RSVP) PATH-related sender information currently in the database

Verifying synchronization is the first step in troubleshooting RSVP. Without synchronization, RSVP has no means to prevent an H.323 gateway from moving into the alerting state and consuming resources that may not be available. To verify synchronization, use the **show call rsvp-sync conf** command.

Example 8-21 shows the output from the **show call rsvp-sync conf** command.

Example 8-21 *The* **show call rsvp-sync** *Command*

```
Router# show call rsvp-sync conf
VoIP QoS: RSVP/Voice Signaling Synchronization config:
Overture Synchronization is ON
Reservation Timer is set to 10 seconds
```

This output tells you that synchronization is enabled and the reservation timer will wait a maximum of 10 seconds for a reservation response.

To display statistics for calls that have attempted RSVP reservation, use the **show call rsvp-sync stats** command.

Example 8-22 shows a sample output from the **show call rsvp-sync stats** command.

Example 8-22 *The* **show call rsvp-sync stats** *Command Output*

```
Router# show call rsvp-sync stats
VoIP QoS:Statistics Information:
Number of calls for which QoS was initiated : 18478
Number of calls for which QoS was torn down : 18478
Number of calls for which Reservation Success was notified : 0
```

Example 8-22 *The* **show call rsvp-sync stats** *Command Output (Continued)*

```
Total Number of PATH Errors encountered : 0
Total Number of RESV Errors encountered : 0
Total Number of Reservation Timeouts encountered : 0
```

The **show call rsvp-sync stats** command offers a quick glance at reservation successes and failures on the router. A high number of errors or timeouts may indicate a network or configuration issue.

You can use the **show ip rsvp installed** command to display information about local interfaces configured for RSVP and the current reservations on the interfaces. In Example 8-23, the **show ip rsvp installed** command shows that Ethernet interface 2/1 has four reservations but serial interface 3/0 has none.

Example 8-23 *The* **show ip rsvp installed** *Command*

```
Router# show ip rsvp installed
RSVP:Ethernet2/1
BPS To From Protoc DPort Sport Weight Conversation
44K 145.20.0.202 145.10.0.201 UDP 1000 1000 0 264
44K 145.20.0.202 145.10.0.201 UDP 1001 1001 13 266
98K 145.20.0.202 145.10.0.201 UDP 1002 1002 6 265
1K 145.20.0.202 145.10.0.201 UDP 10 10 0 264
RSVP:Serial3/0 has no installed reservations
Router#
```

The current amount of reserved bandwidth for this interface is 187 kbps, as shown in the following:

44 kbps + 44 kbps + 98 kbps + 1 kbps = 187 kbps

With this information, you have the ability to compare the actual bandwidth reserved versus the maximum bandwidth configured on Ethernet 2/1 using the **ip rsvp bandwidth** command. Reserved bandwidth approaching the maximum can result in RSVP rejections.

You can obtain more detail about current reservations with the **show ip rsvp installed detail** command. Example 8-24 shows an output of this command.

Example 8-24 *Sample* **show ip rsvp installed detail** *Command Output*

```
Router# show ip rsvp installed detail
RSVP:Ethernet2/1 has the following installed reservations
RSVP Reservation. Destination is 145.20.0.202, Source is 145.10.0.201,
Protocol is UDP, Destination port is 1000, Source port is 1000
Reserved bandwidth:44K bits/sec, Maximum burst:1K bytes, Peak rate:44K bits/sec
Resource provider for this flow:
WFQ on hw idb Se3/0: PRIORITY queue 264. Weight:0, BW 44 kbps
```

continues

Example 8-24 *Sample* **show ip rsvp installed detail** *Command Output (Continued)*

```
Conversation supports 1 reservations
Data given reserved service:316 packets (15800 bytes)
Data given best-effort service:0 packets (0 bytes)
Reserved traffic classified for 104 seconds
Long-term average bitrate (bits/sec):1212 reserved, 0M best-effort
RSVP Reservation. Destination is 145.20.0.202, Source is 145.10.0.201,
Protocol is UDP, Destination port is 1001, Source port is 1001
Reserved bandwidth:44K bits/sec, Maximum burst:3K bytes, Peak rate:44K bits/sec
Resource provider for this flow:
WFQ on hw idb Se3/0: RESERVED queue 266. Weight:13, BW 44 kbps
Conversation supports 1 reservations
Data given reserved service:9 packets (450 bytes)
Data given best-effort service:0 packets (0 bytes)
Reserved traffic classified for 107 seconds
Long-term average bitrate (bits/sec):33 reserved, 0M best-effort
RSVP Reservation. Destination is 145.20.0.202, Source is 145.10.0.201,
Protocol is UDP, Destination port is 1002, Source port is 1002
Router#
```

In this example, the reservation on Ethernet 2/1 has met the criteria of the **voice-like** profile and has been admitted into the priority queue. (A weight of 0 identifies the flows that have matched the **voice-like** profile.) The reservation on Serial 3/0 has not met the criteria of the **voice-like** profile and so has been admitted to a reserved WFQ.

RSVP CAC Summary

Remember the following factors regarding the use of RSVP as a CAC mechanism:

- In current Cisco IOS Software, H.323 synchronization is initiated by default.
- RSVP packets (path and resv) travel as best-effort traffic.
- WFQ must be enabled on an interface/PVC as a basis for LLQ.

RSVP is a true end-to-end CAC mechanism *only* if configured on every interface that a call traverses.

For the unique capability to serve as both an end-to-end CAC mechanisms, and to guarantee the QoS for the entire duration of the call, RSVP does incur some “costs” on the network, as follows:

- Signaling (messaging and processing).
- Per flow state (memory).
- Postdial delays.

- RSVP does not provide for call redirection after call setup if a link in the network should fail. Another mechanism, such as dial-peer preferences, must be configured to serve this function.
- RSVP is not yet supported on the Cisco IP Phones.

Table 8-25 evaluates the RSVP mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-25 *RSVP CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	VoIP/H.323 only
Toll bypass or IP telephony	Currently Toll bypass only
Platforms and releases	Cisco IOS gateways in Release 12.1(5)T and 12.2
PBX trunk types supported	All
End to end, local, or IP cloud	End to end between originating gateway and terminating gatekeeper (provided all intermediate nodes are RSVP configured) Could be used at WAN edge with DiffServ backbone
Per call, interface, or endpoint	Per call
Topology awareness	Yes
Guarantees QoS for duration of call	Yes
Postdial delay	Yes
Messaging network overhead	Path/resv and periodic keepalives

Foundation Summary

The “Foundation Summary” is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures are a convenient way to review the day before the exam.

Figure 8-35 shows the effect of a VoIP network without the use of CAC.

Figure 8-35 *VoIP Network Without CAC*

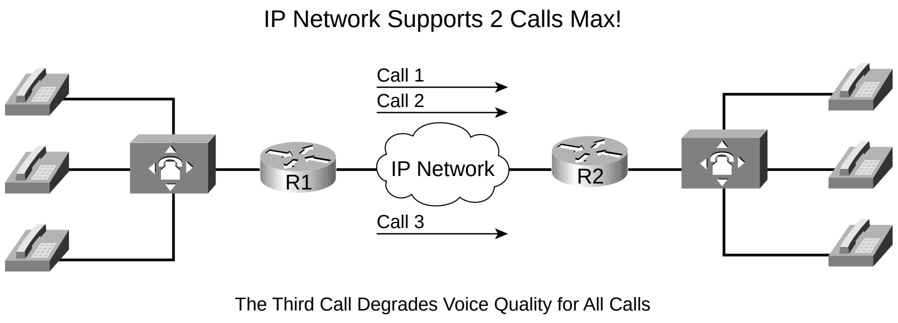

IP Network Supports 2 Calls Max!
Call 1
Call 2
IP Network
R1
R2
Call 3
The Third Call Degrades Voice Quality for All Calls

Figure 8-36 shows how CAC can be used in a legacy VoIP network to redirect a call to the PSTN in the even that sufficient resources are not available to carry the call on the data network.

Figure 8-36 *Legacy VoIP Network with CAC*

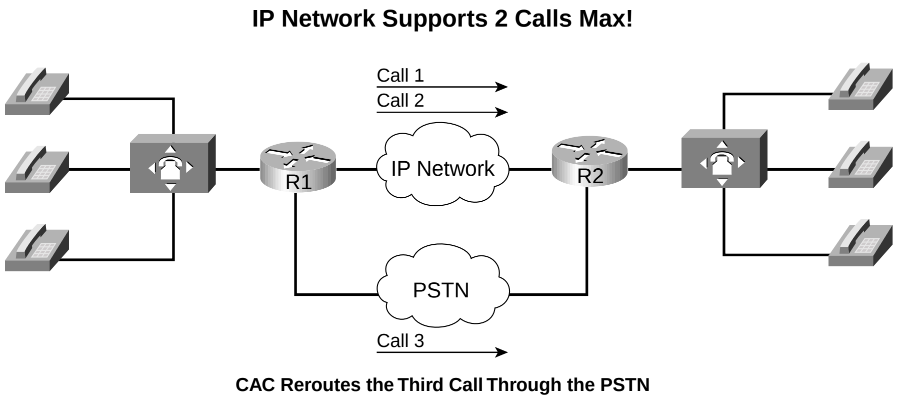

IP Network Supports 2 Calls Max!
Call 1
Call 2
IP Network
R1
R2
PSTN
Call 3
CAC Reroutes the Third Call Through the PSTN

Figure 8-37 shows how CAC can be used in an IP telephony network to redirect a call to the PSTN in the event that sufficient resources are not available to carry the call on the data network.

Figure 8-37 *IP Telephony Network with CAC*

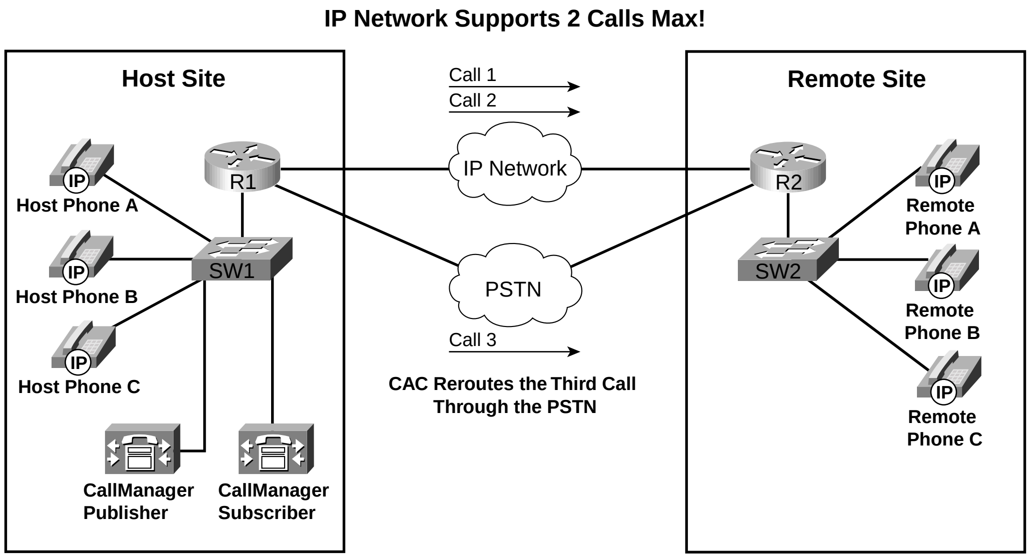


Table 8-26 illustrates a few of the possible G.711 and G.729 bandwidth requirements.

Table 8-26 *Bandwidth Requirements*

Codec	Payload per Packet	IP/UDP/RTP Header Size	L2 Header Type	L2 Header Size	Packets per Second	Bandwidth per Call
G.711	160 bytes	40 bytes	Ethernet	14 bytes	50 pps	85.6 kbps
G.711	240 bytes	40 bytes	Ethernet	14 bytes	33 pps	77.6 kbps
G.711	160 bytes	40 bytes	MLPPP/FR	6 bytes	50 pps	82.4 kbps
G.711	160 bytes	2 bytes (cRTP)	MLPPP/FR	6 bytes	50 pps	67.2 kbps
G.729	20 bytes	40 bytes	Ethernet	14 bytes	50 pps	29.6 kbps
G.729	20 bytes	40 bytes	MLPPP/FR	6 bytes	50 pps	26.4 kbps
G.729	30 bytes	40 bytes	MLPPP/FR	6 bytes	33 pps	20 kbps
G.729	20 bytes	2 bytes (cRTP)	MLPPP/FR	6 bytes	50 pps	11.2 kbps

* For DQOS test takers: These numbers are extracted from the DQOS course, so you can study those numbers. Note, however, that the numbers in the table and following examples do not include the L2 trailer overhead.

Figure 8-38 illustrates the packet structure of the layer 2 and IP/UDP/RTP headers and the payload for a voice packet.

Figure 8-38 *Voice Packet Structure*

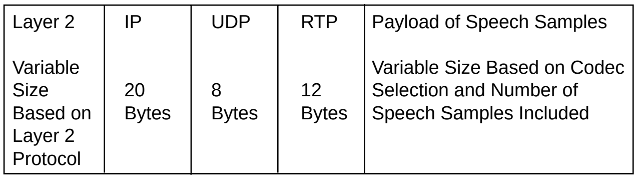

Layer 2
Variable Size Based on Layer 2 Protocol
IP
20 Bytes
UDP
8 Bytes
RTP
12 Bytes
Payload of Speech Samples
Variable Size Based on Codec Selection and Number of Speech Samples Included

Table 8-27 describes the criteria that is used to evaluate the different CAC tools.

Table 8-27 *CAC Feature Evaluation Criteria*

Evaluation Criteria	**Description**
Voice over *X* (Vo*X*) supported	The voice technologies to which the CAC method applies, such as VoIP and VoFR. Some methods apply to a single technology, whereas other methods apply to multiple technologies.
Toll bypass or IP telephony	Whether the method is suitable for use only between voice gateways connected to the PSTN or a PBX (toll bypass), or will the method function with IP Phone endpoints (IP telephony).
Platforms and releases	The Cisco IOS platforms this feature is available on, and the software release in which it was introduced.
PBX trunk types supported	Some CAC features have a dependency on the PSTN or PBX trunk type used in the connection, or act differently with CCS trunks versus CAS trunks.
End-to-end, local, or IP cloud	The scope of visibility of the CAC feature. Some mechanisms work locally on the originating gateway only, others consider the cloud between the source and destination nodes, some consider the destination POTS interface, and some work end to end.
Per call, interface, or endpoint	Different mechanisms involve different elements of the network. Several CAC methods work per call, but some work per interface and some work per endpoint or IP destination.
Topology awareness	Whether the CAC mechanism takes into account the topology of the network, and therefore provides protection for the links and nodes in the topology.
Guarantees QoS for duration of call	Whether the mechanism make a one-time decision before allowing the call, or whether it also protects the QoS of the call for the duration of the call by reserving the required resources.

Table 8-27 *CAC Feature Evaluation Criteria (Continued)*

Evaluation Criteria	Description
Postdial delay	Whether the mechanism imposes an additional postdial delay because it requires extra messaging or processing during call setup.
Messaging network overhead	Whether the method uses additional messaging that must be provisioned in the network to gather the information necessary for the CAC decision.

Figure 8-39 illustrates a network using physical DS0 limitation to provide CAC.

Figure 8-39 *VoIP Physical DS0 Limitation*

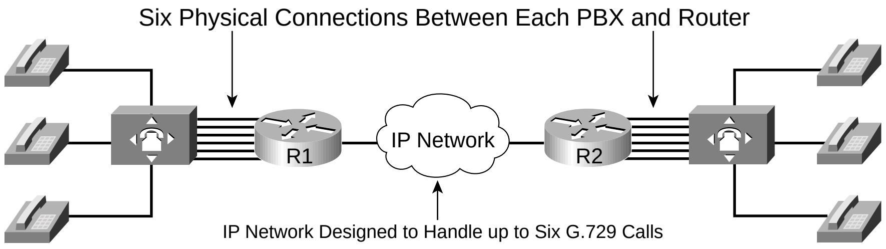

Six Physical Connections Between Each PBX and Router
R1
IP Network
R2
IP Network Designed to Handle up to Six G.729 Calls

Table 8-28 evaluates the physical DS0 limitation mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-28 *DS0 Limitation CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	Independent of the Vo*X* technology used
Toll bypass or IP Telephony	Toll bypass only
Platforms and releases	All voice gateways and all Cisco IOS releases
PBX trunk types supported	All
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per DS0/trunk (per call)
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Figure 8-40 shows a typical VoIP network that can use the **max-conn** command to limit the number of calls between locations.

Figure 8-40 *Max-Connections Multi-Site*

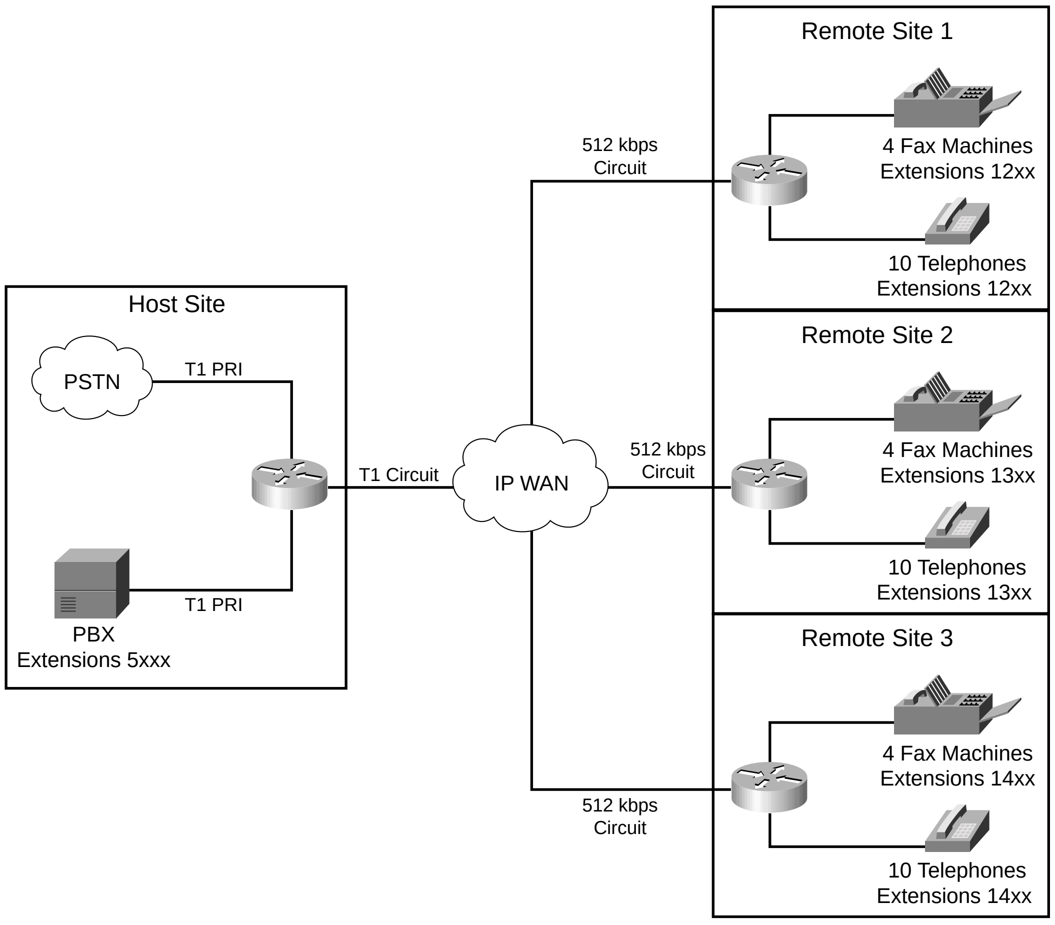


Table 8-29 evaluates the Max-Connections mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-29 *Max-Connections CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	All Vo*X* that use dial peers
Toll bypass or IP telephony	Toll bypass only
Platforms and releases	All voice gateways and all Cisco IOS releases
PBX trunk types supported	All
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per dial peer
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Figure 8-41 shows a typical VoFR network that can use the **frame-relay voice-bandwidth** command to limit the number of calls between locations.

Figure 8-41 *Voice over Frame Relay (VoFR)*

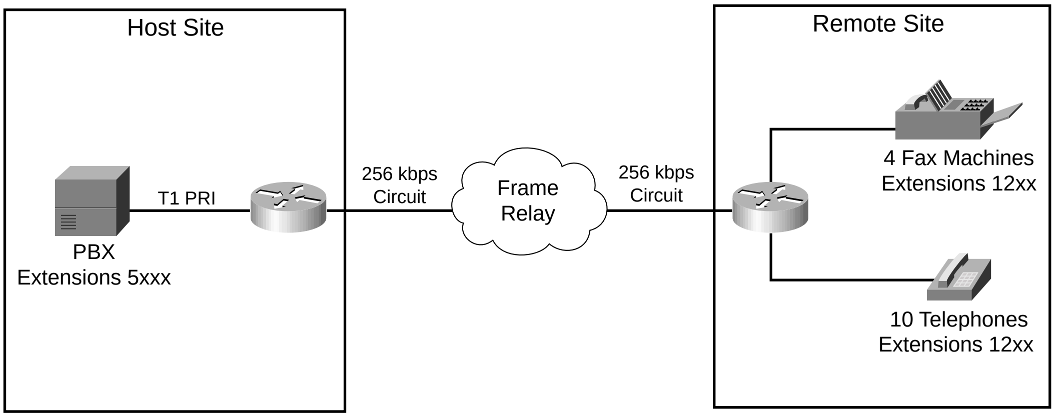


Table 8-30 evaluates the VoFR Voice-Bandwidth mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-30 *VoFR Voice-Bandwidth CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	VoFR
Toll bypass or IP telephony	Toll bypass only
Platforms and releases	Cisco 2600s, 3600s, 3810, and 7200 router; Cisco IOS Release 12.0(4)T
PBX trunk types supported	All
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per call, per PVC
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Figure 8-42 shows a VoIP network using the **connection trunk** command to emulate a circuit switched network.

Figure 8-42 *Trunk Conditioning*

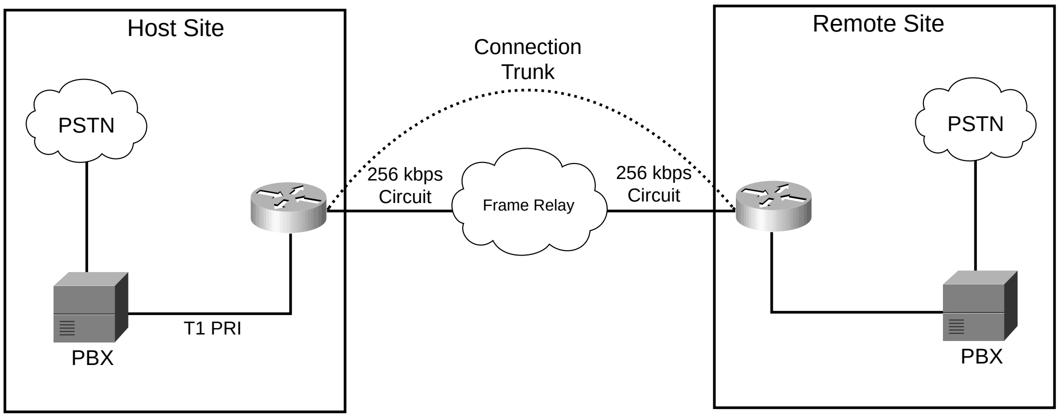


Table 8-31 evaluates the trunk conditioning mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-31 *Trunk Conditioning CAC Evaluation Criteria*

Evaluation Criteria	**Value**
Vo*X* supported	VoIP/H.323, VoFR, VoATM (connection trunk configurations only)
Toll bypass or IP telephony	Toll bypass applications only
Platforms and Releases	Cisco 2600 and 3600 series routers, and Cisco MC3810 multiaccess concentrators; Cisco IOS Release 12.1(3)T
PBX trunk types supported	Analog and CAS
End to end, local, or IP cloud	Local
Per call, interface, or endpoint	Per telephony interface
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None; uses preexisting connection trunk keepalives

Figure 8-43 shows a VoIP network using Local Voice Busyout to provide CAC.

Figure 8-43 *Local Voice Busyout*

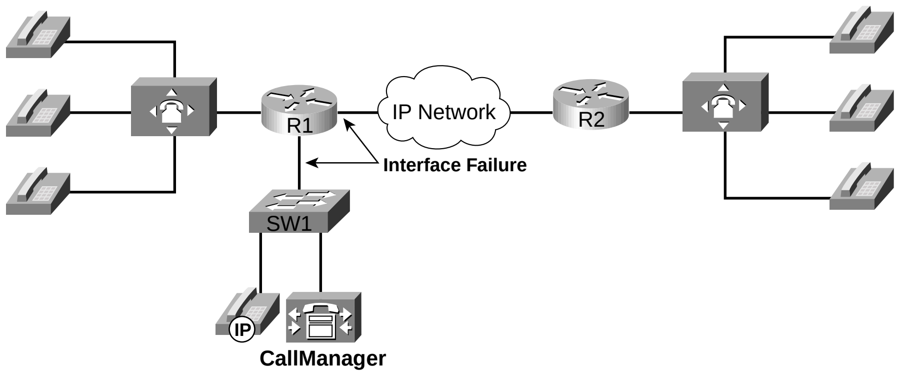


Table 8-32 evaluates the Local Voice Busyout mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-32 *Local Voice Busyout CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	All
Toll bypass or IP telephony	Trunking (calls originating from PBX and terminating to IP telephony destinations)
Platforms and releases	Cisco 2600 and 3600 series routers, MC3810 multiaccess concentrators; Cisco IOS Release 12.1(2)T
PBX trunk types supported	Analog and CAS
End-to-end, local, or IP cloud	Local
Per call, interface, or endpoint	Per WAN, LAN, and telephony interface
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Figure 8-44 shows a VoIP network using Advanced Voice Busyout to provide CAC.

Figure 8-44 *Advanced Voice Busyout*

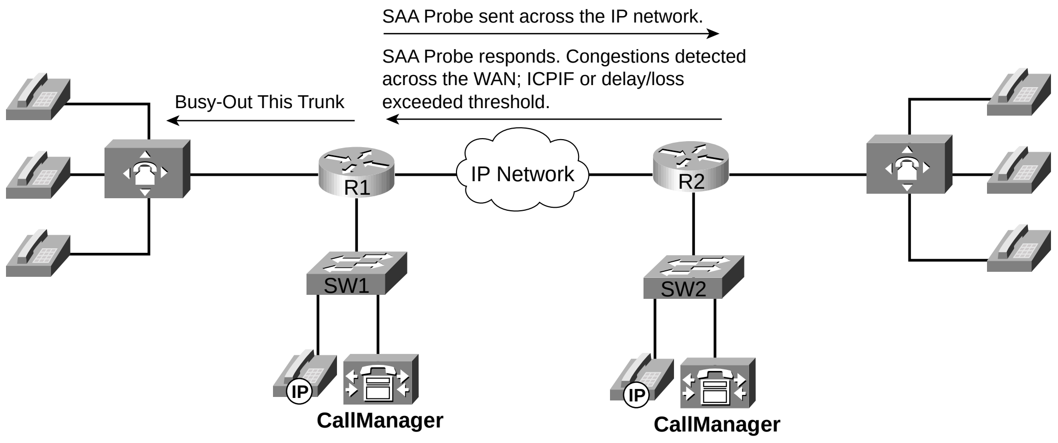

SAA Probe sent across the IP network.
SAA Probe responds. Congestions detected across the WAN; ICPIF or delay/loss exceeded threshold.
Busy-Out This Trunk
R1
IP Network
R2
SW1
SW2
IP
IP
CallManager
CallManager

Table 8-33 evaluates the AVBO mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-33 *Advanced Voice Busyout CAC Evaluation Criteria*

Evaluation Criteria	**Value**
Vo*X* supported	VoIP only
Toll bypass or IP telephony	Toll bypass (calls originating from PBX and terminating to IP telephony destinations)
Platforms and releases	2600s, 3600sand MC3810 with Release 12.1(3)T All router platforms with Release 12.2 Mainline
PBX trunk types supported	Analog and CAS
End to end, local, or IP cloud	IP cloud
Per call, interface, or endpoint	Per IP destination
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	Periodic SAA probes

Figure 8-45 shows a VoIP network using PSTN fallback to provide CAC.

Figure 8-45 *PSTN Fallback*

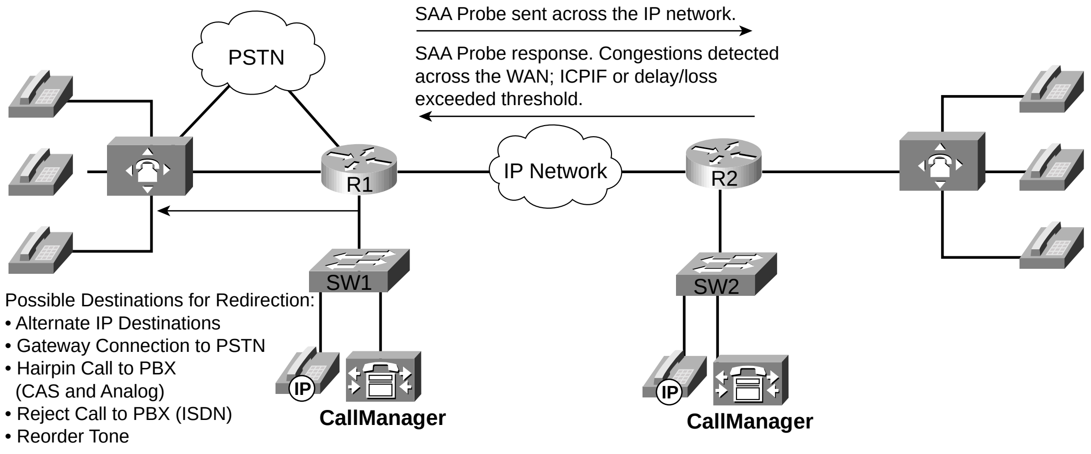

SAA Probe sent across the IP network.
SAA Probe response. Congestions detected across the WAN; ICPIF or delay/loss exceeded threshold.
PSTN
IP Network
R1
R2
SW1
SW2
IP
IP
CallManager
CallManager
Possible Destinations for Redirection:
• Alternate IP Destinations
• Gateway Connection to PSTN
• Hairpin Call to PBX (CAS and Analog)
• Reject Call to PBX (ISDN)
• Reorder Tone

Table 8-34 lists the options and default values of the **call fallback** command.

Table 8-34 *Call Fallback Command*

Call Fallback Command Keyword	Description	Default Value
cache-size *x*	Specifies the call fallback cache size for network traffic probe entries.	128
cache-timeout *x*	Specifies the time after which the cache entries of network conditions are purged.	600s
instantaneous-value-weight *x*	Specifies that the call fallback subsystem take an average from the last two cache entries for call requests.	66
jitter-probe num-packets *x*	Specifies the number of packets in a jitter probe used to determine network conditions.	15
jitter-probe precedence *x*	Specifies the priority of the jitter-probe transmission.	2
jitter-probe priority-queue	Assigns a priority queue for jitter-probe transmissions.	Off
key-chain	Specifies MD5 authentication for sending and receiving SAA probes.	None
map subnet	Specifies that the call fallback router keep a cache table by subnet addresses of distances for several destination peers sitting behind the router.	None
probe-timeout *x*	Sets the timeout for an SAA probe for call fallback purposes.	30s
threshold delay *x* **loss** *y*	Specifies that the call fallback threshold use only packet delay and loss values.	None
icpif *x*	Specifies that the call fallback use the ICPIF threshold.	10

Figure 8-46 illustrates the call setup process for PSTN fallback.

Figure 8-46 *PSTN Fallback Call Setup*

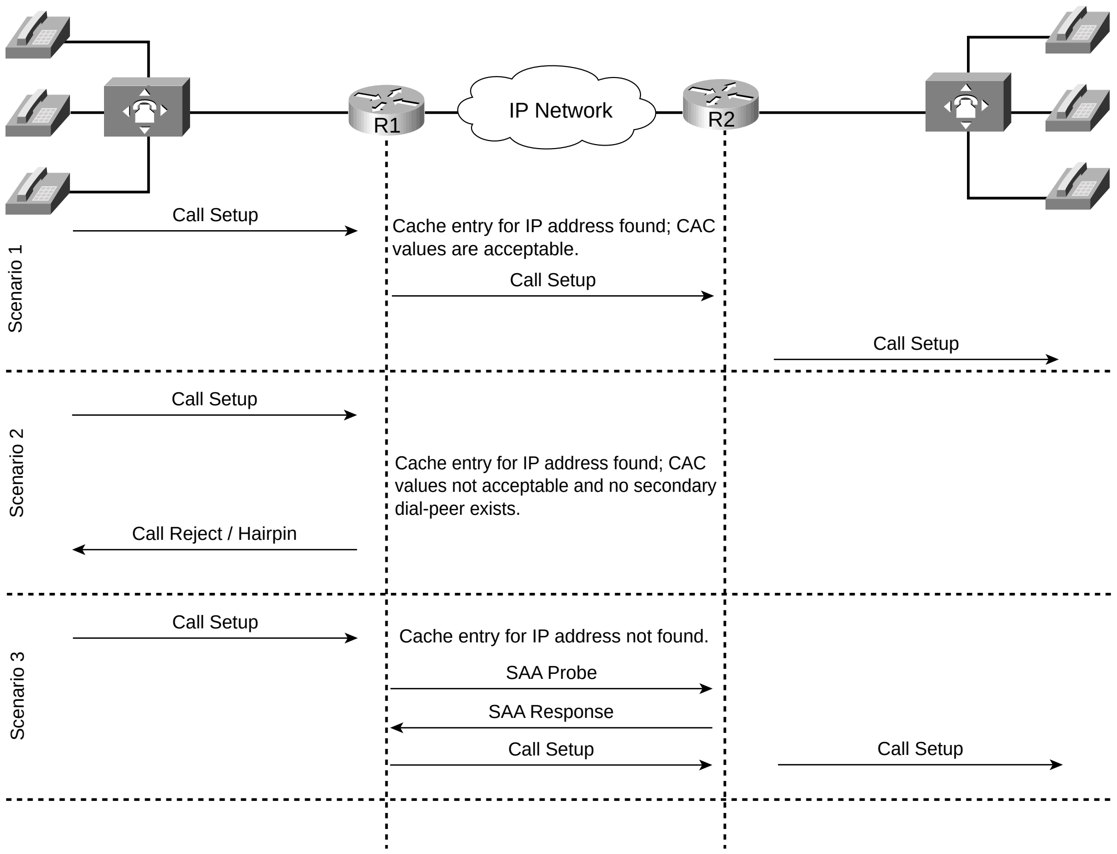

R1
IP Network
R2
Scenario 1
Call Setup
Cache entry for IP address found; CAC values are acceptable.
Call Setup
Call Setup
Scenario 2
Call Setup
Cache entry for IP address found; CAC values not acceptable and no secondary dial-peer exists.
Call Reject / Hairpin
Scenario 3
Call Setup
Cache entry for IP address not found.
SAA Probe
SAA Response
Call Setup
Call Setup

Table 8-35 evaluates the PSTN fallback mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-35 *PSTN Fallback CAC Evaluation Criteria*

Evaluation Criteria	**Value**
Vo*X* supported	VoIP only
Toll bypass or IP telephony	Toll bypass; however, calls originating from a PBX and terminating to IP telephony destinations can be protected
Platforms and releases	Cisco 2600/3600, MC3810: Release 12.1.(3)T AS5300: Release 12.2.(2)T 7200/7500 support SAA responder
PBX trunk types supported	All PBX/PSTN trunk signaling types (analog, digital CAS and CCS) For analog and digital CAS—alternate IP destination, hairpin For digital CCS, reject the call to the PBX or PSTN for rerouting
End to end, local, or IP cloud	IP cloud
Per call, interface, or endpoint	Per active/cached IP destination
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	Only for first call that initiates probe
Messaging network overhead	Periodic SAA probes

Figure 8-47 illustrates how a CAC decision is made with resource availability indication (RAI).

Figure 8-47 *RAI Configuration*

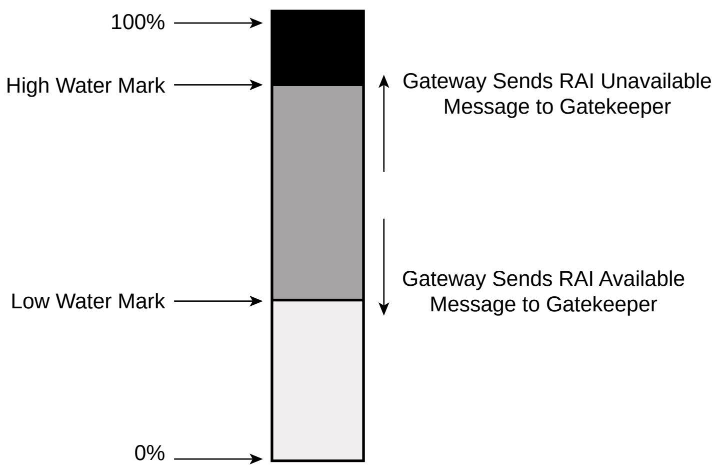

100%
High Water Mark
Low Water Mark
0%
Gateway Sends RAI Unavailable Message to Gatekeeper
Gateway Sends RAI Available Message to Gatekeeper

Table 8-36 evaluates the RAI mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-36 *RAI CAC Evaluation Criteria*

Evaluation Criteria	**Value**
Vo*X* supported	VoIP only
Toll bypass or IP telephony	Toll bypass Potentially IP telephony, but CM does not yet support RAI
Platforms and releases	Cisco AS5300 access server: Cisco IOS Release 12.0(5)T Cisco 2600 and 3600 series routers T1/E1: Cisco IOS Release 12.1(3)T
PBX trunk types supported	All
End to end, local, or IP cloud	Local at the terminating gateway (DSP and DS0 resources; algorithm platform dependent)
Per call, interface, or endpoint	Per gateway
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	Occasional RAI toggle between gateway and gatekeeper

Figure 8-48 illustrates a typical CallManager centralized call-processing model using locations to provide CAC.

Figure 8-48 *CallManager Centralized Call-Processing Model with Regions and Locations Defined*

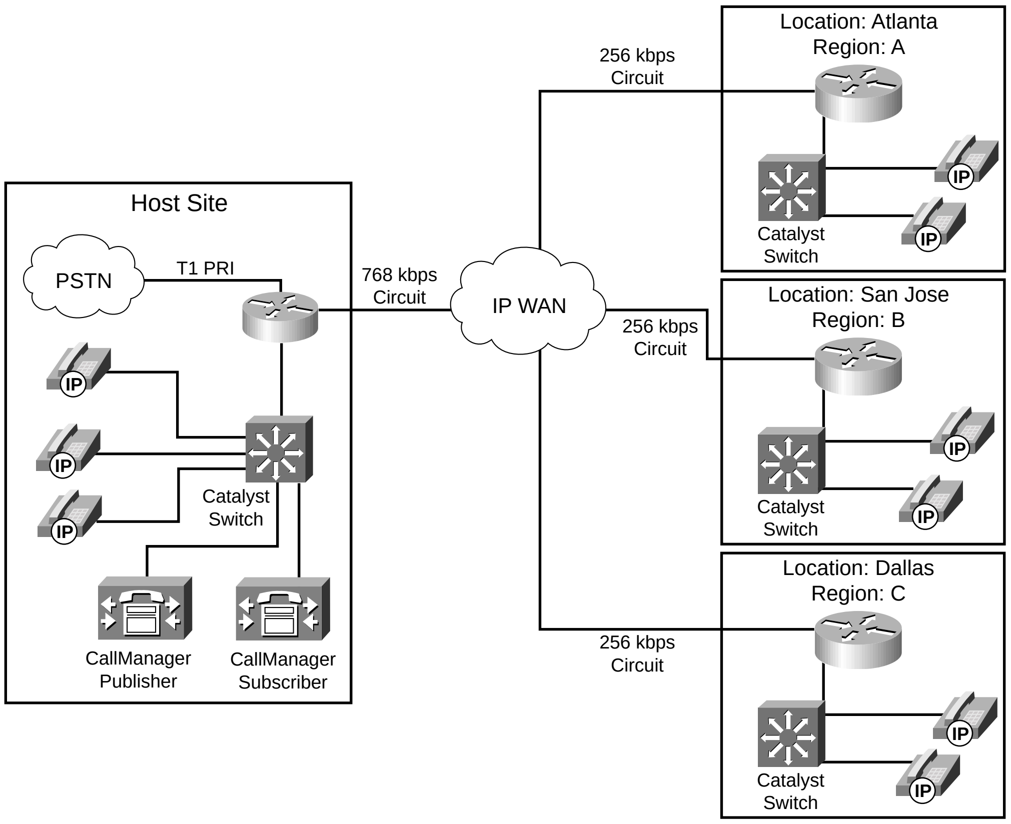


Table 8-37 shows the amount of bandwidth that will be subtracted, per call, from the total allotted bandwidth for a configured region.

Table 8-37 *Location-Based CAC Resource Calculations*

Codec	Bandwidth Reserved
G.711	80 kbps
G.729	24 kbps
G.723	24 kbps
GSM	29 kbps
Wideband	272 kbps

Table 8-38 evaluates location-based CAC against the CAC evaluation criteria described earlier in this chapter.

Table 8-38 *Location-Based CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	VoIP only
Toll bypass or IP telephony	IP Telephony only
Platforms and releases	CallManager 3.0 (AAR was added in CallManager release 3.3.)
PBX trunk types supported	None
End to end, local, or IP cloud	End-to-end between originating and terminating location, although locations have no knowledge of the network topology in between
Per call, interface, or endpoint	Per call
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	None

Figure 8-49 shows a single-zone gatekeeper-controlled VoIP network with two gateways that illustrates gatekeeper CAC in its simplest form.

Figure 8-49 *Simple Single-Zone Topology*

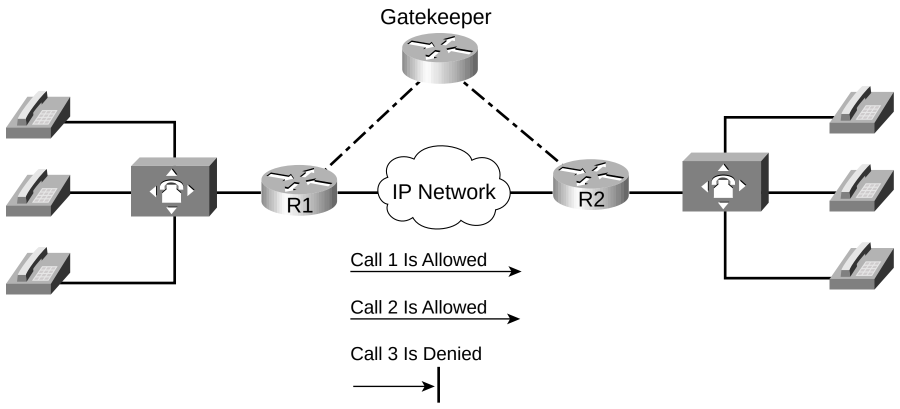


Figure 8-50 shows a more complex enterprise multizone multigatekeeper-controlled VoIP network.

Figure 8-50 *Complex Enterprise Multizone Topology*

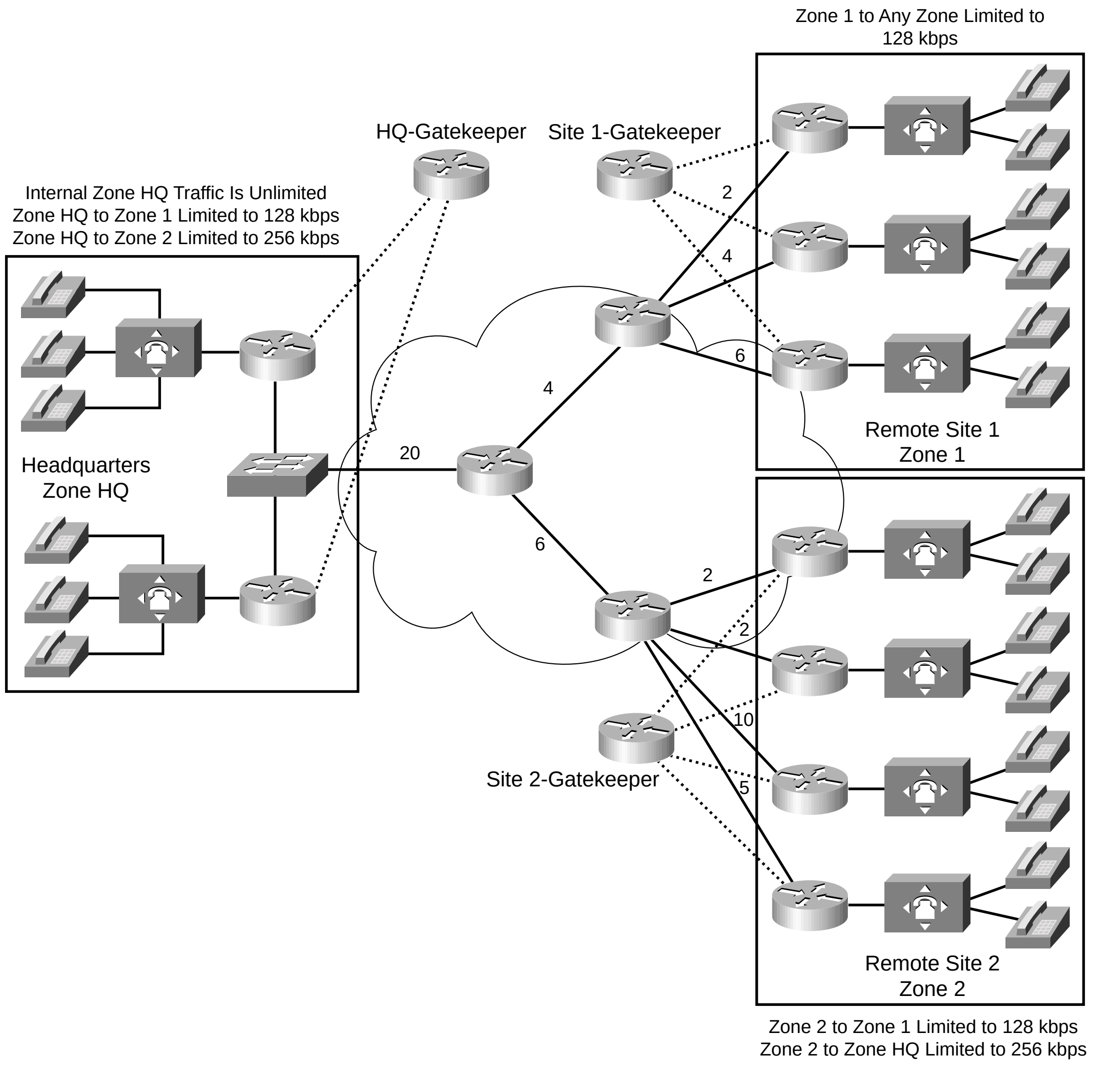
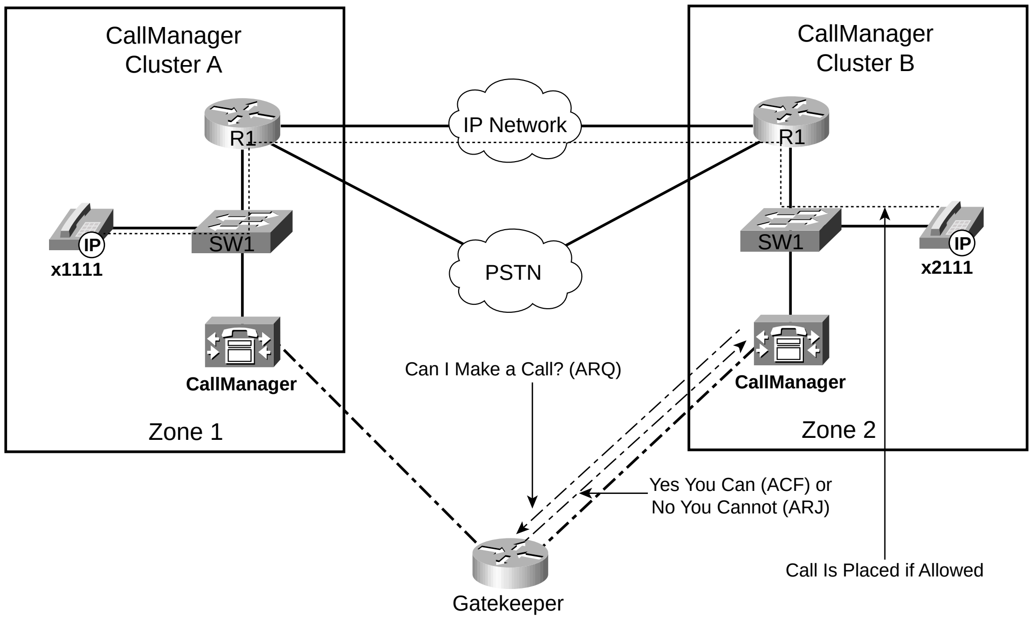

Zone 1 to Any Zone Limited to 128 kbps
HQ-Gatekeeper
Site 1-Gatekeeper
Internal Zone HQ Traffic Is Unlimited
Zone HQ to Zone 1 Limited to 128 kbps
Zone HQ to Zone 2 Limited to 256 kbps
2
4
6
4
20
Headquarters Zone HQ
Remote Site 1 Zone 1
6
2
2
10
5
Site 2-Gatekeeper
Remote Site 2 Zone 2
Zone 2 to Zone 1 Limited to 128 kbps
Zone 2 to Zone HQ Limited to 256 kbps

Figure 8-51 shows a pair of CallManager clusters using a gatekeeper to provide CAC between the clusters.

Figure 8-51 *Gatekeeper in a CallManager Topology*



Tables 8-39 and 8-40 list the gatekeeper commands and options used to configure gatekeeper zone bandwidth.

Table 8-39 *Gatekeeper Bandwidth Command*

Command	**Mode and Function**
bandwidth {**interzone** | **total** | **session**} {**default** | **zone** *zone-name*} *max-bandwidth*	Specifies the gatekeeper zone bandwidth restrictions
bandwidth remote *max-bandwidth*	Specifies the total bandwidth for H.323 traffic between this gatekeeper and any other gatekeeper

Table 8-40 *Gatekeeper Bandwidth Command Options*

Bandwidth Command Options	Function
interzone	Specifies the total amount of bandwidth for H.323 traffic from the zone to any other zone.
total	Specifies the total amount of bandwidth for H.323 traffic allowed in the zone.
session	Specifies the maximum bandwidth allowed for a session in the zone.
default	Specifies the default value for all zones.
zone	
zone-name	Names the particular zone.
max-bandwidth	Maximum bandwidth. For **interzone** and **total**, the range is from 1 to 10,000,000 kbps. For **session**, the range is from 1 to 5000 kbps.

Table 8-41 evaluates the gatekeeper zone bandwidth mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-41 *Gatekeeper Zone Bandwidth CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	VoIP/H.323 only
Toll bypass or IP telephony	Toll bypass and IP telephony (Some caveats exist if both the CallManager and Cisco IOS gateways are used in the same zone.)
Platforms and releases	Cisco IOS gateways since Release 11.3 (CM has recent changes in E.164 registration, and bandwidth requested per call.)
PBX trunk types supported	All
End to end, local, or IP cloud	End-to-end between originating gateway and terminating gateway, although not aware of the network topology in between
Per call, interface, or endpoint	Per call
Topology awareness	None
Guarantees QoS for duration of call	None
Postdial delay	None
Messaging network overhead	Part of the gatekeeper RAS messaging

Figure 8-52 shows the flow of RSVP path and resv messages through the network.

Figure 8-52 *RSVP Path and Resv Messages*

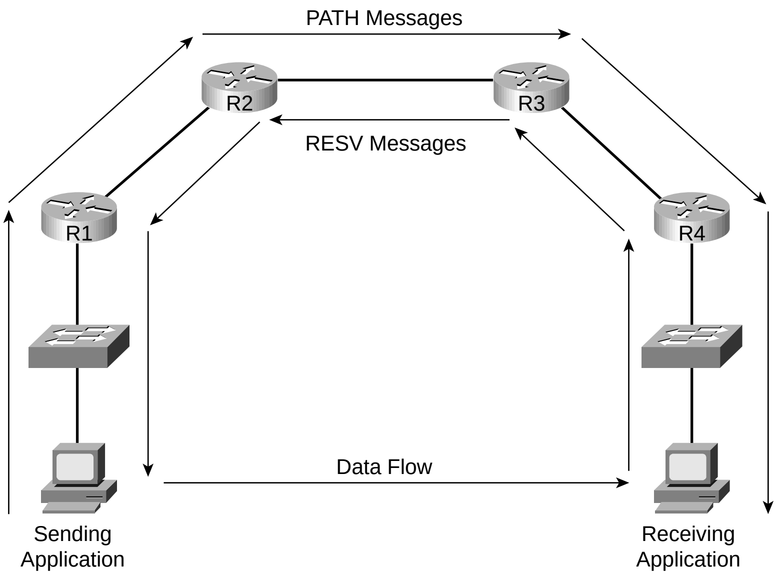


Figure 8-53 shows a call flow of the H.323 call setup messages and the RSVP reservation messages.

Figure 8-53 *RSVP Call Setup for an H.323 Voice Call*

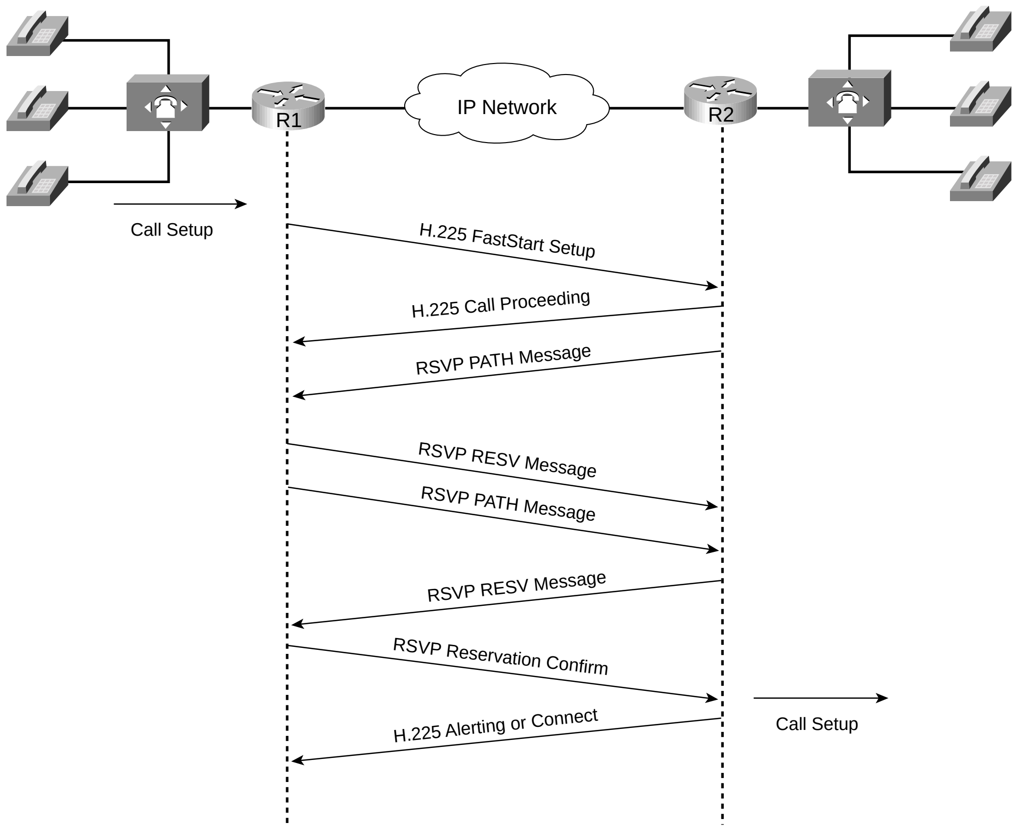


Table 8-42 describes the available options for the **acc-qos** and **req-qos** commands.

Table 8-42 **acc-qos** *and* **req-qos** *Command Options*

acc-qos and req-qos Command Options	Function
best-effort	Indicates that RSVP makes no bandwidth reservation.
controlled-load	Indicates that RSVP guarantees a single level of preferential service, presumed to correlate to a delay boundary. The controlled-load service uses admission (or capacity) control to ensure that preferential service is received even when the bandwidth is overloaded.
guaranteed-delay	Indicates that RSVP reserves bandwidth and guarantees a minimum bit rate and preferential queuing if the bandwidth reserved is not exceeded.

This table was derived from the following:
www.cisco.com/en/US/partner/products/sw/iosswrel/ps1834/products_feature_guide09186a008008045c.html.

Table 8-18 summarizes the results of nine call setup scenarios based on the QoS levels that can be configured in the VoIP dial peers at the originating and terminating gateways.

Figure 8-54 illustrates how RSVP uses the priority queue in LLQ for packets matching the **voice-like** profile.

Figure 8-54 *RSVP Packet-Classification Criteria*

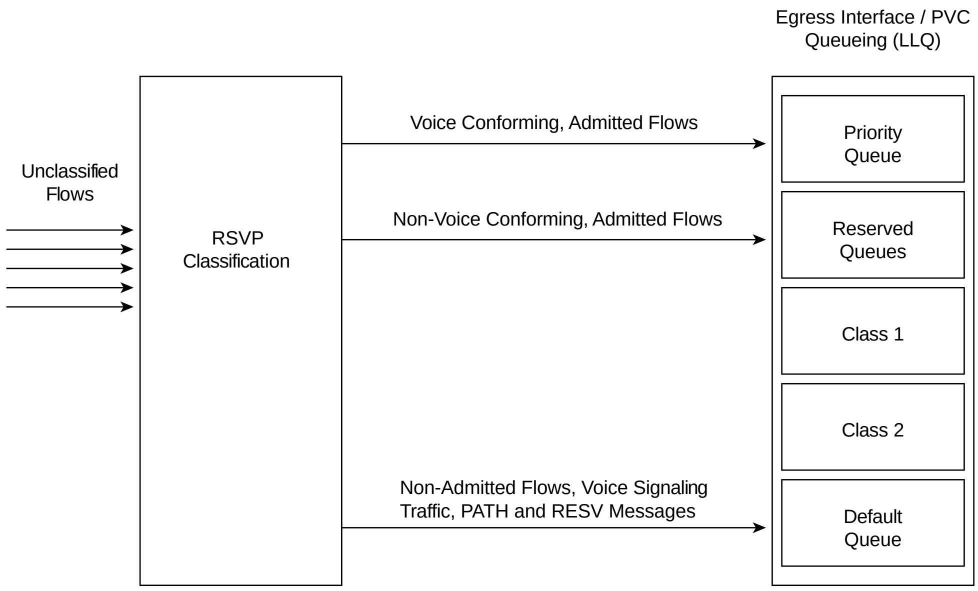

Egress Interface / PVC Queueing (LLQ)
Unclassified Flows
RSVP Classification
Voice Conforming, Admitted Flows
Non-Voice Conforming, Admitted Flows
Non-Admitted Flows, Voice Signaling Traffic, PATH and RESV Messages
Priority Queue
Reserved Queues
Class 1
Class 2
Default Queue

Table 8-43 summarizes the bandwidth RSVP allocates for calls using different Cisco IOS gateway codecs.

Table 8-43 *RSVP Bandwidth Reservations for Voice Codecs*

Codec	**Bandwidth Reserved per Call in LLQ**
G.711 (A-law and µ-law)	80 kbps
G.723.1 and G.723.1A (5.3 kbps)	22 kbps
G.723.1 and G.723.1A (6.3 kbps)	23 kbps
G.726 (16 kbps)	32 kbps
G.726 (24 kbps)	40 kbps
G.726 (32 kbps)	48 kbps
G.728	32 kbps
G.729 (all versions)	24 kbps

Table 8-44 lists the commands used to define and enable RSVP.

Table 8-44 *RSVP Profile,* **req-qos** *and* **acc-qos** *Commands*

Command	Mode and Function
ip rsvp pq-profile	Specifies the criteria for determining which flows go into the priority queue
req-qos {**best-effort** | **controlled-load** | **guaranteed-delay**}	Specifies the desired quality of services requested to be used in reaching a specified VoIP dial peer
acc-qos {**best-effort** | **controlled-load** | **guaranteed-delay**}	Specifies the acceptable quality of service for any inbound and outbound call on a VoIP dial peer

Figure 8-55 shows a managed segment in a Layer 2 domain that interconnects a group of routers.

Figure 8-55 *DSBM Managed Subnet*

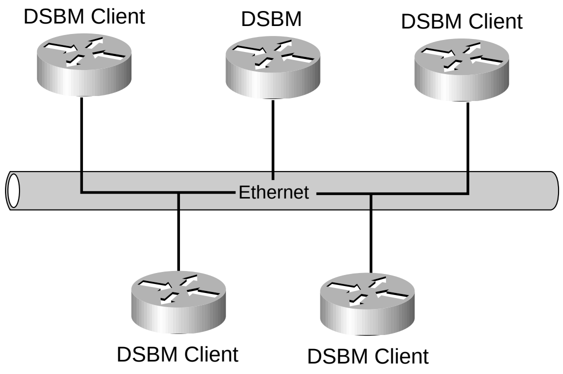


Table 8-45 lists the commands used to enable and define the DSBM in Example 8-18.

Table 8-45 *SBM Commands*

Command	Mode and Function
ip rsvp bandwidth	Enables RSVP on an interface
ip rsvp dsbm candidate [*priority*]	Configures the interface to participate as a contender in the DSBM dynamic election process, whose winner is based on the highest priority
ip rsvp dsbm non-resv-send-limit rate *kbps*	Configures the average rate, in kbps, for the DSBM candidate
ip rsvp dsbm non-resv-send-limit burst *kilobytes*	Configures the maximum burst size, in KB, for the DSBM candidate
ip rsvp dsbm non-resv-send-limit peak *kbps*	Configures the peak rate, in kbps, for the DSBM candidate

Table 8-46 lists other RSVP commands that can be useful in monitoring and troubleshooting RSVP.

Table 8-46 *RSVP Monitoring and Troubleshooting Commands*

Command	Mode and Function
show ip rsvp neighbor [*interface-type interface-number*]	Displays current RSVP neighbors
show ip rsvp request [*interface-type interface-number*]	Displays RSVP-related request information being requested upstream
show ip rsvp reservation [*interface-type interface-number*]	Displays RSVP-related receiver information currently in the database
show ip rsvp sbm [*detail*] [*interface-name*]	Displays information about a SBM configured for a specific RSVP-enabled interface or for all RSVP-enabled interfaces on the router
show ip rsvp sender [*interface-type interface-number*]	Displays Resource Reservation Protocol (RSVP) PATH-related sender information currently in the database

Table 8-47 evaluates the RSVP mechanism against the CAC evaluation criteria described earlier in this chapter.

Table 8-47 *RSVP CAC Evaluation Criteria*

Evaluation Criteria	Value
Vo*X* supported	VoIP/H.323 only
Toll bypass or IP telephony	Currently trunking only
Platforms and releases	Cisco IOS gateways in Release 12.1(5)T and 12.2
PBX trunk types supported	All
End to end, local, or IP cloud	End to end between originating gateway and terminating gatekeeper (provided all intermediate nodes are RSVP configured) Could be used at WAN edge with DiffServ backbone
Per call, interface, or endpoint	Per call
Topology awareness	Yes
Guarantees QoS for duration of call	Yes
Postdial delay	Yes
Messaging network overhead	Path/resv and periodic keepalives

There is little overlap between local CAC mechanisms and those that look ahead to the rest of the network to determine nonlocal conditions. It is easy to understand why the distinct local and nonlocal mechanisms are useful. However, there is considerable overlap between the measurement techniques and the resource reservation techniques of the two nonlocal, lookahead CAC mechanisms. For this reason, there is debate over which is the better method.

Table 8-48 compares the strengths and weaknesses of the measurement-based and resource-based CAC mechanisms. With this information, you can determine the best method for your individual network.

Table 8-48 *Comparison of Measurement-Based and Resource Reservation-Based CAC Features*

Criteria	Measurement-Based Techniques	Resource Reservation-Based Techniques
Network topology	Topology independent. The probe travels to a destination IP address without knowledge of nodes, hops, and bandwidth availability on individual links.	Topology aware. The bandwidth availability on every node and every link is taken into account.
Backbone transparency	Transparent. Probes are IP packets and can be sent over any network, including SP backbones and the Internet.	To be the truly end-to-end method that reservation techniques are intended to be, the feature must be configured on every interface along the path, which means the customer owns the WAN backbone, and all nodes run code that implement the feature. Owning the entire backbone is impractical in some cases, so hybrid topologies may be contemplated—with some compromise to the end-to-end nature of the method.
Postdial delay	An increase in postdial delay exists for the first call only. Information on the destination is cached after the first call, and a periodic probe is sent to the IP destination. Subsequent calls are allowed or denied based on the latest cached information.	An increase in postdial delay exists for every call, because the RSVP reservation must be established before the call setup can be completed.
Industry parity	Several vendors have “ping”-like CAC capabilities. For a customer familiar with this operation, measurement-based techniques are a good fit.	None.
CAC accuracy	The periodic sampling rate of probes can potentially admit calls when bandwidth is insufficient. Measurement-based techniques perform well in networks where traffic fluctuations are gradual.	When implemented on all nodes in the path, RSVP guarantees bandwidth for the call along the entire path for the entire duration of the call. This is the only technique that achieves this level of accuracy.

continues

Table 8-48 *Comparison of Measurement-Based and Resource Reservation-Based CAC Features (Continued)*

Criteria	Measurement-Based Techniques	Resource Reservation-Based Techniques
Protecting voice QoS after admission	The CAC decision is based on probe traffic statistics before the call is admitted. After admission, the call quality is determined by the effectiveness of other QoS mechanisms in the network.	A reservation is established per call before the call is admitted. The quality of the call is therefore unaffected by changes in network traffic conditions.
Network traffic overhead	Periodic probe traffic overhead to a cached number of IP destinations. Both the interval and the cache size can be controlled by the configuration.	RSVP messaging traffic overhead for every call.
Scalability	Sending probes to thousands of individual IP destinations may be impractical in a large network. Probes can be sent to the WAN edge devices, however, which proxy on behalf of many more destinations on a high-bandwidth campus network behind the edge. This provides considerable scalability, because the WAN is much more likely to be congested than the campus LAN.	Individual flow reservation is important on the small-bandwidth links around the edge of the network. However, individual reservations per call flow may not make sense on large-bandwidth links in the backbone such as an OC-12. Hybrid network topologies can solve this need, and additional upcoming RSVP tools in this space will provide further scalability.

Table 8-49 summarizes the 11 different voice CAC mechanisms that have been discussed in chapter. It also lists the first Cisco IOS release in which the feature became available.

Table 8-49 *Summary of CAC Features*

Type	CAC Feature	SW Release
Local		
	Physical DS0 limitation	SW independent
	Max-Connections on the dial peer	11.3
	VoFR Voice-Bandwidth	12.0.(4)T
	Trunk conditioning	12.1.(2)T
	Local Voice Busyout (LVBO)	12.1.(2)T
Measurement based		
	Advanced Voice Busyout (AVBO)	12.1.(3)T
	PSTN fallback	12.1.(3)T

Table 8-49 *Summary of CAC Features (Continued)*

Type	CAC Feature	SW Release
Resource based		
Resource calculation		

	Resource availability indication	12.0.(5)T (AS5300)
		12.1.(3)T (2600/3600)

	CallManager location-based CAC	CallManager 3.0 AAR was added in CallManager release 3.3

	Gatekeeper zone bandwidth	11.(3) (local zone)
		12.1.(5)T (interzone)

Resource reservation		
	Resource Reservation Protocol	12.1.(5)T

Table 8-50 summarizes the voice technologies supported by the CAC methods discussed in this chapter.

Table 8-50 *Summary of Voice Technologies supported*

Feature	VoIP H.323	VoIP SIP	VoIP MGCP	VoFR	VoATM	CM	H.323 Video
Physical DS0 limitation	Yes	Yes	Yes	Yes	Yes	No	No
Max-Connections	Yes	Yes	Yes	Yes	Yes	No	No
Voice-Bandwidth	No	No	No	Yes	No	No	No
Trunk conditioning	Yes	Yes	Yes	Yes	Yes	No	No
Local Voice Busyout	Yes	Yes	Yes	Yes	Yes	No	No
Advanced Voice Busyout	Yes	Yes	Yes	No	No	No	No
PSTN fallback	Yes	Yes	Yes	No	No	No	No
Resource availability indication	Yes	No	No	No	No	No	No
CallManager locations	Yes	No	Yes	Yes	Yes	Yes	No
Gatekeeper zone bandwidth	Yes	No	No	No	No	Yes	Yes
Resource Reservation Protocol	Yes	No	No	No	No	No	No

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD-ROM included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD-ROM nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD-ROM exam to include, or not include, the topics that are only on the CCIP QoS.

Call Admission Control Concepts

1 Why is CAC needed in an environment where LLQ has been properly implemented?

2 Name four possible measures that a CAC mechanism can take in the event that the resources are not available to proceed with the call.

3 How does a common channel signaling circuit, such as PRI, react to CAC

4 How does a channel-associated signaling circuit, such as E&M or T1 CAS, react to CAC?

5 What is the difference between payload bandwidth and the bandwidth per call required for a voice conversation?

Local-Based CAC

6 What is the definition of local-based CAC?

7 Name three of the five available local-based CAC mechanisms.

8 What Cisco IOS command is used to enable physical DS0 limitation?

9 What Cisco IOS command is used to enable CAC on a VoFR network?

10 What type of circuit, CAS or CSS, will Local Voice Busyout (LVB) be most beneficial for?

Measurement-Based CAC

11 What is the definition of measurement-based CAC?

12 What is the difference between LVBO and Advanced LVBO?

13 What is the difference between an SAA packet and a ping packet?

14 What measurements are gathered by SSA probes for PSTN fallback?

15 What Cisco IOS command is used to allow the destination node to participate in measurement based CAC

Resources-Based CAC

16 What is the definition of resource-based CAC?

17 Which resources can resource availability indication (RAI) currently monitor?

18 Which Cisco CallManager call-processing model utilizes location-based CAC?

19 For a gatekeeper to provide CAC, what must be configured for each link that requires protection?

20 What is IntServ and how does it work?

21 Name the messages used by RSVP to provide resource reservation and CAC.

22 What level of service must be in place to provide CAC?

23 Which RSVP profile should be configured to classify traffic for the priority queue within LLQ?

This chapter covers the following exam topics specific to the DQOS and QoS exams:

DQOS Exam Objectives

- Utilize QOS Device Manager to monitor performance, establish baselines, and configure QoS policies.
- Utilize QoS Policy Manager to configure advanced QoS policies, scale policy deployment, upload/verify/roll back policies, and deploy QoS policies by external time-based/event-based scripts.
- Configure Cisco Service Assurance Agent to measure key SLA metrics and monitor network performance between local and remote devices.
- Monitor and troubleshoot network performance with IPM and SMS.
- Design a converged multiservice network to provide proper QoS for voice, video, and data traffic.

CHAPTER 9

Management Tools and QoS Design

Networking course books, including Cisco's, tend to relegate management tools to the end of the book. Likewise, for implementation-oriented classes, specific coverage of design issues tends to happen only during chalk talks with the instructor, unless the course specifically covers design issues. With the Deploying QoS class, however, both topics get specific coverage in two separate chapters at the end of the course.

QoS management tools and QoS design both relate to the information that has already been presented in this book. The QoS management tools enable us to better monitor, configure, update configurations, troubleshoot, and notice trends in a network. Similarly, the design practices help us make better choices on how to implement the QoS tools. But, at the end of it all, both topics involve how best to make use of the large toolbox of QoS tools that were described in the earlier chapters of this book.

The exams describe their coverage of management and design with a short list of very specific exam objectives. The DQOS exam covers the features and functions of the QoS management tools, but does not cover how to point and click to use them. Although the management tools themselves have not been covered elsewhere in this book, the QoS tools they configure and manage have been covered extensively. So, this chapter just points out the features and functions of each tool.

The design treatment in this chapter focuses on voice and video, with many of those concepts already covered in this book in Chapter 1, "QoS Overview." The coverage relates mostly to when to use what QoS tool, along with a four-step process for how to go about the process of QoS design.

"Do I Know This Already?" Quiz

The 12-question quiz, derived from the major sections in the "Foundation Topics" section of this chapter, helps you determine how to spend your limited study time.

Table 9-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 9-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions	Score
1	QoS Management Tools	1 to 6	
2	QoS Design	7 to 12	
All questions		1 to 12	

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **10 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary," and the "Q&A" sections.
- **11 or 12 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, move to the next chapter.

QoS Management Tools Questions

1 What do the following acronyms stand for? IPM, QPM, SAA, QDM

2 What QoS management tool(s) enables you to define QoS policies, after which it creates the appropriate configurations, and deploys the configurations to identified network devices?

3 Which QoS management tool(s) includes as a primary feature the capability to graph real-time information about bit/byte/packet rates and packet loss?

4 What QoS management tool(s) generates packets, called probes, and measures various performance attributes, such as delay and jitter?

5 What QoS management tool(s) includes as a primary function the capability to trouble-shoot and isolate network-wide performance issues by viewing the individual components in a complete end-to-end path through the network?

6 What QoS management tool(s) is embedded into a router?

QoS Design Questions

7 List Step 1 in the QoS design process, and give a short summary of what the step includes.

8 List Step 3 in the QoS design process, and give a short summary of what the step includes.

9 What are the ITU G.114 and Cisco recommendations for maximum one-way delay for voice calls?

10 Summarize the Cisco recommendations for marked values for voice and video traffic.

11 Describe the Cisco recommendation for queuing voice and video traffic.

12 List Step 4 in the QoS design process, and give a short summary of what the step includes.

Foundation Topics

This chapter covers two main topics: management tools and design. In the “QoS Management Tools” section, focus on memorizing the basic features and functions of each tool. In fact, the exam does not cover navigation of the tools’ user interfaces.

The “QoS Design for the Cisco QoS Exams” section covers two key topics. First, a four-step design process covered in the DQOS course is discussed. The second topic covers specific design goals used when applying QoS to networks that utilize voice and video.

QoS Management Tools

The Cisco DQOS exam covers four main management tools. Two of the tools—QoS Device Manager (QDM) and QoS Policy Manager (QPM)—are separate tools. The other “tools”—Internet Performance Monitor (IPM) and Service Management Solutions (SMS)—began as separately sold packages, but are both new features of CiscoWorks2000. A supplementary tool, Service Assurance Agent (SAA), is also covered here, because IPM and SMS make liberal use of SAA.

QoS Device Manager

QoS Device Manager enables the user to sit at a browser window and manage the QoS implementation on a single router or 6000 series switch. Because you only need a browser (Netscape 4.5.1 or Internet Explorer 5.0 or later), QDM is a convenient tool to use from anywhere in the network.

The QDM user can perform two types of tasks. First, the user can configure QoS tools using a graphical interface from a browser. The user can also monitor real-time statistics about QoS behavior inside the single device, including graphs of bit/byte/packets rates, drop rates, queue depth, and so on. In fact, QDM can compare the counters before and after applying a QoS policy, showing you the impact made by the new QoS configurations. If you learn how to use QDM, you’ll have an easy time learning the syntax of the QoS configuration commands because an advantage of using QDM is that you will not have to remember the syntax for Modular QoS command-line interface (MQC)-based QoS commands. Essentially QDM enables you to configure MQC-based QoS commands without having to remember the syntax of each command. You can also forget all the **show** commands, because QDM will extract the same information you see with **show** commands, and graphically present it on the screen.

Some people argue that QDM’s best feature is that it is free! (At least it is free to anyone with a Cisco.com login that allows software downloads.) QDM has some great features for a free tool; however, when deploying QoS in a production network with 50, or 100, or even more routers and switches, you soon yearn for features that are not included in QDM. For instance, QDM enables you to configure a single device. If you want to always configure Low Latency Queuing (LLQ) the same way on every router, giving voice traffic a certain amount of bandwidth, and putting it in the low-latency queue, QDM requires you to repeat every point and click to configure each router. Instead, you could use QPM to define the policy (for instance, voice in the LLQ, *x* percent of bandwidth), and QPM would create and load the configuration into all the routers you want to configure. In addition, although QDM does provide limited trending reports, the reporting capabilities of QDM do not match those of SMS. In short, QDM is useful, but you will want to use QPM and the other tools for typical production QoS implementations.

To support QDM, the user needs a browser, either Netscape 4.5.1 or Internet Explorer 5.0, or higher. QDM code must also be loaded into Flash memory on the router, which takes about 1.6 MB of Flash memory (see www.cisco.com/cgi-bin/tablebuild.pl/qdm). Also the web server feature must be enabled on the router using the **ip http server** IOS command.

You can find the latest release and installation notes for QDM at www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/qdm/.

Table 9-2 lists some of the popular features of the QoS Management tools, with notations about the features QDM does or doesn't support.

Table 9-2 *QDM Features*

Feature	QDM
Supports wide variety of routers	Yes
Supports wide variety of switches	No
Free?	Yes
Allows network-wide QoS policy definition, followed by automatic deployment of appropriate configurations	No
Creates graphs of real-time performance	Yes
Creates graphs of historical performance	Yes*
Creates historical graphs to compare to service levels	No
Requires extra software loaded into router/switch Flash memory	Yes
End-user viewing of reports and configuration using a web browser	Yes
Manages *only* a single device from the browser	Yes
Manages the entire network from one browser window	No
Creates configuration for router probes that measure latency and jitter	No
Implements the actual probes and responses when necessary for measuring network performance	No

* QDM supports historical trending for up to five days of policy history.

QoS Policy Manager

QPM provides many of the features you need when you get serious about deploying QoS across an enterprise. The following list summarizes some of the more important features:

- Enables you to define a policy based on business rules.
- Automatically creates configurations to support the QoS policy, including marking, queuing, shaping, policing, and link fragmentation and interleaving (LFI) tools.
- Loads the correct configurations automatically.
- Enables you to monitor the device configurations to make sure no one has made change to them. In the event that the configurations have been changed, QPM can be used to restore the original configuration.
- Supports a larger variety of hardware than QDM.

In short, if you are looking for an entry-level tool, or if you are still experimenting with QoS, QDM is a good tool. When you get close to actually defining corporate QoS policies, and you want to easily create and manage QoS configuration, you want to upgrade to QPM, and take advantage of its much more robust and scalable features.

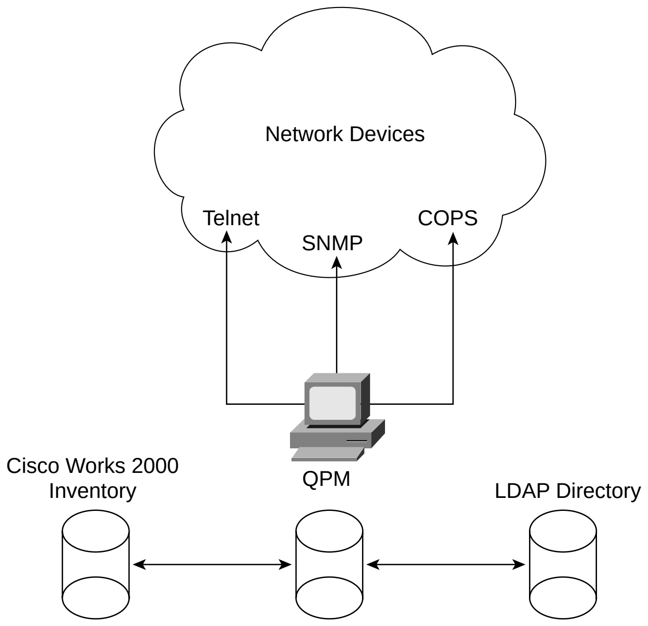
To get a sense for how QPM eases QoS configuration, imagine that you want to create a policy to mark all Voice over IP (VoIP) traffic with differentiated services (DiffServ) code point (DSCP) expedited forwarding (EF) as near to the edge of the network as possible. You just tell QPM what fields to look for in the packet or frame header with a point and click. QPM creates the CB marking configuration and loads it into all the appropriate devices. So, to use QPM, you still need to know what the base QoS tools can do, but you do not have to know the configuration syntax of all 35 different QoS tools, and you do not have to repeat the configuration task on all the devices—QPM takes care of that for you.

QPM runs on a variety of operating systems: Windows 98 (with Windows 2000 patches), Windows 2000, Windows NT Workstation, and Windows NT Server 4.0 Service Pack 5 or higher. (See www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/qos/qpm3_0/qpm30in/kiint.htm#xtocid2 for the hardware and software requirements of the latest release of QPM, version 3.0).

To configure a common QoS policy and push this policy to the network devices, QPM needs to be able to learn which devices are present in the network and communicate with these devices. QPM can use the CiscoWorks database to discover the location of the devices in the network.

Figure 9-1 outlines the overall location and functions between the QPM server and the rest of the network.

Figure 9-1 *QPM Server and Communication with Other Devices*



For QPM to create configurations, load the configurations, and monitor the configurations for changes, QPM must know which devices it should manage. The most convenient way to define the devices for QPM to manage is to use the device list from the CiscoWorks2000 database; in fact, QPM is often loaded on the same machine as CiscoWorks2000. However, you can statically define devices to QPM if you do not have CiscoWorks2000.

To control the devices, QPM uses Telnet to configure the devices, and Simple Network Management Protocol (SNMP) for monitoring changes to the configuration.

NOTE The QPM product line has also gone by the name QPM-PRO in the past. QPM-PRO refers to earlier versions of QPM. References can still be found for QPM-PRO on the Cisco website, but most documents just refer to it as QPM.

QPM provides a vitally necessary tool for networks that deploy QoS extensively. Table 9-3 lists some of the more popular features.

Table 9-3 *QPM Features*

Feature	QPM
Supports wide variety of routers	Yes
Supports wide variety of switches	Yes
Free?	No
Allows network-wide QoS policy definition, followed by automatic deployment of appropriate configurations	Yes
Creates graphs of real-time performance	Yes*
Creates graphs of historical performance	Yes
Creates historical graphs to compare to service levels	No
Requires extra software loaded into router/switch Flash memory	No
End-user viewing of reports and configuration using a web browser	Yes
Manages *only* a single device from the browser	No
Manages the entire network from one browser window	Yes
Creates configuration for router probes that measure latency and jitter	No
Implements the actual probes and responses when necessary for measuring network performance	No

* Through the use of the CB-QOS Management Information Base (MIB)

Service Assurance Agent

The Cisco Service Assurance Agent (SAA) allows a router IOS to measure network performance and availability by sending probes into the network. Based on configuration commands, a router generates and sends packets, called *probes*. The router sends the probes to specified remote destinations, which in some cases are other routers, and in other cases, other hosts. The router or host receiving the probes sends a response packet. Based on the replies, the router that sent the probe can calculate statistics, such as one-way and two-way delay, jitter, and simple availability.

SAA can create simple probes using Internet Control Message Protocol (ICMP) echo, but it's true value shines through with the more advanced probes. For instance, voice delay and jitter can be measured with SAA probes that generate Real Time Protocol (RTP) traffic. SAA supports probes for data-link switching (DLSw) traffic, because SAA was originally created to replace some legacy System Network Architecture (SNA) features. SAA can even generate HTTP traffic when downloading objects from live web servers. In many cases, you can create probes to match each traffic class specified in your QoS policies.

SAA is not a separate product, but rather an IOS feature that was first supported in the mid-1990s. (It was called the Response Time Reporter [RTR] tool then.) You can configure **rtr** commands inside IOS, creating probes and responders (which respond to probes). Then you can use **show** commands to look at the statistics. Because the **rtr** command has many options, it takes some experimentation to learn how to configure the commands effectively.

An easier way to configure SAA is to use IPM or SMS, which both rely on SAA. In other words, configure SAA indirectly. Suppose you want to see a real-time graph of delay and jitter for voice traffic between two routers. You can specify the endpoints as targets in IPM by assigning a target type of Cisco SAA Responder for the routers. IPM can then map an operation type, such as ICMP echo, to each SAA endpoint and test the QoS values between the endpoints. You can also see the same information, but trended over time (to compare today's performance with other days, for instance). Use SMS to do this, which also programs the routers with **rtr** commands, based on your input.

Table 9-4 lists some of the types of SAA probes supported in Cisco IOS Software Release 12.2.

Table 9-4 *Types of SAA Probes*

Type of Probe	Function
ICMP echo	Measures availability between a router and any other devices by sending ICMP echos (pings)
IP Path echo	Measures hop-by-hop response time by pinging each successive device in the route to the true destination of the probe.
TCP connection	Measures the time required to establish a TCP connection to a specified TCP port on a specified IP host.
UDP echo	Measures the round-trip time to send a packet to a specified UDP port number on a specified IP host
UDP jitter	Measures jitter, one-way delay, and packet loss in each direction
FTP	Measures time required to retrieve (FTP get) a specified file from a specified FTP server
HTTP	Tests the time taken to do DNS lookup to find a web server, TCP connection establishment time to the server, or the time required to download an object from the server
DNS	Measures the round-trip time between sending a DNS request and receiving a response
DLSw	Measures DLSw round-trip times, including DLSw stack delays, between a source and destination, without requiring an SAA agent on the destination host

Internetwork Performance Monitor

The Cisco Internetwork Performance Monitor (IPM) provides real-time graphical display of network performance information. Essentially, IPM generates SAA configurations, deploys the configurations, and then reports on the information gathered by the SAA probes. The reported information can describe latency and jitter, download times for static HTTP objects, TCP connection times, ICMP response times, and DNS request response times—all because SAA supports these functions with probes.

IPM provides more than just a pretty and easy way to view data collected by SAA. It can analyze the performance between two endpoints in the network by comparing probes generated and sent from different points in the network. Instead of just knowing that response time is slow, IPM can help pinpoint the slow point in the network. IPM also supports some historical reporting, although SMS has more historical reporting features. More importantly, you can set thresholds with IPM so that when network performance degrades past a certain point, it will generate an SNMP trap.

To purchase IPM, you actually need to purchase CiscoWorks2000 Routed WAN Management Solution, which is an optional CiscoWorks package. IPM began its life as the GUI interface for the Response Time Reporter (RTR) IOS feature, which later became SAA. IPM is now just one of the many features of CiscoWorks2000.

Service Management Solution

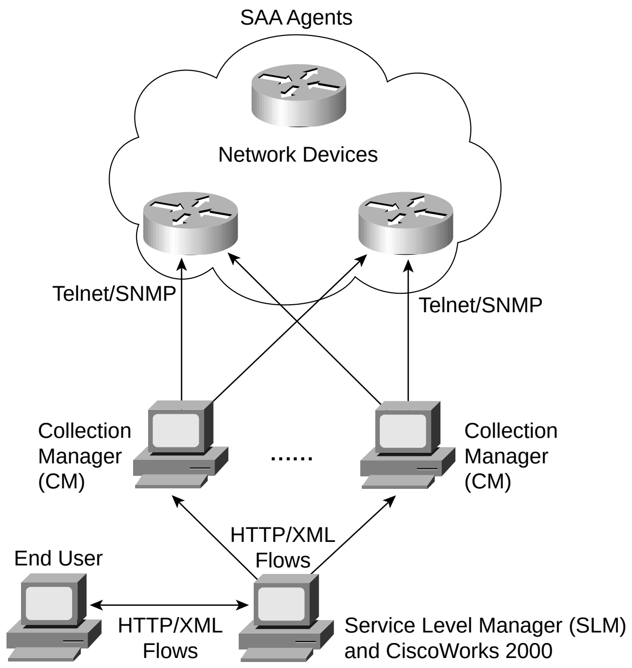
Like IPM, Service Management Solutions (SMS) is a part of CiscoWorks2000. Unlike IPM, SMS is a separate for-fee component of CiscoWorks, just like Router WAN Management Solutions is a different for-fee component of CiscoWorks. (IPM is a part of the Routed WAN Management solution for CiscoWorks2000.)

SMS measures network QoS performance against service-level agreements (SLAs). It provides robust data collection and trend reporting, with the capability to automatically compare actual network performance with defined service levels, over time. The reports aggregate and summarize the information for management reporting, so that the problem spots can be easily recognized. Of course, after management reviews the reports, they will want to know why the problems are occurring—which is why SMS also enables you to drill down inside the aggregated statistics, to see specific data points describing the results of groups of probes, or even for individual probes, to determine the root cause of the problem.

IPM and SMS collect and display the same basic types of information, but SMS needs to collect a much larger amount of information than IPM does. SMS needs to track network performance over time, so it needs to collect data continually. Also most SLAs cover the entire network, so the data must be collected for most, if not all, network devices. Because IPM focuses on troubleshooting a real-time performance, it can enable probes, let you view the information, and then delete or disable the probes, which greatly reduces the amount of data that IPM collects.

To meet the requirement to gather larger amounts of network performance information, SMS includes several components. Figure 9-2 shows the components.

Figure 9-2 *Service Management Solution Components*



SMS includes two main components. First, the Service Level Manager (SLM) is software that runs on the same host as CiscoWorks2000. SLM provides information to the end user of SMS, and generates the configuration of the probes based on end-user input. SMS Collection Managers (CMs) are software agents that run on computers spread around the network for scaling purposes, or a CM can reside on the SLM server for small installations. The CMs actually collect the information that is created by the SAA probes running on the devices in the network. The CMs collectively offload the intensive collection requirement from the SLM host, creating a more scalable and available architecture. If the SLM host goes down, for instance, the CMs can still collect the data. The SLM will asynchronously collect the data from the CMs, most typically during off-hours when the network is much less busy.

NOTE Cisco formerly sold a product called the Cisco Management Engine 1100, which was a piece of hardware on which to run CM. Although it isn't not sold anymore, the course upon which the exam is based still mentions the product, so you should at least be aware of it.

QoS Management Tool Summary

Table 9-5 lists many of the most important features of the various QoS management products, and identifies which products support the features.

Table 9-5 *Summary of QoS Management Tool Features*

Feature	QDM	QPM	IPM	SMS	SAA
Supports a wide variety of routers	Yes	Yes	Yes	Yes	Yes
Supports a wide variety of switches	No	Yes	No	No	No
Free?	Yes	No	No	No	No
Allows network-wide QoS policy definition, followed by automatic deployment of appropriate configurations	No	Yes	No	No	No
Creates graphs of real-time performance	Yes	Yes**	Yes	No	No
Creates graphs of historical performance	Yes*	Yes	Yes	Yes	No
Creates historical graphs to compared to service levels	No	No	No	Yes	No
Requires extra software loaded into router/switch Flash memory	Yes	No	No	No	No
End-user viewing of reports and configuration using a web browser	Yes	Yes	Yes	Yes	No
Manages *only* a single device from a browser	Yes	No	No	No	No
Manages entire network from one browser window	No	Yes	Yes	Yes	No
Creates configuration for router probes that measure latency and jitter	No	No	Yes	Yes	No
Implements the actual probes and responses when necessary for measuring network performance	No	No	No	No	Yes

* QDM supports historical trending for up to five days of policy history.

** Through the use of the CB-QOS MIB

QoS Design for the Cisco QoS Exams

QoS design covers concepts as high level as thinking about how each application is used inside a company, all the way down to how to choose reasonable numbers for committed burst (Bc) when performing traffic shaping. Any way you look at it, QoS design involves manipulating QoS characteristics—bandwidth, delay, jitter, and loss—for overall benefit in a network.

Notice, however, that this section is not just titled "QoS Design," but rather "QoS Design for the Cisco QoS Exams." QoS design, unbounded, could require several chapters. Interestingly, there is only one objective for the DQOS exam that mentions design, and yes, the number of questions in the exam question database is somewhat proportional to the number of objectives. Therefore, the specific treatment of design in this chapter focuses on the issues relating to what could be on the exam.

You might see two main variations on QoS design on the exam. One variation is a four-step QoS design process that is recommended by the Cisco DQOS course. Recommendations for how to apply QoS tools with voice and video are also covered.

Four-Step QoS Design Process

Figure 9-3 outlines the four-step design process that is recommended in the Cisco DQOS course.

Figure 9-3 *Four Steps for QoS Design*

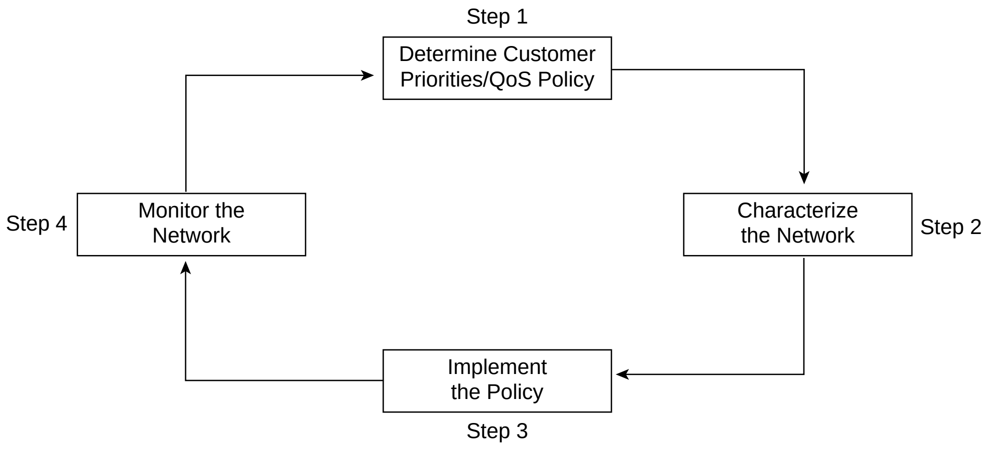


Figure 9-3 shows four tasks, each leading to the next in a continual circle. The process begins with determining priorities—what traffic should get more bandwidth? Less loss, jitter, and delay? The policies, however, typically define classes of traffic in general terms; you need to know the specifics of the traffic patterns so that you can configure the QoS tools to classify the traffic correctly. At Step 2, you characterize the traffic, which enables you to know how to configure the various classification features of the QoS tools. Then you can proceed with Step 3, where you actually configure the QoS tools. Finally, you need to monitor the network (Step 4) to determine whether you met the stated policy goals you determined in Step 1. The process continues over time, with the quality of the QoS implementation improving with each cycle. (Or at least you hope so!)

The following four sections provide more information about each step in the process.

Step 1: Determine Customer Priorities/QoS Policy

One of the hardest parts of the four-step process is to reach agreement about which traffic is important, and which is not. The "problem statement" in Step 1 presumes that you work for a vendor, or a reseller, or a consulting company, and you are helping a customer. Even if you are making QoS design choices for your own company, you need to involve the people who know the applications and the business priorities in the network, and not just the guys who operate and engineer the network.

You may need to hold planning sessions with representatives from many departments and divisions from the same company, or you may just get the most capable people together in one room and come up with a plan. Regardless of how formally or informally you attack the problem, you should finish the process with a set of general statements like the following:

- The traffic needs to be classified into five different categories.
- Category 1 should get 30 percent of the bandwidth, and have low delay and low jitter. It consists of voice traffic only. The delay budget is 150 ms maximum one-way delay.
- Category 2 should get 15 percent of the bandwidth, with moderate delay, and high jitter allowed. Loss can also be as high as needed. Web traffic to the customer server farm is placed in this category.
- And so on . . .

Statements with this level of detail provide you with a good idea of what needs to be implemented. However, you may not know exactly how to configure the parameters for each QoS tool based on this level of depth. For instance, you probably memorized the range of UDP port numbers used for VoIP traffic after reading this book, but most people simply do not memorize such trivia—so on to Step 2!

Step 2: Characterize the Network

Now you can send most of your fellow QoS policy planners back to their normal jobs, and start analyzing the network. In short, at this step, you should perform a network audit. The results of the audit should be that you have an understanding of how much traffic is occurring regularly in the network, and what QoS behavior they are experiencing in regards to bandwidth, delay, jitter, and loss. The audit should answer several questions, such as the following:

- How can I uniquely identify the packets that belong to each category defined in the QoS policies?
- How much bandwidth is currently being used by each application? By each QoS policy class?
- How have the bandwidth requirements been growing per application, and per class?
- What are the voice call traffic patterns? What are the busy hour statistics?

- What are the overall packet drop rates per link?
- What is the current link utilization on each link?

Although this list of questions is a short sample, the questions illustrate some of the general issues you must know before deploying QoS policies. If the policy calls for 15 percent of the link for Category 2, as stated earlier, but those packets currently consume 60 percent of an almost fully utilized link today, you need to reconsider the policy, consider increasing bandwidth, enable compression, and so on.

You have several tools at your disposal for performing the network audit. First, you can enable network-based application recognition (NBAR) to start gathering statistics and packet/byte/bit rates, and information about the types of traffic running through the network. You can also enable NetFlow statistic gathering in Cisco routers, which is another IOS tool, similar to NBAR, which gathers statistics for types of packets. With NetFlow, the statistics can be gathered to a server for historical reporting, which can be very helpful in this repetitive four-step process. Finally, you can always plug in your favorite network analyzer, many of which provide statistics of types of packet flows as well as the typical packet capture and analysis used when troubleshooting.

At the end of this step, you should have the same list of categories from Step 1, but now with enough information about how to go configure the QoS tools. This includes a classification and marking strategy, with each class specified, and the corresponding DSCP or precedence value identified. You should now be ready to implement specific QoS tools at Step 3.

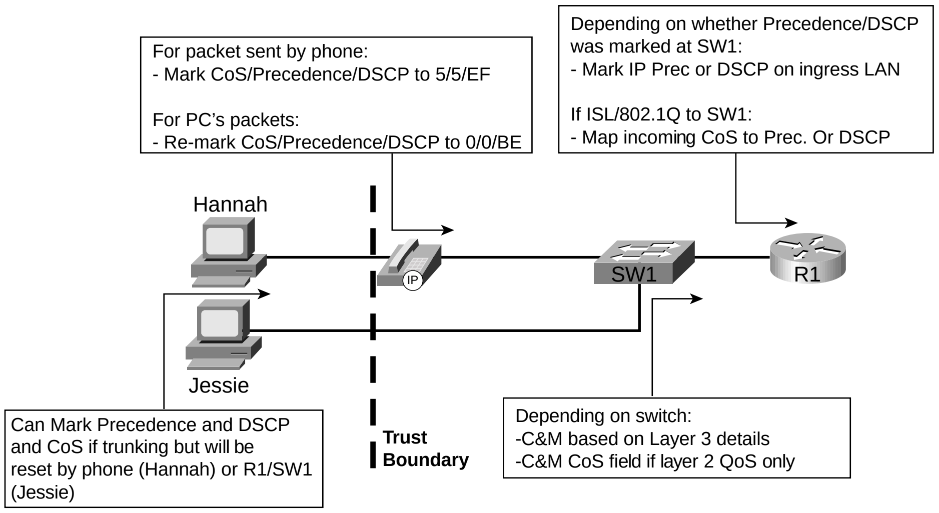
Step 3: Implement the Policy

During this step of the design process, you have two main subtasks:

1. To implement the trust boundary by using classification and marking
2. To choose and implement the tools that affect bandwidth, delay, jitter, and loss

Recall from Chapter 3, “Classification and Marking,” that you should attempt to mark the packet closest to the source of the packet. By doing so, all other QoS tools can more efficiently classify the packet based on its specific marking, such as the IP Precedence or IP DSCP fields. Of course, you must also choose a trust boundary—the boundary between the devices from which you can trust the precedence or DSCP values, and the devices that you cannot. Figure 9-4 may remind you of some of the typical choices for trust boundaries.

Figure 9-4 *QoS Trust Boundaries*


For packet sent by phone:
- Mark CoS/Precedence/DSCP to 5/5/EF
For PC's packets:
- Re-mark CoS/Precedence/DSCP to 0/0/BE
Depending on whether Precedence/DSCP was marked at SW1:
- Mark IP Prec or DSCP on ingress LAN
If ISL/802.1Q to SW1:
- Map incoming CoS to Prec. Or DSCP
Hannah
IP
SW1
R1
Jessie
Can Mark Precedence and DSCP and CoS if trunking but will be reset by phone (Hannah) or R1/SW1 (Jessie)
Trust Boundary
Depending on switch:
-C&M based on Layer 3 details
-C&M CoS field if layer 2 QoS only

Typically, at remote sites with no Layer 3 switching capabilities, you end up performing the classification and marking function on ingress to a WAN edge router. In larger campuses, classification and marking typically occur in a switch capable of performing Layer 3 marking. Marking can also occur in Cisco IP Phones.

Before implementing the rest of the QoS features, you need to decide which tools meet the requirements. For instance, you may have decided to shape traffic on WAN links, so you could use generic traffic shaping (GTS), class-based (CB) shaping, or Frame Relay traffic shaping (FRTS). If you use a Frame Relay WAN, and you want to perform FR fragmentation, however, you may have to use FRTS, because GTS and CB shaping do not allow FR fragmentation at the same time on all platforms. Therefore, you must next pick the tools that enable you to configure the defined policies.

Finally, you just need to configure the tools at the various points in the network. When completed, you need to know whether the configurations work, which brings you to Step 4.

Step 4: Monitor the Network

The monitoring step requires tools as well as effort. The tools have already been covered in this chapter—namely QDM, QPM, IPM, and SMS. With these tools, you can configure and monitor the network performance in real time and historically. However, creating reports is not enough—you must read them, interpret them, and begin the whole four-step process over again—at least if you want to do what is suggested by the DQOS course!

QoS Design Guidelines for Voice and Video

This final section of this chapter reviews some of the most important considerations when planning QoS for voice and video, along with some recommendations for how to use QoS tools for voice and video. Voice and video differ from data applications significantly in terms of their needs for bandwidth, delay, jitter, and loss; the QoS policy design should consider these differences. Most of the information in this section is scattered throughout the book, so most of the individual concepts are for review purposes. However, now that you are familiar with the concepts of the QoS required for managing delay, jitter, and loss, you need to be able to apply them in a network, and answer questions about them on the Cisco QoS exams.

Voice and Video: Bandwidth, Delay, Jitter, and Loss Requirements

Voice calls need a constant amount of bandwidth, with low delay, low jitter, and low loss. In other words, voice calls need to experience excellent treatment for the calls to sound good. One of the biggest challenges for the network is to provide consistent (low-jitter) delay for the voice calls, so delay tends to be the big focus when designing QoS for voice. You must decide on a delay budget for the voice calls, and then examine all the call paths to decide whether the delay budget can be met. Table 9-6 and Table 9-7 list the delay components covered in Chapter 1, along with recommended overall delay budgets.

Table 9-6 *One-Way Delay Budget Guidelines for Voice*

One Way Delay (ms)	Description
0–150	ITU G.114 recommended acceptable range
0–200	Cisco's recommended acceptable range
150–400	ITU G.114's recommended range for degraded service
400+	ITU G.114's range of unacceptable delay in all cases

Table 9-7 *Delay Components, Variable and Fixed*

Delay Component	Fixed or Variable	Comments
Codec	Fixed	Varies slightly based on codec and processing load; considered fixed in course books (and probably on exams). Typically around 10 ms.
Packetization	Fixed	Some codecs require a 30-ms payload, but packetization delay does not vary for a single codec. Typically 20 ms, including when using G.711 and G.729.

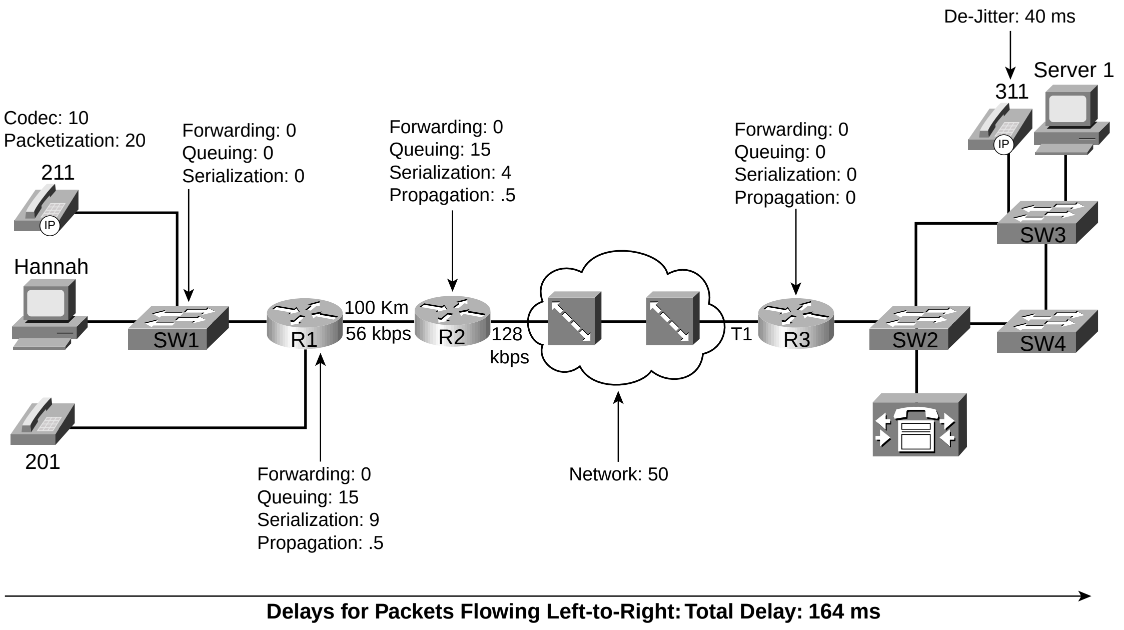
continues

Table 9-7 *Delay Components, Variable and Fixed (Continued)*

Delay Component	Fixed or Variable	Comments
Propagation	Variable	Varies based on length of circuit. About 5ms/100 km.
Queuing	Variable	This is the most controllable delay component for packet voice.
Serialization	Fixed	It is fixed for voice packets, because all voice packets are of equal length. It is variable based on packet size for all packets. The delay is based on the clock speed of the WAN circuit.
Network	Variable	Least controllable variable component. Latency is potentially higher in a packet-switched network than in a leased line.
De-jitter buffer (initial playout delay)	Variable	This component is variable because it can be configured for a different value. However, that value, once configured, remains fixed for all calls until another value is configured. In other words, the initial playout delay does not dynamically vary.

The challenge with voice delay relates to the overall budget versus the fact that only some of the delay components can be lowered using QoS tools. Looking at Figure 9-5, for example, you can see typical values for the voice delay components for a call from one IP Phone to another IP Phone. The variable delay components have actually been constrained pretty well in this case. The first point of delay is the originating IP Phone. In this example, a 10-ms delay occurs for the codec and a 20-ms delay occurs for packetization, bringing the initial delay to 30 ms. For the purposes of this example, assume that there is no delay experienced on the Ethernet interfaces of the switches or routers. The next point of delay is experienced on the egress interface of R1. Here there is a 15-ms queuing delay, 9-ms serialization delay, and a .5-ms propagation delay, bringing the delay experienced up to this point to 54.5 ms. The next point of delay is experienced on the egress interface of R2. Here there is a 15-ms queuing delay, 4-ms serialization delay, and a .5-ms propagation delay, bringing the delay experienced up to this point to 74 ms. The next point of delay is experienced as the packet crosses the IP network. Here a 50-ms delay occurs, bringing the total delay up to this point to 124 ms. The last point of delay that the packet experiences is in the jitter buffer of the remote IP Phone. Here the delay experienced is 40 ms, bringing the one-way delay to 164 ms end to end.

Figure 9-5 *Complete End-to-End Voice Delay Example*



The delay has crept beyond the acceptable limits of one-way delay, according to G.114, but is slightly under the limit of 200 ms suggested by Cisco. Without the additional voice delays, the 150-ms delay budget seemed attainable. With 30 ms of codec and packetization delay, and a (reasonable) default of 40-ms de-jitter delay (actually, de-jitter initial playout delay), however, 70 ms of that 150/200-ms delay is consumed. So, what can you do to stay within the desired delay budget? You attack the variable components of delay, as listed in Table 9-7.

The other big consideration for voice QoS relates to how much bandwidth voice needs. The amount of bandwidth needed varies based on which codec is used, and whether Voice Activity Detection (VAD) is used. QoS designs typically assume that VAD is not used. Table 9-8 lists some of the popular codecs, and the bandwidth required.

Table 9-8 *Bandwidth Requirements for Various Types of Voice Calls*

Codec	Payload Bandwidth	IP/UDP/RTP Header Size	L2 Header Type	L2 Header Size	Total Bandwidth
G.711	64 kbps	40 bytes	Ethernet	14	85.6
G.711	64 kbps	40 bytes	MLPPP/FR	6	82.4
G.711	64 kbps	2 bytes (cRTP)	MLPPP/FR	6	67.2

continues

Table 9-8 *Bandwidth Requirements for Various Types of Voice Calls (Continued)*

Codec	Payload Bandwidth	IP/UDP/RTP Header Size	L2 Header Type	L2 Header Size	Total Bandwidth
G.729	8 kbps	40 bytes	Ethernet	14	29.6
G.729	8 kbps	40 bytes	MLPPP/FR	6	26.4
G.729	8 kbps	2 bytes (cRTP)	MLPPP/FR	6	11.2

For DQOS test takers: These numbers are extracted from the DQOS course.

Interactive (two-way) video requires delay, jitter, and loss behavior similar to voice traffic. Video uses more bandwidth than voice, in some cases far more bandwidth, and the amount of bandwidth varies over time. Table 9-9 displays some of the bandwidth requirements for video of some popular formats.

Table 9-9 *Video Codecs and Required Bandwidth*

Video Codec	Application	Required Bandwidth
MPEG-4	Over WANs	28.8–400 kbps
H.261	Low motion	100–400 kbps
MPEG-1	VHS quality	500–500 kbps
MPEG-2	DVD QUALITY	1.5–10 Mbps

For one-way streaming video, the QoS characteristics differ when compared to two-way video and voice. The de-jitter buffer can expand from tens of milliseconds to tens of seconds, removing most of the jitter consideration. In addition, with very large de-jitter buffers, and because there is no need for interactive responses, one-way delay can be very large. For one-way video, as long as the video gets enough bandwidth, and there is low loss, the video stream works well.

Voice and Video QoS Design Recommendations

Cisco makes recommendations for how to apply QoS to voice and video in the QoS course books and in several documents on the Cisco website. This section summarizes many of the recommendations made in the QoS courses, because the QoS exams are based on the courses. As with all recommendations, these are not the only ways to apply QoS for voice and video—but they are the ways specifically listed in the design chapter of the Cisco DQOS course. For those of you who want more information, a particularly good source is the Cisco AVVID Network Infrastructure Enterprise QoS Design Guide, which you can find at www.cisco.com/warp/customer/771/srnd/qos_srnd.pdf.

The QOS recommendations for voice and video are listed by category of QoS tool:

- **Classification and marking**—Cisco suggests three classes for voice and video traffic:
  - — One class for voice payload, marked with DSCP EF/CoS 5/precedence 5
  - — Another class for video payload, marked with DSCP DSCP AF41/CoS 4/precedence 4
  - — A third class for both voice and video signaling, marked with DSCP AF31/CoS 3/precedence 3

  Table 1-13 in Chapter 1 outlines the protocols to watch for when classifying voice and video payload and signaling.
- **Queuing**—The biggest dilemma when thinking of voice and video in the same network is whether video is placed into a low-latency queue with the voice. The course book recommends that voice only be placed into the low-latency queue, with video payload being placed into another queue. The general recommendations are as follows:
  - — Use Class-Based Weighted Fair Queuing (CBWFQ) with LLQ; if the IOS does not support it, use IP RTP Priority for voice.
  - — Place voice payload (DSCP EF) into the low-latency queue.
  - — Place video payload into another class queue.
- **Shaping and policing**—Shaping slows down traffic when congestion occurs, and policing discards packets when congestion occurs. Voice and video both do not like packet loss or delay. (The exception is one-way video, which tolerates delay.) However, shaping may be usefully applied to a site to throttle the transmission of traffic to prevent overrunning the ingress circuit at the remote site. If you apply shaping to traffic on a Frame Relay virtual circuit (VC), also apply queuing to the shaping queue, to minimize the extra delay added for voice. The following list summarizes the recommendations:
  - — Use FRTS as necessary, particularly if FR fragmentation is also needed.
  - — Set Tc to 10 ms (Tc = CIR/Bc).
  - — Set excess burst (Be) to 0 (no excess burst).
  - — Shape strictly to committed information rate (CIR) to avoid drops in the Frame Relay cloud.
  - — When using adaptive shaping, set mincir to a large enough value to support all voice and video traffic.
  - — Avoid egress blocking pitfalls.
  - — Use queuing on shaping queues to improve delay, jitter, and loss for voice and video.

- **Link efficiency**—Link fragmentation and interleaving (LFI) reduces one of the variable delay components, namely serialization delay, while compression techniques reduce the bandwidth required. Both types of tools can help. The following list summarizes the recommendations:
  - — Fragment to a size equal to or slightly larger than the size of the voice packet.
  - — Use LFI on links with clock rates of 768 kbps or less (1500-byte frame at 768 kbps takes about 15 ms).
  - — Ensure that the voice packets are smaller than the fragmentation size, so that the voice packets are not fragmented.
  - — If several VCs use the same access link, and at least one has voice, fragment on all VCs using that access link.
  - — Use Real Time Protocol header compression (cRTP).
- **Call admission control**—Without CAC, all the other work to design and implement QoS can be wasted. If you choose your configuration options expecting one range of concurrent calls and video conferences, and twice as many happen, the users and the network will suffer. Use any and all methods available, depending on the topology, as described in Chapter 8, "Call Admission Control and QoS Signaling."

Foundation Summary

The “Foundation Summary” is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final prep before the exam, these tables and figures are a convenient way to review the day before the exam.

QPM needs to be able to communicate with the routers and switches, as well as learn about what devices are in the network. Figure 9-6 outlines the overall location and functions between the QPM server and the rest of the network.

Figure 9-6 *QPM Server and Communication with Other Devices*

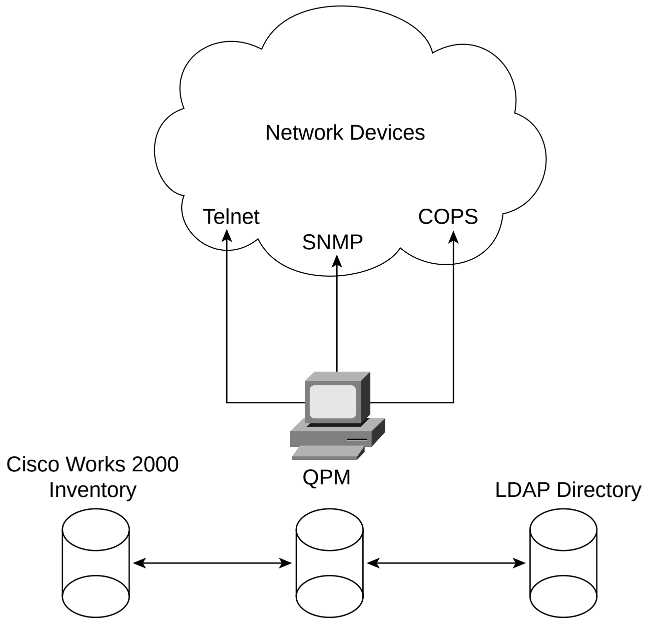


Table 9-10 lists some of the types of SAA probes supported in IOS 12.2.

Table 9-10 *Types of SAA Probes*

Type of Probe	Function
ICMP echo	Measures availability between a router and any other devices by sending ICMP echos (pings)
IP Path echo	Measures hop-by-hop response time by pinging each successive device in the route to the true destination of the probe.
TCP connection	Measures the time required to establish a TCP connection to a specified TCP port on a specified IP host.
UDP echo	Measures the round-trip time to send a packet to a specified UDP port number on a specified IP host
UDP jitter	Measures jitter, one-way delay, and packet loss in each direction
FTP	Measures time required to retrieve (FTP get) a specified file from a specified FTP server
HTTP	Tests the time taken to do DNS lookup to find a web server, TCP connection establishment time to the server, or the time required to download an object from the server
DNS	Measures the round-trip time between sending a DNS request and receiving a response
DLSw	Measures DLSw round-trip times, including DLSw stack delays, between a source and destination, without requiring an SAA agent on the destination host

To meet the requirement to gather larger amounts of network performance information, SMS includes several components. Figure 9-7 shows the components, with an explanation of each component following the figure.

Figure 9-7 *Service Management Solution Components*

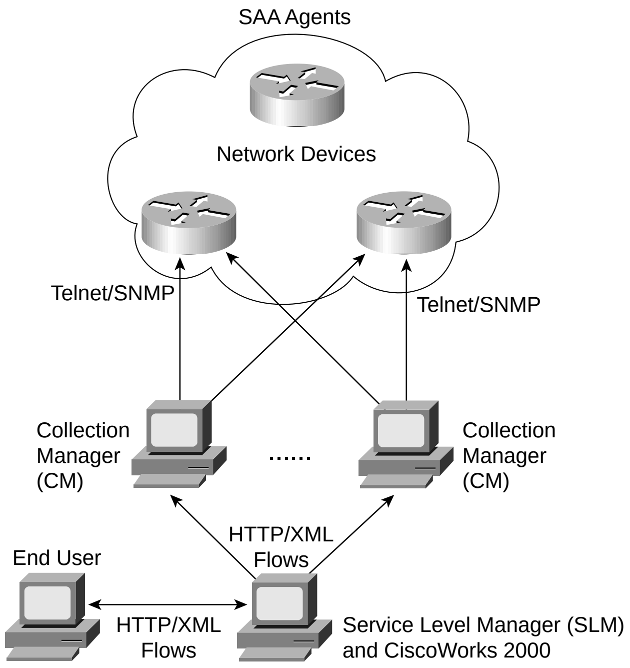


Table 9-11 lists many of the most important features of the various QoS management products and identifies which products support the features.

Table 9-11 *Summary of QoS Management Tool Features*

Feature	QDM	QPM	IPM	SMS	SAA
Supports a wide variety of routers	Yes	Yes	Yes	Yes	Yes
Supports a wide variety of switches	No	Yes	No	No	No
Free?	Yes	No	No	No	No
Allows network-wide QoS policy definition, followed by automatic deployment of appropriate configurations	No	Yes	No	No	No
Creates graphs of real-time performance	Yes	Yes**	Yes	No	No
Creates graphs of historical performance	Yes*	Yes	Yes	Yes	No

continues

Table 9-11 *Summary of QoS Management Tool Features (Continued)*

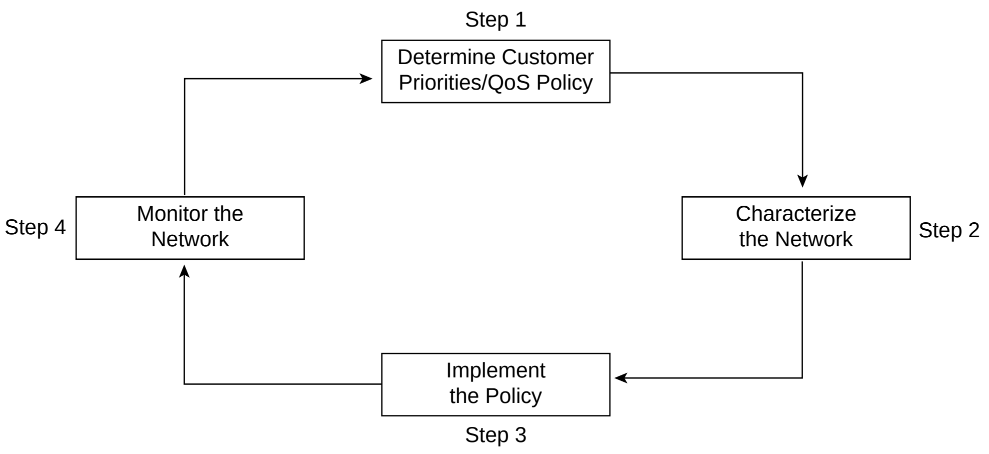
Feature	**QDM**	**QPM**	**IPM**	**SMS**	**SAA**
Creates historical graphs to compared to service levels	No	No	No	Yes	No
Requires extra software loaded into router/switch Flash memory	Yes	No	No	No	No
End-user viewing of reports and configuration using a web browser	Yes	Yes	Yes	Yes	No
Manages *only* a single device from a browser	Yes	No	No	No	No
Manages entire network from one browser window	No	Yes	Yes	Yes	No
Creates configuration for router probes that measure latency and jitter	No	No	Yes	Yes	No
Implements the actual probes and responses when necessary for measuring network performance	No	No	No	No	Yes

* QDM supports historical trending for up to five days of policy history.

** Through the use of the CB-QOS MIB

Figure 9-8 outlines the four-step design process as suggested in the Cisco DQOS course.

Figure 9-8 *Four Steps for QoS Design*



You must decide on a delay budget for the voice calls, and then examine all the call paths to decide whether the delay budget can be met. The delay components, along with recommended overall delay budgets, are listed in Tables 9-12 and 9-13.

Table 9-12 *One-Way Delay Budget Guidelines for Voice*

One Way Delay (ms)	Description
0–150	ITU G.114 recommended acceptable range
0–200	Cisco's recommended acceptable range
150–400	ITU G.114's recommended range for degraded service
400+	ITU G.114's range of unacceptable delay in all cases

Table 9-13 *Delay Components, Variable and Fixed*

Delay Component	Fixed or Variable	Comments
Codec	Fixed	Varies slightly based on codec and processing load; considered fixed in course books (and probably on exams). Typically around 10 ms.
Packetization	Fixed	Some codecs require a 30-ms payload, but packetization delay does not vary for a single codec. Typically 20 ms, including when using G.711 and G.729.
Propagation	Variable	Varies based on length of circuit. About 5ms/100 km.
Queuing	Variable	This is the most controllable delay component for packet voice.
Serialization	Fixed	It is fixed for voice packets, because all voice packets are of equal length. It is variable based on packet size for all packets. The delay is based on the clock speed of the WAN circuit.
Network	Variable	Least controllable variable component. Latency is potentially higher in a packet-switched network than in a leased line.
De-jitter buffer (initial playout delay)	Variable	This component is variable because it can be configured for a different value. However, that value, once configured, remains fixed for all calls until another value is configured. In other words, the initial playout delay does not dynamically vary.

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD-ROM included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD-ROM nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD-ROM exam to include, or not include, the topics that are only on the CCIP QoS.

QoS Management

1. What do the following acronyms stand for? IPM, QPM, SAA, QDM
2. What QoS Management tool is actually a component of the CiscoWorks2000 Routed WAN Management Solution?
3. What QoS management tool is free?
4. What QoS management tool(s) enables you to define QoS policies, after which it creates the appropriate configurations, and deploys the configurations to identified network devices?
5. Which QoS management tool(s) includes as a primary feature the capability to graph real-time information about bit/byte/packet rates and packet loss?
6. Describe the two primary components of SMS and their general roles.
7. What QoS management tool(s) generates packets, called probes, and measures various performance attributes, such as delay and jitter?
8. What QoS management tool(s) includes as a primary function the capability to produce historical performance reports as compared to service-level agreements?
9. What QoS management tool(s) includes as a primary function the capability to troubleshoot and isolate network-wide performance issues by viewing the individual components in a complete end-to-end path through the network?
10. Does QDM or QPM support more devices?

11 Among QPM, QDM, SMS, and IPM, which tool(s) originally had a limited capability to create real-time or historical graphs of performance?

12 What QoS management tool(s) is embedded into a router?

13 Which QoS management tools are available in CiscoWorks2000 bundles?

QoS Design

14 List Step 1 in the QoS design process, and give a short summary of what the step includes.

15 List Step 2 in the QoS design process, and give a short summary of what the step includes.

16 List Step 3 in the QoS design process, and give a short summary of what the step includes.

17 List Step 4 in the QoS design process, and give a short summary of what the step includes.

18 What are the ITU G.114 and Cisco recommendations for maximum one-way delay for voice calls?

19 What delay component represents the most controllable variable delay in a voice call?

20 How much bandwidth does a G.729 call require if ignoring the Layer 2 header/trailer? How much bandwidth does it require if using Frame Relay?

21 Summarize the Cisco recommendation for marked values for voice and video traffic.

22 Describe the Cisco recommendation for queuing voice and video traffic.

23 Summarize the Cisco recommendations for tuning shaping parameters when shaping must be enabled on VCs carrying voice.

This chapter covers LAN QoS topics for which there are no specific exam topics on either the DQOS exam or the QoS exam. This chapter is included in the book, however, because LAN QoS topics do remain an important part of QoS design and implementation.

CHAPTER 10

LAN QoS

"Why would I go through the hassle of configuring and managing QoS on my LAN, which is already overprovisioned? If I have to, I will just add more bandwidth!"

This is a common statement. Conventional wisdom dictates, "If you have enough bandwidth, you do not need QoS. QoS is only used for WAN links that do not have enough bandwidth. LANs have plenty of bandwidth, so you are safe."

Although this sounds reasonable, it is not entirely accurate. Bandwidth is only one of the factors that need to be taken into consideration for a network that supports real-time applications.

This chapter discusses the capabilities of Cisco Catalyst LAN switches and examines some recommended configurations designed to prioritized real-time applications across your local-area network (LAN).

"Do I Know This Already?" Quiz

The purpose of the "Do I Know This Already?" quiz is to help you decide whether you really need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 18-question quiz, derived from the major sections in the "Foundation Topics" section of the chapter, helps you determine how to spend your limited study time.

Table 10-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 10-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions	Score
1	LAN QoS Concepts	1 to 6	
2	Catalyst 6500	7 to 10	
3	Catalyst 4500/4000	11 to 14	
4	Catalyst 3550/3524	15 to 18	
All questions		1 to 18	

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **14 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary," and the "Q&A" sections.
- **15 or 16 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, move to the next chapter.

1. What is instantaneous buffer overrun?
2. What is meant by the term "1p1q4t?"
3. What packet or frame marking(s) can a Layer 2 switch use to prioritize traffic?
4. What must be done for a Layer 2 switch to properly classify a packet received from a host across the WAN?
5. What is a drop threshold?
6. What is a trust boundary, and where should it be set?
7. Explain why the first packet in a flow may not retain the proper CoS values when a Catalyst 6500 is configured with a PFC.
8. What is the difference between Hybrid mode and Native mode in a 6500 series switch?
9. What queue and threshold is recommended for call control traffic on a Catalyst 6500?

10 Which series of Ethernet line cards are preferred in the Catalyst 6500 series for QoS? Why?

11 Which supervisor engine is preferred on the Catalyst 4500/4000 series? Why?

12 What trust state are the ports of a 4500 with a Supervisor III in when QoS is enabled?

13 In which queue number does the priority queue reside on a Catalyst 4500 or 4000 with a Supervisor III Engine?

14 What keyword(s) is used to enable the priority queue on a Catalyst 4500 or 4000 with a Supervisor III module?

15 In which queue number does the priority queue reside on a Catalyst 3550?

16 What command enables QoS on the Catalyst 3550?

17 Name the four methods a Catalyst 3550 can use to classify traffic.

18 What command enables QoS on the Catalyst 3524?

Foundation Topics

The Need for QoS on the LAN

LAN quality of service (QoS) is often misunderstood and overlooked. Thanks to conventional wisdom, most network administrators think that they do not require LAN QoS. If your plans include the addition of real-time applications, such as IP telephony or video conferencing, you should include a strategy for LAN QoS while you are in the planning stages of your project. This up-front planning can result in the perceived success or failure of your project in the eyes of the end users.

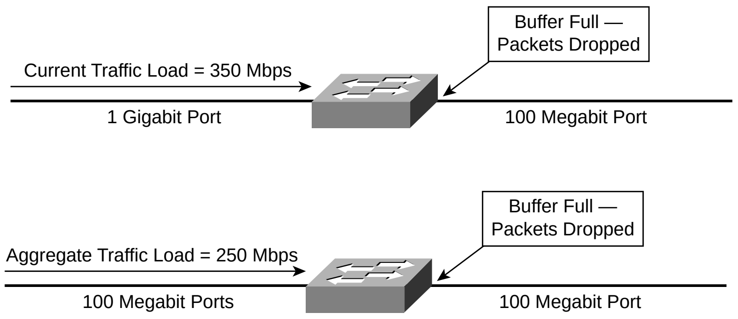
This chapter explores the need for LAN QoS and discusses the following options available to you:

- Layer 2 priority (CoS)
- Layer 3-to-Layer 2 (DSCP-to-CoS) mapping
- Layer 2 queues
- Drop thresholds
- Trust boundaries
- Cisco switch configurations for LAN QoS

Buffer Overflow (Overrun)

Suppose that is it 8:30 on a Monday morning. All of your fellow employees report to work, simultaneously power on their computers, and begin their day. Their traffic flows through the access layer switches and converge on the uplink port to the distribution layer switch. In the event that the uplink port is smaller than the input port, or the uplink port is oversubscribed, the buffer on the uplink port begins to fill, as shown in Figure 10-1.

Figure 10-1 *Buffer Overflow*


Buffer Full — Packets Dropped
Current Traffic Load = 350 Mbps
1 Gigabit Port
100 Megabit Port
Buffer Full — Packets Dropped
Aggregate Traffic Load = 250 Mbps
100 Megabit Ports
100 Megabit Port

For an instant, the buffer of the uplink port can become full, potentially causing packets to drop. In a typical TCP/IP data-networking environment, this is not a concern because the packet is retransmitted. In an environment comprised of real-time applications, such as IP telephony and video conferencing, instantaneous buffer overruns (overflows) can affect the quality of the voice or video streams.

In a Cisco IP telephony environment, a G.729 digital signal processor (DSP) can rebuild up to 30 ms of lost voice. If the Cisco standard 20 ms per packet has been deployed, a single packet can be dropped without degrading voice quality; if two consecutive voice packets are lost, resulting in 40 ms of lost voice conversation, however, the algorithm cannot compensate, and a clip is heard in the conversation. In the event that the Real Time Protocol (RTP) stream carries a fax or modem conversation, a single packet results in a modem retrain, whereas two consecutive packets result in a dropped connection.

By classifying the real-time applications on your LAN and scheduling the desired level of priority for each real-time application, you can avoid these problems; however, you cannot remedy the problem by adding bandwidth. QoS tools are required to manage these buffers to minimize loss, delay, and jitter. You must properly enable and configure QoS to set priority by matching a traffic flow with a desired queue or threshold.

Bandwidth is *not* a substitute for LAN QoS! LAN QoS is a buffer management issue.

Marking and Classification

As discussed in Chapter 3, “Classification and Marking,” marking at Layer 2 takes place in the 3-bit User Priority field called the Class of Service (CoS), which resides inside an Ethernet header. The CoS field only exists inside Ethernet frames when trunking (either 802.1Q or Inter-Switch Link [ISL]) is used. The field can be used to set eight different binary values, which the classification features of other QoS tools can use.

Figure 10-2 shows the general location of the CoS field inside the 802.1q and ISL headers.

As discussed in Chapter 3, Layer 3 marking takes place in the Type of Service (ToS) or Differentiated Services (DS) field in the IP Header. The IP Precedence and Differentiated Services Code Point (DSCP) fields can be marked with any valid binary value of either 3 or 6 bits, respectively. Chapter 2, “QoS Tools and Architectures,” contains detailed discussion of the recommended values used in these two fields. Figure 10-3 outlines the two fields and their positions inside an IP header.

Figure 10-2 *Class of Service Fields*

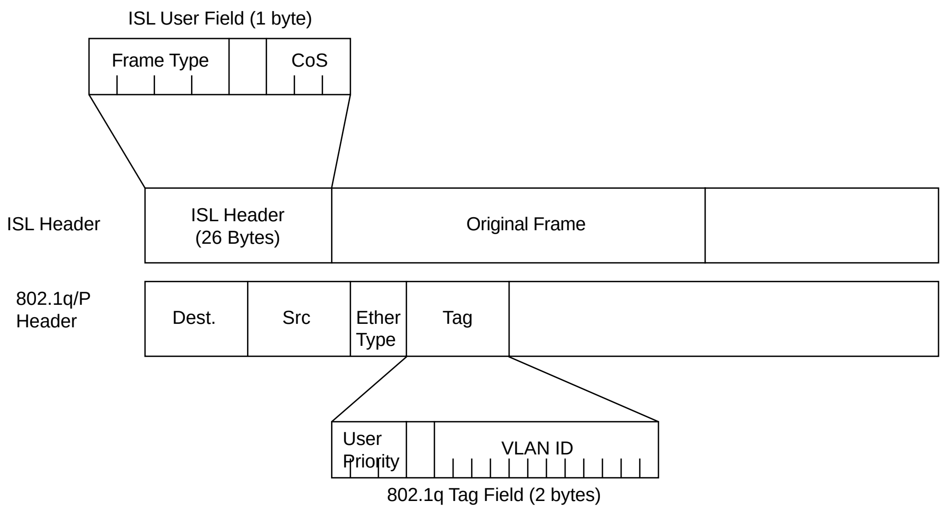
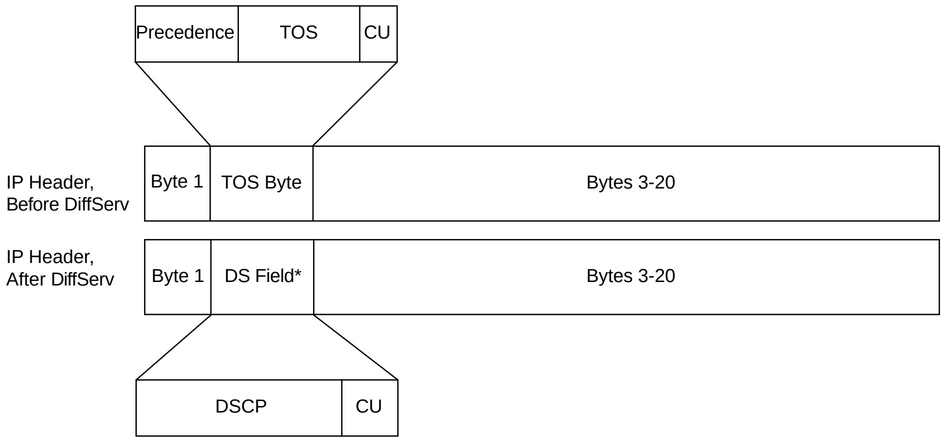


Figure 10-3 *IP Precedence and IP DSCP Fields*



Layer 3-to-Layer 2 Classification Mapping

Layer 2 switches perform QoS features based on the CoS field within the Ethernet headers without regard for the markings within the IP header. As discussed in Chapter 3, when a packet exits a router interface that is not configured as an 802.1Q trunk, a CoS value does not exist. The receiving switch marks the packet with the default CoS value configured on the receiving port, typically 0, although the DSCP value may be expedited forwarding (EF). If the switch receiving the packet cannot classify on DSCP markings, this packet is classified as a best-effort packet, even if the intent was to provide prioritization.

To solve this problem, you must map the Layer 3 marking to Layer 2 marking on the router, and perform trunking between the switch and the router, so that the Layer 2 switch can properly classify the received traffic. Without remapping, L2 switches cannot differentiate between traffic flows.

Example 10-1 shows the configuration necessary to map IP DSCP to CoS that enables the Layer 2 switch to prioritize traffic. This example also shows a CoS-to-IP DSCP map that allows the router to match incoming CoS values with DSCP values.

Example 10-1 *DSCP-to-CoS Mapping*

```
class-map cos3
 match cos 3
!
class-map cos5
 match cos 5
!
class-map EF
 match ip dscp EF
!
class-map AF31
 match ip dscp AF31
policy-map map-cos-to-dscp
 class cos5
  set ip DSCP EF
 class cos3
  set ip dscp af31
 class class-default
  set ip dscp default
!
policy-map map-dscp-to-cos
 class EF
  set cos 5
 class AF31
  set cos 3
 class class-default
  set cos 0
!
interface FastEthernet0/0
```

continues

Example 10-1 *DSCP-to-CoS Mapping (Continued)*

```
!
interface FastEthernet0/0.1
 encapsulation dot1Q 102
service-policy input map-cos-to-dscp
service-policy output map-dscp-to-cos
!
interface FastEthernet0/0.2
 encapsulation dot1Q 2 native
```

After classification has been established, QoS tools can be used to differentiate the traffic and direct the desired classification to the proper queue.

Layer 2 Queues

Although queues on LAN switches behave similarly as compared with router queues, a slightly different perspective helps with understanding the problems solved by queuing on a LAN switch. Think of a Layer 2 queue as a bucket that buffers packets until they are transported. The larger the bucket, the more packets it holds. In a switch that has one queue, refered to as 1q, for example, all traffic is placed in this queue regardless of traffic classification and traffic is serviced on a first-in, first-out (FIFO) basis. If a packet arrives at the queue during a period of congestion, the packet may be dropped if the queue cannot hold additional packets.

Using multiple queues on a switch interface allocates some of the finite number of switch buffers in each queue. You can protect the voice queue against the possibility of running out of buffers by putting delay-sensitive voice into one queue, and all else into another. A switch that has two queues, refered to as 2q, for example, has the capability to direct traffic matching a specific classification, such as voice traffic with a CoS value of 5, into one queue while directing other traffic that does not meet this critera into the other queue. Because voice packets are smaller and more predictable than data packets, classifying and scheduling high-priority traffic into this second queue decreases the likelihood that the second queue will expreience buffer overrun and discard the high-priority traffic.

Keep in mind that each port has a finite amount of buffer space to support the buckets. One queue will take all of the buffer space, for instance, two queues will divide the buffer space into two parts, three queues divide the buffer space into three parts, and so on. If the buffer space is too small, it will not be effective in momentarily holding the traffic before transport. Because nonpriority queues are serviced in either a round-robin or a Weighted Round-Robin manner, there is no guarantee that the traffic in the buffer is transported next. This limitation can lead to instaneous buffer overrun for the classified traffic of the last example due to the smaller buffer size available.

One remmedy for this situation is the introduction of a single strict-priority queue, refered to as *1p*—meaning “one-priority queue.” In a 1p queue, all traffic within the priority queue is prioritized over the traffic in a standard queue by being transported as it is received. For example, a switch configured with one priority queue and one standard queue, called *1p1q*, can be configured to direct traffic matching a specific classification, such as voice traffic with a CoS value of 5, into the priority queue, ensuring immediate transport, while directing other traffic that does not meet this critera into the standard queue.

Table 10-2 defines a few of the Layer 2 queues available in Cisco Catalyst switches.

Table 10-2 *Layer 2 Queues*

Layer 2 Queue	Description
1q	A single Layer 2 queue. All traffic crossing the interface flows through this queue.
2q	2 Layer 2 queues. Traffic can be directed to the desired queue based on classification.
1p1q	1 priority Layer 2 queue and 1 standard Layer 2 queue. Traffic can be directed to the priority queue based on classification. Other traffic can be directed to the standard queue.
1p2q	1 priority Layer 2 queue and 2 standard Layer 2 queues. Traffic can be directed to the priority queue based on classification. Traffic can be directed to the desired standard queues based on additional classification.

It is highly recommended that you place real-time applications into the priority queue.

Drop Thresholds

Drop thresholds define the amount of the total Layer 2 buffer use that must be reached before a specified class of traffic is dropped. In other words, this is how much the bucket needs to fill before a decision is made to begin dropping traffic of a specific class. Some switches have one priority queue and one standard queue, for instance, with four drop thesholds on the nonpriority queue, refered to as *1p1q4t*. For the traffic placed into the nonpriority queue, the four drop thresholds indicate which class of traffic, based on CoS, should be dropped most agresssively in the event that a percentage of the bucket fills.

Table 10-3 illustrates a possible configuration of the theshold values from the previous example.

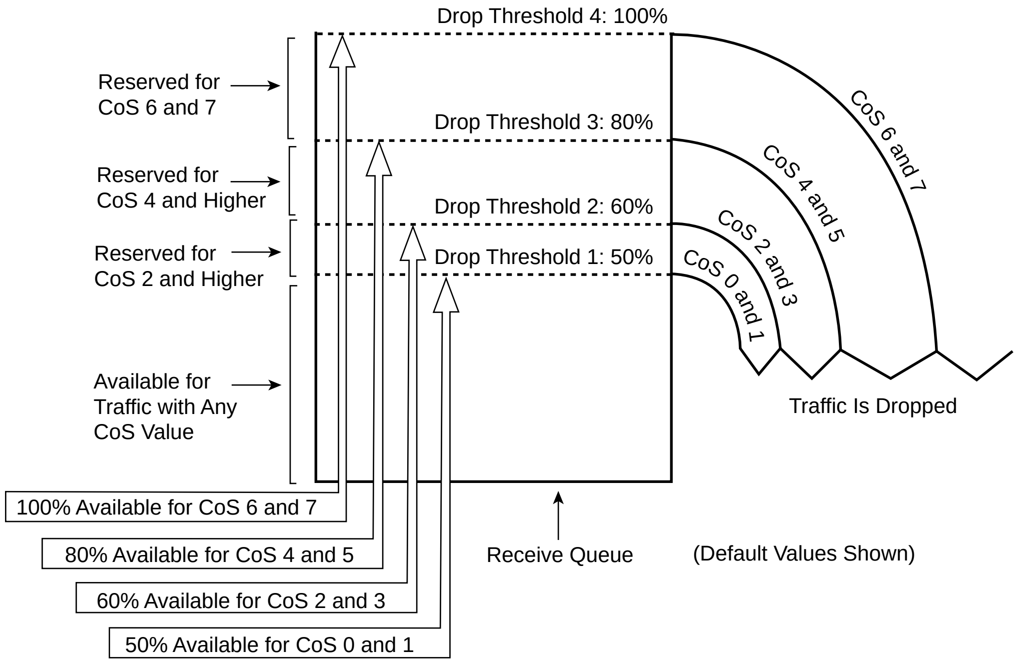
Table 10-3 *Drop Thesholds*

Queue Use Threshold	Traffic to Drop
50%	CoS 0–1
60%	CoS 2–3
80%	CoS 4–5
100%	CoS 6–7

After the queue has reached 50 percent of capacity, any traffic classified with CoS of 0 or 1 becomes drop candidates to avoid congestion. If the queue continues to fill in spite of the drops, at 60 percent of capacity any traffic classified with a CoS of 0, 1, 2, or 3 becomes drop candidates to avoid congestion. If the queue still continues to fill in spite of the drops, at 80 percent of capacity any traffic classified with a CoS of 0, 1, 2, 3, 4, or 5 becomes drop candidates to avoid congestion. At 100 percent of capacity, all traffic, regardless of classification, becomes drop candidates.

Figure 10-4 illustrates the drop thresholds.

Figure 10-4 *Drop Thresholds*

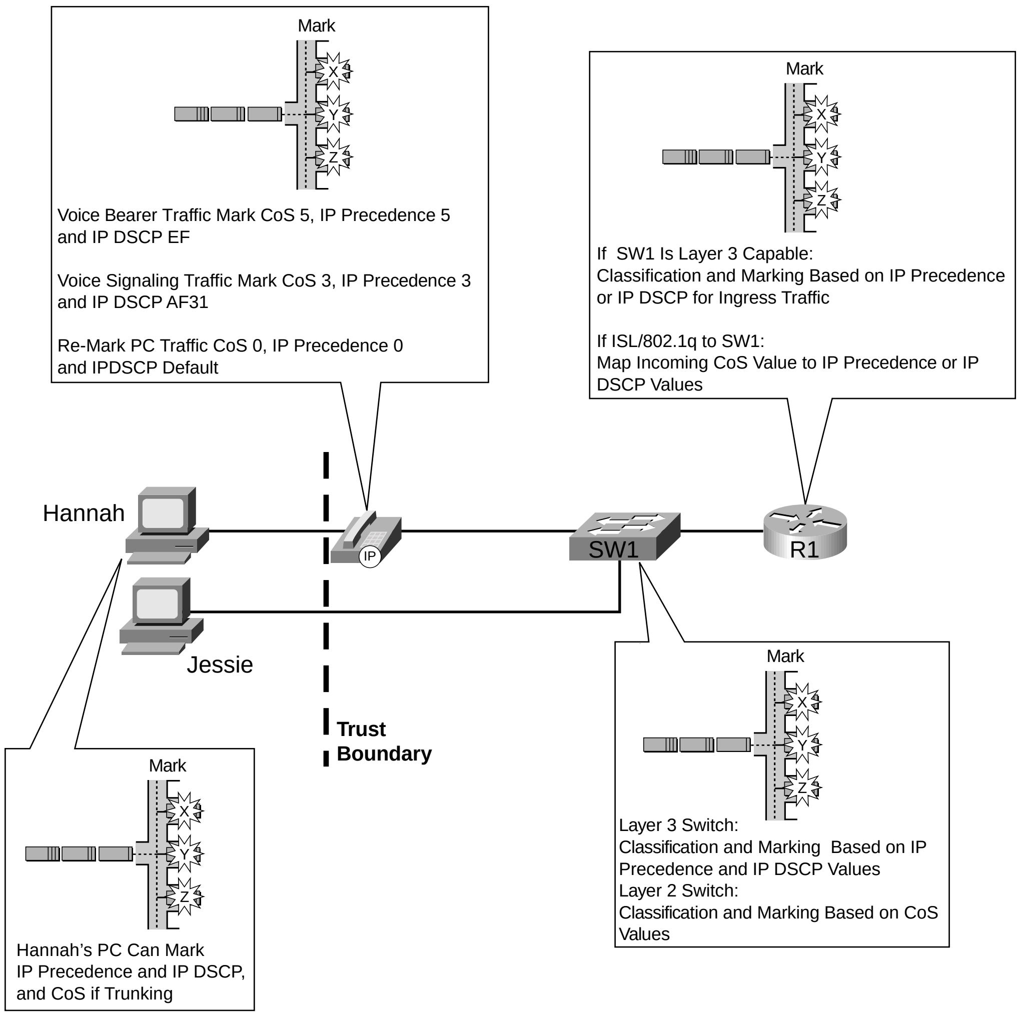

Drop Threshold 4: 100%
Drop Threshold 3: 80%
Drop Threshold 2: 60%
Drop Threshold 1: 50%
Reserved for CoS 6 and 7
Reserved for CoS 4 and Higher
Reserved for CoS 2 and Higher
Available for Traffic with Any CoS Value
CoS 6 and 7
CoS 4 and 5
CoS 2 and 3
CoS 0 and 1
Traffic Is Dropped
100% Available for CoS 6 and 7
80% Available for CoS 4 and 5
60% Available for CoS 2 and 3
50% Available for CoS 0 and 1
Receive Queue
(Default Values Shown)

Thesholds provide an alternative to dividing the buffer space to add more standard queues, eliminating the risk of starving one queue while flooding another. Setting a drop theshold allows the entire buffer space to be used, decreasing the potential for instaneous buffer overrun for higher-priority traffic in the standard queue.

Trust Boundries

As discussed in Chapter 3, trust boundaries represent the point in the network that you begin to trust the packet markings. Establishing a trust boundary becomes increasingly important as PC network interface cards (NICs) gain the capability to mark traffic and alter the desired QoS design of your network. Figure 10-5 shows a network that uses a trust boundary at the IP Phone. A trust boundary can also be configured on an access switch in the event that an IP Phone is not present.

Figure 10-5 *Trust Boundaries*


Mark
X
Y
Z
Voice Bearer Traffic Mark CoS 5, IP Precedence 5 and IP DSCP EF
Voice Signaling Traffic Mark CoS 3, IP Precedence 3 and IP DSCP AF31
Re-Mark PC Traffic CoS 0, IP Precedence 0 and IPDSCP Default
Mark
X
Y
Z
If SW1 Is Layer 3 Capable: Classification and Marking Based on IP Precedence or IP DSCP for Ingress Traffic
If ISL/802.1q to SW1: Map Incoming CoS Value to IP Precedence or IP DSCP Values
Hannah
IP
SW1
R1
Jessie
Trust Boundary
Mark
X
Y
Z
Layer 3 Switch: Classification and Marking Based on IP Precedence and IP DSCP Values
Layer 2 Switch: Classification and Marking Based on CoS Values
Mark
X
Y
Z
Hannah's PC Can Mark IP Precedence and IP DSCP, and CoS if Trunking

This allows the network administrator to mark the packet as close to the end of the network without trusting all the applications on the PCs. You can gain more granular control of QoS tools by Specifying which devices are trusted.

Cisco Catalyst Switch QoS Features

Cisco Catalyst switches offer a wide array of QoS tools to help configure QoS in the LAN to support real-time applications. The following sections discuss the QoS capabilities, and hardware components if applicable, of these Cisco Catalyst switches:

- Catalyst 6500 series
- Catalyst 4500/4000 series
- Catalyst 3550 series
- Catalyst 3524 series

Catalyst 6500 QoS Features

This section is not intended to be an exhaustive look at the Cisco Catalyst 6500. Although many options exist for this product, the scope of this discussion covers only the Qos features of the Catalyst 6500 as they relate to the prioritization of real-time applications such as IP telepehony.

The 6500 architecture distributes the QoS features among the line cards, the Supervisor card, and the Policy Feature Card (PFC). To appreciate what is really happening in the 6500, you must first have a basic understanding of the line cards and Supervisor cards.

Supervisor and Switching Engine

The type of supervisor and switching engine installed in your Catalyst 6500 determines the QoS feature set. As of this writing, there are four supervisor/switching engine combinations. Table 10-4 outlines these.

Table 10-4 *Supervisor and Switching Engine Combinations*

Supervisor	Switching Engine
Supervisor II (WS-X6K-SUP2-2GE)	Layer 3 Switching Engine II (WS-F6K-PFC2—PFC2)
Supervisor Engine I (WS-X6K-SUP1A-2GE) or (WS-X6K-SUP1-2GE)	Layer 3 Switching Engine (WS-F6K-PFC)

Table 10-4 *Supervisor and Switching Engine Combinations (Continued)*

Supervisor	**Switching Engine**
Supervisor Engine I (WS-X6K-SUP1A-2GE) or (WS-X6K-SUP1-2GE)	Layer 2 Switching Engine II (WS-F6020A)
Supervisor Engine I (WS-X6K-SUP1A-2GE) or (WS-X6K-SUP1-2GE)	Layer 2 Switching Engine I (WS-F6020)

Enter the **show module** command for the supervisor engine to display your switching engine configuration. The next several examples just provide a reference so that you can determine which cards are in your particular 6500 switch. Example 10-2, for instance, shows the response of a Catalyst 6500 configured with a Supervisor II card, which includes the Policy Feature Card (PFC2) Layer 3 switching engine.

Example 10-2 **show module** *of Catalyst 6500 with a Supervisor II*

```
Cat6500>(enable)  show module
Mod  Slot Ports Module-Type                Model               Sub Status
--- ---- ----- -------------------------  ------------------- --- --------
1   1    2     1000BaseX Supervisor       WS-X6K-SUP2-2GE     yes ok
15  1    1     Multilayer Switch Feature  WS-F6K-MSFC2        no  OK
2   2    2     1000BaseX Supervisor       WS-X6K-SUP2-2GE     yes standby
16  2    1     Multilayer Switch Feature  WS-F6K-MSFC2        no  OK

Mod Module-Name         Serial-Num
--- ------------------- -----------
1                       SAD051307ER
15                      SAD050814J3
2                       SAD0421058D
16                      SAD042106PB

Mod MAC-Address(es)                        Hw     Fw         Sw
--- -------------------------------------- ------ ---------- -----------------
1   00-01-64-75-eb-ce to 00-01-64-75-eb-cf 2.2    6.1(3)     6.2(2)
00-01-64-75-eb-cc to 00-01-64-75-eb-cd
00-05-5f-0f-ec-80 to 00-05-5f-0f-ec-bf
15  00-05-5e-da-ee-00 to 00-05-5e-da-ee-3f 1.2    12.1(8a)E5 12.1(8a)E5
2   00-01-64-f8-38-ac to 00-01-64-f8-38-ad 0.310  6.1(2)     6.3(3)
00-01-64-f8-38-ae to 00-01-64-f8-38-af
16  00-02-fd-b1-0f-00 to 00-02-fd-b1-0f-3f 1.1    12.1(8a)E5 12.1(8a)E5

Mod Sub-Type                Sub-Model           Sub-Serial  Sub-Hw
--- ----------------------- ------------------- ----------- ------
1   L3 Switching Engine II  WS-F6K-PFC2         SAD051405TV 1.3
2   L3 Switching Engine II  WS-F6K-PFC2         SAD04110B5E 0.305
```

Example 10-3 shows the response of a Catalyst 6500 configured with a Supervisor IA card and the optional PFC card.

Example 10-3 **show module** *of Catalyst 6500 with a Supervisor IA and Layer 3 PFC Switching Engine*

```
Cat6500> (enable) show module

Mod Slot Ports Module-Type               Model               Sub Status
--- ---- ----- ------------------------- ------------------- --- --------
1   1    2     1000BaseX Supervisor      WS-X6K-SUP1A-2GE    yes OK
15  1    1     Multilayer Switch Feature WS-F6K-MSFC2        no  OK
2   2    2     1000BaseX Supervisor      WS-X6K-SUP1A-2GE    yes standby
16  2    1     Multilayer Switch Feature WS-F6K-MSFC2        no  OK

Mod Module-Name          Serial-Num
--- -------------------- -----------
1                        SAD0433088P
15                       SAD04360AJ8
2                        SAD05030UEW
16                       SAD05030Z4W

Mod MAC-Address(Es)                        Hw     Fw         SW
--- -------------------------------------- ------ ---------- -----------------
1   00-d0-d3-3d-d2-3a to 00-d0-d3-3d-d2-3b 3.2    5.3(1)     6.3(3)
00-d0-d3-3d-d2-38 to 00-d0-d3-3d-d2-39
00-30-7b-4e-64-00 to 00-30-7b-4e-67-ff
15  00-03-6b-f1-2a-40 to 00-03-6b-f1-2a-7f 1.1    12.1(8a)E5  12.1(8a)E5
2   00-02-7e-f5-c8-7e to 00-02-7e-f5-c8-7f 7.1    5.3(1)     6.2(2)
00-02-7e-f5-c8-7c to 00-02-7e-f5-c8-7d
16  00-04-dd-f1-f0-80 to 00-04-dd-f1-f0-bf 1.2    12.1(8a)E5  12.1(8a)E5

Mod Subtype                Sub-Model           Sub-Serial  Sub-Hw
--- ---------------------- ------------------- ----------- ------
1   L3 Switching Engine     WS-F6K-PFC          SAD04330KWZ 1.1
2   L3 Switching Engine     WS-F6K-PFC          SAD050315AR 1.1
```

Example 10-4 shows the response of a Catalyst 6500 configured with a Supervisor IA card, which includes the Layer 2 switching engine.

Example 10-4 **show module** *of Catalyst 6500 with a Supervisor IA and Layer 2 Switching Engine*

```
Cat6500>(enable)show module

Mod Slot Ports Module-Type               Model               Sub Status
--- ---- ----- ------------------------- ------------------- --- --------
1   1    2     1000BaseX Supervisor      WS-X6K-SUP1A-2GE    yes ok
2   2    2     1000BaseX Supervisor      WS-X6K-SUP1A-2GE    yes standby

Mod Module-Name         Serial-Num
--- ------------------- -----------
1                       SAD050404KM
```

Example 10-4 **show module** *of Catalyst 6500 with a Supervisor IA and Layer 2 Switching Engine (Continued)*

```
2                        SAD05040EC2

Mod MAC-Address(es)                        Hw     Fw         Sw
--- ---------------------------------- ------ ---------- ----------------
1   00-02-7e-27-17-f6 to 00-02-7e-27-17-f7 7.0    5.3(1)     5.5(9)
00-02-7e-27-17-f4 to 00-02-7e-27-17-f5
00-d0-03-8c-9c-00 to 00-d0-03-8c-9f-ff
2   00-01-64-75-80-16 to 00-01-64-75-80-17 7.0    5.3(1)     5.5(9)
00-01-64-75-80-14 to 00-01-64-75-80-15

Mod Sub-Type                Sub-Model          Sub-Serial  Sub-Hw
--- ----------------------- ------------------ ----------- ------
1   L2 Switching Engine II  WS-F6020A          SAD05030WR5 2.0
2   L2 Switching Engine II  WS-F6020A          SAD05030VZH 2.0
```

Policy Feature Card

Cisco 6500 QoS features differ based on whether a PFC is installed and on other configuration options. For instance, the Layer 2 switching engine on the Supervisor I or Supervisor IA classifies, marks, and schedules traffic based on destination Media Access Control (MAC) address or VLAN tag information. Without the optional PFC, all marking of packets by the Layer 2 switching engine takes place in the CoS field. A Layer 2 switching engine does not examine the IP header, and so markings in the IP Precidence and DSCP fields are ignored.

With the addition of the PFC, the Catalyst 6500 is capable of enabling advanced QoS tools such as packet classification and marking, scheduling, and congestion avoidance based on either Layer 2, 3, or 4 header information. Specifically, the PFC can perform classification and marking, and policing. The line cards perform queuing and packet-drop logic. The PFC performs the actual packet-forwarding process, effectively making it a Layer 3 switching engine. The PFC can examine the Layer 2 and Layer 3 headers as the packet arrives. If the destination IP adress exists in the flow cache, for PFC cards, or the Cisco Express Forwarding (CEF) Forwarding Information Base (FIB), for PFC2 cards, the PFC can rewrite the following five header fields:

- Layer 2 (MAC) Destination address
- Layer 2 (MAC) Source address
- Layer 3 IP Time-to-Live (TTL)
- Layer 3 Checksum

The Layer 2 (MAC) checksum (also called the frame checksum or FCS) allows the packet to retain the original Layer 2 and Layer 3 QoS markings.

In the event that the IP destination does not exist in the flow cache, or CEF FIB, the first packet of the flow is forwarded to the Multilayer Switch Feature Card (MSFC) to route the packet to the correct interface. The routing decision populates the flow cache or CEF FIB for subsequent packets; however, the first routed packet loses the original CoS marking. All subsequent packets are switched by the PFC and retain their original Layer 2 and Layer 3 markings.

Although the PFC and PFC2 are both Layer 3 switching engines, they differ in a few ways, as listed in Table 10-5.

Table 10-5 *PFC and PFC2 Differences*

PFC	PFC2
Available as an option on Supervisor Engine IA only	Premounted on all Supervisor Engine II modules
Performs Layer 2/3/4 services according to on a flow-based architecture	Required for Layer 2/3/4 services in a CEF-based architecture
Performs certain Cisco IOS features such as PBR, standard and extended access lists, and reflexive ACLs in hardware	Performs certain Cisco IOS features such as PBR, unicast RPF, TCP intercept, standard and extended access lists, and reflexive ACLs in hardware, and incorporates significant performance improvements
Centralized forwarding mechanism based on a flow-caching mechanism	Distributed forwarding mechanism based on a distributed Cisco Express Forwarding (DCEF) architecture. Used for IP unicast and multicast traffic and Layer 2 forwarding

After a flow has been established, the PFC has the capability to use a QoS access list to police traffic to reduce the flow of traffic to a predefined limit. Traffic in excess of that limit can be dropped or have the DSCP value in the frame marked down to a lower value.

A QoS access-control list (ACL), consisting of a list of acess-control entries (ACEs), defines a set of QoS rules that the PFC uses to process incoming frames. ACEs are similar to a router ACL. The ACE defines classification, marking, and policing criteria for an incoming frame. If an incoming frame matches the criteria set in the ACE, the QoS engine processes the frame.

Ethernet Interfaces

The QoS role of Ethernet interfaces in a Catalyst 6500 includes the scheduling of packets and congestion management, as well as providing inline power to support the addition of real-time applications such as IP telephony. It is important to understand the capabilities of the Ethernet modules in your Catalyst to properly provision your network for the addition of real-time applications. Table 10-6 illustrates the QoS advantage that the current generation of Ethernet modules, typically the WS-X65*xx* series of cards, offers over the past generation of

Ethernet modules, typically the WS-X63*xx* series of cards. Table 10-6 shows a comparision of the 48-port Ethernet modules.

Table 10-6 *Ethernet Buffers, Queues, and Thresholds*

Ethernet Module	Total Buffer Size	RX Buffer Size	TX Buffer Size	RX Queue	TX Queue
WS-X6548-RJ-45	1116 KB	28 KB	1088 KB	1p1q0t	1p3q1t
WS-X6348-RJ-45	128 KB	16 KB	112 KB	1q4t	2q2t

The **show port capabilities** command enables you to display QoS information about a particular port or modules. Example 10-5 displays the QoS capabilities of a supervisor IA card and a WS-X6248-RJ-45 card.

Example 10-5 *The* **show port capabilities** *Command*

```
CAT6K> (enable) show port capabilities 1/1
Model                    WS-X6K-SUP1A-2GE
Port                     1/1
Type                     1000BaseSX
Speed                    1000
Duplex                   full
Trunk encap type         802.1Q,ISL
Trunk mode               on,off,desirable,auto,nonegotiate
Channel                  yes
Broadcast suppression    percentage(0-100)
Flow control             receive-(off,on,desired),send-(off,on,desired)
Security                 yes
Dot1x                    yes
Membership               static,dynamic
Fast start               yes
QOS scheduling           rx-(1p1q4t),tx-(1p2q2t)
CoS rewrite              yes
ToS rewrite              DSCP
UDLD                     yes
Inline power             no
AuxiliaryVlan            no
SPAN                     source,destination
COPS port group          1/1-2
Link debounce timer      yes
CAT6K> (enable)
CAT6K> (enable) show port capabilities 2/1
Model                    WS-X6248-RJ-45
Port                     2/1
Type                     10/100BaseTX
Speed                    auto,10,100
Duplex                   half,full
Trunk encap type         802.1Q,ISL
Trunk mode               on,off,desirable,auto,nonegotiate
```

continues

Example 10-5 *The* **show port capabilities** *Command (Continued)*

```
Channel                  yes
Broadcast suppression    percentage(0-100)
Flow control             receive-(off,on),send-(off)
Security                 yes
Dot1x                    yes
Membership               static,dynamic
Fast start               yes
QOS scheduling           rx-(1q4t),tx-(2q2t)
CoS rewrite              yes
ToS rewrite              DSCP
UDLD                     yes
Inline power             no
AuxiliaryVlan            1..1000,1025..4094,untagged,dot1p,none
SPAN                     source,destination
COPS port group          2/1-48
Link debounce timer      yes
CAT6K> (enable)
```

The WS-X65*xx* series of Ethernet cards introduce a priority queue and extend the buffer sizes to better accommodate traffic during times of congestion. In a side-by-side comparison, it is plain to see that the WS-65*xx* series of Ethernet line cards are the better choice for QoS.

The uplink Gigabit Ethernet ports on the Supervisor I support a single receive queue with four thresholds (1q4t) and two transmit queues with two thresholds (2q2t). The uplink Gigabit Ethernet ports on the Supervisor IA and the Supervisor II cards include enhanced QoS features that provide an additional priority queue for both ingress (1p1q4t) and egress (1p2q2t) interfaces. These queues are serviced in a WRR method, except for the priority queue, which is always serviced as soon as frames have entered the queue. Table 10-7 lists the TX and RX queues supported by each supervisor engine.

Table 10-7 *Supervior Queues and Thresholds*

Supervisor	RX Queue	TX Queue
Supervisor II (WS-X6K-SUP2-2GE)	1p1q4t	1p2q2t
Supervisor Engine IA (WS-X6K-SUP1A-2GE)	1p1q4t	1p2q2t
Supervisor Engine I (WS-X6K-SUP1-2GE)	1q4t	2q2t

The addition of the priority queue on the Supervisor IA and Supervisor II offers an advantage for providing QoS for real-time applications over Supervisor I.

Ingress and egress scheduling are always based on the CoS value associated with the frame. By default, higher CoS values are mapped to higher queue numbers. CoS 5 traffic, typically associated with Voice over IP (VoIP) traffic, is mapped to the strict-priority queue, if present.

In addition to the different queues, each standard queue has one or more drop thresholds. There are two types of drop thresholds:

- Tail-drop thresholds

  On ports with tail-drop thresholds, frames of a given CoS value are admitted to the queue until the drop threshold associated with that CoS value is exceeded; subsequent frames of that CoS value are discarded until the threshold is no longer exceeded.

  If CoS 1 is assigned to Queue 1, threshold 2, for example, and the threshold 2 watermark is 60 percent, frames with CoS 1 are not dropped until Queue 1 is 60 percent full. All subsequent CoS 1 frames are dropped until the queue is less than 60 percent full.

- WRED-drop thresholds

  On ports with Weighted Random Early Detection (WRED)-drop thresholds, frames of a given CoS value are admitted to the queue based on a random probability designed to avoid buffer congestion. The probability of a frame with a given CoS being admitted to the queue or discarded depends on the weight and threshold assigned to that CoS value.

  If CoS 2 is assigned to Queue 1, threshold 2, for example, and the threshold 2 watermarks are 40 percent (low) and 80 percent (high), frames with CoS 2 are not dropped until Queue 1 is at least 40 percent full. As the queue depth approaches 80 percent, frames with CoS 2 have an increasingly higher probability of being discarded rather than being admitted to the queue. When the queue is more than 80 percent full, all CoS 2 frames are dropped until the queue is less than 80 percent full. The frames that the switch discards when the queue level is between the low and high thresholds are picked at random, rather than on a per-flow or FIFO basis. This method works well with protocols such as TCP that are capable of adjusting to periodic packet drops by backing off and adjusting their transmission window size.

Enter the **show qos info config** [*mod*/*port*] command to determine whether your Catalyst switch port supports WRED. In Example 10-6, the **show qos info config** command shows that port 2/1 does not support WRED.

Example 10-6 *The* **show qos info config** *Command*

```
Cat6500> show qos info config 2/1
QoS setting in NVRAM:
QoS is enabled
Port 2/1 has 2 transmit queue with 2 drop thresholds (2q2t).
Port 2/1 has 1 receive queue with 4 drop thresholds (1q4t).
Interface type:vlan-based
ACL attached:
The qos trust type is set to untrusted.
```

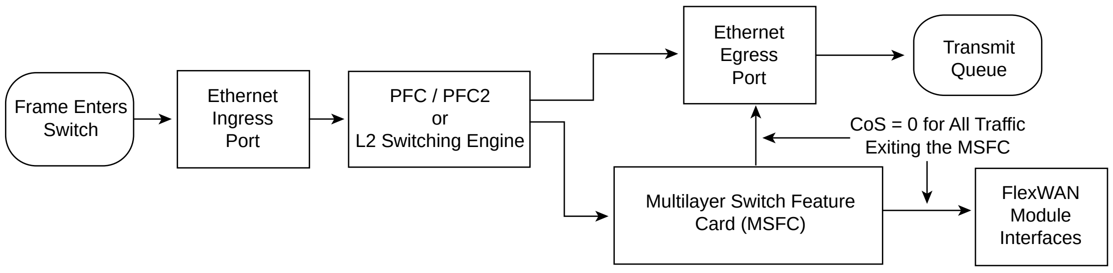
Example 10-6 *The* **show qos info config** *Command (Continued)*

```
Default CoS = 0
Queue and Threshold Mapping:
Queue Threshold CoS
----- --------- --------------
1     1         0 1
1     2         2 3
2     1         4 5
2     2         6 7
Rx drop thresholds:
Rx drop thresholds are disabled for untrusted ports.
Queue #  Thresholds - percentage (abs values)
-------  -------------------------------------
1        50% 60% 80% 100%
Tx drop thresholds:
Queue #  Thresholds - percentage (abs values)
-------  -------------------------------------
1        40% 100%
2        40% 100%
Tx WRED thresholds:
WRED feature is not supported for this port_type.
Queue Sizes:
Queue #  Sizes - percentage (abs values)
-------  -------------------------------------
1        80%
2        20%
WRR Configuration of ports with speed 1000MBPS:
Queue #  Ratios (abs values)
-------  -------------------------------------
1        100
2        255
```

QoS Flow on the Catalyst 6500

Now that you have an understanding of the various hardware options available on the Catalyst 6500 as they relate to QoS, we can begin to discuss the QoS flow as the frames are received and forwarded through the switch. Figure 10-6 shows the QoS flow overview.

Figure 10-6 *QoS Flow Overview*


Frame Enters Switch
Ethernet Ingress Port
PFC / PFC2 or L2 Switching Engine
Ethernet Egress Port
Transmit Queue
CoS = 0 for All Traffic Exiting the MSFC
Multilayer Switch Feature Card (MSFC)
FlexWAN Module Interfaces

The frame arrives at the switch Ethernet ingress port. The existing frame marking, Ethernet ingress queuing structure, and the current trust state of the ingress Ethernet port determine the level of service the frame receives. After the ingress queue and threshold have been determined, the ingress port forwards the frame to the Layer 2 switching engine, or the PFC Layer 3 switching engine if one is installed.

When the supervisor does not have a PFC, the Layer 2 switching engine forwards the packet to the egress Ethernet port. The egress switch port uses the CoS marking on the frame to determine the queue that is used for the frame as it exits the switch.

When the supervisor does have a PFC, the PFC consults its cache to determine whether it has a known path to the destination address. If the path exists in the cache, the PFC forwards the packet to the correct egress port. Because the PFC can operate at Layer 3, it has the capability to switch the packet between VLANs without losing the Layer 2 CoS markings, so the frame retains its CoS marking, and the packet retains its DSCP or IP precedence marking.

In the event that the PFC does not have the path in cache, the packet is forwarded to the MSFC to determine which Ethernet egress port to use. When the MFSC makes a routing decision, and determines the correct Ethernet egress port needed to reach the destination address, the PFC populates its cache with this information for future use. This means the first packet in a flow loses the CoS value, because the first packet is routed at Layer 3 across the MSFC. However, subsequent packets are switched by the PFC and retain their original marked values. After the egress Ethernet port has been identified, the packet is assigned to the proper queue and threshold indicated by the Layer 2 or Layer 3 markings on the packet.

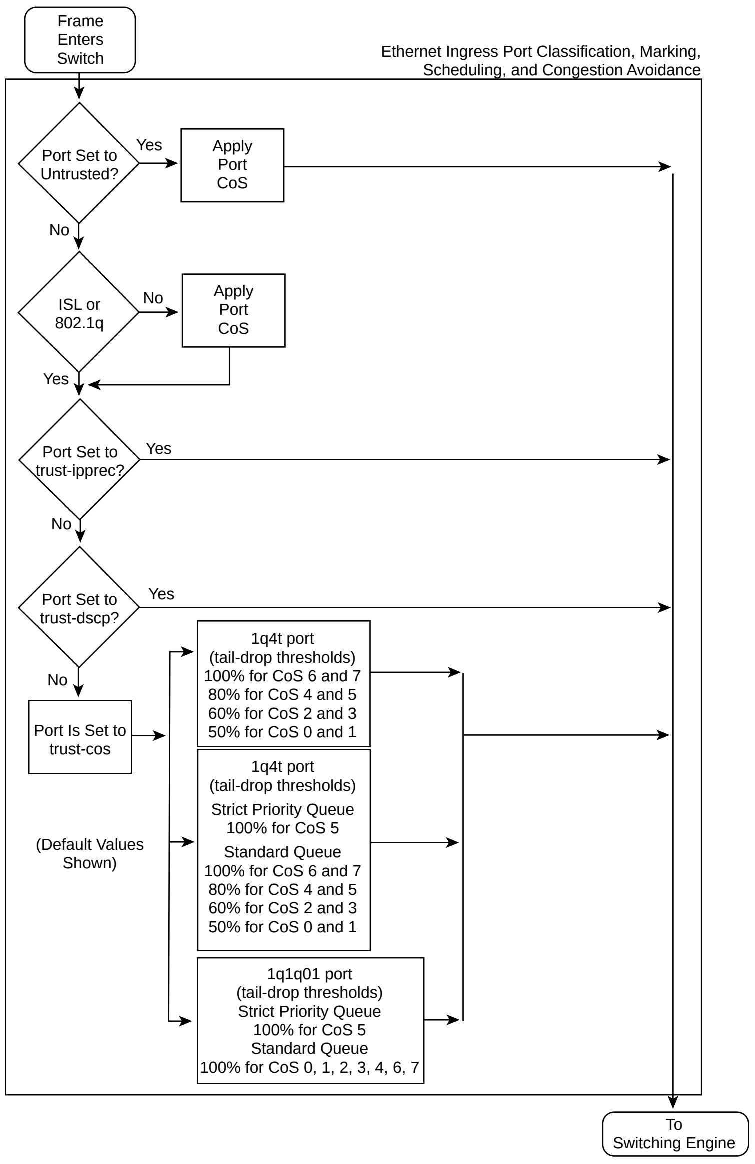
Ingress Queue Scheduling

The first opportunity for applying any kind of QoS tool to an incoming frame occurs when the frame enters the switch. Even before the supervisor or PFC forwards the frame to the egress line card, ingress QoS features occur. For instance, ingress queue scheduling defines the treatment that a frame or packet receives as it enters the switch. If QoS has been enabled, and the ingress port is trusted, the switch uses receive queue drop thresholds to schedule the arriving frames. Figure 10-7 shows the treatment received by the frame or packet in the ingress queue.

The first variable encountered as the frame enters the switch is trust. This asks the questions “Where is our trust boundary? Do we want to trust the markings on a frame as it enters the switch on this port?” If the port is untrusted, there is no need to continue to classify the frame. The ingress port just applies the configured port QoS value, typically 0, and forwards the frame to the switching engine.

If the port is trusted, the next questions asked are “Is this an 802.1Q or ISL trunk port? Does the incoming frame have an 802.1Q or ISL header?” If this ingress Ethernet port is not a trunk port, the configured port CoS value is assigned to the frame. If this ingress Ethernet port is configured as a trunk port, the CoS value marked on the frame is trusted.

Figure 10-7 *Ethernet Ingress Port Treatment*


Frame Enters Switch
Ethernet Ingress Port Classification, Marking, Scheduling, and Congestion Avoidance
Port Set to Untrusted?
Yes
Apply Port CoS
No
ISL or 802.1q
No
Apply Port CoS
Yes
Port Set to trust-ipprec?
Yes
No
Port Set to trust-dscp?
Yes
No
Port Is Set to trust-cos
1q4t port (tail-drop thresholds) 100% for CoS 6 and 7 80% for CoS 4 and 5 60% for CoS 2 and 3 50% for CoS 0 and 1
1q4t port (tail-drop thresholds) Strict Priority Queue 100% for CoS 5 Standard Queue 100% for CoS 6 and 7 80% for CoS 4 and 5 60% for CoS 2 and 3 50% for CoS 0 and 1
(Default Values Shown)
1q1q01 port (tail-drop thresholds) Strict Priority Queue 100% for CoS 5 Standard Queue 100% for CoS 0, 1, 2, 3, 4, 6, 7
To Switching Engine

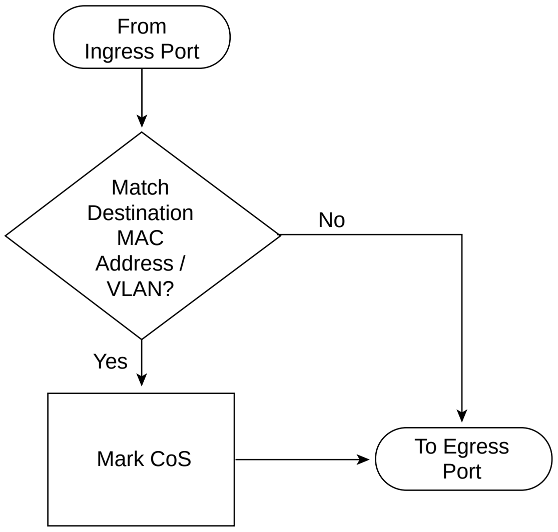
The next question depends on the hardware installed in the switch. If a PFC has been installed, the switch has the capability to classify and mark traffic based on Layer 3 markings, and so the question is asked “Do we want to trust IP precedence or DSCP from packets entering this Ethernet ingress port?” If the answer is yes, the packet is forward to the PFC. If we do not want to trust the Layer 3 markings, the switch relies on the CoS value to determine the correct queue and threshold.

After the proper queue and threshold has been identified, the question is asked “Is there room in the ingress queue for this frame, or has the drop threshold for this CoS value been met?” If the threshold has been met, the frame is dropped, otherwise the frame is forwarded on to the switching engine. If a PFC is not installed in the Catalyst switch, the frame flows to the Layer 2 switching engine, otherwise the frame is passed to the Layer 3 switching engine (PFC).

Layer 2 Switching Engine QoS Frame Flow

The Layer 2 switching engine has few options for classification and marking because it operates on the Layer 2 network. Figure 10-8 shows the options available.

Figure 10-8 *Layer 2 Switching Engine Classification and Marking*



The Layer 2 switching engine receives the frame from the ingress Ethernet port. The only variable that the Layer 2 switching engine has is the destination MAC address / VLAN match. If the arriving/received frame matches this criteria, the CoS value can be overwritten. If the received frame does not match this criteria, the frame is forwarded to the egress Ethernet port, retaining the received CoS value.

Layer 3 Switching Engine QoS Packet Flow

The PFC has the capability to classify, mark, and police traffic based on both Layer 2 and Layer 3 elements. The PFC also introduces the concept of QoS ACLs, which can be used for granular control over QoS configurations. Figure 10-9 illustrates the flow of a frame or packet through a PFC.

As the packet arrives at the PFC, the ingress interface or the ingress VLAN is examined to see whether an ACL is applied. If an ACL is not applied to either the ingress port or the ingress VLAN, the packet uses the default ACL. The default IP ACL consists of a single ACE that has a configurable marking rule and configurable policing rules. If the default ACL has not been configured, the ACE in the default ACL applies a DSCP value of 0 to all traffic passed from ingress ports configured with the untrusted port keyword. If an ACL is applied to the port or VLAN, the ACL examines the ACEs to see whether any entries match the arriving packet.

The PFC uses a DSCP setting for each IP packet that it processes. The PFC asks itself, “Do we trust the DSCP, IP precedence, or CoS values on the frame?” If these values are not trusted, the DSCP value specified in the default ACL is used. If these values are trusted, the switch uses the DSCP values received. For IP precedence and CoS values, the PFC uses an internal DSCP value generated from the IP precedence-to-DSCP or the CoS-to-DSCP mappings, respectively.

If policing has been enabled, the PFC determines the current traffic rate and decides to drop the packet, re-mark the packet, or place the packet in the egress queue unchanged.

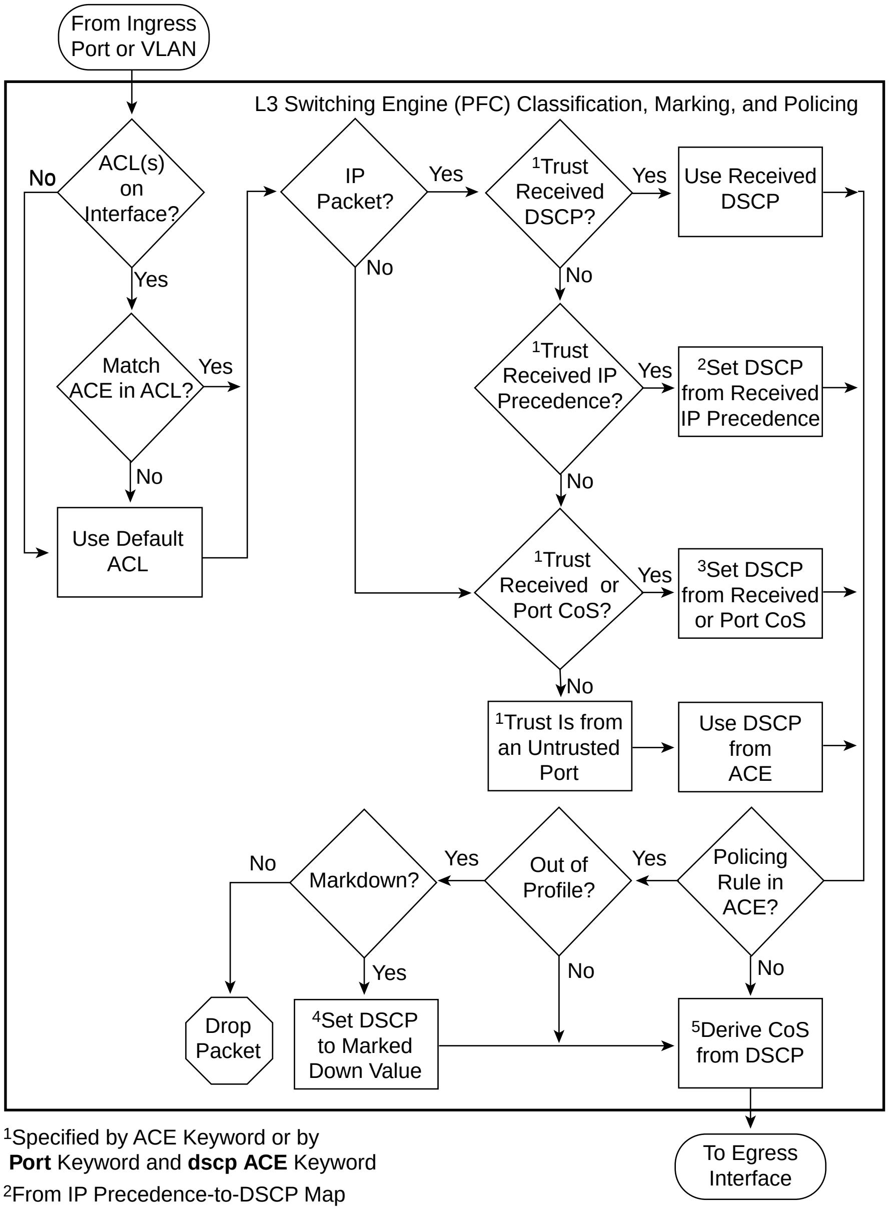
If the packet is destined for the egress queue, the CoS value is derived from the DSCP value before the packet is delivered.

Egress Queue Scheduling

Egress queue scheduling defines the treatment that a frame or packet receives as it exits the switch. Figure 10-10 shows the treatment received by the frame or packet in the egress queue.

As the frame arrives at the egress queue, the CoS value is examined to determine the proper queue and threshold for the egress interface type. In this example, the switch has three different queues that a frame can be placed in: 2q2t, 1p2q2t, and 1p3q1t. The CoS value determines the proper egress queue. After the proper queue and threshold has been identified, the question is asked “Is there room in the egress queue for this frame or has the drop threshold for this CoS value been met?” If the threshold has been met, the frame is dropped; otherwise the frame is placed in a queue to await transport.

Figure 10-9 *Layer 3 Switching Engine Classification, Marking, and Policing*


From Ingress Port or VLAN
L3 Switching Engine (PFC) Classification, Marking, and Policing
ACL(s) on Interface?
No
Yes
Match ACE in ACL?
Yes
No
Use Default ACL
IP Packet?
Yes
No
1Trust Received DSCP?
Yes
No
Use Received DSCP
1Trust Received IP Precedence?
Yes
No
2Set DSCP from Received IP Precedence
1Trust Received or Port CoS?
Yes
No
3Set DSCP from Received or Port CoS
1Trust Is from an Untrusted Port
Use DSCP from ACE
Policing Rule in ACE?
Yes
No
Out of Profile?
Yes
No
Markdown?
Yes
No
Drop Packet
4Set DSCP to Marked Down Value
5Derive CoS from DSCP
To Egress Interface

[1]Specified by ACE Keyword or by **Port** Keyword and **dscp ACE** Keyword

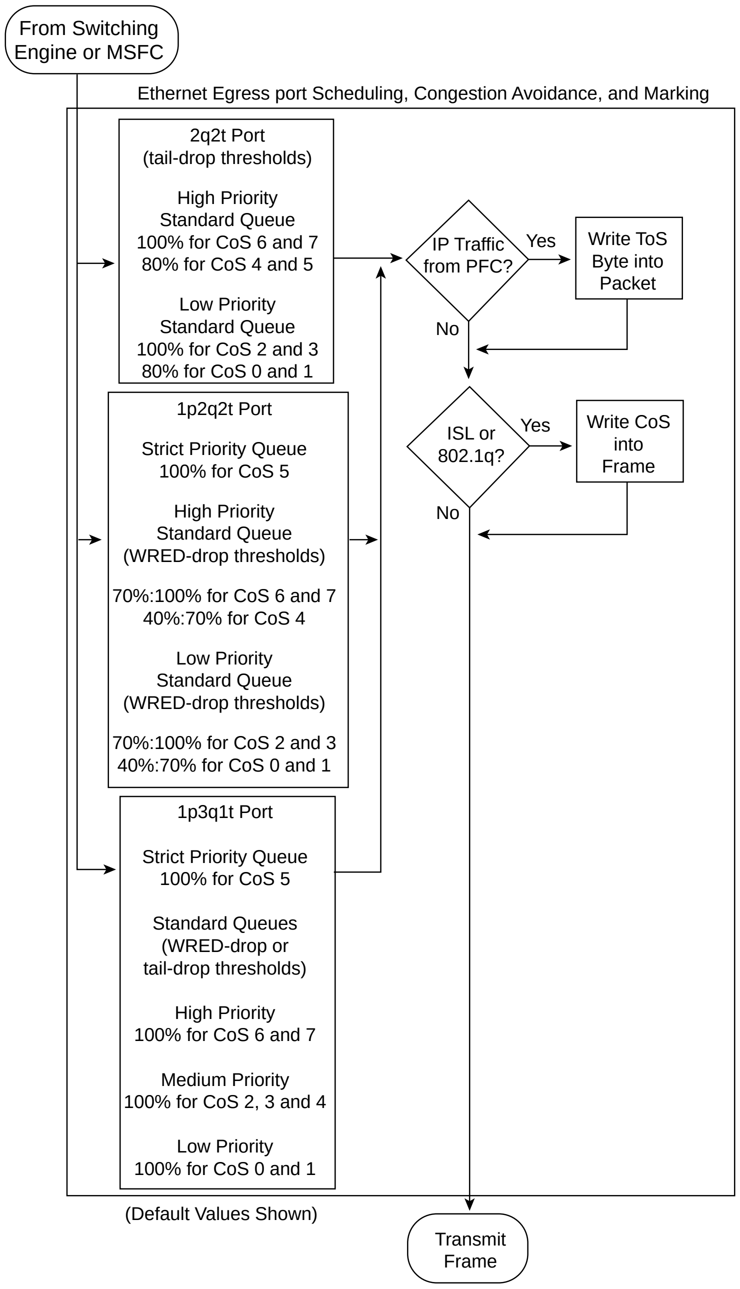
[2]From IP Precedence-to-DSCP Map

[3]From CoS-to-DSCP Map

[4]From DSCP Markdown Map

[5]From DSCP-to-CoS Map

Figure 10-10 *Ethernet Egress Port Treatment*



If the PFC has forwarded the frame into the egress queue, the IP DCSP value and CoS value is set as the frame is placed in the proper transmit queue. If the frame is an 802.1Q or ISL trunk frame, only the CoS value is set as the frame is placed in the proper transmit queue.

Catalyst 6500 QoS Summary

The numerous hardware options available on the Catalyst 6500 offer a wide range of customization. This section summarizes the impact of the hardware choices on the QoS performance of a Catalyst 6500 series switch.

Table 10-8 summarizes the QoS processes of the Catalyst 6500 and lists the component responsible for the function.

Table 10-8 *QoS Processess of the Catalyst 6500*

QoS Process	Catalyst 6500 Component That Performs This Function
Input scheduling	Performed by port application-specific integrated circuits (ASICs) Layer 2 only with or without PFC
Classification	Performed by supervisor or PFC L2 only is done by supervisor L2/3 is done by PFC
Policing	Performed by PFC via L3 forwarding engine
Packet rewrite	Performed by port ASICs L2/L3 based on classification
Output scheduling	Performed by port ASICs L2/L3 based on classification

Table 10-9 describes the supported receive queues of the Catalyst 6500.

Table 10-9 *Supported Receive Queues of the Catalyst 6500*

Catalyst 6500 Receive Queues	Description
1q4t	1 standard queue with 4 tail-drop thresholds
1p1q4t	1 strict-priority queue, 1 standard queue with 4 tail-drop thresholds
1p1q0t	1 strict-priority queue, 1 standard queue with one nonconfigurable (100%) tail-drop threshold
1p1q8t	1 strict-priority queue, 1 standard queue with 8 configurable WRED-drop thresholds and 1 nonconfigurable (100%) tail-drop threshold

Table 10-10 describes the supported transmit queues of the Catalyst 6500.

Table 10-10 *Supported Transmit Queues of the Catalyst 6500*

Catalyst 6500 Transmit Queues	Description
2q2t	2 standard queues with 2 tail-drop thresholds per queue
1p2q2t	1 strict-priority queue, 2 standard queues with 2 WRED-drop thresholds per queue
1p3q1t	1 strict-priority queue, 3 standard queues with 1 WRED-drop threshold and 1 nonconfigurable tail-drop threshold per queue
1p2q1t	1 strict-priority queue, 2 standard queues with 1 WRED-drop threshold and 1 nonconfigurable (100%) tail-drop threshold per queue

Table 10-11 lists the supported receive and transmit queues of the Ethernet modules for the Catalyst 6500.

Table 10-11 *Supported Receive and Transmit Queues of the Ethernet Modules for the Catalyst 6500*

Ethernet Modules	Module Description	RX Queue Type	TX Queue Type	RX Queue Size	TX Queue Size
WS-X6K-S2U-MSFC2 WS-X6K-S2-MSFC2 WS-X6K-S2-PFC2	All Supervisor Engine II uplink ports	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6K-S1A-MSFC2 WS-X6K-SUP1A-MSFC WS-X6K-SUP1A-PFC WS-X6K-SUP1A-2GE	All Supervisor Engine IA uplink ports	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6K-SUP1-2GE	Supervisor Engine II uplink ports	1q4t	2q2t	Q1: 80 KB	Q2: 80 KB Q1: 352 KB
WS-X6524-100FX-MM	24-port 100BASE-FX fabric-enabled with MT-RJ connectors	1p1q0t	1p3q1t	PQ: 6 KB Q1: 22 KB	PQ: 272 KB Q3: 272 KB Q2: 272 KB Q1: 272 KB
WS-X6548-RJ-21	48-port 10/100BASE-TX fabric-enabled with RJ-21 connectors	1p1q0t	1p3q1t	PQ: 6 KB Q1: 22 KB	PQ: 272 KB Q3: 272 KB Q2: 272 KB Q1: 272 KB

Table 10-11 *Supported Receive and Transmit Queues of the Ethernet Modules for the Catalyst 6500 (Continued)*

Ethernet Modules	Module Description	RX Queue Type	TX Queue Type	RX Queue Size	TX Queue Size
WS-X6548-RJ-45	48-port 10/100BASE-TX fabric-enabled with RJ-45 connectors	1p1q0t	1p3q1t	PQ: 6 KB Q1: 22 KB	PQ: 272 KB Q3: 272 KB Q2: 272 KB Q1: 272 KB
WS-X6324-100FX-MM	24-port 100BASE-FX with MT-RJ connectors	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6324-100FX-SM	24-port 100BASE-FX with MT-RJ connectors	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6348-RJ-45	48-port 10/100BASE-TX with RJ-45 connectors	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6348-RJ21V	48-port 10/100BASE-TX with RJ-21 connectors and inline power	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6348-RJ-45V	48-port 10/100BASE-TX with RJ-45 connectors and inline power	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6224-100FX-MT	24-port 100BASE-FX with MT-RJ connectors	1q4t	2q2t	Q1: 8 KB	Q2: 16 KB Q1: 40 KB
WS-X6248-RJ-45	48-port 10/100BASE-TX with RJ-45 connectors	1q4t	2q2t	Q1: 8 KB	Q2: 16 KB Q1: 40 KB
WS-X6248-TEL	48-port 10/100BASE-TX with RJ-21 connectors	1q4t	2q2t	Q1: 8 KB	Q2: 16 KB Q1: 40 KB

continues

Table 10-11 *Supported Receive and Transmit Queues of the Ethernet Modules for the Catalyst 6500 (Continued)*

Ethernet Modules	Module Description	RX Queue Type	TX Queue Type	RX Queue Size	TX Queue Size
WS-X6248A-TEL	48-port 10/100BASE-TX with RJ-21 connectors	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6148-RJ-45V	48-port 10/100BASE-TX with RJ-45 connectors and inline power	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6148-RJ21V	48-port 10/100BASE-TX with RJ-21 connectors and inline power	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6024-10FL-MT	24-port 10BASE-FL with MT-RJ connectors	1q4t	2q2t	Q1: 8 KB	Q2: 16 KB Q1: 40 KB
WS-X6816-GBIC	16-port 1000BASE-X dual-fabric with GBIC connectors and onboard DFC	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6516-GBIC	16-port 1000BASE-X with GBIC connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6516-GE-TX	16-port 10/100/ 100BASE-T with RJ-45 connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6408-GBIC	8-port 1000BASE-X with GBIC connectors	1q4t	2q2t	Q1: 80 KB	Q2: 80 KB Q1: 352 KB
WS-X6408A-GBIC	8-port 1000BASE-X with GBIC connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB

Table 10-11 *Supported Receive and Transmit Queues of the Ethernet Modules for the Catalyst 6500 (Continued)*

Ethernet Modules	Module Description	RX Queue Type	TX Queue Type	RX Queue Size	TX Queue Size
WS-X6416-GBIC	16-port 1000BASE-X with GBIC connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6416-GE-MT	16-port 1000BASE-SX with MT-RJ connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6316-GE-TX	16-port 1000BASE-T with RJ-45 connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6502-10GE	1-port 10 Gigabit Ethernet with OIM connectors	1p1q8t	1p2q1t	51 KB 205 KB	PQ: 15.3 MB Q2: 17.9 MB Q1: 30.7 MB
WS-X6501-10GEX4	1-port 10 Gigabit Ethernet with SC connectors	1p1q8t	1p2q1t	51 KB 205 KB	PQ: 15.3 MB Q2: 17.9 MB Q1: 30.7 MB
OSM	All optical services modules (Layer 2 Gigabit Ethernet ports only)	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB

Cisco Catalyst 4500/4000 QoS Features

The Catalyst 4500 and 4000 series switches are modular and scalable, making these switches an excellent choice for wiring-closet installations, branch offices, Layer 3 distribution, or Layer 3 core installations for small and medium-sized networks. The modular design enables you to configure the switch based on the needs of your network. The addition of inline power positions the Catalyst 4500 and 4000 series switch as a natural choice for supporting devices that can take advantage of this feature, such as IP telephony and wireless access points.

QoS configuration on a Catalyst 4500 or Catalyst 4000 series switch depends on the supervisor engine module installed in the switch. Four supervisor engine modules are currently available for the Catalyst 4500 and Catalyst 4000 series of switches. Table 10-12 lists the 4500/4000 Catalyst switches that support each supervisor engine model.

Table 10-12 *Catalyst 4500/4000 Supervisor Engine Matrix*

Catalyst Switches That Support the Supervisor Engine I	Catalyst Switches That Support the Supervisor Engine II	Catalyst Switches That Support the Supervisor Engine III	Catalyst Switches That Support the Supervisor Engine IV
Catalyst 4003	Catalyst 4506	Catalyst 4506	Catalyst 4507
	Catalyst 4503	Catalyst 4503	Catalyst 4506
	Catalyst 4006	Catalyst 4006	Catalyst 4503
			Catalyst 4006

Supervisor Engine I and II

The Supervisor Engine I and Supervisor Engine II modules use the Cisco Catalyst operating system.

With the Supervisor Engine I and Supervisor Engine II module, the Catalyst 4500 and 4000 series switches have a single receive queue serviced in a FIFO order. The transmit queue consists of two standard queues with a single drop threshold (2q1t). The two standard queues are scheduled on a round-robin basis. Admission to the queues is based on the 802.1p CoS value of the frame; however, the Catalyst switch with the Supervisor Engine I or the Supervisor Engine II module only examines 2 of the 3 CoS bits to determine the CoS value of a frame. This limitation means that CoS values must be mapped in pairs. To map a CoS value of 5 to the second queue, for instance, both CoS values of 4 and 5 must be mapped to Queue 2.

When QoS is enabled globally, but without any other configuration, CoS values 0 though 7 are assigned to Queue 1, whereas broadcast and multicast traffic is assigned to Queue 2. This can lead to performance degradation, because most user traffic (unicast traffic) ends up in a single queue. Therefore, it is useful to configure the switch to put some unicast traffic in the second queue. (The configuration required to remap CoS values to the second queue is discussed in the "CoS-to-Egress Queue Mapping for the Catalyst OS Switch" section of this chapter.) Table 10-13 lists the default queue assignment for the Catalyst 4500/4000 with a Supervisor I or II after QoS has been enabled.

Table 10-13 *Catalyst 4500/4000 Supervisor I or II Default Queue Admission*

Classification	Queue
CoS values 0 through 7	1
Multicast and broadcast traffic	2

Trust on a Catalyst 4500/4000 Supervisor I or II is based on the QoS state. If QoS is enabled, the switch trusts all CoS values received on all ports.

Table 10-14 lists the available QoS features of a Catalyst 4500/4000 with a Supervisor II Engine.

Table 10-14 *Catalyst 4500/4000 Supervisor II QoS Features*

QoS Feature	Setting on 4000/4500 with Sup II
QoS Feature	Setting on 4000/4500 with Sup II
TX queue	2q1t
Scheduling	Round-robin
Trust	Switch wide
Extend trust	Not available
Classification	CoS
Operating system	Catalyst OS

Supervisor Engine III and IV

The Supervisor Engine III and Supervisor Engine IV modules use the Cisco IOS operating system.

The Supervisor Engine III and the Supervisor Engine IV modules introduce many enhanced QoS features to the Catalyst 4000 and Catalyst 4500 series switches. The transmit queue on both the Supervisor Engine III module and the Supervisor Engine IV module includes one priority queue and three standard queues with a single threshold (1p3q1t). By default, voice bearer traffic (typically marked as CoS 5 or DSCP 46) is mapped to the strict-priority queue, which resides in Queue 3. The standard queues are serviced on a WRR basis.

Admission to the queues is based on internal IP DSCP values. The Supervisor Engine III and the Supervisor Engine IV modules support a global mapping of these internal DSCP values—determined by input packet QoS state, port trust configuration, and policing—to output queues. Table 10-15 lists the default queue assignment for the Catalyst 4500/4000 with a Supervisor III or IV.

Table 10-15 *Catalyst 4500/4000 Supervisor III or IV Default Queue Admission*

Classification	Queue
DSCP 0–15 CoS 0–1	1
DSCP 16–31 CoS 2–3	2

continues

Table 10-15 *Catalyst 4500/4000 Supervisor III or IV Default Queue Admission (Continued)*

Classification	Queue
DSCP 32–47 CoS 4–5	3
DSCP 48–63 CoS 6–7	4

By default, all ports on a catalyst 4500/4000 are in the untrusted state after QoS has been enabled. Trust is enabled on a per-port basis, allowing more granular QoS control over the Supervisor I or II Engines. The Supervisor III and IV also add the capability to extend trust to the ASICs in the IP Phones, allowing the switch to trust the IP Phones, without having to trust the attached PC.

Because the Catalyst 4500/4000 Supervisor III and IV use the Cisco IOS operating system, they have the capability to classify traffic based on standard and extended IOS ACLs. This allows the switch to define traffic flows based on characteristics other than CoS, IP DSCP, or IP precedence, such as TCP port number, source address, and destination address.

The Supervisor Engine IV introduces an additional QoS feature called *dynamic buffer limiting* (DBL). DBL manages misbehaving traffic flows by tracking the buffering for each traffic flow and limiting the flow if excessive demands are placed on a buffer. A misbehaving flow is defined as a flow that uses all available bandwidth, consumes switch buffers, fills output queues, and does not respond to congestion feedback such as Random Early Detection (RED). A flow is defined as a source IP address, destination IP address, IP protocol, Layer 4 Transmission Control Protocol/User Datagram Protocol (TCP/UDP) ports, and VLAN. If the buffer usage of a flow exceeds a dynamically computed buffer limit, DBL constrains the flow's buffering until the flow reaches a lower threshold. DBL is implemented at wire speed on all ports in the Catalyst family with Supervisor Engine IV.

The Supervisor Engine III and Supervisor Engine IV modules offer more granular control of QoS than their predecessors, which makes the newer modules a preferred choice for networks that carry real-time applications. Table 10-16 lists the available QoS features of a Catalyst 4500/4000 with a Supervisor III of IV engine.

Table 10-16 *Catalyst 4500/4000 Supervisor III or IV QoS Features*

QoS Feature	Setting on 4000/4500 with Sup III or IV
TX queue	1p3q1t
Scheduling	1 priority queue 3 WWR Queues
Trust	Per port
Extend trust	Per port to IP Phone

Table 10-16 *Catalyst 4500/4000 Supervisor III or IV QoS Features (Continued)*

QoS Feature	Setting on 4000/4500 with Sup III or IV
Classification	CoS IP precendence IP DSCP ACLs
Operating system	IOS

Cisco Catalyst 3550 QoS Features

The Catalyst 3550 family of switches supports enhanced QoS features that can be used in the access and distribution layers. The transmit queue on the 3550 series of switches includes one priority queue, which resides in Queue 4, and three standard queues with a single threshold (1p3q1t). Scheduling is done on a WRR basis, where each queue is given a relative weight, while the priority queue is serviced exhaustively. The default is WRR only. If priority scheduling is required, it must be explicitly enabled. Admission to the queues is based on IP DSCP, 802.1p CoS, or port-priority CoS. Table 10-17 describes the default queue admission criteria.

Table 10-17 *Catalyst 3550 Default Queue Admission*

Classification	Queue
DSCP 0–15 CoS 0–1	1
DSCP 16–31 CoS 2–3	2
DSCP 32–47 CoS 4–5	3
DSCP 48–63 CoS 6–7	4

Similar to the Catalyst 4500/4000 Supervisor III or IV, all ports on the Catalyst 3550 are in the untrusted state after QoS has been enabled. Trust is enabled on a per-port basis. The Catalyst 3550 also has the capability to extend trust to the ASICs in the IP Phones, allowing the switch to trust the IP Phones without having to trust the attached PC.

The Catalyst 3550 also has the capability to classify and mark traffic on ingress to the network using standard and extended IOS ACLs. The Catalyst 3550's capability to identify traffic flows at Layer 3 and Layer 4 using ACLs makes it very powerful as an access layer switch.

Table 10-18 lists the available QoS features of a Catalyst 3550.

Table 10-18 *Catalyst 3550 QoS Features*

QoS Feature	Setting on 3550
TX queue	1p3q1t
Scheduling	1 priority queue 3 WWR queues
Trust	Per port
Extend trust	Per port to IP Phone
Classification	CoS IP precendence IP DSCP ACLs
Operating system	IOS

Cisco Catalyst 3524 QoS Features

The available inline power, ease of configuration, and priority queuing capabilities have made the Catalyst 3524 switch a popular Layer 2 switch for IP telephony deployments. Although the 3524 has been replaced with the 3550-24PWR for future deployments, the number of deployed 3524 switches warrants a brief discussion of its QoS features.

QoS is enabled by default on the Catalyst 3524 switch. The transmit queue on the Catalyst 3524 series of switches includes one priority queue and one standard queue (1p1q). By default, voice bearer traffic (typically marked as CoS 5) is mapped to the strict-priority queue, which resides in Queue 2. Table 10-19 describes the default queue admission criteria.

Table 10-19 *Catalyst 3524 Default Queue Admission*

CoS Value	Queue
0–3	1
4–7	2

The 3524 trusts the CoS values received on each port by default and places the traffic in the appropriate queue based on the CoS value received. However, the Catalyst 3524 does offer the capability to extend the trust to the ASICs on IP Phones, thereby trusting the IP Phones without having to trust the attached PC.

Table 10-20 lists the available QoS features of a Catalyst 3524.

Table 10-20 *Catalyst 3524 QoS Features*

QoS Feature	Setting on 3524
TX queue	1p1q
Scheduling	1 priority queue 3 WWR queues
Trust	Switch wide
Extend trust	Per port to IP Phone
Classification	CoS
Operating system	IOS

QoS Configurations on Catalyst Switches

Several factors determine the configuration of a Cisco Catalyst switch. It is important to fully understand the QoS requirements that are needed to support the applications on your network before adding real-time applications such as voice and video. Ask yourself the following questions:

- What are the QoS requirements for the LAN running the current applications?
- What are the QoS requirements of the LAN if a real-time application is added?
- How many queues are present on the switches that comprise the LAN?
- Can the access layer switches support the QoS mechanisms that are needed to support the addition of real-time applications?
- Can the distribution layer switches support the QoS mechanisms that are needed to support the addition of real-time applications?
- Do the LAN switches support a priority queue?
- Can the LAN switches classify traffic on Layer 3 markings?
- Will the LAN need to be upgraded to handle the addition of real-time applications?
- Do the LAN switches use Catalyst OS or IOS operating system?

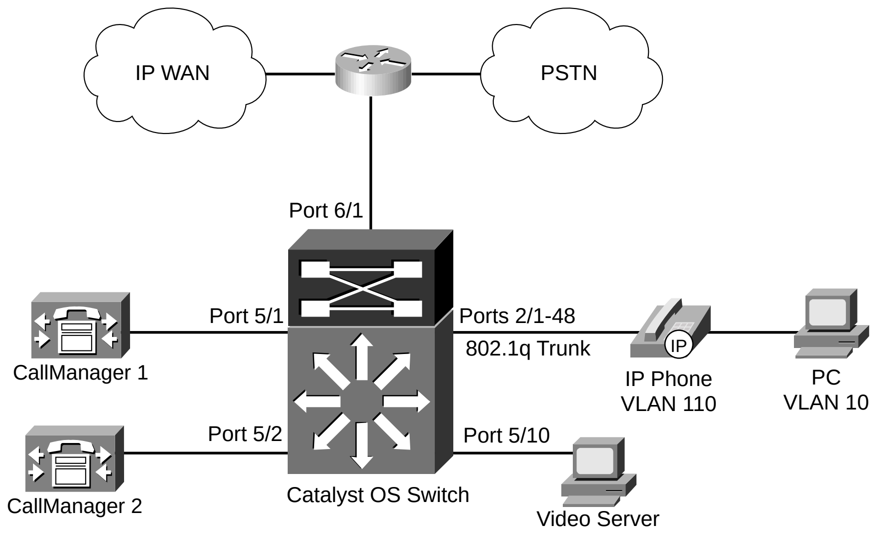
After you understand the QoS LAN requirements, you can begin to examine the configuration steps necessary to meet the requirements. The following configuration sections describe QoS configurations on Catalyst OS and IOS switches in both the distribution and access layer of your network. These configurations are based on Cisco QoS Solution Reference Network Design (SRND), which you can find on the Cisco website at the following link:

www.cisco.com/warp/customer/771/srnd/qos_srnd.pdf

Configuration of a Catalyst Switch Using Catalyst OS

This section explores the configuration of a Catalyst OS switch in the access and distribution layers. Assume that your IP telephony network is configured as depicted in Figure 10-11.

Figure 10-11 *Catalyst OS Switch*



In this example, CallManager 1 is connected to port 5/1, CallManager 2 is connected to port 5/2, and 48 IP Phones are connected to ports 2/1 through 2/48, respectively. Each IP Phone has a PC attached to the IP Phone switch port. An IP video server is also connected to port 5/10.

The Catalyst 6500 and 4000 with Supervisor I or II are examples of Catalyst OS switches. The QoS required to minimize delay, jitter, and packet drops for these switches can be broken down into the following tasks:

- Configuring auxiliary VLANs for a Catalyst OS switch
- Configuring QOS for the Catalyst OS switch
- CoS-to-egress queue mapping for the Catalyst OS switch
- Layer 2-to-Layer 3 mapping
- Configuring trust boundaries for a Catalyst OS switch
- Configuring untagged frames for the Catalyst OS switch
- Configuring QoS access lists in the Catalyst OS switch
- Connecting a Catalyst OS switch to WAN segments

Configuring Auxiliary VLANs for a Catalyst OS Switch

The first step in configuring QoS for the access layer is separating the voice traffic from the data traffic. Cisco IP Phones have the capability to use 802.1Q trunks to accomplish this task. The IP Phone can tag the voice traffic with a VLAN identifier, while leaving the data traffic in the native, or untagged, VLAN. The switch must be configured to participate in the 802.1Q trunk from the IP Phone. This is achieved with the **set port auxiliaryvlan** command. Example 10-7 shows the auxiliary VLAN of 110 being set for ports 2/1 through 2/48.

Example 10-7 *Creating Auxiliary VLANs*

```
CatOS> (enable) set port auxiliaryvlan 2/1-48 110

AuxiliaryVlan    Status    Mod/Ports
110             active     2/1-48
CatOS> (enable)
```

In this example, switch ports 2/1 through 2/48 have been configured to participate in the 802.1Q trunk from the phones using VLAN 110 for the tagged VLAN that will carry the voice traffic.

Configuring QoS for the Catalyst OS Switch

QoS must be enabled globally within the switch to use the multiple queues desired. This is achieved with the **set qos enable** command, as shown in Example 10-8.

Example 10-8 *Enabling QoS*

```
CatOS> (enable) set qos enable
QoS is enabled.
CatOS> (enable)
```

In this example, QoS has been enabled on the Catalyst OS switch.

After QoS has been enabled, it can be configured on a specific port or on a per-VLAN basis. Configuring QoS on a port applies QoS to only the specified port, whereas configuring QoS on a VLAN applies the QoS configuration to all ports that reside in that VLAN.

The **set port qos** command is used to associate QoS configuration with a specific port or a VLAN. Example 10-9 assigns VLAN QoS configuration to ports 2/1 through 2/48 based on VLAN membership.

Example 10-9 *Associating QoS with a VLAN*

```
CatOS> (enable) set port qos 2/1-48 vlan-based
Hardware programming in progress...
QoS interface is set to vlan-based for ports 2/1-48.
CatOS> (enable)
```

In Figure 10-11, all IP Phones reside on ports 2/1 through 2/48. The **set port qos 2/1-48 vlan-based** command associates all QoS configuration for these ports on a per-VLAN basis. Any previous QoS configuration attached to the specific ports are removed.

Enabling QoS based on VLAN membership works well for IP telephony because IP Phones typically share the same VLAN membership. However, what if you need to enable QoS for a separate device that does not share the same VLAN, yet needs to have QoS enabled, such as a video server in port 5/10? Example 10-10 assigns QoS configuration to the specific port 5/10.

Example 10-10 *Associating QoS with a Port*

```
CatOS> (enable) set port qos 5/10 port-based
Qos interface is set to port-based for ports 5/10
CatOS> (enable)
```

CoS-to-Egress Queue Mapping for the Catalyst OS Switch

By default, Cisco IP Phones mark voice bearer traffic with a CoS of 5 and voice signaling traffic with a CoS value of 3. The Catalyst 6500 places traffic marked with a CoS value of 5 into the priority queue on 1p2q2t interfaces and Queue 2 on 2q2t interfaces when QoS is enabled. However, you must perform the additional step of configuring the Catalyst 6500 CoS queue mapping to ensure that traffic with a CoS of 3 is placed into the second queue.

Example 10-11 maps call control traffic, a CoS value of 3, to Queue 2 threshold 1 on a 2q2t interface.

Example 10-11 *Map CoS Value of 3 to Queue 2 Threshold 1 for 2q2t*

```
Cat65k> (enable) set qos map 2q2t tx 2 1 cos 3
QoS tx priority queue and threshold mapped to cos successfully.
Cat65k> (enable)
```

Example 10-12 shows call control traffic, a CoS value of 3, to Queue 2 threshold 1 on a 1p2q2t interface.

Example 10-12 *Map CoS Value of 3 to Queue 2 Threshold 1 for 1p2q2t*

```
Cat65k> (enable) set qos map 1p2q2t tx 2 1 cos 3
QoS tx priority queue and threshold mapped to cos successfully.
Cat65k> (enable)
```

Unlike the Catalyst 6500, the Catalyst 4000 places all packets, regardless of CoS values, in the first queue after QoS has been enabled. Also the Catalyst 4000 must map CoS values to a queue in pairs. In Example 10-13, CoS values of 0 and 1 have been mapped to Queue 1, whereas CoS values of 2 through 7 have been mapped to Queue 2.

Example 10-13 *Mapping CoS Values 2 Through & to Queue 2 Threshold 1*

```
Cat4k> (enable) set qos map 2q1t 1 1 cos 0-1
Cat4k> (enable) set qos map 2q1t 2 1 cos 2-3
Cat4k> (enable) set qos map 2q1t 2 1 cos 4-5
Cat4k> (enable) set qos map 2q1t 2 1 cos 6-
```

Layer-2-to-Layer 3 Mapping

Cisco follows the recommended standard for setting the DSCP classification values for both the VoIP call control traffic and VoIP bearer traffic. The recommended settings are DSCP = AF31 (or decimal 26) for VoIP call control and DSCP = EF (or decimal 46) for VoIP bearer traffic. By default, the CoS-to-DSCP mapping does not match this recommendation, as shown in Table 10-21.

Table 10-21 *Default CoS-to-DSCP Mapping*

CoS Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

To map the Layer 2 CoS and Layer 3 IP precedence settings correctly to these DSCP values, you must modify the default CoS/ToS-to-DSCP mappings.

Example 10-14 demonstrates the configuration required for the CoS-to-DSCP mappings.

Example 10-14 *Modifying the CoS-to-DSCP Mappings*

```
CatOS> (enable) set qos cos-dscp-map 0 8 16 26 34 46 48 56
QoS cos-dscp-map set successfully.
CatOS> (enable)
```

Example 10-15 demonstrates the configuration required for the IP precedence-to-DSCP mappings.

Example 10-15 *Modifying the IP Precedence-to-DSCP Mappings*

```
CatOS> (enable) set qos ipprec-dscp-map 0 8 16 26 32 46 48 56
QoS ipprec-dscp-map set successfully.
Console > (enable)
```

Configuring Trust Boundaries for a Catalyst OS Switch

Trust boundaries define the point in the network where CoS, IP precedence, or DSCP markings are trusted. Typically this trust is established at the ingress switch port of the access layer switch. After QoS has been enabled, all ports on a Catalyst 6500 are in the untrusted state. This means that any CoS value received by the switch is re-marked with a value of 0. By contrast, after QoS has been enabled on a Catalyst 4000 switch, all ports are in a trusted state, meaning that all CoS values received are trusted on every port. Therefore, the rest of this section discussing trust excludes the Catalyst 4000.

Example 10-16 demonstrates the configuration required to instruct the Catalyst 6500 access layer switch to trust the CoS values received on the 802.1Q trunks that connect to the IP Phones.

Example 10-16 *Enabling Trust of CoS*

```
CatOS> (enable) set port qos 2/1-48 trust trust-cos
Trust type trust-cos not supported on port(s) 2/1-48
Receive thresholds enabled on ports(s) 2/1-48
Trust type set to untrusted on port(s) 2/1-48
CatOS> (enable)
```

On all 63*xx* series 10/100 Ethernet line cards with 1q4t ports, the **trust-cos** port keyword displays an error message stating that **trust-cos** is not supported. This configuration must be entered to activate the receive queue drop thresholds; however, the trust state of the ports will remain untrusted. You must configure a **trust-cos** ACL to match the ingress traffic to successfully apply the **trust-cos** trust state. This ACL is discussed in the "Configuring QoS Access Lists on a Catalyst OS Switch" section. The 65*xx* series 10/100 Ethernet line cards do not have this issue.

A Layer 3 6500 switch offers the capability to classify traffic flows based on IP header information, such as IP precedence and DSCP markings. This can prove useful for end devices that do not have the capability to use 802.1Q or ISL trunks, but can manipulate the IP header fields. In Figure 10-11, CallManager 1 is connected to port 5/1, and CallManager 2 is connected to port 5/2. Either of the following configurations shown in Examples 10-17 or 10-18 can be added to the switch to prioritize CallManager traffic using IP hearer information.

Example 10-17 *Enabling Trust of DSCP*

```
CatOS> (enable) set port qos 5/1-2 trust trust-dscp
Port  5/1-2 qos set to trust-dscp.
CatOS> (enable)
```

Example 10-18 *Enabling Trust of IP Precedence*

```
CatOS> (enable) set port qos 5/1-2 trust trust-ipprec
Port  5/1-2 qos set to trust-ipprec.
CatOS> (enable)
```

The **trust-ipprec** and **trust-dscp** keywords are supported only with a Layer 3 switching engine.

Although the **trust** command determines whether the received markings are trusted on the ingress switch port, the **trust-ext** command has little to do with the ingress switch port. The **trust-ext** command determines how the ASIC on the IP Phone handles the CoS value presented by the attached PC. If the **trust-ext** command is trusted, the original CoS value presented by the PC to the phone is passed to the switch intact. If the **trust-ext** command is untrusted, the CoS value presented by the PC is rewritten to a value of 0.

In a typical IP telephony deployment, trust should not be extended to a PC. Be cautious if your design includes extending trust to a PC. If you trust one application on a PC, by default you trust all applications on that PC. This can lead to unintentional voice-quality degradation. Example 10-19 shows the **trust-ext** configuration of a typical IP telephony deployment.

Example 10-19 *Configuring* **trust-ext**

```
CatOS> (enable) set port qos 2/1-48 trust-ext untrusted
Qos trust extension for ports 2/1-48 set to untrusted.
CatOS> (enable)
```

Configuring Untagged Frames for the Catalyst OS Switch

What if the end device does not support 802.1Q or ISL trunks? How do you prioritize the traffic of this device? The addition of a Layer 3 switching engine allows the Catalyst to use IP precedence or DSCP to classify this traffic, but what if you do not have a Layer 3 switching engine or you would still prefer to mark the traffic with a CoS value?

You can achieve this by setting the CoS value on the ingress switch port. By doing this, you are telling the switch to mark any frame received on this port with a specific CoS value. Even though the actual received frame may not have even had an ISL or 802.1Q header, the switch can process the frame as if it had a CoS value set, based on configuration. In the sample network, for instance, CallManager 1 is connected to port 5/1, and CallManager 2 is connected to port 5/2. Example 10-20's configuration can be added to the switch to prioritize CallManager traffic by setting any inbound traffic received on ports 5/1 or 5/2 to a CoS value of 3.

Example 10-20 *Assigning a CoS Value to a Specific Port*

```
CatOS> (enable) set port qos 5/1-2 cos 3
Port 5/1-2 qos cos set to 3
CatOS> (enable)
```

Because this CoS value is assigned to the port, there is no regard for the end device connected to this port. In other words, any device connected to ports 5/1 or 5/2 receive the configured CoS value. Additionally, all traffic from that end device receive this CoS value. In this configuration, for example, web browsing from the CallManager server receives the same priority across your network as call control traffic. For this reason, it is recommended just to trust DSCP on ports connected to a CallManager, allowing the CallManager to identify the proper QoS classification

To remove the CoS value applied to a specific port, use the **clear port qos** command, as shown in Example 10-21.

Example 10-21 *Clearing a CoS Value from a Specific Port*

```
CatOS> (enable) clear port qos 5/1-2 cos
Port 5/1-2 qos cos setting cleared.
CatOS> (enable)
```

Configuring QoS Access Lists in the Catalyst OS Switch

As discussed in the "Configuring Trust Boundaries for a Catalyst OS Switch" section, the 63*xx* series line cards with 1q4t ports do not apply the **trust-cos** trust state to ingress traffic. An ACL must be configured to enable the port to trust the received CoS values. The **set qos acl** command is used to accomplish this task. Example 10-22 demonstrates the configuration of a QoS ACL.

Example 10-22 *Configuring a QoS ACL*

```
Cat65k> (enable) set qos acl ip IP-PHONES trust-cos any
IP_PHONES editbuffer modified. Use 'commit' command to apply changes.
Cat65k> (enable)
```

In this example, a QoS access list named IP-PHONES was created. The IP-PHONES QoS ACL is configured to trust the CoS value from any device.

After the ACL has been created, it must be committed to hardware. QoS ACLs reside in hardware ASICs for optimum performance. The **commit qos acl** command enables you to accomplish this task. Example 10-23 demonstrates the configuration required to commit a QoS ACL to hardware.

Example 10-23 *Committing a QoS ACL to the ASIC*

```
Cat65k> (enable) commit qos acl IP-PHONES
QoS ACL 'IP_PHONES' successfully committed.
Cat65k> (enable)
```

After the QoS ACL has been committed to hardware, you can map it to the desired VLAN. This allows the 63*xx* 10/100 line cards to get around the trust state issue. The **set qos acl map** command enables you to accomplish this task. Example 10-24 demonstrates the configuration of a QoS ACL map.

Example 10-24 *Mapping the QoS ACL to a VLAN*

```
Cat65k> (enable) set qos acl map IP_PHONES 110
ACL IP_PHONES is successfully mapped to vlan 110.
Cat65k> (enable)
```

Connecting a Catalyst OS Switch to WAN Segments

In Figure 10-11, a WAN router is connected to port 6/1. Typically, you will trust the Layer 3 markings from a WAN device. To configure QoS for a port that connects the distribution layer switch to a WAN segment, use port-based QoS with a trust state of **trust-ipprec**, as shown in Example 10-25.

Example 10-25 *Associating QoS with Port 6/1 and Enabling IP Precedence Trust State*

```
cat6k-distrib> (enable) set port qos 6/1 port-based
cat6k-distrib> (enable) set port qos 6/1 trust trust-ipprec
Trust type trust-ipprec not supported on port(s) 6/1
Receive thresholds enabled on ports(s) 6/1
```

Trust type set to untrusted on port(s) 6/1As discussed in the “Configuring Trust Boundaries” section, with all 63*xx* series 10/100 Ethernet line cards with 1q4t ports, the **trust-ipprec** port keyword displays an error message stating that **trust-ipprec** is not supported. This configuration must be entered to activate the receive queue drop thresholds; however, the trust state of the ports remains untrusted. You must configure a **trust-ipprec** ACL to match the ingress traffic to successfully apply the **trust-ipprec** trust state. The 65*xx* series 10/100 Ethernet line cards do not have this issue.

Example 10-26 creates the trust state ACL, commits the ACL to hardware, and applies the ACL to port 6/1.

Example 10-26 *Trust ACL for WAN Segments*

```
catOS> (enable) set qos acl ip ACL_TRUST-WAN trust-ipprec any
catOS> (enable) commit qos acl ACL_TRUST-WAN
catOS> (enable) set qos acl map ACL_TRUST-WAN 6/1
```

Displaying QoS Settings for the Catalyst OS Switch

Several **show** commands enable you to verify the QoS configuration of the Catalyst OS switch. The **show port qos** command provides a great deal of information about the QoS configuration of the specified port, as shown in Example 10-27.

Example 10-27 *The* **show port qos** *Command*

```
CatOS> (enable) show port qos 2/1
QoS is enabled for the switch.
QoS policy source for the switch set to local.

Port    Interface Type       Interface Type       Policy Source       Policy Source
        config                runtime              config               runtime
-----   --------------       --------------       -------------       -------------
 2/1     vlan-based            vlan-based          COPS                 local

Port   TxPort Type     RxPort Type    Trust Type   Trust Type   Def CoS    Def CoS
                                      config       runtime      config   runtime
----- ------------     ------------ ------------ ------------ -------    -------
 2/1         2q2t            1q4t       trust-cos    trust-cos*        0         0

Port    Ext-Trust      Ext-Cos
-----   ---------      -------
 2/1    untrusted           0

(*)Runtime trust type set to untrusted.

Config:
Port  ACL name                         Type
----- -------------------------------- ----
No ACL is mapped to port 2/1.

Runtime:
Port  ACL name                         Type
----- -------------------------------- ----
No ACL is mapped to port 2/1.

CatOS> (enable)
```

From the **show port qos 2/1** commands, you can see that the QoS is enabled and applied to this port on a VLAN basis. This is a result of the command **set port qos 2/1-48 vlan-based** entered earlier in the examples to assign QoS to the IP Phones connected to ports 2/1 through 2/48. The TX port type and RX port type list the queues and thresholds in use for interface 2/1. The runtime trust state is currently untrusted. As discussed in the "Trust Boundaries" section, a QoS ACL resolves this issue.

In the "Configuring QoS ACLs" section, a QoS ACL named IP-PHONES was created and associated with VLAN 110. Port 2/1 is a member of this VLAN, so why is the **show port qos 2/1** saying that there is no ACL mapped to port 2/1? The answer to this question is that the IP-PHONES ACL was mapped to the VLAN and not to port 2/1. An ACL is listed only if the ACL was mapped to the specific port selected.

As seen in Example 10-28, the **show qos info runtime** command provides much of the same information obtained with the **show port qos command**.

Example 10-28 *The* **show qos info runtime** *Command*

```
CatOS> (enable) show qos info runtime 2/1
Run time setting of QoS:
QoS is enabled
Policy Source of port 2/1: Local
Tx port type of port 2/1 : 2q2t
Rx port type of port 2/1 : 1q4t
Interface type: vlan-based
ACL attached:
The qos trust type is set to trust-cos.
Warning: Runtime trust type set to untrusted.
Default CoS = 0
Queue and Threshold Mapping for 2q2t (tx):
Queue Threshold CoS
----- --------- --------------
1     1         0 1
1     2         2
2     1         3 4 5
2     2         6 7
Queue and Threshold Mapping for 1q4t (rx):
Queue Threshold CoS
----- --------- --------------
1     1         0 1
1     2         2
1     3         3 4 5
1     4         6 7
. . .

CatOS> (enable)
```

The commands have a few notable differences. The **show qos info runtime** command lists the default CoS, typically set to 0, and the active CoS to transmit and receive queue mappings. In this example, any frame that arrives with a CoS value of 3, 4, or 5 is placed in Queue 2, threshold 1. This matches the configuration from the "CoS-to-Egress Queue Mapping" section.

The **show qos statistics l3stats** command shows a summary of all IP packets that have had their CoS/ToS fields modified by the Layer 3 switching engine as well as any packet drops resulting from policing, as shown in Example 10-29.

Example 10-29 *Show QoS Layer 3 Statistics*

```
CatOS> (enable) show qos statistics l3stats
Packets dropped due to policing:              0
IP packets with ToS changed:              10473
IP packets with CoS changed:               4871
Non-IP packets with CoS changed:              0
CatOS> (enable)
```

The **show qos statistics** command displays detailed information about the number of packets that have been dropped from each queue and threshold. In Example 10-30, all packets dropped are in the first drop threshold of TX Queue 1.

Example 10-30 *The* **show qos statistics** *Command*

```
CatOS> (enable) show qos statistics 2/1

Tx port type of port 2/1 : 2q2t
Q #     Threshold #:Packets dropped
---     -----------------------------------------------
1       1:393210 pkts, 2:0 pkts
2       1:0 pkts, 2:0 pkts

Rx port type of port 2/1 : 1q4t
Q #     Threshold #:Packets dropped
---     -----------------------------------------------
1       1:0 pkts, 2:0 pkts, 3:0 pkts, 4:0 pkts

CatOS> (enable)
```

The **show qos map runtime** command enables you to display the CoS/DSCP/IP precedence mappings within the Catalyst 6500. You can now display the mappings configured in the "Layer 2-to-Layer 3 Mapping" section of this chapter.

Example 10-31 show the CoS value of 3 is mapped to the DSCP decimal value of 26 (AF31), whereas the CoS value of 5 is mapped to the DSCP decimal value of 46 (EF).

Example 10-31 *The* **show qos map runtime cos-dscp-map** *Command*

```
Cat65k-Access> (enable) show qos map runtime cos-dscp-map
CoS - DSCP map:
CoS DSCP
--- ----
0   0
1   8
2   16
3   26 -> 26 = AF31
4   32
5   46 -> 46 = EF
6   48
7   56
Console > (enable)
```

Example 10-32 shows the DSCP decimal values 24 through 31 are mapped to the CoS value of 3, whereas the DSCP decimal values 40 through 47 are mapped to the CoS value of 5.

Example 10-32 *The* **show qos map runtime dscp-cos-map** *Command*

```
CatOS> (enable) show qos map runtime dscp-cos-map
DSCP - CoS map:
DSCP   CoS
---    ---
0-7    0
8-15   1
16-23  2
24-31  3 -> 26 = AF31
32-39  4
40-47  5 -> 46 = EF
48-55  6
56-63  7
CatOS> (enable)
```

Example 10-33 shows the IP precedence value of 3 mapped to the DSCP value of 26 (AF31), whereas the IP precedence value of 5 is mapped to the DSCP value of 46 (EF).

Example 10-33 *The* **show qos map runtime ipprec-dscp-map** *Command*

```
Cat65k-Access> (enable) show qos map runtime ipprec-dscp-map
IP-Precedence - DSCP map:
IP-Prec   DSCP
-------   ----
      0   0
      1   8
      2   16
```

continues

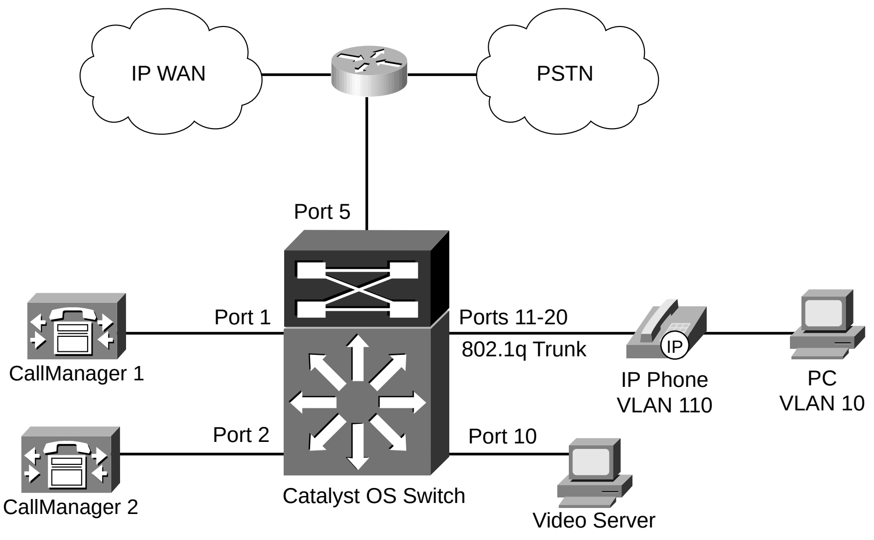
Example 10-33 *The* **show qos map runtime ipprec-dscp-map** *Command (Continued)*

```
        3   26 -> 26 = AF31
        4   32
        5   46 -> 46 = EF
        6   48
        7   56
Cat65k-Access> (enable)
```

Configuration of a Catalyst Switch Using IOS

This section explores the configuration of a Catalyst IOS switch in the access and distribution layers. Assume that your IP telephony network is configured as depicted in Figure 10-12.

Figure 10-12 *Catalyst IOS Switch*



In this example, CallManager 1 is connected to port 1, CallManager 2 is connected to port 2, and 10 IP Phones are connected to ports 11 through 20 respectively. Each IP Phone has a PC attached to the IP Phone switch port. And an IP video server is also connected to port 10.

The Catalyst 6500, 4500 with the Supervisor III or IV Engine, 3550, and 3524 are examples of Catalyst IOS switches. The QoS required to minimize delay, jitter, and packet drops for these switches can be broken down into the following tasks:

- Configuring voice VLANs for a Catalyst IOS switch

- Configuring QOS for the Catalyst IOS switch
- Enabling priority queuing for the Catalyst IOS switch
- CoS to egress queue mapping for the Catalyst IOS switch
- Layer 2-to-Layer 3 mapping
- Configuring trust boundaries for a Catalyst IOS switch
- Configuring untagged frames for the Catalyst IOS switch
- Configuring QoS access lists in the Catalyst IOS switch
- Connecting a Catalyst OS switch to WAN segments

Configuring Voice VLANs for a Catalyst IOS Switch

The first step in configuring QoS for the access layer is separating the voice traffic from the data traffic. Cisco IP Phones have the capability to use 802.1Q trunks to accomplish this task. The IP Phone can tag the voice traffic with a VLAN identifier, while leaving the data traffic in the native, or untagged, VLAN. The switch must be configured to participate in the 802.1Q trunk from the IP Phone. Assuming that the voice VLAN has been configured for 110 and the data VLAN has been configured for 10, Example 10-34 shows the configurations necessary on the Catalyst 6500, 4500, 3550, and 3524 switches.

Example 10-34 *Creating Voice VLANs*

```
CatIOS (config)#interface FastEthernet0/11
CatIOS (config)#switchport access vlan 10
CatIOS (config)#switchport voice vlan 110
CatIOS (config)#spanning-tree portfast
```

In this example, switch port 0/11, the first IP Phone port in Figure 10-12, has been configured to participate in the 802.1Q trunk from the IP Phones using VLAN 110 for the tagged VLAN that will carry the voice traffic and VLAN 10 for the untagged traffic.

Enabling QoS for the Catalyst IOS Switch

QoS must be enabled globally within the switch to use the multiple queues desired. The command to enable QoS depends on the Catalyst IOS switch. The Catalyst 6500 and 3550 both use the **mls qos** command to enable QoS, whereas the Catalyst 4500 uses the **qos** command. QoS on the catalyst 3524 is enabled by default.

In this example, QoS has been enabled on the Catalyst IOS switch.

Enabling Priority Queuing for the Catalyst IOS Switch

The Catalyst 6500, with selected line cards that support priority queuing, and 3524 transmit frames with a CoS value of 5, typically voice traffic, using the priority queue by default. Although the Catalyst 4500 and the 3550 have a transmit priority queue, they are not configured when QoS is enabled.

The priority queue on a 3550 interface resides in Queue 4. To enable this priority queue, use the **priority-queue out** interface command, as shown in Example 10-35.

Example 10-35 *Enabling the Priority Queue on a Catalyst 3550*

```
3550 (config)#interface fastethernet 0/11
3550 (config-if)#priority-queue out
3550 (config-if)#wrr-queue cos-map 4 5
```

The priority queue on a 4500 interface resides in Queue 3. Example 10-36 shoes how to configure this priority queue.

Example 10-36 *Enabling the Priority Queue on a Catalyst 4500*

```
4500 (config)# interface fastethernet 1/1
4500 (config-if-tx-queue)# tx-queue 3
4500 (config-if-tx-queue)# priority high
```

Table 10-22 summarizes the priority queues available on the Catalyst IOS switches.

Table 10-22 *Priority Queues*

Platform	Priority Queue	Configuration Command
6500	1p	On by default
4500	3	**tx-queue 3** **priority high**
3550	4	**priority-queue out**
3524	2	On by default

CoS-to-Egress Queue Mapping for the Catalyst IOS Switch

By default, Cisco IP Phones mark voice bearer traffic with a CoS of 5 and voice signaling traffic with a CoS value of 3. The Catalyst 3550 uses transmit Queue 4 for a priority queue; therefore, the CoS value of 5 needs to be mapped to Queue 4 for voice traffic to take advantage of the priority queue. Example 10-37 shows the configuration needed to map the CoS value of 5 into the priority queue, which resides in Queue 4.

Example 10-37 *Mapping CoS 5 to the Priority Queue in a Catalyst 3550*

```
3550 (config)#interface fastethernet 0/11
3550 (config-if)#priority-queue out
3550 (config-if)#wrr-queue cos-map 4 5
```

The Catalyst 6500 and 4500 place traffic marked with a CoS value of 5 into the priority queue when QoS is enabled and the priority queue has been activated. However, you must perform the additional step of configuring the Catalyst IOS switches' CoS queue mapping to ensure that traffic with a CoS of 3 is placed into the second queue on the Catalyst 6500 and 4500. Example 10-38 shows the configuration needed to place the call control traffic (CoS 3) in Queue 2, threshold 1, of a Catalyst 6500 switch. CoS 4 is also placed in the same queue.

Example 10-38 *Map CoS Value of 3 to Queue 2 Threshold 1*

```
CatIOS (config)# interface fastethernet 0/11
CatIOS (config-if)#wrr-queue cos-map 2 1 3 4
```

The Catalyst 4500 Supervisor III or IV uses a slightly different configuration for this task. From the global configuration mode, the **qos map dscp 3 to tx-queue 3** command is entered to direct frames with a CoS value of 3 to the high-priority queue, Queue 3, on a Catalyst 4500. The queue assignment on a Catalyst 3524 cannot be changed.

Layer 2-to-Layer 3 Mapping

Cisco follows the recommended standard for setting the DSCP classification values for both the VoIP call control traffic and VoIP bearer traffic. The recommended settings are DSCP = AF31 (or decimal 26) for VoIP call control and DSCP = EF (or decimal 46) for VoIP bearer traffic. By default the CoS-to-DSCP mapping does not match this recommendation, as shown in Table 10-23.

Table 10-23 *Default CoS-to-DSCP Mapping*

CoS Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

To map the Layer 2 CoS and Layer 3 IP precedence settings correctly to these DSCP values, you must modify the default CoS/ToS-to-DSCP mappings.

Example 10-39 demonstrates the configuration required for the CoS-to-DSCP mappings in a Catalyst 6500 and 3550.

Example 10-39 *Modifying the CoS-to-DSCP Mappings*

```
CatIOS (config)# mls qos map cos-dscp 0 8 16 26 32 46 48 56
```

Example 10-40 demonstrates the configuration required for the CoS-to-DSCP mappings in a Catalyst 4500.

Example 10-40 *Modifying the CoS-to-DSCP Mappings*

```
Cat4500 (config)# qos map cos 3 to dscp 26
Cat4500 (config)# qos map cos 5 to dscp 46
```

After these command have been applied, the Catalyst 6500, 4500, and 3550 map a CoS value of 3 to IP DSCP 26 (AF31) and a CoS value of 5 to IP DSPC 46 (EF). The Catalyst 3524 operates on Layer 2 tags and does not have the capability to classify or mark based on IP DSCP.

Configuring Trust Boundaries for a Catalyst IOS Switch

Trust boundaries define the point in the network where CoS, IP precedence, or DSCP markings are trusted. Typically this trust is established at the ingress switch port of the access layer switch. The Catalyst 6500, 4500, and 3550 switches establish trust on a per-port basis. By default all listed switches reside in the untrusted state. In other words, any CoS or DSCP value received by the switch is re-marked to a CoS or DSCP value of 0. In Figure 10-12, CallManager 1 is connected to the Catalyst IOS switch on port 1. Assuming that the switch is a 6500 or a 3500, to configure the switch to trust the DSCP values received on port 1, you must place the **mls qos trust dscp** command on interface 1. If the switch is a 4500, the **qos trust dscp** command is configured on interface 1. These configurations allow the DSCP values received on the configured interface to be trusted by the switch. In the example, assume that the video server in port 10 can only use CoS values and does not have the capability to classify or mark based on IP DSCP. The Catalyst 6500 and 3550 can be configured to trust the receive CoS value by placing the **mls qos trust cos** command on interface 10. The Catalyst 4500 uses the **qos trust cos** command on interface 10 to accomplish the same goal.

The **switchport priority extend cos** command enables you to overwrite the CoS value presented to an IP Phone from the attached PC. In a typical IP telephony deployment, any CoS value presented by an attached PC should be overwritten with a value of 0. Be cautious if your design includes extending trust to a PC. If you trust one application on a PC, by default, you trust all applications on that PC. This can lead to unintentional voice-quality degradation. Example 10-41 shows the **switchport priority extend cos** configuration of a typical IP telephony deployment.

Example 10-41 *Configuring* **trust-ext**

```
CatIOS (config)#interface FastEthernet0/11
CatIOS (config)#switchport access vlan 10
CatIOS (config)#switchport voice vlan 110
CatIOS (config)#switchport priority extend 0
CatIOS (config)#spanning-tree portfast
```

Configuring Untagged Frames for the Catalyst IOS Switch

What if the end device does not support 802.1Q trunks? How do you prioritize the traffic of this device? The addition of a Layer 3 switching engine allows the Catalyst to use IP precedence or DSCP to classify this traffic, but what if you do not have a Layer 3 switching engine or you would still prefer to mark the traffic with a CoS value?

You can accomplish this by setting the CoS value on the ingress switch port. By doing this, you are telling the switch to mark any frame received on this port with a specific CoS value. Even though the actual received frame may not have even had an ISL or 802.1Q header, the switch can process the frame as if it had a CoS value set, based on configuration. In Figure 10-12, for example, CallManager 1 is connected to port 1, and CallManager 2 is connected to port 2. The **mls qos cos 3** command can be added to interface 1 and 2 on a Catalyst 6500 and a 3550 to mark all inbound traffic with a CoS value of 3. A Catalyst 4500 Supervisor III or IV can use the **qos cos 3** command to accomplish the same results, whereas a Catalyst 3524 can use the **switchport priority default 3** command.

Because this CoS value is assigned to the port in this example, there is no regard for the end device connected to this port. In other words, any device connected to interface 1 or 2 receive the configured CoS value. Additionally, all traffic from that end device receive this CoS value. In this configuration, for example, web browsing from the CallManager server receives the same priority across your network as call control traffic. For this reason, it is recommended to trust IP DSCP whenever possible, allowing the CallManager to identify the proper QoS classification.

Configuring QoS Access Lists in the Catalyst IOS Switch

The Catalyst 6500, 4500, and 3550 IOS switches enable you to classify traffic based on standard or extended access lists typically deployed on routers. This allows the switches to classify traffic flows by examining Layer 3 and Layer 4 information, providing a much more granular control of QoS.

In the Example 10-42, three IP access lists (GOLD-DATA, VOICE, and VOICE-CONTROL) are created to specify interesting traffic flows.

Example 10-42 *Creating the ACL*

```
CatIOS (config)#ip access-list extended GOLD-DATA
CatIOS (config-ext-nacl)#remark Match IP Address of the application server
CatIOS (config-ext-nacl)#permit ip any host 192.168.100.1
CatIOS (config-ext-nacl)#permit ip host 192.168.100.1 any
CatIOS (config)#ip access-list extended VOICE
CatIOS (config-ext-nacl)#remark Match the UDP ports that VoIP Uses for Bearer Traffic
CatIOS (config-ext-nacl)#permit udp any any range 16384 32767
CatIOS (config)#ip access-list extended VOICE-CONTROL
CatIOS (config-ext-nacl)#remark Match VoIP Control Traffic
CatIOS (config-ext-nacl)#remark SCCP
CatIOS (config-ext-nacl)#permit tcp any any range 2000 2002
CatIOS (config-ext-nacl)#remark H323 Fast Start
CatIOS (config-ext-nacl)#permit tcp any any eq 1720
CatIOS (config-ext-nacl)#remark H323 Slow Start
CatIOS (config-ext-nacl)#permit tcp any any range 11000 11999
CatIOS (config-ext-nacl)#remark H323 MGCP
CatIOS (config-ext-nacl)#permit udp any any eq 2427
```

After the traffic flows have been specified, you use the **class-map** command to identify these traffic flows. Example 10-43 shows a **class-map** for each of the IP access lists created in Example 10-42.

Example 10-43 *Create Classes Based on the ACL*

```
CatIOS (config)#class-map match-all GOLD-DATA
CatIOS (config-cmap)#description Mission Critical Traffic
CatIOS (config-cmap)#match access-group name GOLD-DATA
CatIOS (config)#class-map match-all VOICE
CatIOS (config-cmap)#description VoIP Bearer Traffic
CatIOS (config-cmap)#match access-group name VOICE
CatIOS (config)#class-map match-all VOICE-CONTROL
CatIOS (config-cmap)#description VoIP Control Traffic (SCCP, H225, H254, MGCP)
CatIOS (config-cmap)#match access-group name VOICE-CONTROL
```

Next, the behavior of the traffic flow can be altered. In Example 10-44, the **policy-map** command is used to overwrite the DSCP value of all traffic identified in the three classes.

Example 10-44 *Set the DSCP Value for the Classes*

```
CatIOS (config)#policy-map CAT-IOS-IN
CatIOS (config-pmap)#description Set DSCP Value for Each Class
CatIOS (config-pmap)#class VOICE-CONTROL
CatIOS (config-pmap-c)#set ip dscp 26
CatIOS (config-pmap)#class VOICE
```

Example 10-44 *Set the DSCP Value for the Classes (Continued)*

```
CatIOS (config-pmap-c)#set ip dscp 46
CatIOS (config-pmap)#class GOLD-DATA
CatIOS (config-pmap-c)#set ip dscp 18
```

In this example, all traffic matching the GOLD-DATA access list is marked with a DSCP value of 18, all traffic matching the VOICE-CONTROL access list is marked with a DSCP value of 26, and all traffic matching the VOICE access list is marked with a DSCP value of 46.

Finally, the **service-policy** command is used to assign the policy map to the input traffic on a specific interface. In Example 10-45, the policy map configuration is applied to interface Gigabit Ethernet 0/2.

Example 10-45 *Applying the Policy to the Interface*

```
CatIOS (config)#int gigabitethernet 0/2
CatIOS (config-if)#service-policy input CAT-IOS-IN
```

Connecting a Catalyst IOS Switch to Distribution Switches or WAN Segments

In Figure 10-12, a WAN router is connected to port 5. Typically you will trust the Layer 3 markings from a WAN device. Assuming that the switch is a 6500 or a 3500, to configure the switch to trust the DSCP values received on port 5, you must place the **mls qos trust dscp** command on interface 5. If the switch is a 4500, the **qos trust dscp** command is configured on interface 5.

Displaying QoS Settings for the Catalyst IOS Switch

Several **show** commands enable you to verify the QoS configuration of the Catalyst OS switch. The **show mls qos interface queueing** command enables you to display the QoS queuing strategy for the Catalyst 3550 switch. In Example 10-46, egress expedited queuing (priority queuing) has been enabled on interface gigabit 0/1, causing Queue 4 to become the priority queue. All frames with a CoS value of 5, 6, or 7 are placed in this queue.

Example 10-46 *The* **show mls qos interface queueing** *Command*

```
3550 #show mls qos interface queueing
GigabitEthernet0/1
Ingress expedite queue: dis
Egress expedite queue: ena
wrr bandwidth weights:
qid-weights
 1 - 25
 2 - 25
```

continues

Example 10-46 *The* **show mls qos interface queueing** *Command*

```
 3 - 25
 4 - 25 when expedite queue is disabled
Dscp-threshold map:
d1 : d2 0  1  2  3  4  5  6  7  8  9
---------------------------------------
0 :     01 01 01 01 01 01 01 01 01 01
1 :     01 01 01 01 01 01 01 01 01 01
2 :     01 01 01 01 01 01 01 01 01 01
3 :     01 01 01 01 01 01 01 01 01 01
4 :     01 01 01 01 01 01 01 01 01 01
5 :     01 01 01 01 01 01 01 01 01 01
6 :     01 01 01 01
Cos-queue map:
cos-qid
 0 - 1
 1 - 1
 2 - 2
 3 - 2
 4 - 3
 5 - 4
 6 - 4
 7 – 4
```

Note that at the end of the command output, the CoS values' mapping to the four queues is listed. CoS 5 maps to Queue 4, the priority queue, automatically as a result of the configuration to use priority queuing.

The **show mls qos** interface command enables you to display QoS information for a specified interface. In Example 10-47, the QoS configuration of Fast Ethernet port 0/1 on a Catalyst 3550 switch displays. In the earlier example, interface Fast Ethernet 0/1 was configured to trust the DSCP values received.

Example 10-47 *The* **show mls qos interface** *Command F0/1*

```
3550 t#show mls qos interface fastethernet0/1
FastEthernet0/1
trust state: trust dscp
COS override: dis
default COS: 0
DSCP Mutation Map: Default DSCP Mutation Map
```

In Example 10-48, the QoS configuration of Fast Ethernet 0/10 on the Catalyst 3550 switch displays. In the earlier example, interface Fast Ethernet 0/10 was configured to trust the CoS values received.

Example 10-48 *The* **show mls qos interface** *Command F0/10*

```
3550 t#show mls qos int g0/10
GigabitEthernet0/10
trust state: trust cos
COS override: dis
default COS: 0
DSCP Mutation Map: Default DSCP Mutation Map
```

The **show qos interface** command enables you to display the current QoS configuration information about the specified interface in a Catalyst 4500. In Example 10-49, QoS has been enabled globally in the switch as well as on this port; however, the trust state for port 4/5 is configured for untrusted. Although QoS has been enabled, this port transmits packets without regard to the CoS or DSCP values on the packets.

Example 10-49 *The* **show qos interface** ***Command****: Untrusted*

```
4500 #show qos interface fastethernet 4/5
QoS is enabled globally
Port QoS is enabled
Port Trust State: 'untrusted'
Default DSCP: 0 Default CoS: 0
Tx-Queue       Bandwidth      ShapeRate      Priority       QueueSize
 (bps)           (bps)                      (packets)
1                N/A          disabled         N/A            240
2                N/A          disabled         N/A            240
3                N/A          disabled         high            240
4                N/A          disabled         N/A            24
```

In Example 10-50, port 3/1 on a Catalyst 4500 has been configured to trust the DSCP values of packets. Any packet that matches the DSCP classification for admittance into the priority queue is transmitted immediately.

Example 10-50 *The* **show qos interface** ***Command****: Trust DSCP*

```
4500 #show qos interface fastethernet 3/1
QoS is enabled globally
Port QoS is enabled
Port Trust State: 'dscp'
Default DSCP: 0 Default CoS: 0
Tx-Queue       Bandwidth      ShapeRate      Priority       QueueSize
 (bps)           (bps)                       (packets)
```

continues

Example 10-50 *The* **show qos interface** ***Command****: Trust DSCP (Continued)*

```
1              250000000      disabled         N/A           1920
2              250000000      disabled         N/A           1920
3              250000000      disabled         high          1920
4              250000000      disabled         N/A           1920
```

In Example 10-51, port 3/2 on a Catalyst 4500 has been configured to trust the CoS values of frames. Any frame that matches the CoS classification for admittance into the priority queue is transmitted immediately.

Example 10-51 *The* **show qos interface** ***Command****: Trust CoS*

```
4500 #show qos interface fastethernet 3/2
QoS is enabled globally
Port QoS is enabled
Port Trust State: 'cos'
Default DSCP: 0 Default CoS: 0
Tx-Queue Bandwidth ShapeRate Priority QueueSize
 (bps)   (bps) (packets)
1250000000 disabled   N/A  1920
2 250000000 disabled   N/A  1920
3 250000000 disabled   high  1920
4 250000000 disabled   N/A  1920
```

The **show mls qos map cos** command on the Catalyst 6500 and 3550 enables you to display the current CoS-to-DSCP mapping in use by the switch. The **show qos map cos** command on the Catalyst 4500 displays the same results. In Example 10-52, CoS values of 3, 4, and 5 have been mapped to the DSCP values of 26, 34, and 46, respectively.

Example 10-52 *The* **show qos map cos** *Command*

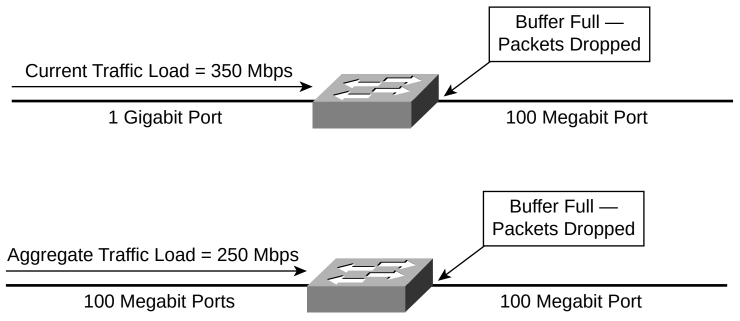
```
CatIOS #show mls qos map cos
CoS-DSCP Mapping Table
CoS:    0      1       2       3       4       5     6      7
--------------------------------------------------------------
DSCP:   0     8       16      26       34      46    48     56
```

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary could help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final preparation before the exam, these tables and figures are a convenient way to review the day before the exam.

Why is QoS important on a LAN when bandwidth is abundant? As you have learned, bandwidth is not the only important factor to consider when determining how traffic will flow across your LAN infrastructure. Figure 10-13 illustrates the concept of buffer overrun.

Figure 10-13 *Buffer Overflow*


Buffer Full — Packets Dropped
Current Traffic Load = 350 Mbps
1 Gigabit Port
100 Megabit Port
Buffer Full — Packets Dropped
Aggregate Traffic Load = 250 Mbps
100 Megabit Ports
100 Megabit Port

You must understand the behavior of the real-time applications present on your network and devise a strategy to support these real-time applications.

The first step in devising your strategy is achieving an understanding of the real-time applications that will reside on your LAN. Following are questions you should consider:

- Will voice traffic reside on your LAN?
- Will video traffic reside on your LAN?
- Will you need to prioritize specific data applications on your LAN?
- What are the bandwidth, delay, and jitter requirements for the expected applications?

After you have an understanding of the real-time applications that will be supported on your LAN, you can begin to design the network to meet your needs. For QoS support in a LAN environment, multiple queues are required on all interfaces to guarantee that loss, delay, and jitter do not affect voice, video, and mission-critical data.

Table 10-24 defines a few of the Layer 2 queues available in Cisco Catalyst switches.

Table 10-24 *Layer 2 Queues*

Layer 2 Queue	Description
1q	A single Layer 2 queue. All traffic crossing the interface flows through this queue.
2q	2 Layer 2 queues. Traffic can be directed to the desired queue based on classification.
1p1q	1 priority Layer 2 queue and 1 standard Layer 2 queue. Traffic can be directed to the priority queue based on classification. Other traffic can be directed to the standard queue.
1p2q	1 priority Layer 2 queue and 2 standard Layer 2 queues. Traffic can be directed to the priority queue based on classification. Traffic can be directed to the desired standard queues based on additional classification.

Figure 10-14 illustrates the proccess that drop thresholds follow. When the queue has reached 50 percent of capacity, any traffic classified with CoS of 0 or 1 becomes drop candidates to avoid congestion. If the queue continues to fill in spite of the drops, at 60 percent of capacity any traffic clssified with a CoS of 0, 1, 2, or 3 becomes drop candidates to avoid congestion. If the queue still continues to fill in spite of the drops, at 80 percent of capacity any traffic clssified with a CoS of 0, 1, 2, 3, 4, or 5 becomes drop candidates to avoid congestion. At 100 percent of capacity, all traffic, requardless of classification, becomes drop candidates.

Table 10-25 describes the possible supervisor and switching engine combinations on a Catalyst 6500.

Table 10-25 *Supervisor and Switching Engine Combinations*

Supervisor	Switching Engine
Supervisor II (WS-X6K-SUP2-2GE)	Layer 3 Switching Engine II (WS-F6K-PFC2—PFC2)
Supervisor Engine I (WS-X6K-SUP1A-2GE) or (WS-X6K-SUP1-2GE)	Layer 3 Switching Engine (WS-F6K-PFC)
Supervisor Engine I (WS-X6K-SUP1A-2GE) or (WS-X6K-SUP1-2GE)	Layer 2 Switching Engine II (WS-F6020A)
Supervisor Engine I (WS-X6K-SUP1A-2GE) or (WS-X6K-SUP1-2GE)	Layer 2 Switching Engine I (WS-F6020)

Figure 10-14 *Thresholds*

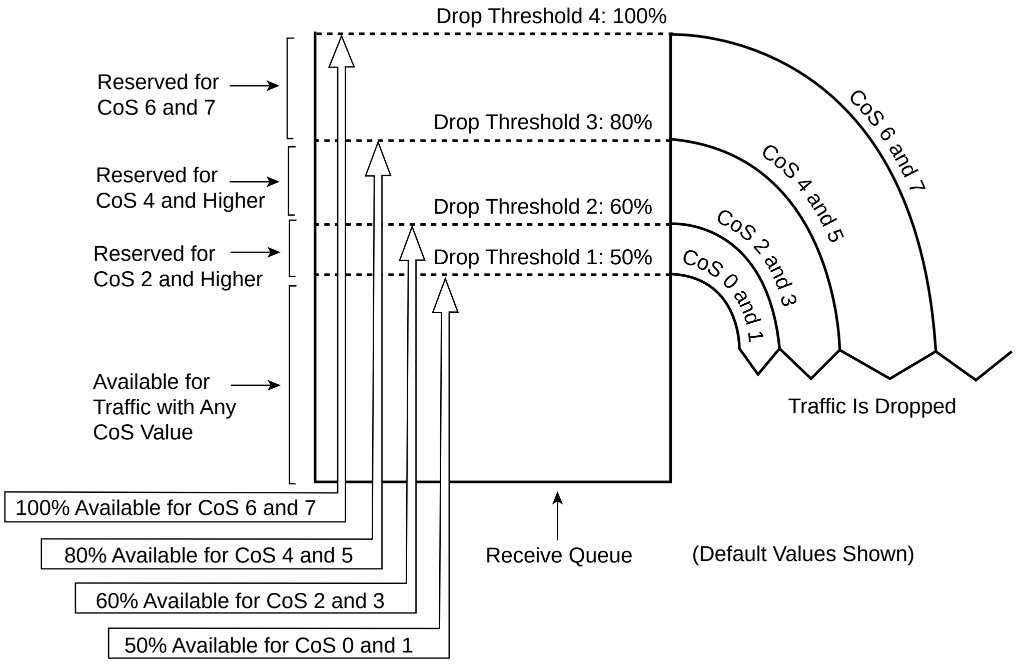

Drop Threshold 4: 100%
Drop Threshold 3: 80%
Drop Threshold 2: 60%
Drop Threshold 1: 50%
Reserved for CoS 6 and 7
Reserved for CoS 4 and Higher
Reserved for CoS 2 and Higher
Available for Traffic with Any CoS Value
CoS 6 and 7
CoS 4 and 5
CoS 2 and 3
CoS 0 and 1
Traffic Is Dropped
100% Available for CoS 6 and 7
80% Available for CoS 4 and 5
60% Available for CoS 2 and 3
50% Available for CoS 0 and 1
Receive Queue
(Default Values Shown)

Table 10-26 summarizes the QoS processes of the Catalyst 6500 and lists the component responsible for the function.

Table 10-26 *QoS Processess of the Catalyst 6500*

QoS Process	**Catalyst 6500 Component That Performs This Function**
Input scheduling	Performed by port application-specific integrated circuits (ASICs) Layer 2 only with or without PFC
Classification	Performed by supervisor or PFC L2 only is done by supervisor L2/3 is done by PFC
Policing	Performed by PFC via L3 forwarding engine
Packet rewrite	Performed by port ASICs L2/L3 based on classification
Output scheduling	Performed by port ASICs L2/L3 based on classification

Table 10-27 describes the supported receive queues of the Catalyst 6500.

Table 10-27 *Supported Receive Queues of the Catalyst 6500*

Catalyst 6500 Receive Queues	Description
1q4t	1 standard queue with 4 tail-drop thresholds
1p1q4t	1 strict-priority queue, 1 standard queue with 4 tail-drop thresholds
1p1q0t	1 strict-priority queue, 1 standard queue with one nonconfigurable (100%) tail-drop threshold
1p1q8t	1 strict-priority queue, 1 standard queue with 8 configurable WRED-drop thresholds and 1 nonconfigurable (100%) tail-drop threshold

Table 10-28 describes the supported transmit queues of the Catalyst 6500.

Table 10-28 *Supported Transmit Queues of the Catalyst 6500*

Catalyst 6500 Transmit Queues	Description
2q2t	2 standard queues with 2 tail-drop thresholds per queue
1p2q2t	1 strict-priority queue, 2 standard queues with 2 WRED-drop thresholds per queue
1p3q1t	1 strict-priority queue, 3 standard queues with 1 WRED-drop threshold and 1 nonconfigurable tail-drop threshold per queue
1p2q1t	1 strict-priority queue, 2 standard queues with 1 WRED-drop threshold and 1 nonconfigurable (100%) tail-drop threshold per queue

Table 10-29 lists the supported receive and transmit queues of the Ethernet modules for the Catalyst 6500.

Table 10-29 *Supported Receive and Transmit Queues of the Ethernet Modules for the Catalyst 6500*

Ethernet Modules	Module Description	RX Queue Type	TX Queue Type	RX Queue Size	TX Queue Size
WS-X6K-S2U-MSFC2 WS-X6K-S2-MSFC2 WS-X6K-S2-PFC2	All Supervisor Engine II uplink ports	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6K-S1A-MSFC2 WS-X6K-SUP1A-MSFC WS-X6K-SUP1A-PFC WS-X6K-SUP1A-2GE	All Supervisor Engine IA uplink ports	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB

Table 10-29 *Supported Receive and Transmit Queues of the Ethernet Modules for the Catalyst 6500 (Continued)*

Ethernet Modules	Module Description	RX Queue Type	TX Queue Type	RX Queue Size	TX Queue Size
WS-X6K-SUP1-2GE	Supervisor Engine II uplink ports	1q4t	2q2t	Q1: 80 KB	Q2: 80 KB Q1: 352 KB
WS-X6524-100FX-MM	24-port 100BASE-FX fabric-enabled with MT-RJ connectors	1p1q0t	1p3q1t	PQ: 6 KB Q1: 22 KB	PQ: 272 KB Q3: 272 KB Q2: 272 KB Q1: 272 KB
WS-X6548-RJ-21	48-port 10/100BASE-TX fabric-enabled with RJ-21 connectors	1p1q0t	1p3q1t	PQ: 6 KB Q1: 22 KB	PQ: 272 KB Q3: 272 KB Q2: 272 KB Q1: 272 KB
WS-X6548-RJ-45	48-port 10/100BASE-TX fabric-enabled with RJ-45 connectors	1p1q0t	1p3q1t	PQ: 6 KB Q1: 22 KB	PQ: 272 KB Q3: 272 KB Q2: 272 KB Q1: 272 KB
WS-X6324-100FX-MM	24-port 100BASE-FX with MT-RJ connectors	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6324-100FX-SM	24-port 100BASE-FX with MT-RJ connectors	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6348-RJ-45	48-port 10/100BASE-TX with RJ-45 connectors	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6348-RJ21V	48-port 10/100BASE-TX with RJ-21 connectors and inline power	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6348-RJ-45V	48-port 10/100BASE-TX with RJ-45 connectors and inline power	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6224-100FX-MT	24-port 100BASE-FX with MT-RJ connectors	1q4t	2q2t	Q1: 8 KB	Q2: 16 KB Q1: 40 KB

continues

Table 10-29 *Supported Receive and Transmit Queues of the Ethernet Modules for the Catalyst 6500 (Continued)*

Ethernet Modules	Module Description	RX Queue Type	TX Queue Type	RX Queue Size	TX Queue Size
WS-X6248-RJ-45	48-port 10/100BASE-TX with RJ-45 connectors	1q4t	2q2t	Q1: 8 KB	Q2: 16 KB Q1: 40 KB
WS-X6248-TEL	48-port 10/100BASE-TX with RJ-21 connectors	1q4t	2q2t	Q1: 8 KB	Q2: 16 KB Q1: 40 KB
WS-X6248A-TEL	48-port 10/100BASE-TX with RJ-21 connectors	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6148-RJ-45V	48-port 10/100BASE-TX with RJ-45 connectors and inline power	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6148-RJ21V	48-port 10/100BASE-TX with RJ-21 connectors and inline power	1q4t	2q2t	Q1: 16 KB	Q2: 22 KB Q1: 90 KB
WS-X6024-10FL-MT	24-port 10BASE-FL with MT-RJ connectors	1q4t	2q2t	Q1: 8 KB	Q2: 16 KB Q1: 40 KB
WS-X6816-GBIC	16-port 1000BASE-X dual-fabric with GBIC connectors and onboard DFC	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6516-GBIC	16-port 1000BASE-X with GBIC connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6516-GE-TX	16-port 10/100/ 100BASE-T with RJ-45 connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB

Table 10-29 *Supported Receive and Transmit Queues of the Ethernet Modules for the Catalyst 6500 (Continued)*

Ethernet Modules	Module Description	RX Queue Type	TX Queue Type	RX Queue Size	TX Queue Size
WS-X6408-GBIC	8-port 1000BASE-X with GBIC connectors	1q4t	2q2t	Q1: 80 KB	Q2: 80 KB Q1: 352 KB
WS-X6408A-GBIC	8-port 1000BASE-X with GBIC connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6416-GBIC	16-port 1000BASE-X with GBIC connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6416-GE-MT	16-port 1000BASE-SX with MT-RJ connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6316-GE-TX	16-port 1000BASE-T with RJ-45 connectors	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB
WS-X6502-10GE	1-port 10 Gigabit Ethernet with OIM connectors	1p1q8t	1p2q1t	51 KB 205 KB	PQ: 15.3 MB Q2: 17.9 MB Q1: 30.7 MB
WS-X6501-10GEX4	1-port 10 Gigabit Ethernet with SC connectors	1p1q8t	1p2q1t	51 KB 205 KB	PQ: 15.3 MB Q2: 17.9 MB Q1: 30.7 MB
OSM	All optical services modules (Layer 2 Gigabit Ethernet ports only)	1p1q4t	1p2q2t	PQ: 9 KB Q1: 64 KB	PQ: 64 KB Q2: 64 KB Q1: 311 KB

Table 10-30 lists the 4500/4000 Catalyst switches that support each supervisor engine model.

Table 10-30 *Catalyst 4500/4000 Supervisor Engine Matrix*

Catalyst Switches That Support the Supervisor Engine I	Catalyst Switches That Support the Supervisor Engine II	Catalyst Switches That Support the Supervisor Engine III	Catalyst Switches That Support the Supervisor Engine IV
Catalyst 4003	Catalyst 4506	Catalyst 4506	Catalyst 4507
	Catalyst 4503	Catalyst 4503	Catalyst 4506

continues

Table 10-30 *Catalyst 4500/4000 Supervisor Engine Matrix (Continued)*

Catalyst Switches That Support the Supervisor Engine I	Catalyst Switches That Support the Supervisor Engine II	Catalyst Switches That Support the Supervisor Engine III	Catalyst Switches That Support the Supervisor Engine IV
	Catalyst 4006	Catalyst 4006	Catalyst 4503
			Catalyst 4006

Table 10-31 lists the default queue assignment for the Catalyst 4500/4000 with a Supervisor I or II after QoS has been enabled.

Table 10-31 *Catalyst 4500/4000 Supervisor I or II Default Queue Admission*

Classification	Queue
CoS values 0 through 7	1
Multicast and broadcast traffic	2

Table 10-32 lists the available QoS features of a Catalyst 4500/4000 with a Supervisor II Engine.

Table 10-32 *Catalyst 4500/4000 Supervisor II QoS Features*

QoS Feature	Setting on 4000/4500 with Sup II
QoS Feature	Setting on 4000/4500 with Sup II
TX queue	2q1t
Scheduling	Round-robin
Trust	Switch wide
Extend trust	Not available
Classification	CoS
Operating system	Catalyst OS

Table 10-33 lists the default queue assignments for the Catalyst 4500/4000 with a Supervisor III or IV.

Table 10-33 *Catalyst 4500/4000 Supervisor III or IV Default Queue Admission*

Classification	Queue
DSCP 0–15 CoS 0–1	1
DSCP 16–31 CoS 2–3	2
DSCP 32–47 CoS 4–5	3
DSCP 48–63 CoS 6–7	4

Table 10-34 lists the available QoS features of a Catalyst 4500/4000 with a Supervisor III of IV Engine.

Table 10-34 *Catalyst 4500/4000 Supervisor III or IV QoS Features*

QoS Feature	Setting on 4000/4500 with Sup III or IV
TX queue	1p3q1t
Scheduling	1 priority queue 3 WWR Queues
Trust	Per port
Extend trust	Per port to IP Phone
Classification	CoS IP precendence IP DSCP ACLs
Operating system	IOS

Table 10-35 describes the default queue admission criteria.

Table 10-35 *Catalyst 3550 Default Queue Admission*

Classification	Queue
DSCP 0–15 CoS 0–1	1
DSCP 16–31 CoS 2–3	2
DSCP 32–47 CoS 4–5	3
DSCP 48–63 CoS 6–7	4

Table 10-36 lists the available QoS features of a Catalyst 3550.

Table 10-36 *Catalyst 3550 QoS Features*

QoS Feature	Setting on 3550
TX queue	1p3q1t
Scheduling	1 priority queue 3 WWR queues
Trust	Per port
Extend trust	Per port to IP Phone
Classification	CoS IP precendence IP DSCP ACLs
Operating system	IOS

Table 10-37 describes the default queue admission criteria.

Table 10-37 *Catalyst 3524 Default Queue Admission*

CoS Value	Queue
0–3	1
4–7	2

Table 10-38 lists the available QoS features of a Catalyst 3524.

Table 10-38 *Catalyst 3524 QoS Features*

QoS Feature	Setting on 3524
TX queue	1p1q
Scheduling	1 priority queue 3 WWR queues
Trust	Switch wide
Extend trust	Per port to IP Phone
Classification	CoS
Operating system	IOS

Table 10-39 lists the default CoS-to-DSCP mapping in Catalyst switches. To match the recommended settings of DSCP = AF31 (or decimal 26) for VoIP call control and DSCP = EF (or decimal 46) for VoIP bearer traffic, these DSCP values must be remapped to CoS values of 3 for VoIP call control and 5 for VoIP bearer traffic.

Table 10-39 *Default CoS-to-DSCP Mapping*

CoS Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

Table 10-40 summarizes the priority queues available on the Catalyst IOS switches.

Table 10-40 *Priority Queues*

Platform	Priority Queue	Configuration Command
6500	1p	On by default
4500	3	**tx-queue 3** **priority high**
3550	4	**priority-queue out**
3524	2	On by default

Table 10-41 summarizes the receive queues and transmit queues present on the switches that have been discussed in this chapter. For the purposes of QoS, a priority queue is always preferred over a standard queue. Use a switch or module that supports priority queues in your designs whenever possible.

Table 10-41 *Summary of RX and TX Queues*

Switch or Module	Receive Queue	Transmit Queue
Catalyst 6500 Supervisor Engine II or Ia	1p1q4t	1p2q2t
Catalyst 6500 Supervisor Engine II or Ia	1q4t	2q2t
Catalyst 6500 65*xx* series line cards	1p1q0t	1p3q1t
Catalyst 6500 63*xx* series line cards	1q4t	2q2t
Catalyst 4500/4000 with Supervisor Engine III or IV	1q	1p3q1t
Catalyst 4500/4000 with Supervisor Engine II	1q	2q1t
Catalyst 3550 series	1q	1p3q1t
Catalyst 3524	1q	2q1t

Understand the different roles of the distribution and access layer switch. Table 10-25 lists common network layer positioning for Cisco switches in real-time application environments.

Table 10-42 *Network Layer Postitioning*

Access Layer Switch	Distribution Layer Switch
Catalyst 6500 with Layer 2 or Layer 3 (PFC) switching engine	Catalyst 6500 with PFC
Catalyst 4500/4000 with Supervisor Engine II, III or IV	Catalyst 4500/4000 with Supervisor Engine III or IV
Catalyst 3550 / 3524 series	

The distribution layer switch provides aggregation for traffic from the access layer switches. CoS or DSCP values received from the access layer switches is trusted by the distribution layer.

It is the responsibility of the access layer switch to mark the traffic flow with the desired CoS or DSCP value. Trust the CoS values received on the access layers switch from attached IP Phones; however, be sure to rewrite the CoS value received from the attached PC to 0. Set the DSCP and CoS classification marking on the access layer switch for those devices that cannot set these values on their own, such as video servers.

After you have designed your network around your real-time application, you can trace the path of the real-time traffic flow and examine the effects of your queuing strategy. Consider the answers to these questions:

- Are your real-time applications correctly classified?
- Do your real-time applications use the priority queue?
- Are there any link mismatches or oversubscribed links that may cause instantaneous buffer overrun?

Remember that QoS in the LAN is not a bandwidth management issue as much as it is a buffer management issue. TX queue congestion can cause packet loss, which can adversely affect performance of applications that are sensitive to loss, delay, and jitter.

Q&A

As mentioned in the Introduction, you have two choices for review questions. The questions that follow next give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD nearer to the end of your preparation, for practice with the actual exam format. You can even customize the CD exam to include, or not include, the topics that are only on the CCIP QoS.

LAN QoS Concepts

1. What is instantaneous buffer overrun?
2. What is meant by the term “1p1q4t?”
3. What packet or frame marking(s) can a Layer 2 switch use to prioritize traffic?
4. What packet or frame marking(s) can a Layer 3 switch use to prioritize traffic?
5. What must be done for a Layer 2 switch to properly classify a packet received from a host across the WAN?
6. For a Layer 2 switch to receive a CoS value from a router, what must exist?
7. What type of router interface supports an 802.1Q trunk to a Layer 3 switch?
8. What is a drop threshold?
9. What is a trust boundary, and where should it be set?

Catalyst 6500 Series of Switches

10. Name three of the five fields that the Policy Feature Card (PFC) rewrites to perform Layer 3 switching.
11. Explain why the first packet in a flow may not retain the proper CoS values when a Catalyst 6500 is configured with a PFC.
12. What is the difference between Hybrid mode and Native mode?
13. What command is used on a Catalyst 6500 running in Hybrid mode to place ports 2/1 through 2/10 in VLAN 10?

14 What command enables QoS on a 6500 running in Native mode?

15 What queue and threshold is recommended for call control traffic on a Catalyst 6500?

16 Which series of Ethernet line cards are preferred in the Catalyst 6500 series for QoS? Why?

Catalyst 4500/4000 Series of Switches

17 Which supervisor engine is preferred on the Catalyst 4500/4000 series? Why?

18 In which queue number does the priority queue reside on a Catalyst 4500 or 4000 with a Supervisor II Engine?

19 In which queue number does the priority queue reside on a Catalyst 4500 or 4000 with a Supervisor III Engine?

20 What keyword(s) is used to enable the priority queue on a Catalyst 4500 or 4000 with a Supervisor III module?

21 Name the four methods a Catalyst 4500 with a Supervisor III uses to classify traffic.

22 What command enables you to rewrite a the CoS value of a PC attached to an IP Phone with Catalyst IOS switches?

23 What trust state are the ports if a 4500 with a Supervisor III in when QoS is enabled?

24 What is dynamic buffer limiting?

Catalyst 3550/3524 Series of Switches

25 In which queue number does the priority queue reside on a Catalyst 3550?

26 What command enables QoS on the Catalyst 3550?

27 What trust state are the ports of a 3550 in when QoS is enabled?

28 Name the four methods a Catalyst 3550 can use to classify traffic.

29 What command enables QoS on the Catalyst 3524?

30 When is a GigaStack GBIC module an acceptable QoS choice for cascading Catalyst 3524 switches together?

APPENDIX A

Answers to the "Do I Know This Already?" Quizzes and Q&A Sections

Chapter 1

"Do I Know This Already?" Quiz

QoS: Tuning Bandwidth, Delay, Jitter, and Loss

1 List the four traffic characteristics that QoS tools can affect.

Bandwidth, delay, jitter, and loss are the four characteristics that affect QoS tools.

2 Describe some of the characteristics of video traffic when no QoS is applied in a network.

Picture displays erratically; picture shows jerky movements; audio not in sync with video; movements slow down; video stream stops.

3 Define bandwidth. Compare and contrast bandwidth concepts over point-to-point links versus Frame Relay.

Bandwidth refers to the number of bits per second that can reasonably be expected to be successfully delivered across a network. With point-to-point networks, bandwidth is equal to the speed of the link—the clock rate. With Frame Relay, the actual bandwidth is difficult to define. Typically, the minimum bandwidth equals the committed information rate (CIR) of a virtual circuit (VC). However, engineers at the provider and at the customer typically expect more than CIR to get through the network. The maximum bandwidth would be bounded by the slower of the two access rates on the access links.

4 List the categories of delay that could be experienced by all three types of traffic: data, voice, and video.

Serialization, propagation, queuing, forwarding/processing, shaping, network. Note that coder/decoder (codec), packetization, and de-jitter delays are unique to voice and video, so technically these delays should not have been part of your answer for this question.

5 Define jitter. Give an example that shows a packet without jitter, followed by a packet with jitter.

Jitter measures the change in delay experienced by consecutive packets. For instance, if a PC sends four packets one after the other, practically at the same time, say 1 ms apart, so the departure times are T=0, T=1, T=2, and T=3. The packets arrive at T=70, T=71, T=80, T=81, respectively. The second packet was sent 1 ms after the first, and arrived 1 ms after the first packet—so no jitter was experienced. However, the third packet arrived 9 ms after the second packet, after being sent 1 ms after the second packet—so 8 ms of jitter was experienced.

6 Define packet loss, and describe the primary reason for loss for which QoS tools can help.

Packet loss means that a packet, which has entered the network, does not get delivered to the endpoint—it is lost in transit. Routers and switches drop packets for many reasons. However, QoS tools can affect the behavior of loss when packets will be lost due to queues being too full. When a queue is full, and another packet needs to be added to the queue, tail drop occurs.

Traffic Characteristics of Voice, Video, and Data

7 Describe the contents of an IP packet carrying the payload for a G.729 Voice over IP (VoIP) call.

The IP packet contains an IP header, a UDP header, an RTP header, and the voice payload. With G.729, the payload uses 20 bytes, with an 8-byte UDP header, and a 12-byte RTP header. The IP header is 20 bytes long.

8 Describe the amount of bandwidth required for G.711 and G.729 VoIP calls, ignoring data-link header/trailer overhead.

G.711 consumes 64 kbps for payload, for an 80-kbps stream with IP, UDP, and RTP headers. G.729 consumes 8 kbps for payload, plus another 16 kbps for IP, UDP, and RTP headers, for a total of 24 kbps.

9 Define the meaning of the term "packetization delay" in relation to a voice call.

Voice must be converted from sound waves to analog electrical signals, and finally to digital signals, and then placed into a packet. Before 20 ms of voice digital payload can be placed into a packet, the speaker must speak for 20 ms. Packetization delay refers to the (default) 20 ms of delay—waiting for the speaker to speak long enough to fill the packet with the correctly sized payload.

10 Define the term "codec delay" and discuss the two components when using a G.729 codec.

Voice calls incur codec delay when the codec converts the analog signal into digital voice payload. Every codec requires some time to process the incoming signal, which adds delay. With G.729, because it is predictive, it must also wait for some additional

incoming voice to arrive, because it is algorithm-processing the voice sample to be encoded, plus a part of the next sample that will be encoded. The delay waiting for the additional voice is called "look-ahead" delay.

11 Describe a typical video payload flow in terms of packet sizes and packet rates.

Video payloads use variable-length packets. The packet rates are also typically variable.

12 Contrast the QoS characteristics needed by interactive data applications to the QoS needs of voice payload flows.

Answering such a question requires one to understand that QoS requirements for data applications are more subjective than those for voice. Generally, interactive data wants consistent delay (low jitter); relative to voice, however, more jitter is tolerable. Bandwidth demands vary greatly for data applications, whereas a single voice call uses a static amount of bandwidth. Delay for interactive data can be relatively longer than for voice, but the key measurement for data is application response time, which includes round-trip packet delays. Finally, data applications are much more tolerant of packet loss—because either the application will resend the data, or rely on TCP to resend the data, or just not care if some data is lost.

Q&A

1 List the four traffic characteristics that QoS tools can affect.

Bandwidth, delay, jitter, and loss.

2 Describe some of the characteristics of voice traffic when no QoS is applied in a network.

Voice is hard to understand; voice breaks up, sounds choppy; calls are disconnected; large delays make it difficult to know when the other caller has finished talking.

3 Describe some of the characteristics of video traffic when no QoS is applied in a network.

Picture displays erratically; picture shows jerky movements; audio not in sync with video; movements slow down; video stream stops.

4 Describe some of the characteristics of data traffic when no QoS is applied in a network.

Data arrives too late to be useful; erratic response times cause users to stop using application; customer care agents waiting on screen refresh, so customer waits.

5 Interpret the meaning of the phrase, “QoS is both ‘managed fairness,’ and at the same time ‘managed unfairness’.”

QoS tools improve QoS characteristics for particular packets. However, improving one packet’s behavior typically comes at the expense of another packet. The terms “managed fairness” and “managed unfairness” just refer to the fact that QoS policies may be fair to one packet but unfair to another.

6 Define bandwidth. Compare and contrast bandwidth concepts over point-to-point links versus Frame Relay.

Bandwidth refers to the number of bits per second that can reasonably be expected to be successfully delivered across a network. With point-to-point networks, bandwidth is equal to the speed of the link—the clock rate. With Frame Relay, the actual bandwidth is difficult to define. Typically, the minimum bandwidth equals the CIR of a VC. However, engineers at the provider and at the customer typically expect more than CIR to get through the network. The maximum bandwidth would be bounded by the slower of the two access rates on the access links.

7 Compare and contrast bandwidth and clock rate in relation to usage for QoS.

Bandwidth refers to the router’s perceived bandwidth on the interface/subinterface, and is referenced by QoS tools. Clock rate defines the physical encoding rate on router interfaces that provide clocking; QoS tools ignore the clock rate setting.

8 List the QoS tool types that affect bandwidth, and give a brief explanation of why each tool can affect bandwidth.

Compression, CAC, and queuing affect bandwidth. Compression reduces the number of bits needed to transmit a frame, allowing more frames to be sent over the same amount of bandwidth. CAC reduces the overall load of voice and video traffic in the network by disallowing new calls. Queuing can reserve subsets of the bandwidth on a link for a particular queue, guaranteeing a minimum amount of bandwidth for that queue.

9 Define delay, compare/contrast one-way and round-trip delay, and characterize the types of packets for which one-way delay is important.

Delay is the time taken from when a frame/packet is sent until it is received on the other side of the network. One-way delay just measures the delay for a packet from one endpoint in the network to its destination. Round-trip delay measures the time it takes to send a packet to one destination and for a response packet to be received. Voice and video are concerned with one-way delay.

10 List the categories of delay that could be experienced by all three types of traffic: data, voice, and video.

Serialization, propagation, queuing, forwarding/processing, shaping, network. Note that codec, packetization, and de-jitter delays are unique to voice and video, so technically these delays should not have been part of your answer for this question.

11 Define, compare, and contrast serialization and propagation delay.

Serialization delay defines the time it takes to encode a frame onto the physical link. For instance, on a point-to-point link of 56 kbps, a bit is encoded every 1/56,000 seconds; therefore, a frame that is 1000 bits long takes 1000/56000 seconds to encode on the link. So, serialization delay is a function of link speed and length of the frame. Propagation delay defines the time taken for a single bit to be delivered across some physical medium, and is based solely on the length of the physical link, and the speed of energy across that medium. If that same point-to-point link were 1000 km (approximately 620 miles) in length, the propagation delay would be 1,000,000m/2.1 * 10^8 ms, or 4.8 milliseconds.

12 Define network delay.

Network delay refers to the delay incurred by a packet inside a packet network, like ATM, Frame Relay, or MPLS networks. Because the customer does not know the details of these networks, and because many customers' packets share the carrier network, variable delays occur.

13 List the QoS tool types that affect delay and give a brief explanation of why each tool can affect delay.

Queuing, link fragmentation and interleaving, compression, and traffic shaping. Queuing methods use an algorithm to choose from which queue to take the next packet for transmission, which can decrease delay for some packets and increase delay for others. LFI tools break large frames into smaller frames, so that smaller delay-sensitive frames can be sent after the first short fragment, instead of having to wait for the entire large original frame to be sent. Compression helps delay because it reduces the overall load in the network, reducing congestion, reducing queue lengths, and reducing serialization delays. Finally, traffic shaping actually increases delay, but it can be applied for one type of traffic, allowing other traffic to be sent with less delay.

14 Define jitter. Give an example that shows a packet without jitter, followed by a packet with jitter.

Jitter measures the change in delay experienced by consecutive packets. If a PC sends four packets one after the other, practically at the same time, say 1 ms apart, so the departure times are T=0, T=1, T=2, and T=3, for instance, packets arrive at T=70, T=71, T=80, T=81, respectively. The second packet was sent 1 ms after the first,

and arrived 1 ms after the first packet—so no jitter was experienced. However, the third packet arrived 9 ms after the second packet, after being sent 1 ms after the second packet—so 8 ms of jitter was experienced.

15 List the QoS tool types that affect jitter and give a brief explanation of why each tool can affect jitter.

Queuing, link fragmentation and interleaving, compression, and traffic shaping. These same QoS tools can be used for addressing delay issues. Queuing can always be used to service a jitter-sensitive queue first if packets are waiting, which greatly reduces delay and jitter. LFI decreases jitter by removing the chance that a jitter-sensitive packet will be waiting behind a very large packet. Compression helps by reducing overall delay, which has a net effect of reducing jitter. Traffic shaping may actually increase jitter, so it should be used with care—but if shaping is applied to jitter-insensitive traffic only, jitter-sensitive traffic will actually have lower delays and jitter.

16 Define packet loss and describe the primary reason for loss for which QoS tools can help.

Packet loss means that a packet, which has entered the network, does not get delivered to the endpoint—it is lost in transit. Routers and switches drop packets for many reasons. However, QoS tools can affect the behavior of loss when packets will be lost due to queues being too full. When a queue is full, and another packet needs to be added to the queue, tail drop occurs.

17 List the QoS tool types that affect loss and give a brief explanation of why each tool can affect loss.

Queuing and RED. Queuing tools allow definition of a longer or shorter maximum queue length; the longer the queue, the less likely that drops will occur. Also by placing traffic into different queues, more variable traffic may experience more loss, because those queues will be more likely to fill. RED tools preemptively discard packets before queues fill, hoping to get some TCP connections to slow down, which reduces the overall load in the network—which shortens queues, reducing the likelihood of packet loss.

18 Describe the contents of an IP packet carrying the payload for a G.729 VoIP call.

The IP packet contains an IP header, a UDP header, an RTP header, and the voice payload. With G.729, the payload uses 20 bytes, with an 8-byte UDP header, and a 12-byte RTP header. The IP header is 20 bytes long.

19 Describe the amount of bandwidth required for G.711 and G.729 VoIP calls, ignoring data-link header/trailer overhead.

G.711 consumes 64 kbps for payload, for an 80-kbps stream with IP, UDP, and RTP headers. G.729 consumes 8 kbps for payload, plus another 16 kbps for IP, UDP, and RTP headers, for a total of 24 kbps.

20 List the delay components that voice calls experience, but which data-only flows do not experience.

Codec delay, packetization delay, and de-jitter initial playout delay.

21 Define the meaning of the term "packetization delay" in relation to a voice call.

Voice must be converted from sound waves to analog electrical signals, and finally to digital signals, and then placed into a packet. Before 20 ms of voice digital payload can be placed into a packet, the speaker must speak for 20 ms. Packetization delay refers to the (default) 20 ms of delay, waiting for the speaker to speak long enough to fill the packet with the correctly sized payload.

22 List the different one-way delay budgets as suggested by Cisco and the ITU.

The ITU in document G.114 suggests a budget of up to 150 ms for quality voice calls; Cisco suggests a delay budget of up to 200 ms one-way if you cannot meet the 150-ms goal.

23 Define the term "codec delay" and discuss the two components when using a G.729 codec.

Voice calls incur codec delay when the codec converts the analog signal into digital voice payload. Every codec requires some time to process the incoming signal, which adds delay. With G.729, because it is predictive, it must also wait for some additional incoming voice to arrive, because it is algorithm-processing the voice sample to be encoded, plus a part of the next sample that will be encoded. The delay waiting for the additional voice is called "look-ahead" delay.

24 Describe the affects of a single lost packet versus two consecutive lost packets, for a G.729 voice call.

Lost voice packets result in the receiver having a period of silence corresponding the length of voice payload inside the lost packet(s). With two consecutive G.729 packets lost, 40 ms of voice is lost; the G.729 codec cannot predict and generate replacement signals when more than 30 ms of consecutive voice is lost. A single lost G.729 packet would only cause a 20-ms break in the voice, which could be regenerated. So, a single lost packet is not perceived as loss in a G.729 call.

25 Describe a typical video payload flow in terms of packet sizes and packet rates.

Video payloads use variable-length packets. The packet rates are also typically variable.

26 Discuss the delay requirements of video traffic.

Interactive video (video conferencing, for instance) requires low delay because it is interactive. Delay budgets up to 200 ms are the norm. However, streaming video—one-way video—can tolerate long delays. When playing an e-learning video, for instance, the playout may start after 30 seconds of video has been received into a de-jitter buffer—but each packet may have experienced several seconds of delay.

27 List the basic differences between TCP and UDP traffic.

TCP performs error recovery, whereas UDP does not. TCP also uses dynamic windowing to perform flow control, whereas UDP does not. Both use port numbers to multiplex among various applications running on a single computer.

28 Contrast the QoS characteristics needed by interactive data applications, as compared to the QoS needs of voice payload flows.

Answering such a question requires one to understand that QoS requirements for data applications are more subjective than those for voice. Generally, interactive data wants consistent delay (low jitter), but relative to voice, more jitter is tolerable. Bandwidth demands vary greatly for data applications, whereas a single voice call uses a static amount of bandwidth. Delay for interactive data can be relatively longer than for voice, but the key measurement for data is application response time, which includes round-trip packet delays. Finally, data applications are much more tolerant of packet loss, because either the application will resend the data, or rely on TCP to resend the data, or just not care whether some data is lost.

Chapter 2

"Do I Know This Already?" Quiz

QoS Tools

1 List four queuing tools, including the full names and popular acronyms.

Priority Queuing (PQ), Custom Queuing (CQ), Weighted Fair Queuing (WFQ), IP RTP Priority, Class-Based WFQ (CBWFQ), Low Latency Queuing (LLQ), Modified Deficit Round-Robin (MDRR).

2 List four link-efficiency tools, including the full names and popular acronyms.

Payload compression, RTP header compression (cRTP), TCP header compression, Multilink PPP fragmentation and interleaving (MLPPP LFI), Frame Relay fragmentation (FRF), link fragmentation and interleaving for Frame Relay and ATM virtual circuits (VCs).

3 Which of the following tools can be used for classification and marking? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

CB marking, NBAR, CAR, and QPPB.

4 Which of the following tools can be used for policing? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

CAR.

Differentiated Services

5 Define the DiffServ term "DSCP," including what the acronym stands for.

According to RFC 2475, DSCP refers to "a specific value of the DSCP portion of the DS field, used to select a PHB." The acronym stands for differentiated services code point. It is the 6-bit field in the redefined ToS byte in the IP header used for marking packets for DiffServ.

6 Define the DiffServ term "PHB," including what the acronym stands for.

According to RFC 2475, PHB refers to "the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate." The acronym stands for per-hop behavior. It is the collection of QoS actions that occur at one router (hop) in a network for a particular behavior aggregate (BA).

7 Compare and contrast the terms "shaper," "meter," and "dropper," according to DiffServ specifications. Suggest typical points in the network where each is used.

The meter function measures the packet/bit rate of ingress or egress traffic, as well as the sizes of bursts of traffic, with the goal of determining whether the traffic meets the agreed-upon traffic contract. A shaper delays packets on egress so that the bit rate and burst size do not exceed a configured level. Droppers, called policers by Cisco IOS, drop packets that exceed a configured bit rate and burst size. Shapers are typically used on egress from one differentiated services domain; policers are typically used on ingress boundary nodes.

8 Compare and contrast the contents of the IP ToS byte before and after the advent of DiffServ.

Before DiffServ, the ToS byte contained a 3-bit Precedence field, 4 bits in a ToS field, and 1 reserved bit. DiffServ redefined the ToS byte to contain a 6-bit DSCP field, which contains the DSCP values, and 2 reserved bits.

Integrated Services

9 Imagine an enterprise network, connected to an Internet service provider (ISP), that is connected to a second ISP, which is then connected to another enterprise network. The second ISP does not support IntServ directly. Discuss the two options that allow the other three networks to support IntServ for flows that pass through the nonsupporting ISP.

Two options exist for support of RSVP and IntServ. Both require ISP2 to pass the RSVP messages through it just like any other packet. Additionally, with one option, no guarantees are made in ISP2; the packets in the reserved flows are treated as best-effort traffic. The other option maps the RSVP flows to a DSCP, so ISP2 can provide DiffServ QoS treatment.

10 Compare and contrast DiffServ and IntServ in terms of using classes, flows, and scalability.

IntServ applies to individual flows, whereas DiffServ differentiates traffic into classes. With large networks and the Internet, the number of IntServ-managed flows does not scale, because information retained about each flow, and the RSVP signaling messages for each flow, continue throughout the life of each flow. DiffServ uses classes, and the number of classes does not increase when packet volumes increase, which allows better scalability.

11 Describe the two options available to a router to perform IntServ admission control.

Routers can perform local admission control by just examining the amount of currently reserved bandwidth, versus the maximum allowed, on each of its interfaces. Routers can also use a Common Open Policy Server (COPS) policy decision point (PDP), asking it to decide whether enough bandwidth is available in the network.

12 What is the QoS framework, and what does it define?

The QoS framework is a drawing and description that lists the major components of QoS functions in a network.

Q&A

1 List four classification and marking tools, including the full names and popular acronyms.

Committed access rate (CAR), policy-based routing (PBR), class-based marking (CB marking), dial peers, network-based application recognition (NBAR), QoS policy propagation with BGP (QPPB).

2 List four queuing tools, including the full names and popular acronyms.

Priority Queuing (PQ), Custom Queuing (CQ), Weighted Fair Queuing (WFQ), IP RTP Priority, Class-Based WFQ (CBWFQ), Low Latency Queuing (LLQ), Modified Deficit Round-Robin (MDRR).

3 List four policing and shaping tools, including the full names and popular acronyms.

Committed access rate (CAR), class-based policing (CB policing), Frame Relay traffic shaping (FRTS), generic traffic shaping (GTS), distributed traffic shaping (DTS), class-based shaping (CB shaping).

4 List three congestion-avoidance tools, including the full names and popular acronyms.

Random Early Detection (RED), Weighted RED (WRED), Flow-Based RED (FRED).

5 List four link-efficiency tools, including the full names and popular acronyms.

Payload compression, RTP header compression (cRTP), TCP header compression, Multilink PPP fragmentation and interleaving (MLPPP LFI), Frame Relay fragmentation (FRF), link fragmentation and interleaving for Frame Relay and ATM VCs.

6 List seven VoIP CAC tools, including the full names and popular acronyms.

Physical DS0 limitation, Max-connections, voice-bandwidth for Frame Relay, trunk conditioning, Local Voice Busy-Out (LVBO), Advanced Voice Busy-Out (AVBO), PSTN Fallback, Resource Availability Indicator (RAI), Gatekeeper Zone Bandwidth (GK Zone Bandwidth), Resource Reservation Protocol (RSVP).

7 List four QoS management tools, including the full names and popular acronyms.

QoS Device Manager (QDM), QoS Policy Manager (QPM), Service Assurance Agent (SAA), Internetwork Performance Monitor (IPM), Service Management Solution (SMS).

8 List the QoS tools that perform some classification function.

This is a bit of a trick question. Almost all IOS QoS tools perform classification—for instance, to place two different types of packets into two different queues, the queue tool performs classification.

9 Which of the following tools can be used for classification and marking? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

CB marking, NBAR, CAR, and QPPB.

10 Which of the following tools can be used for queuing? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

WFQ, LLQ, PQ, CQ.

11 Which of the following tools can be used for policing? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

CAR

12 Which of the following tools can be used for shaping? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

CB shaping, FRTS, GTS

13 Which of the following tools can be used for link efficiency? CAR, CB marking, PQ, CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

FRF, cRTP, MLPPP LFI

14 Which of the following tools can be used for network management? CAR, CB marking, PQ CB shaping, QDM, WFQ, WRED, FRTS, LLQ, GTS, RAI, FRF, RSVP, SAA, MLPPP LFI, AVBO, CQ, NBAR, QPM, CAR, FRED, QPPB, cRTP

QPM, QDM, SAA

15 Define the DiffServ term "behavior aggregate."

According to RFC 2475, a behavior aggregate is "a collection of packets with the same DS code point crossing a link in a particular direction." The key points are that the DSCP has been set; the packets all move the same direction; and the packets collectively make up a class.

16 Define the DiffServ term "DSCP," including what the acronym stands for.

According to RFC 2475, DSCP refers to "a specific value of the DSCP portion of the DS field, used to select a PHB." The acronym stands for differentiated services code point. It is the 6-bit filed in the redefined ToS byte in the IP header used for marking packets for DiffServ.

17 Define the DiffServ term "PHB," including what the acronym stands for.

According to RFC 2475, PHB refers to "the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate." The acronym stands for per-hop behavior. It is the collection of QoS actions that occur at one router (hop) in a network for a particular BA.

18 Define the DiffServ term "MF classifier," including what the acronym stands for.

According to RFC 2475, an MF classifier is "a multi-field (MF) classifier which selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field, protocol ID, source port and destination port." It is the classification function used to classify packets before the DSCP has been set.

19 Define the DiffServ term "DS ingress node," including what the acronym stands for.

According to RFC 2475, a DS ingress node is "a DS boundary node in its role in handling traffic as it enters a DS domain." DS stands for differentiated services. The term defines a node at which packets enter the DiffServ domain.

20 Compare and contrast the terms "BA classifier" and "MF classifier," according to DiffServ specifications. Suggest typical points in the network where each is used.

A classifier is a DiffServ function that classifies or categories packets based on the contents of fields in the packet headers. A BA classifier performs this function only based on the DSCP field. An MF classifier can look at many fields in the packet header. MF classifiers typically classify ingress traffic near the edge of a network, and work with markers to set the DSCP field. BA classifiers are used at points in the network after an MF classifier and marker have set the DSCP field values.

21 Compare and contrast the terms "shaper," "meter," and "dropper," according to DiffServ specifications. Suggest typical points in the network where each is used.

The meter function measures the packet/bit rate of ingress or egress traffic, as well as the sizes of bursts of traffic, with the goal of determining whether the traffic meets the agreed-upon traffic contract. A shaper delays packets on egress so that the bit rate and burst size do not exceed a configured level. Droppers, called policers by Cisco IOS, drop packets that exceed a configured bit rate and burst size. Shapers are typically used on egress from one DS domain; policers are typically used on ingress boundary nodes.

22 Compare and contrast the contents of the IP ToS byte before and after the advent of DiffServ.

Before DiffServ, the ToS byte contained a 3-bit Precedence field, 4 bits in a ToS field, and 1 reserved bit. DiffServ redefined the ToS byte to contain a 6-bit DSCP field, which contains the DSCP values, and 2 reserved bits.

23 Describe the QoS behavior at a single DS node when using the AF PHB. Also explain what the acronym "AF PHB" represents and identify the RFC that defines it.

The assured forwarding per-hop behavior, as defined in RFC 2597, defines a PHB with two components. The first part defines four BAs or classes, each which should be placed in a separate queue and given a configured guaranteed minimum amount of bandwidth. The second component provides three different drop probabilities for a congestion-avoidance tool such as RED.

24 Explain (by comparing and contrasting) whether AF and CS PHB DSCPs conform to the concept that "bigger DSCP values are better than smaller values."

CS uses values that have three binary 0s at the end, and the eight IP precedence values for the first three bits. In other words, CS includes the eight binary values for a 6-bit number for which the last three digits are 0s. CS conforms to the idea that a bigger value is better, to be backward compatible with IP precedence. AF uses 12 different values. Of the three AF DSCPs in each class, the highest of the three values actually receives the worst drop preference.

25 Describe the QoS behavior at a single DS node when using the EF PHB. Also explain what the acronym "EF PHB" represents and identify the RFC that defines it.

The expedited forwarding per-hop behavior, as defined in RFC 2598, defines a PHB with two components. The first part defines queuing, with features that reserve bandwidth for a single BA, with the added feature on minimizing latency, delay, and loss. The other action of the PHB provides a policing/dropper function, disallowing traffic beyond a configured maximum bandwidth for the class.

26 Describe the process used by RSVP to reserve bandwidth in a network.

A host signals to the network using an RSVP reservation request using an RSVP path message. The request passes along the route to the destination host; at each intermediate router, if that router can guarantee the right bandwidth, the request is forwarded. When received by the destination host, it replies with an RSVP resv message. The process is reversed, with each router passing the reserve message if it can guarantee the bandwidth in the opposite direction. If the original host receives the reservation message, the bandwidth has been reserved.

27 Imagine an enterprise network, connected to an Internet service provider (ISP), that is connected to a second ISP, which is then connected to another enterprise network. The second ISP does not support IntServ directly. Discuss the two options that allow the other three networks to support IntServ for flows that pass through the nonsupporting ISP.

Two options exist for support of RSVP and IntServ. Both require ISP2 to pass the RSVP messages through it just like any other packet. Additionally, with one option, no guarantees are made in ISP2; the packets in the reserved flows are treated as best-effort traffic. The other option maps the RSVP flows to a DSCP, so ISP2 can provide DiffServ QoS treatment.

28 List and describe the two main features of IntServ.

IntServ provides resource reservation and admission control. Resource reservation uses the RSVP protocol to reserve bandwidth and set a maximum delay value. To prevent too many reserved flows from occurring, however, routers can reject new reservations, which is called admission control.

29 Compare and contrast DiffServ and IntServ in terms of using classes, flows, and scalability.

IntServ applies to individual flows, whereas DiffServ differentiates traffic into classes. With large networks and the Internet, the number of IntServ-managed flows does not scale, because information retained about each flow, and the RSVP signaling messages for each flow, continues throughout the life of each flow. DiffServ uses classes, and the number of classes does not increase when packet volumes increase, which allows better scalability.

30 Describe the two options available to a router to perform IntServ admission control.

Routers can perform local admission control by just examining the amount of currently reserved bandwidth, versus the maximum allowed, on each of its interfaces. Routers can also use a Common Open Policy Server (COPS) policy decision point (PDP), asking it to decide whether enough bandwidth is available in the network.

31 What is the QoS framework, and what does it define?

The QoS framework is a drawing and description that lists the major components of QoS functions in a network.

32 Which five categories of QoS tools are shown in the bottom half of the Cisco QoS framework?

Classification and marking, congestion management, link efficiency, traffic conditioners, congestion avoidance.

Chapter 3

"Do I Know This Already?" Quiz

Classification and Marking Concepts

1 Describe the difference between classification and marking.

Classification processes packet headers, or possibly other information, to differentiate between multiple packets. Marking changes a field inside the frame or packet header.

2 Describe, in general, how a queuing feature could take advantage of the work performed by a classification and marking feature.

Queuing features can perform their own classification function to place different packets into different queues. After a classification and marking tool has marked a packet, the queuing feature can look for the marked value when classifying packets.

3 Which of the following QoS marking fields are carried inside an 802.1Q header: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

CoS and User Priority. CoS is the more general name, with User Priority specifically referring to the 3-bit field in the 802.1P header.

4 Which of the following QoS marking fields are carried inside an IP header: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

ToS Byte, ToS bits, Precedence, DSCP, DS field.

CAR, PBR, and CB Marking

5 Define the meaning of MQC, and spell out what the acronym stands for.

Modular QoS command-line interface (MQC) is a relatively new syntax structure for IOS configuration commands used in configuring several QoS features. MQC features the separation of classification logic from the actual QoS behavior (for instance, marking or queuing), and with yet another command for enabling the QoS feature on an interface. MQC is often described as a new QoS CLI, although it is really just a new set of commands.

6 What configuration command lists the marking details when configuring CB marking? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The set command defines what value to mark in the frame or packet header after a packet has been classified. The command is a subcommand under the class command, which is a subcommand under the policy-map command, which in turn is a global configuration command.

7 What configuration command lists the marking details when configuring CAR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The rate-limit command defines what value to mark in the frame or packet header after a packet has been classified. The marking details are included in the same rate-limit command as the classification details. The command is a subcommand under the interface command, which is a global configuration command.

8 What configuration command lists the classification details when configuring PBR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The match command defines the details of what must be matched to classify a packet. The command, which differs from the CB marking match command, is a subcommand under the route-map command, which is a global configuration command.

Q&A

1 Describe the difference between classification and marking.

Classification processes packet headers, or possibly other information, to differentiate between multiple packets. Marking changes a field inside the frame or packet header.

2 Describe, in general, how a queuing feature could take advantage of the work performed by a classification and marking feature.

Queuing features can perform their own classification function to place different packets into different queues. After a classification and marking tool has marked a packet, the queuing feature can look for the marked value when classifying packets.

3 Characterize what must be true before the CoS field may be useful for marking packets.

CoS only exists in 802.1P/Q headers and ISL headers. In turn, these headers are used only on Ethernet links that use trunking. Therefore, the CoS field can only be marked or reacted to for Ethernet frames that cross an 802.1Q or ISL trunk.

4 Most other QoS tools, besides classification and marking tools, also have a classification feature. Describe the advantage of classification, in terms of overall QoS design and policies, and explain why classification and marking is useful, in spite of the fact that other tools also classify the traffic.

Classification and marking, near the ingress edge of a network, can reduce the amount of work required for classification by other QoS tools. In particular, many QoS tools can classify based on marked fields without using an ACL, which reduces overhead for each QoS tool. By marking packets near the ingress edge, QoS policies can be more consistently applied. In addition, configurations for most other QoS tools become simpler, which can reduce configuration errors in the network.

5 Which of the following classification and marking tools can classify based on the contents of an HTTP URL: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?

NBAR actually performs the classification based on HTTP header contents. CB marking is the only tool that marks based on NBAR’s match of the URL string.

6 Describe the differences between IP extended ACLs as compared with NBAR for matching TCP and UDP port numbers.

You can use both tools to match packet based on well-known port numbers. However, some higher-layer protocols allocate dynamic port numbers, making the use of extended ACLs difficult at best. NBAR can look further into the packet contents to identify what dynamic ports are currently in use by certain protocols, and match packets using those dynamic ports.

7 Which of the following classification and marking tools can classify based on the outgoing interface of the route used for a packet: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?

PBR is the only tool that can classify and mark based on route information.

8 Which of the following classification and marking tools can classify based on the destination TCP port number of a packet, without using an IP ACL: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?

None! To classify on a TCP port number, an IP ACL must be used.

9 Which of the following classification and marking tools can classify based on the DSCP, without using an IP ACL: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?

CAR, CB marking, and dial peers. Dial peers can set the DSCP value for VoIP traffic created as a result of the dial peer (as of IOS 12.2T), but that is not really classification. (Note that the DQOS exam, at time of publication of this book, covers 12.2 mainline, but not 12.2T train features.)

10 Which of the following classification and marking tools can classify based on either the source or destination MAC address: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial peers?

CAR and CB marking. Only CB marking can examine the destination MAC.

11 Which of the following classification and marking tools can classify based on the even numbered UDP ports used for RTP traffic, with or without using an IP ACL: class-based marking (CB marking), policy-based routing (PBR), committed access rate (CAR), network-based application recognition (NBAR), or dial-peers?

CB marking. Tools that use IP ACLs can match port-number ranges, but because RTP uses only the even-numbered ports, IP ACLs cannot easily match just the even-numbered ports. Dial peers can set the DSCP value for VoIP payload (even numbered ports) created as a result of the dial peer as of IOS 12.2T, but that is not really a classification feature. It is, instead, a marking feature.

12 Which of the following QoS marking fields are carried inside an 802.1Q header: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

CoS and User Priority. CoS is the more general name, with User Priority specifically referring to the 3-bit field in the 802.1P header.

13 Which of the following QoS marking fields are carried inside an IP header: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

ToS byte, ToS bits, Precedence, DSCP, DS.

14 Which of the following QoS marking fields are never marked inside a frame that exits a router: QoS, CoS, DE, ToS byte, User Priority, ToS bits, CLP, Precedence, QoS Group, DSCP, MPLS Experimental, or DS?

QoS Group is only used for internal purposes in GSR and 7500 series routers.

15 Describe the goal of marking near the edge of a network in light of the meaning of the term “trust boundary.”

Good QoS design calls for classification and marking, based on well-defined QoS policies, as near to the ingress edge of the network as possible. However, packets marked in devices near the edge of the network may be able to be re-marked by devices whose administrators cannot be trusted. A packet can be marked by the end-user PC, for instance, but the end user can configure the value to be marked. An IP Phone, however, can mark packets, and the marked values cannot be overridden by the user of the phone. Therefore, the goal of marking near the edge must be tempered against the fact that some devices can be reconfigured for QoS by those outside the group responsible for QoS.

16 Define the meaning of MQC, and spell out what the acronym stands for.

Modular QoS command-line interface (MQC) is a relatively new syntax structure for IOS configuration commands used in configuring several QoS features. MQC features the separation of classification logic from the actual QoS behavior (for

instance, marking or queuing), and with yet another command for enabling the QoS feature on an interface. MQC is often described as a new QoS CLI, although it is really just a new set of commands.

17 What configuration command lists the classification details when configuring CB marking? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The match command defines the details of what must be matched to classify a packet. The command is a subcommand under the class-map command, which is a global configuration command.

18 What configuration command lists the marking details when configuring CB marking? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The set command defines what value to mark in the frame or packet header once a packet is classified. The command is a subcommand under the class command, which is a subcommand under the policy-map command, which in turn is a global configuration command.

19 What configuration command enables CB marking? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The service-policy command enables CB marking for either input or output packets on an interface. The command refers to the policy map, which in turn refers to the class maps. The command is a subcommand under the interface command, which is a global configuration command.

20 What configuration command lists the classification details when configuring CAR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The rate-limit command defines the details of what must be matched to classify a packet. The command is a subcommand under the interface command, which is a global configuration command.

21 What configuration command lists the marking details when configuring CAR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The rate-limit command defines what value to mark in the frame or packet header when a packet is classified. The marking details are included in the same rate-limit command as the classification details. The command is a subcommand under the interface command, which is a global configuration command.

22 What configuration command enables CAR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The rate-limit command enables CAR on an interface, either for input or output packets. The command is a subcommand under the interface command, which is a global configuration command.

23 What configuration command lists the classification details when configuring PBR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The match command defines the details of what must be matched to classify a packet. The command, which his different from the CB marking match command, is a subcommand under the route-map command, which is a global configuration command.

24 What configuration command lists the marking details when configuring PBR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The set command defines what value to mark in the frame or packet header when a packet is classified. The command, which differs from the CB marking set command, is a subcommand under the route-map command, which is a global configuration command.

25 What configuration command enables PBR? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The ip policy command enables PBR on an interface, always for input packets. This command refers to the route map under which the match and set commands are configured. The command is a subcommand under the interface command, which is a global configuration command.

26 Describe the process dial peers use to classify and mark traffic.

At the IOS levels being referenced as of the publication of this book (IOS 12.2 mainline), dial peers can mark all VoIP packets created by the dial peer with a configured IP precedence value. As of 12.2T, dial peers can classify voice payload and voice signaling traffic using the media option and the signaling option, respectively, on the ip qos dscp command.

27 What configuration command(s) lists the marking details when configuring dial peers? What configuration mode must you use to configure the command? What commands must you issue to place the configuration mode user into that mode?

The ip precedence command (or the ip qos dscp command at IDS 12.2T) is used to define the precedence value (or DSCP value) of VoIP packets to the specified dial peer. The command is a subcommand of the dial-peer global configuration command.

28 What QoS values can a dial peer mark?

At 12.2 mainline IOS, only the IP precedence. As of 12.2T, DSCP can also be marked.

Chapter 4

“Do I Know This Already?” Quiz

Queuing Concepts

1 Describe the benefits of having a single FIFO output queue.

The most basic benefit of queuing is to provide a means to hold a packet while the interface is busy. Without at least a single FIFO queue, routers would have to discard packets if the outgoing interface were busy.

2 Explain the purpose of a TX Ring and TX Queue in a Cisco router.

By design, routers want to be able to immediately begin sending the next packet when the preceding packet’s last bit is sent. To do this, the interface hardware must have access to a queue structure with the next packet, and not be impeded by waiting on service from other processes. On Cisco routers, the TX Ring and TX Queue provide queue structures that are available to the interface directly, without relying on the main processor.

3 Assume a queuing tool has been enabled on interface S0/0. Describe the circumstances under which the queuing tool would actually be used.

Congestion must occur on the interface first, which causes packets to be held in the TX Ring/Queue. When the TX Ring/Queue fills, IOS enables the queuing function on the interface.

4 Explain the circumstances under which it would be useful to enable a queuing tool on a subinterface.

Queues only form on subinterfaces when traffic shaping is enabled on the subinterface.

WFQ and IP RTP Priority

5 Characterize the effect the WFQ scheduler has on different types of flows.

Lower-volume flows get relatively better service, and higher-volume flows get worse service. Higher-precedence flows get better service than lower-precedence flows. If lower-volume flows are given higher precedence values, the bandwidth, delay, jitter, and loss characteristics improve even more.

6 Describe the WFQ scheduler process. Include at least the concept behind any formulas, if not the specific formula.

Each new packet is assigned a sequence number (SN), which is based on the previous packet's SN, the length of the new packet, and the IP precedence of the packet. The formula is as follows:

(Previous SN + weight) * New packet length

The scheduler just takes the lowest SN packet when it needs to de-queue a packet.

7 You previously disabled WFQ on interface S0/0. List the minimum number of commands required to enable WFQ on S0/0.

Use the fair-queue interface subcommand.

8 What commands list statistical information about the performance of WFQ?

The show interfaces and the show queueing fair commands list statistics about WFQ.

CBWFQ and LLQ

9 Describe the CBWFQ scheduler process, both inside a single queue and among all queues.

The scheduler provides a guaranteed amount of bandwidth to each class. Inside a single queue, processing is FIFO, except for the class-default queue. In class-default, Flow-Based WFQ can be used, or FIFO, inside the queue.

10 Describe how LLQ allows for low latency while still giving good service to other queues.

LLQ is actually a variation of CBWFQ, in which a single class is always serviced first—in other words, the low-latency queue is a strict-priority queue. To prevent the low-latency queue from dominating the link, and to continue to guarantee bandwidth amounts to other queues, the LLQ class is policed.

11 Compare and contrast IP RTP Priority and LLQ. In particular, mention what other queuing tools can be used concurrently with each, how each classifies packets, and which is recommended by Cisco.

LLQ is actually a feature of CBWFQ and is always used in conjunction with CBWFQ. IP RTP Priority can be used with WFQ or CBWFQ to add a Low Latency Queue option. IP RTP Priority classifies packets based on even-number UDP ports,

in a specified range. LLQ can classify on anything that can be matched using MQC classification commands, making it much more flexible. Given a choice, Cisco recommends LLQ.

12 Compare and contrast the CBWFQ command that configures the guaranteed bandwidth for a class with the command that enables LLQ for a class.

The bandwidth command enables you to define a specific bandwidth, or a percentage bandwidth. The priority command, which enables LLQ in a class, appears to also reserve an amount or percentage of bandwidth. However, it actually defines the policing rate, to prevent the low-latency queue from dominating the link. The priority command enables you to set the policing burst size as well.

Comparing Queuing Options

13 Which of the following queuing tools allows for WRED inside a single queue? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

CBWFQ

14 Which of the following queuing tools can always service a particular queue first, even when other queues have packets waiting? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

PQ, LLQ, and IP RTP Priority

15 Which of the following queuing tools allows for a percentage bandwidth to be assigned to each queue? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

CBWFQ and LLQ.

16 Which queuing tools could be configured to provide the lowest possible latency for voice traffic? Of these, which does Cisco recommend as the best option for voice queuing today?

PQ, IP RTP Priority, and LLQ. PQ would probably not be a good option in many networks today, but it could provide the lowest possible latency for voice. Cisco currently recommends LLQ.

Q&A

Conceptual Questions

1. Describe the benefits of having a single FIFO output queue.

   The most basic benefit of queuing is to provide a means to hold a packet while the interface is busy. Without at least a single FIFO queue, routers would have to discard packets if the outgoing interface were busy.

2. Explain the effects of changing a single FIFO queue's length to twice its original value. Include comments about how the change affects bandwidth, delay, jitter, and loss.

   With a longer queue, more packets can be enqueued before the queue fills. Therefore, the tail-drop process drops packets less often. However, with more packets in the queue, the average delay increases, which also can increase jitter. There is no impact on bandwidth.

3. Explain the purpose of a TX Ring and TX Queue in a Cisco router.

   By design, routers want to be able to begin immediately sending the next packet when the preceding packet's last bit is sent. To do this, the interface hardware must have access to a queue structure with the next packet, and not be impeded by waiting on service from other processes. On Cisco routers, the TX Ring and TX Queue provide queue structures that are available to the interface directly, without relying on the main processor.

4. Explain how a long TX Ring might affect the behavior of a queuing tool.

   Output queuing does not occur until the TX Ring is full. If the TX Ring is long, the queuing tool may not be enabled. Because the TX Ring always uses FIFO logic, packets will not be reordered. With a short TX Ring, output queuing may be queuing the packets, and have an opportunity to reorder the packet exit sequence based on the queuing scheduling algorithm.

5. Describe the command output that identifies the length of the TX Ring or TX Queue, and whether the length was automatically lowered by IOS.

   The show controllers command lists output that includes the output line that reads something like "tx_limited=0(16)." The first number is 0 or 1, with 0 meaning that the statically-configured value is being used, and the number in parenthesis representing the length of the TX Ring/TX Queue. If the first number is 1, the TX Ring/TX Queue has been automatically shortened by the IOS as a result of having a queuing tool enabled on the interface.

6 Explain under what circumstances the TX Ring, interface output queues, and subinterface output queues both fill and drain, and to where they drain.

The TX Ring fills when the packets needing to exit an interface exceed the line (clock) rate of the interface. When the TX Ring fills, the interface output queue(s) begin to fill. The subinterface output queue(s) only fill if traffic shaping is enabled on the subinterfaces or individual VCs, and if the offered traffic on a subinterface or VC exceeds the shaped rate. The VC or subinterface queues drain into the interface queue(s), the interface queue(s) into the TX Ring, and the TX Ring onto the physical interface.

Priority Queuing and Custom Queuing

7 Assume a queuing tool has been enabled on interface S0/0. Describe the circumstances under which the queuing tool would actually be used.

Congestion must occur on the interface first, which causes packets to be held in the TX Ring/TX Queue. When the TX Ring/TX Queue fills, IOS enables the queuing function on the interface.

8 Explain the circumstances under which it would be useful to enable a queuing tool on a subinterface.

Queues only form on subinterfaces when traffic shaping is enabled on the subinterface.

9 Describe the classification feature of Priority Queuing, including the list of items that can be examined for classification decisions.

PQ can classify packets using access-control lists (ACLs) for most Layer 3 protocols, matching anything allowed by any of the types of ACLs. PQ can also directly match, without using an ACL, the incoming interface, packet length, and TCP and UDP ports numbers.

10 Describe the classification feature of Custom Queuing, including the list of items that can be examined for classification decisions.

CQ performs class-based queuing, in that it classifies packets on a large variety of packet header fields. CQ can classify packets using access-control lists (ACLs) for most Layer 3 protocols, matching anything allowed by any of the types of ACLs. CQ can also directly match, without using an ACL, the incoming interface, packet length, and TCP and UDP ports numbers.

11 List the classification options available to Custom Queuing that are not also available to Priority Queuing.

There are none. PQ and CQ classification options are identical.

12 Describe the process and end result of the scheduling feature of Priority Queuing.

Always service higher-priority queues first; the result is great service for the High queue, with the potential for 100 percent of link bandwidth. Service degrades quickly for lower-priority queues.

13 Describe the process and end result of the scheduling feature of Custom Queuing.

Services packets from a queue until a byte count is reached; round-robins through the queues, servicing the different byte counts for each queue. The effect is to reserve a percentage of link bandwidth for each queue.

14 List the maximum number of queues used by Priority Queuing and Custom Queuing.

PQ has 4 queues, and CQ has 16. CQ also has a reserved queue, into which only IOS can schedule packets.

WFQ

15 Characterize the effect the WFQ scheduler has on different types of flows.

Lower-volume flows get relatively better service, and higher-volume flows get worse service. Higher-precedence flows get better service than lower-precedence flows. If lower-volume flows are given higher precedence values, the bandwidth, delay, jitter, and loss characteristics improve even more.

16 Describe the WFQ scheduler process. Include at least the concept behind any formulas, if not the specific formula.

Each new packet is assigned a sequence number, which is based on the previous packet's SN, the length of the new packet, and the IP precedence of the packet. The formula is as follows:

(Previous SN + weight) * New packet length

The scheduler just takes the lowest SN packet when it needs to de-queue a packet.

17 You previously disabled WFQ on interface S0/0. List the minimum number of commands required to enable WFQ on S0/0.

Use the fair-queue interface subcommand.

18 What commands list statistical information about the performance of WFQ?

The show interfaces and the show queueing fair commands list statistics about WFQ.

19 Define what comprises a flow in relation to WFQ.

A flow consists of all packets with the same source and destination IP address, transport layer protocol, and transport layer source and destination port. Some references also claim that WFQ includes the ToS byte in the definition of a flow.

20 You just bought and installed a new 3600 series router. Before adding any configuration to the router, you go ahead and plug in the new T1 Frame Relay access link to interface S0/0. List the minimum number of commands required to enable WFQ on S0/0.

No commands are required. WFQ is the default on E/1 and slower interfaces in a Cisco router.

CBWFQ, LLQ, IP RTP Priority

21 Describe the CBWFQ scheduler process, both inside a single queue and among all queues.

The scheduler provides a guaranteed amount of bandwidth to each class. Inside a single queue, processing is FIFO, except for the class-default queue. In class-default, Flow-Based WFQ can be used, or FIFO, inside the queue.

22 Describe how LLQ allows for low latency while still giving good service to other queues.

LLQ is actually a variation of CBWFQ, in which the LLQ classes are always serviced first—in other words, the low-latency queues are a strict-priority queues. To prevent the low-latency queues from dominating the link, and to continue to guarantee bandwidth amounts to other queues, the LLQ classes are policed.

23 Compare and contrast IP RTP Priority and LLQ. In particular, mention what other queuing tools can be used concurrently with each, how each classifies packets, and which is recommended by Cisco.

LLQ is actually a feature of CBWFQ, and is always used in conjunction with CBWFQ. IP RTP Priority can be used with WFQ or CBWFQ to add a low-latency queue option. IP RTP Priority classifies packets based on even-number UDP ports, in a specified range. LLQ can classify on anything that can be matched using MQC classification commands, making it much more flexible. Given a choice, Cisco recommends LLQ.

24 Compare and contrast the CBWFQ command that configures the guaranteed bandwidth for a class with the command that enables LLQ for a class.

The bandwidth command enables you to define a specific bandwidth, or a percentage bandwidth. The priority command, which enables LLQ in a class, appears to reserve an amount or percentage of bandwidth as well. However, it actually defines the policing rate, to prevent the LLQ from dominating the link. The priority command enables you to set the policing burst size as well.

25 Describe the CBWFQ classification options. List at least five fields that can be matched without using an ACL.

CBWFQ uses the Modular QoS CLI, and therefore can match on any fields that can be matched with other MQC tools, like CB marking. Other than referring to an ACL, CBWFQ can classify based on incoming interface, source/destination MAC, IP Precedence, IP DSCP, LAN CoS, QoS group, MPLS Experimental bits, and anything recognizable by NBAR.

26 Name the two CBWFQ global configuration commands that define classification options, and then the per-hop behaviors, respectively. Also list the command that enables CBWFQ on an interface.

The class-map command names a class map and places the user into class-map configuration mode. Classification parameters can be entered at that point. The policy-map command names a policy and enables you to refer to class maps and then define actions. The service-policy command enables the policy map for packets either entering or exiting the interface.

27 List the command used to configure IP RTP Priority on a serial link.

ip rtp priority *starting-rtp-port-number port-number-range bandwidth*

28 Characterize the type of traffic that can be queued using both IP RTP priority and LLQ. List specific port numbers and IP addresses as applicable, and describe the type of traffic.

Both tools can classify and queue traffic using even-numbered UDP ports from 16384 to 32767. These UDP ports are used to transport RTP traffic, which holds voice payload, but not signaling, traffic.

29 Examine the following configuration (Example 4-10). Which of the five policy maps would certainly enable LLQ for voice payload traffic, based only of the information in the configuration?

Example 4-10 *Exhibit for CBWFQ Configuration Questions*

```
!
class-map match-all class1
  match ip rtp 16384 16383
class-map match-all class2
  match access-group 101
class-map match-all class3
  match ip rtp 16384 32767
class-map match-all class4
  match ip dscp ef
class-map match-all class5
  match access-group 102
!
policy-map pmap1
```

continues

Example 4-10 *Exhibit for CBWFQ Configuration Questions (Continued)*

```
 class class1
  priority 60
policy-map pmap2
 class class2
  priority 60
policy-map pmap3
 class class3
  priority 60
policy-map pmap4
 class class4
  priority 60
policy-map pmap5
 class class5
  priority 60
!
interface Serial0/0
 service-policy output ?????
!
access-list 101 permit udp any gt 16383 any gt 16383
access-list 102 permit udp any range 16383 32767 any range 16383 32767
!
```

All the policy maps except pmap4 would perform LLQ on voice payload. In some cases, the policy map would match more than just voice payload. Only pmap1 would match just RTP voice payload traffic.

30 Using the same exhibit as in the preceding example, describe what must also be true for pmap4 to queue voice payload traffic successfully, and only voice payload traffic, in a low-latency queue.

If some other classification and marking tool were configured, and it marked all voice payload traffic as DSCP EF, pmap4 would match all voice packets in the low-latency queue.

Comparing Queuing Tool Options

31 Which of the following queuing tools allows for WRED inside a single queue? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

CBWFQ.

32 Which of the following queuing tools can always service a particular queue first, even when other queues have packets waiting? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

PQ, LLQ, and IP RTP Priority.

33 Which of the following queuing tools allows for a percentage bandwidth to be assigned to each queue? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

CBWFQ and LLQ. CQ effectively does this as well, but you cannot specify the exact percentage.

34 Which queuing tools could be configured to provide the lowest possible latency for voice traffic? Of these, which does Cisco recommend as the best option for voice queuing today?

PQ, IP RTP Priority, and LLQ. PQ would probably not be a good option in many networks today, but it could provide the lowest possible latency for voice. Cisco currently recommends LLQ.

35 Which of the following queuing tools can use flow-based classification? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

WFQ and CBWFQ in the class-default queue.

36 Which of the following queuing tools uses the Modular QoS CLI? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

CBWFQ, LLQ.

37 Which of the following queuing tools allows for a value to be configured, which then results in a specific number of bytes being taken from each queue during a round-robin pass through the queues? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

CQ.

38 Which of the following queuing tools allow the largest number of queues for a flow-based tool? For a class-based tool? What are the maximum values? First-In, First-Out Queuing (FIFO); Priority Queuing (PQ); Custom Queuing (CQ); Weighted Fair Queuing (WFQ); Class-Based WFQ (CBWFQ); Low Latency Queuing (LLQ); and IP RTP Priority.

WFQ allows 4096 (flow based), and CBWFQ allows 64 queues (class based).

39 Which queuing tools could be configured to provide the lowest possible latency for voice signaling traffic?

PQ and LLQ. PQ would probably not be a good option in many networks today. IP RTP Priority does not match voice signaling traffic.

Chapter 5

“Do I Know This Already?” Quiz

Shaping and Policing Concepts

1 Explain the points during the process of a single router receiving and forwarding traffic at which shaping and policing can be enabled on a router.

Shaping can be enabled for packets exiting an interface, subinterface, or individual VC. Policing can be performed both on packets entering an interface or exiting an interface.

2 Compare and contrast the actions that shaping and policing take when a packet exceeds a traffic contract.

Shaping delays packets when the shaping rate is exceeded. Policing either discards the packet, just transmits the packet, or it re-marks a QoS field before transmitting the packet.

3 If a router has a shaping tool configured, with a shaping rate of 256 kbps, and a Bc of 16,000 bits, what Tc value does the shaping tool use?

Because Tc = Bc/CIR, the answer is 16,000/256,000, or 62.5 ms.

4 Define the terms Tc, Bc, Be, and CIR.

Tc: time interval, measured in milliseconds, over which the committed burst (Bc) can be sent.

Bc: committed burst size, measured in bits. This is the amount of traffic that can be sent during every interval Tc. Typically also defined in the traffic contract.

Be: excess burst size, in bits. This is the number of bits beyond Bc that can be sent in the first Tc after a period of inactivity.

CIR: committed information rate, in bits per second, defines the amount of bandwidth that the provider has agree to provide as defined in the traffic contract.

Policing with CAR and CB Policer

5 List the command, with the correct syntax, that sets a policed rate of 512 kbps, a Bc of 1 second's worth of traffic, and a Be of an additional 0.5 second's worth of traffic, when using CAR. Do not assume any defaults; explicitly set the values in the command. You can choose any other settings needed for the command.

rate-limit input 512000 64000 96000 conform-action transmit exceed-action drop

6 Which policing tools allow for three categories of actions to take?

CB policer.

7 Explain the concept behind re-marking policed packets versus discarding the packets.

By re-marking the packets, you can increase the packet's likelihood of being dropped later. For instance, WRED reacts to the precedence or DSCP value, discarding certain marked values more aggressively. By re-marking, if no congestion occurs, the packet may still get through the network. If congestion does occur, the packet that the policer marked down has a greater chance of being dropped.

8 CB policing has been configured under subinterface s0/0.1. What **show** command lists statistics for CB policing behavior just for that subinterface?

show policy-map interface s0/0.1

Shaping with FRTS, GTS, DTS, and CB Shaping

9 Along with the **class-map**, **policy-map**, and **service-policy** commands, CB shaping requires one specific command that actually sets values used for the shaping function. List the command, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using CB shaping. Do not assume any defaults; explicitly set the values in the command.

shape 128000 8000 8000

10 Compare and contrast the use of the **class-map** and **map-class** commands in terms of how each is used by FRTS and CB shaping.

FRTS uses map-class to create a common set of shaping parameters, which are then applied to an interface, subinterface, or VC. FRTS does not use the class-map command for shaping. Likewise, CB shaping does not use the map-class command. It does, however, use the class-map command for defining classification options. Interestingly, FRTS can use the class-map command indirectly, when using CBWFQ or LLQ. (See Example 5-9 later in the book for a sample configuration.)

11 DTS has been configured under subinterface s0/0.1. What **show** command displays statistics for DTS behavior just for that subinterface?

show policy-map interface s0/0.1

12 Which of the traffic-shaping tools can be enabled on each VC on a Frame Relay multipoint subinterface?

FRTS.

Q&A

Shaping and Policing Concepts

1 Explain the points during the process of a single router receiving and forwarding traffic at which shaping and policing can be enabled on a router.

Shaping can be enabled for packets exiting an interface, subinterface, or individual VC. Policing can be performed both on packets entering an interface or exiting an interface.

2 Compare and contrast the actions that shaping and policing take when a packet exceeds a traffic contract.

Shaping queues packets when the shaping rate is exceeded. Policing either discards the packet, just transmits the packet, or it re-marks a QoS field before transmitting the packet.

3 Compare and contrast the effects that shaping and policing have on bandwidth, delay, jitter, and loss.

Shaping places packets into queues when the actual traffic rate exceeds the traffic contract, which causes more delay, and more jitter. Policing when making a simple decision to either discard or forward each packet causes more packet loss, but less delay and jitter for the packets that do make it through the network.

4 Describe the typical locations to enable shaping and policing in an internetwork.

Shaping is typically performed before sending packets into a network that is under some other administrative control. For instance, shaping is typically performed before sending packets from an enterprise into a service provider’s Frame Relay network. Policing, although supported as both an input and output function, is typically performed at ingress points, once again at the edge between two administrative domains.

5 Describe the reasons behind egress blocking in a Frame Relay network with a T1 access link at the main site, 128-kbps access links at each of 20 remote sites, with 64-kbps CIR VCs from the main site to each remote site.

Egress blocking can occur for frames leaving the Frame Relay network going to the main site, because the sum of the access rates of the 20 sites exceeds the access rate at the main site. Egress blocking occurs for packets leaving the Frame Relay network going to an individual remote site, because the access rate at the main site exceeds the access rate at each remote site.

6 If a router has a shaping tool configured, with a shaping rate of 256 kbps, and a Bc of 16,000 bits, what Tc value does the shaping tool use?

Because Tc = Bc/CIR, the answer is 16,000/256,000, or 62.5 ms.

If a router has a shaping tool configured, with a shaping rate of 512 kbps, and a Be of 16,000 bits, what Tc value does the shaping tool use?

Tc is not calculated based on Be, so you cannot know the answer with this limited amount of information. The formula to calculate Tc is as follows: Tc = Bc/CIR, where CIR is the shaping rate. If you know Bc and CIR, you can calculate the Tc value.

7 Define the terms Tc, Bc, Be, and CIR.

Tc: Time interval, measured in milliseconds, over which the committed burst (Bc) can be sent.

Bc: committed burst size, measured in bits. This is the amount of traffic that can be sent during every interval Tc. Typically also defined in the traffic contract.

Be: Excess burst size, in bits. This is the number of bits beyond Bc that can be sent in the first Tc after a period of inactivity.

CIR: committed information rate, in bits per second, defines the amount of bandwidth that the provider has agree to provide as defined in the traffic contract.

8 Explain the goal of the Da and Dc debt processes when using CAR.

CAR attempts to reduce congestion by getting the senders of the traffic to slow down, similar to WRED.

9 Describe the concept of traffic-shaping adaption, explaining the two triggers that cause shaping to adapt.

Adaption causes the shaper to reduce the shaping rate during congestion. Shaping can react to frames with the BECN bit set, or to Cisco Foresight congestion messages.

10 Describe the difference between interface output queues and shaping queues, and explain where the queues could exist on a router with 1 physical interface and 20 subinterfaces.

Output queues exist on the physical interface, and can be controlled with queuing tools such as CBWFQ, CAR, and WFQ. Shaping queues exist when traffic shaping is enabled; the shaping queue is associated with the particular instance of shaping. If shaping has been enabled on 20 subinterfaces on a single physical interface, for instance, 20 sets of shaping queues exist, all feeding into the single set of physical interface output queues.

Traffic Shaping

11 What do the following intialisms stand for? FRTS, GTS, DTS, and CB shaping

Frame Relay traffic shaping, generic traffic shaping, distributed traffic shaping, and class-based shaping.

12 List the command, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using GTS. Do not assume any defaults; explicitly set the values in the command.

traffic-shape rate 128000 8000 8000

13 Along with the **class-map**, **policy-map**, and **service-policy** commands, CB shaping requires one specific command that actually sets values used for the shaping function. List the command, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using CB shaping. Do not assume any defaults; explicitly set the values in the command.

shape 128000 8000 8000

14 Along with the **class-map**, **policy-map**, and **service-policy** commands, DTS requires one specific command that actually sets values used for the shaping function. List the command, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using DTS. Do not assume any defaults; explicitly set the values in the command.

shape 128000 8000 8000

15 Many commands are needed to configure FRTS, but the actual shaping parameters, such as CIR and Bc, are set under the **map-class frame-relay** command. List the **map-class** subcommands, with the correct syntax, that sets a shaped rate of 128 kbps, a Bc of 8000, and a Be of 8000, when using FRTS, and using multiple commands. Do not assume any defaults; explicitly set the values in the commands.

frame-relay bc 8000

frame-relay be 8000

frame-relay cir 128000

16 Compare and contrast the use of the **class-map** and **map-class** commands in terms of how each is used by FRTS and CB shaping.

FRTS uses map-class to create a common set of shaping parameters, which are then applied to an interface, subinterface, or VC. FRTS does not use the class-map command for shaping. Likewise, CB shaping does not use the map-class command. It does, however, use the class-map command for defining classification options. Interestingly, FRTS can use the class-map command indirectly, when using CBWFQ or LLQ. (Example 5-9 shows a sample configuration.)

17 Describe the context inside the configuration mode under which the **service-policy** command can be used to enable LLQ on an FRTS shaping queue. ("Context" means what part of configuration mode—for instance, global-configuration mode, interface configuration mode, and so on.)

You can use the service-policy command inside map-class configuration mode.

18 Explain the context inside the configuration mode under which the **service-policy** command can be used to enable LLQ on a CB shaping queue. ("Context" means what part of configuration mode—for instance, global-configuration mode, interface configuration mode, and so on.)

CB shaping requires a policy map, with class commands inside the policy map. Inside class configuration mode inside the CB shaping policy map, the service-policy command can refer to another policy map, which could enable LLQ for the class.

19 Explain the context inside the configuration mode under which the **service-policy** command can be used to enable LLQ on a GTS shaping queue. ("Context" means what part of configuration mode—for instance, global-configuration mode, interface configuration mode, and so on.)

GTS does not support LLQ as one of its shaping queues.

20 GTS has been configured under subinterface s0/0.1. What **show** command lists statistics for GTS behavior just for that subinterface?

show traffic-shape statistics interface s0/0.1

21 DTS has been configured under subinterface s0/0.1. What **show** command lists statistics for DTS behavior just for that subinterface?

show policy-map interface s0/0.1

22 CB shaping has been configured under subinterface s0/0.1. What **show** command lists statistics for CB shaping behavior just for that subinterface?

show policy-map interface s0/0.1

23 FRTS has been configured under subinterface s0/0.1. What **show** command lists statistics for FRTS behavior just for that subinterface?

show traffic-shape statistics interface s0/0.1

24 Which of the traffic-shaping tools can be enabled on PPP interfaces?

GTS, DTS, and CB shaping.

25 Which of the traffic-shaping tools can be enabled on each VC on a Frame Relay multipoint subinterface?

FRTS.

26 Which of the traffic-shaping tools support adaptive shaping?

FRTS, GTS, DTS, and CB shaping.

27 For which of the traffic-shaping tools does IOS perform shaping processing on 7500 VIP cards?

DTS.

28 Which of the traffic-shaping tools can classify traffic to shape subsets of the traffic on a subinterface?

GTS, DTS, and CB shaping.

29 Which shaping tools do not support WFQ and PQ to be used for the shaping queues?

GTS, CB shaping, DTS.

30 Which shaping tools do not allow WFQ or LLQ to be used for the interface output queuing at the same time as the shaping tool is enabled on the same interface?

FRTS.

Traffic-Policing Tools

31 Explain the use of the **continue** keyword as part of CAR policing actions, and the feature CAR provides through the keyword.

CAR allows nested, or cascaded, rate-limit commands. A single packet can match the classification logic of a rate-limit command, but if the action includes the continue keyword, the packet is compared to any additional rate-limit commands, until another one is matched. CAR supports the ability to police a superset of the traffic, and then police a subset of the traffic at a different rate, using the continue keyword.

32 CAR has been configured under subinterface s0/0.1. What **show** command lists statistics for CAR behavior just for that subinterface?

show interfaces s 0/0.1 rate-limit

Chapter 6

"Do I Know This Already?" Quiz

Congestion-Avoidance Concepts and RED

1 Describe the process of TCP slow start and discuss when it occurs.

TCP slow start governs the growth of the TCP congestion window after the window has been lowered in reaction to a packet drop. Slow start increases the window by one for each positively acknowledged packet received.

2 Describe the meaning of the term "global synchronization," and discuss what causes it.

Global synchronization describes a condition in which many TCP connections have their congestion windows lowered due to unacknowledged or lost segments at around the same instant in time. The connections all grow CWND at about the same rate, re-creating the same congestion levels again, causing more drops, which in turn reduces again the TCP congestion windows. Global synchronization is caused by a large number of packet drops in a very short period, typically caused by tail drops.

3 Define the meaning of the term "tail drop."

When a queue fills, and a new packet must be placed into the queue, the packet is dropped. Because the packet would be placed into the end, or tail, of the queue, it is called tail drop.

4 Does RED compare the actual queue depth or the average queue depth to queue thresholds when deciding whether it should discard a packet? Why this one, and not the other?

RED uses average queue depth. By using the average, rather than the actual queue depth, RED behaves more consistently, rather than more erratically, which helps prevent synchronization of TCP flows.

WRED

5 List the queuing tools that can enable WRED for use with some or all of their queues, effectively enabling WRED concurrently with the queuing tool.

CBWFQ and LLQ.

6 Describe how WRED "weights" packets.

WRED weights packets based on precedence or DSCP by assigning different minimum threshold, maximum threshold, and mark probability denominator values for each precedence or DSCP.

7 Taking as many defaults as possible, list the configuration commands needed to configure precedence-based WRED on interface S1/1.

```
interface serial 1/1
    random-detect
```

8 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based WRED inside **class class1**, inside **policy map my-policy**. (You can assume that the CBWFQ configuration has already been completed, and you just entered global configuration mode. Assume that you need just to enable WRED in **class class1**.)

```
policy-map my-policy
    class class1
     random-detect dscp-based
```

FRED

9 Identify the most significant difference between FRED operation and WRED operation.

FRED discards packets more aggressively for flows that try to consume a relatively large amount of space in a queue, whereas WRED does not. By doing so, UDP flows can be prevented from taking too much of the space in the queue.

10 List the three categories of flows defined by FRED, and identify which category has its packets discarded most aggressively.

Robust flows, fragile flows, and nonadaptive flows. Nonadaptive flows are discarded most aggressively.

11 Taking as many defaults as possible, list the configuration commands needed to configure precedence-based FRED on interface S1/1.

```
interface serial 1/1
    random-detect flow-based
```

By using the random-detect flow-based command, Cisco IOS Software adds the random-detect command automatically.

12 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based FRED on interface S1/1.

```
interface serial 1/1
    random-detect dscp-based
    random-detect flow-based
```

If you had just used the random-detect flow-based command, and not the random-detect dscp-based command, Cisco IOS Software would have added the random-detect command automatically, which enables precedence-based WRED.

Q&A

Congestion-Avoidance Concepts and Random Early Detection (RED)

1 Describe the function of the congestion window in TCP, and how it is changed as a result of packet loss.

The TCP congestion window, or CWND, is one of two windowing mechanisms that limit TCP senders. CWND can be split in half as a result of packet loss, slowing the sending rate. CWND can also be *slammed shut* to the size of a single segment in some cases.

2 Identify the two TCP windowing mechanisms, and describe when each is used.

The TCP congestion window, or CWND, and the TCP receiver window, are the two windowing mechanisms. The lower of the two values is used at all times.

3 Describe the process of TCP slow start, and when it occurs.

TCP slow start governs the growth of the TCP congestion window after the window has been lowered in reaction to a packet drop. Slow start increases the window by one for each positively acknowledged packet received.

4 Describe the process of TCP congestion avoidance, and when it occurs.

TCP congestion avoidance then governs the growth of the TCP congestion window after the slow start phase has been completed.

5 Describe the meaning of the term “global synchronization,” and discuss what causes it.

Global synchronization describes a condition in which many TCP connections have their congestion windows lowered due to unacknowledged or lost segments at around the same instant in time. The connections all grow CWND at about the same rate, re-creating the same congestion levels again, causing more drops, which in turn reduces again the TCP congestion windows. Global synchronization is caused by a large number of packet drops in a very short period, typically the result of tail drops.

6 Define the meaning of the term “tail drop.”

When a queue fills, and a new packet must be placed into the queue, the packet is dropped. Because the packet would be placed into the end, or tail, of the queue, it is called tail drop.

7 Define the meaning of the term “TCP starvation.”

When packets are dropped, TCP connections slow down, but UDP flows do not slow down. UDP packets can consume a disproportionate amount of queue space as a result, which could get to the point that the TCP connections simply get little or no queue space; this is called TCP starvation.

8 Does RED compare the actual queue depth or the average queue depth to queue thresholds when deciding whether it should discard a packet? Why this one, and not the other?

RED uses average queue depth. By using the average, rather than the actual queue depth, RED behaves more consistently, rather than more erratically, which helps prevent synchronization of TCP flows.

9 Describe how RED uses actual queue depth to calculate average queue depth. Do not list the formula, but just describe the general idea.

RED calculates the average by adjusting the previously calculated average a small amount based on the current actual queue depth. By default, the current queue depth is weighted at about .2 percent in the formula.

10 Assume the RED minimum threshold is 20, the maximum threshold is 40, and the mark probability denominator is 10. What must be true for RED to discard all new packets?

The average queue depth must be above 40.

11 Assume the RED minimum threshold is 20, the maximum threshold is 40, and the mark probability denominator is 10. What must be true for RED to discard 5 percent of all new packets?

The average queue depth must be at 30. Because the discard percentage grows linearly from 0 percent to 10 percent (in this case), between average queue depth of 20 through 40, average queue depth of 30 would mean that the discard percentage had grown to 5 percent.

12 Define how RED uses the mark probability denominator. Give one example.

RED calculates the discard percentage based on the formula 1/MPD. For instance, with an MPD of 20, the discard percentage is 1/20, or 5 percent.

13 Define the term “exponential weighting constant.” If the value is lowered compared to the default setting of 9, how does RED behave differently?

The exponential weighting constant defines how quickly the average queue depth changes, by determining how much the actual queue depth affects the rolling average queue depth. If EWC is lowered, the average changes more quickly, because the formula weights the current actual queue depth more than before.

Weighted RED (WRED)

14 Spell out the words represented by the initialisms RED, WRED, and FRED.

Random Early Detection (RED), Weighted Random Early Detection (WRED), Flow-Based Weighted Random Early Detection (FRED).

15 List the queuing tools that can be concurrently supported on an interface when WRED has been enabled directly on a serial interface.

FIFO Queuing only.

16 Identify the most important difference between RED operation and WRED operation.

WRED weights its discard decisions based on precedence or DSCP, whereas RED ignores precedence and DSCP.

17 Describe how WRED "weights" packets.

WRED weights packets based on precedence or DSCP by assigning different minimum threshold, maximum threshold, and mark probability denominator values for each precedence or DSCP.

18 List the queuing tools that can enable WRED for use with some or all of their queues, effectively enabling WRED concurrently with the queuing tool.

CBWFQ and LLQ.

19 What command enables you to look at WRED drop statistics when WRED is configured inside an MQC class?

show policy-map interface

20 Taking as many defaults as possible, list the configuration commands needed to configure precedence-based WRED on interface S1/1.

```
interface serial 1/1
    random-detect
```

21 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based WRED on interface S1/1.

```
interface serial 1/1
    random-detect dscp-based
```

22 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based WRED inside **class class1**, inside **policy map my-policy**. (You can assume that the CBWFQ configuration has already been completed, and you just entered global configuration mode. Assume that you need just to enable WRED in **class class1**.)

```
policy-map my-policy
    class class1
     random-detect dscp-based
```

23 List the command needed to set the minimum threshold to 25, the maximum threshold to 50, and the mark probability denominator to 4, for precedence 2.

random-detect precedence 2 25 50 4

24 What **show** command lists detailed statistics about random drops on interface S1/1?

show queueing interface s1/1

Flow-Based WRED (FRED)

25 List the queuing tools that can be concurrently supported on an interface when FRED has been enabled directly on a serial interface.

FIFO Queuing only.

26 Identify the most significant difference between FRED operation and WRED operation.

FRED discards packets more aggressively for flows that try to consume a relatively large amount of space in a queue, whereas WRED does not. By doing so, UDP flows can be prevented from taking too much of the space in the queue.

27 List the three categories of flows defined by FRED, and identify which category has its packets discarded most aggressively.

Robust flows, fragile flows, and nonadaptive flows. Nonadaptive flows are discarded most aggressively

28 Describe how FRED prevents TCP starvation.

FRED prevents hungry UDP flows from consuming too much of a queue, which in turn prevents TCP starvation.

29 List the queuing tools that can enable FRED for use with some or all queues, effectively enabling FRED concurrently with the queuing tool.

None.

30 Suppose that an interface has five active flows, with Flow 1 consuming 20 queue entries, and a maximum queue size of 40. Describe the terms “maximum per-flow queue depth,” and give an example of how it is calculated with this example. Use default values for any information not stated in the question.

The maximum per-flow queue depth defines the dividing line between fragile flows and nonadaptive flows with FRED. First, the average per-flow queue depth is calculated as 40/5 in this case, or 8. The average per-flow queue depth is multiplied by a scaling factor, which defaults to 4, giving a maximum per-flow queue depth of 32. Flow 1 would be considered a fragile flow, because it has fewer than 32 packets in the queue.

31 Taking as many defaults as possible, list the configuration commands needed to configure precedence-based FRED on interface S1/1.

```
interface serial 1/1
    random-detect flow-based
```

By using the random-detect flow-based command, Cisco IOS Software adds the random-detect command automatically.

32 Taking as many defaults as possible, list the configuration commands needed to configure DSCP-based FRED on interface S1/1.

```
interface serial 1/1
    random-detect dscp-based
    random-detect flow-based
```

If you had just used the random-detect flow-based command, and not the random-detect dscp-based command, Cisco IOS Software would have added the random-detect command automatically, which enables precedence-based WRED.

Chapter 7

"Do I Know This Already?" Quiz

Compression

1 Describe what is compressed, and what is not compressed, when using payload compression. Be as specific as possible regarding headers and data.

Payload compression does not compress the data-link header and trailer, but it does compress all the headers and data between the two. Specifically, the IP, TCP, UDP, RTP headers as appropriate, and the user data, are compressed.

2 Describe what is compressed, and what is not compressed, when using TCP header compression. Be as specific as possible regarding headers and data.

IP packets that also have TCP headers are compressed. The compression algorithm does not compress the data-link header or trailer. It does compress both the IP and TCP headers. It does not compress any user data that follows the TCP header.

3 Describe what is compressed, and what is not compressed, when using RTP header compression. Be as specific as possible regarding headers and data.

IP packets that also have RTP headers are compressed. The compression algorithm does not compress the data-link header or trailer. It does compress the IP, UDP, and RTP headers. It does not compress any user data that follows the RTP header.

4 List the three point-to-point payload compression options available in IOS.

Stacker, Microsoft Point-to-Point Compression (MPPC), and Predictor.

5 When TCP header compression is used, what is the range of sizes of the part of the frame that can be compressed, and what is the range of sizes for this field of the frame after compression?

TCP header compression compresses the 20-byte IP header and 20-byte TCP header, with the combined field size of 40 bytes. The compressed field will be between 3 and 5 bytes.

6 When RTP header compression is used, what is the range of sizes of the part of the frame that can be compressed, and what is the range of sizes for this field of the frame after compression?

RTP header compression compresses the 20-byte IP header, 8-byte UDP header, and 12-byte RTP header, with the combined field size of 40 bytes. The compressed field will be between 2 and 4 bytes.

Link Fragmentation and Interleave

7 List the words represented by the abbreviation LFI.

Link fragmentation and interleaving.

8 To achieve a 20-ms serialization delay on a 128-kbps link, how long can the fragments be?

The formula is max-delay * bandwidth, which is .02 * 128,000 = 2560 bits, or 320 bytes.

9 What queuing tools can you enable directly on a serial interface when using multilink Point-to-Point Protocol with link fragmentation and interleaving (MLP LFI), as compared to when you are just using PPP?

All the queuing tools available for use with PPP are also available with MLP LFI. There are no restrictions.

10 What command can you use to determine the fragment size used for MLP LFI? What is the only parameter of the command?

The ppp multilink fragment-delay command sets the maximum serialization delay in milliseconds. IOS calculates the fragment size using the formula max-delay * bandwidth.

11 What command enables FRF and sets the fragment size?

The frame-relay fragment *fragment_size* command.

12 What other QoS feature for Frame Relay must you enable when you configure FRF.12 as well?

Frame Relay traffic shaping (FRTS).

Q&A

Compression Tools

1 Describe what is compressed, and what is not compressed, when using payload compression. Be as specific as possible regarding headers and data.

Payload compression does not compress the data-link header and trailer, but it does compress all the headers and data between the two. Specifically, the IP, TCP, UDP, RTP headers as appropriate, and the user data, are compressed.

2 Describe what is compressed, and what is not compressed, when using TCP header compression. Be as specific as possible regarding headers and data.

IP packets that also have TCP headers are compressed. The compression algorithm does not compress the data link header or trailer. It does compress both the IP and TCP headers. It does not compress any user data that follows the TCP header.

3 Describe what is compressed, and what is not compressed, when using RTP header compression. Be as specific as possible regarding headers and data.

IP packets that also have RTP headers are compressed. The compression algorithm does not compress the data-link header or trailer. It does compress the IP, UDP, and RTP headers. It does not compress any user data that follows the RTP header.

4 List the three point-to-point payload compression options available in IOS.

Stacker, Microsoft Point-to-Point Compression (MPPC), and Predictor.

5 Suppose a packet is sent across a network with no compression. Later, a packet of the exact same size and contents crosses the network, but payload compression is used on the one serial link in the network. Describe the difference in bandwidth and delay in the network between these two packets.

The compressed packet experiences longer delay, all other things being equal, because the compression algorithm does take some time to execute. The compressed packet consumes less bandwidth on the link, because the compression algorithm has made it smaller. An overall reduction in queue sizes can occur as well, which can actually then reduce delay and jitter.

6 List the three options for Frame Relay payload compression in IOS. Which of the three is not Cisco proprietary?

Packet by packet, data stream, and FRF.9 are the three options. FRF.9 is not Cisco proprietary.

7 Which payload compression tool in IOS supports Link Access Procedure, Balanced (LAPB), High-Level Data Link Control (HDLC) and Point-to-Point Protocol (PPP) encapsulations?

Stacker.

8 How much bandwidth should a G.729 call require over Frame Relay, and how much should be required with cRTP?

A single G.729 call requires 26.4 kbps over Frame Relay, but it only needs 11.2 kbps using cRTP.

9 When TCP header compression is used, what is the range of sizes of the part of the frame that can be compressed, and what is the range of sizes for this field of the frame after compression?

TCP header compression compresses the 20-byte IP header and 20-byte TCP header, with the combined field size of 40 bytes. The compressed field will be between 3 and 5 bytes.

10 When RTP header compression is used, what is the range of sizes of the part of the frame that can be compressed, and what is the range of sizes for this field of the frame after compression?

RTP header compression compresses the 20-byte IP header, 8-byte UDP header, and 12-byte RTP header, with the combined field size of 40 bytes. The compressed field will be between 2 and 4 bytes.

11 To configure Stacker payload compression on a point-to-point link, what command(s) is used, and in what configuration modes?

The compress stac interface subcommand.

12 To configure Stacker payload compression on a point-to-point Frame Relay subinterface, what command(s) is used, and in what configuration modes?

The frame-relay compress stac subinterface subcommand.

13 To configure FRF.9 payload compression on a point-to-point Frame Relay subinterface, what command(s) is used, and in what configuration modes?

The frame-relay payload-compress FRF9 stac subinterface subcommand.

14 What command lists compression statistics for payload compression on a point-to-point link?

The show compress command.

15 What command lists compression statistics for payload compression on a Frame Relay point-to-point subinterface?

The show compress command.

16 To configure TCP header compression on a point-to-point Frame Relay subinterface, what command(s) is used, and in what configuration modes?

The frame-relay ip tcp header-compression subinterface subcommand.

17 To configure RTP header compression on a point-to-point link, what command(s) is used, and in what configuration modes?

The ip rtp header-compression interface subcommand.

LFI Tools

18 List the words represented by the abbreviation LFI.

Link fragmentation and interleaving.

19 Describe the main motivation for LFI tools in relation to the support of data, voice, and video traffic.

LFI tools interleave some packets between the fragments of other packets. Voice and two-way video traffic are particularly sensitive to delay. LFI reduces the delay for voice and video packets by interleaving voice and video packets between fragments of the data packets.

20 To achieve a 20-ms serialization delay on a 128-kbps link, how long can the fragments be?

The formula is max-delay * bandwidth, which is .02 * 128,000 = 2560 bits, or 320 bytes.

21 To achieve a 10-ms serialization delay on a 64-kbps link, how long can the fragments be?

The formula is max-delay * bandwidth, which is .01 * 64,000 = 640 bits, or 80 bytes.

22 Suppose that a 1500-byte packet exits a 56-kbps serial interface, and LFI is not used. How long is the serialization delay?

The formula is packet length/link speed, which is 1500 * 8/56,000, or .214 seconds. The units used in the formula are bits and bits per second, respectively.

23 Which queuing tools can you enable directly on a serial interface when using multilink Point-to-Point Protocol with link fragmentation and interleaving (MLP LFI), as compared to when you are just using PPP?

All the queuing tools available for use with PPP are also available with MLP LFI. There are no restrictions.

24 Which queuing tools can you enable for shaping queues when using FRF.12? Which ones actually interleave the traffic?

Weighted Fair Queuing (WFQ), Class-Based WFQ (CBWFQ), Low Latency Queuing (LLQ), and IP RTP Priority can be enabled, with LLQ and IP RTP Priority actually interleaving packets.

25 Explain the popularly stated scheduling logic, which is consistent with the Cisco QoS courses, that defines how FRF.12 determines which packets can be interleaved in front of fragments of other packets.

Unfragmented packets can be interleaved, with fragmented packets not being interleaved.

26 Explain the scheduling logic used by MLP LFI to determine which packets can be interleaved in front of fragments of other packets.

MLP LFI does not define scheduling logic. Instead, it relies on the scheduler of the queuing tool enabled on the interface to decide which packets to send next. If LLQ were used, for instance, packets from the low-latency queue would be interleaved in front of packets from other queues.

27 Suppose a 1500-byte packet arrives and needs to be sent over an MLP bundle that has two active links. LFI has not been configured. Which link does the packet flow across to achieve MLP load balancing?

MLP fragments the packet into two equal-sized fragments, and sends one over one link, and one over the other.

28 What command can you use to determine the fragment size used for MLP LFI? What is the only parameter of the command?

The ppp multilink fragment-delay command sets the maximum serialization delay in milliseconds. IOS calculates the fragment size using the formula max-delay * bandwidth.

29 What command enables the interleaving feature of MLP LFI?

The ppp multilink interleave command.

30 What commands list counters for the number of interleaved packets using MLP LFI?

The show queue and show interfaces commands.

31 What other QoS feature for Frame Relay must you enable when you also configure FRF.12?

Frame Relay traffic shaping (FRTS).

32 What command enables FRF and sets the fragment size?

The frame-relay fragment *fragment_size* command.

33 What command lists counters for the numbers of packets and bytes that were fragmented and unfragmented by FRF.12?

The show frame-relay fragment interface command.

34 What command lists counters for the numbers of packets and bytes that would have been sent if FRF.12 fragmentation had not been performed?

The show frame-relay fragment interface command.

35 How do FRF.12 and FRF.11-C differ in terms of deciding which packets can be interleaved, and which cannot?

FRF.11-C allows only voice frames to be interleaved. FRF.12 actually interleaves packets that were in a low-latency queue (either with LLQ or IP RTP Priority) in a shaping queue. However, the Cisco courses (upon which the exam is based) states that the FRF.12 classification logic simply interleaves unfragmented frames.

Chapter 8

"Do I Know This Already?" Quiz

1 Why is call admission control needed in an environment where LLQ has been properly implemented?

LLQ provides classification, marking, and prioritization of voice packets, but does not differentiate between voice streams. The priority queue can become overburdened with a larger number of voice conversations than expected, leading to degradation in all voice conversations. CAC provides the protection needed to guarantee the quality of service for established voice conversations by limiting the number of simultaneous voice conversations allowed in the priority queue. In short, LLQ protects voice from data, whereas CAC protects voice from voice.

2 How does a channel-associated signaling circuit, such as E&M or T1 CAS, react to call admission control?

CAS circuits carry signaling in band. This forces the call to be hairpinned back on another channel in the event that a CAC mechanism indicates that the resources are not available to proceed with the call.

3 Name four possible measures that a CAC mechanism can take in the event that the resources are not available to proceed with the call.

- **Reroute call to alternate LAN/WAN link.**
- **Reroute call to alternate PSTN link.**
- **Return call to originating PBX for rerouting.**
- **Return a reorder tone, or fast-busy, to the caller.**

4 What is the definition of local-based CAC?

Local CAC mechanisms base the availability of network resources on local nodal information such as the state of the outgoing LAN or WAN link.

5 What Cisco IOS command is used to enable CAC on a VoFR network?

frame-relay voice-bandwidth *bandwidth-in-bps*

6 What Cisco IOS command is used to enable physical DS0 limitation?

Physical DS0 limitation is not an IOS command; it is a design methodology that limits the number of physical DS0 connections into the gateway.

7 What is the definition of measurement-based CAC?

Measurement-based CAC techniques look into the packet network to gauge the current state of the network.

8 What is the difference between an SAA packet and a ping packet?

SAA packets can be IP/TCP, or IP/UDP, or (most common) IP/UDP/RTP packets with sizes, frequency, and ToS all set to accurately synthesize the protocol being observed; whereas a ping packet is an ICMP best-effort packet that does not resemble a voice packet in size or protocol.

9 What Cisco IOS command is used to allow the destination node to participate in measurement-based CAC?

rtr responder

10 What is the definition of resource-based CAC?

Resource-based CAC calculates the resources required to facilitate and protect the call on each leg the call traverses toward the destination and then attempts to reserve these resources for use by the voice conversation.

11 For a gatekeeper to provide CAC, what must be configured for each link that requires protection?

Each link must be associated with a zone for a gatekeeper to provide CAC.

12 What is IntServ and how does it work?

The integrated services (IntServ) model includes provisions for best-effort traffic, real-time traffic, and controlled-link sharing. In the IntServ model, an application requests a specific level of service from the network before sending a data flow. If the requested level of service can be achieved from the network, a reservation is placed in each intermediary router to guarantee the requested level of service for the life of the flow.

13 Name the messages used by RSVP to provide resource reservation and CAC.

Path and resv message

14 What level of service must be in place to provide CAC?

Guaranteed-delay

Q&A

Call Admission Control Concepts

1 Why is CAC needed in an environment where LLQ has been properly implemented?

LLQ provides classification, marking, and prioritization of voice packets, but does not differentiate between voice streams. The priority queue can become overburdened with a larger number of voice conversations than expected, leading to degradation in all voice conversations. CAC provides the protection needed to guarantee the quality of service for established voice conversations by limiting the number of simultaneous voice conversations allowed in the priority queue. In short, LLQ protects voice from data, whereas CAC protects voice from voice.

2 Name four possible measures that a CAC mechanism can take in the event that the resources are not available to proceed with the call.

- **Reroute call to alternate LAN/WAN link.**
- **Reroute call to alternate PSTN link.**
- **Return call to originating PBX for rerouting.**
- **Return a reorder tone, or fast-busy, to the caller.**

3 How does a common channel signaling circuit, such as PRI, react to CAC?

CCS circuits have the capability to send out-of-band signaling to tear down the circuit and reroute the call in the event that a CAC mechanism indicates that the resources are not available to proceed with the call.

4 How does a channel-associated signaling circuit, such as E&M or T1 CAS, react to CAC?

CAS circuits carry signaling in band. This forces the call to be hairpinned back on another channel in the event that a CAC mechanism indicates that the resources are not available to proceed with the call.

5 What is the difference between payload bandwidth and the bandwidth per call required for a voice conversation?

Payload bandwidth is the amount of bandwidth required to digitize the analog voice stream, whereas the bandwidth per call for a voice conversation takes the Layer 2 overhead, IP, UDP, and RTP headers, and the payload into consideration.

Local-Based CAC

6 What is the definition of local-based CAC?

Local CAC mechanisms base the availability of network resources on local nodal information such as the state of the outgoing LAN or WAN link.

7 Name three of the five available local-based CAC mechanisms.

- **Physical DS0 limitation**
- **Max-Connections dial-peer configuration**
- **VoFR, Voice-Bandwidth configuration**
- **Trunking condition**
- **Local Voice Busyout**

8 What Cisco IOS command is used to enable physical DS0 limitation?

Physical DS0 limitation is not an IOS command; it is a design methodology that limits the number of physical DS0 connections into the gateway.

9 What Cisco IOS command is used to enable CAC on a VoFR network?

frame-relay voice-bandwidth *bandwidth-in-bps*

10 What type of circuit, CAS or CSS, will Local Voice Busyout (LVB) be most beneficial for?

A CAS circuit is better suited to use LVBO because a CAS circuit can reroute the call in the event that the resources are not available to proceed with the call.

Measurement-Based CAC

11 What is the definition of measurement-based CAC?

Measurement-based CAC techniques look into the packet network to gauge the current state of the network

12 What is the difference between LVBO and Advanced LVBO?

Whereas LVBO provides for busyout based on local conditions of the originating gateway, AVBO adds the capability to trigger an SAA probe to one or more configured IP destinations.

13 What is the difference between an SAA packet and a ping packet?

SAA packets can be IP/TCP, or IP/UDP or (most common) IP/UDP/RTP packets with sizes, frequency, and ToS all set to accurately synthesize the protocol being observed; whereas a ping packet is an ICMP best-effort packet that does not resemble a voice packet in size or protocol.

14 What measurements are gathered by SSA probes for PSTN fallback?

Delay and loss measurements

15 What Cisco IOS command is used to allow the destination node to participate in measurement based CAC

rtr responder

Resources-Based CAC

16 What is the definition of resource-based CAC?

Resource-based CAC calculates the resources required to facilitate and protect the call on each leg the call traverses toward the destination and then attempts to reserve these resources for use by the voice conversation.

17 Which resources can resource availability indication (RAI) currently monitor?

DS0 and DSP resources.

18 Which Cisco CallManager call-processing model utilizes location-based CAC?

A centralized call-processing model.

19 For a gatekeeper to provide CAC, what must be configured for each link that requires protection?

Each link must be associated with a Zone for gatekeeper to provide CAC.

20 What is IntServ and how does it work?

The integrated services (IntServ) model includes provisions for best-effort traffic, real-time traffic, and controlled-link sharing. In the IntServ model, an application requests a specific level of service from the network before sending a data flow. If the requested level of service can be achieved from the network, a reservation is placed in each intermediary router to guarantee the requested level of service for the life of the flow.

21 Name the messages used by RSVP to provide resource reservation and CAC.

Path and resv message

22 What level of service must be in place to provide CAC?

Guaranteed-delay

23 Which RSVP profile should be configured to classify traffic for the priority queue within LLQ?

The ip rsvp pq-profile uses the voice-like profile to classify voice traffic.

Chapter 9

"Do I Know This Already?" Quiz

QoS Management Tools

1 What do the following acronyms stand for? IPM, QPM, SAA, QDM

Internetwork Performance Monitor, QoS Policy Manager, Service Assurance Agent, and QoS Device Manager.

2 What QoS management tool(s) enables you to define QoS policies, after which it creates the appropriate configurations, and deploys the configurations to identified network devices?

QPM

3 Which QoS management tool(s) includes as a primary feature the capability to graph real-time information about bit/byte/packet rates and packet loss?

QDM, IPM, and QPM

4 What QoS management tool(s) generates packets, called probes, and measures various performance attributes, such as delay and jitter?

SAA

5 What QoS management tool(s) includes as a primary function the capability to troubleshoot and isolate network-wide performance issues by viewing the individual components in a complete end-to-end path through the network?

IPM

6 What QoS management tool(s) is embedded into a router?

QDM

QoS Design

7 List Step 1 in the QoS design process, and give a short summary of what the step includes.

Step 1 is to "determine customer priorities/QoS policy." It includes discussions with appropriate personnel to determine what level of quality each type of traffic should get.

8 List Step 3 in the QoS design process, and give a short summary of what the step includes.

Step 3 is to "implement policy." It includes actual choices of QoS tools, and then the configuration of the tools.

9 What are the ITU G.114 and Cisco recommendations for maximum one-way delay for voice calls?

The ITU recommends 150 ms, with Cisco stating it can be as high as 200 ms.

10 Summarize the Cisco recommendations for marked values for voice and video traffic.

Cisco suggests that voice payload be marked with DSCP EF, CoS 5, and/or precedence 5; video payload with DSCP AF41, CoS 4, and/or precedence 4; and voice and video signaling with DSCP AF31, CoS 3, and/or precedence 3.

11 Describe the Cisco recommendation for queuing voice and video traffic.

Cisco recommends using LLQ, placing the voice payload into the low-latency queue, and the video payload into a separate low-latency queue. If the IOS does not support LLQ, Cisco suggests enabling IP RTP Priority.

12 List Step 4 in the QoS design process, and give a short summary of what the step includes.

Step 4 is to "monitor the network." It includes using the QoS management tools to see historical trends, and the process of reviewing the reports versus policies to allow the four-step process to repeat, improving the quality of the QoS implementation over time.

Q&A

QoS Management

1 What do the following acronyms stand for? IPM, QPM, SAA, QDM

Internetwork Performance Monitor, QoS Policy Manager, Service Assurance Agent, and QoS Device Manager.

2 What QoS Management tool is actually a component of the CiscoWorks2000 Routed WAN Management Solution?

IPM.

3 What QoS management tool is free?

QDM.

4 What QoS management tool(s) enables you to define QoS policies, after which it creates the appropriate configurations, and deploys the configurations to identified network devices?

QPM.

5 Which QoS management tool(s) includes as a primary feature the capability to graph real-time information about bit/byte/packet rates and packet loss?

QDM, IPM, and QPM.

6 Describe the two primary components of SMS and their general roles.

SMS includes the Service Level Manager, to which the user connects. The SMS Collection Managers (CMs) actually collect the statistics, which are then asynchronously fed back to the SLM.

7 What QoS management tool(s) generates packets, called probes, and measures various performance attributes, such as delay and jitter?

SAA.

8 What QoS management tool(s) includes as a primary function the capability to produce historical performance reports as compared to service-level agreements?

SMS.

9 What QoS management tool(s) includes as a primary function the capability to troubleshoot and isolate network-wide performance issues by viewing the individual components in a complete end-to-end path through the network?

IPM.

10 Does QDM or QPM support more devices?

QPM.

11 Among QPM, QDM, SMS, and IPM, which tool(s) originally had a limited capability to create real-time or historical graphs of performance?

QPM.

12 What QoS management tool(s) is embedded into a router?

QDM.

13 Which QoS management tools are available in CiscoWorks2000 bundles?

IPM and SMS.

QoS Design

14 List Step 1 in the QoS design process, and give a short summary of what the step includes.

Step 1 is to "determine customer priorities/QoS policy." It includes discussions with appropriate personnel to determine what level of quality each type of traffic should get.

15 List Step 2 in the QoS design process, and give a short summary of what the step includes.

Step 2 is to "characterize the network." It includes a network audit, which is used to plan any needed upgrades of bandwidth, and to determine all the details needed in order to successfully classify the traffic in that particular network.

16 List Step 3 in the QoS design process, and give a short summary of what the step includes.

Step 3 is to "implement policy." It includes actual choices of QoS tools, and then the configuration of the tools.

17 List Step 4 in the QoS design process, and give a short summary of what the step includes.

Step 4 is to "monitor the network". It includes using the QoS management tools to see historical trends, and the process of reviewing the reports versus policies to allow the four-step process to repeat, improving the quality of the QoS implementation over time.

18 What are the ITU G.114 and Cisco recommendations for maximum one-way delay for voice calls?

The ITU recommends 150 ms, with Cisco stating it can be as high as 200 ms.

19 What delay component represents the most controllable variable delay in a voice call?

Queuing delay.

20 How much bandwidth does a G.729 call require if ignoring the Layer 2 header/trailer? How much bandwidth does it require if using Frame Relay?

The call requires 24 kbps discounting the data-link framing; adding the Frame Relay headers and trailers, the call takes 26.4 kbps.

21 Summarize the Cisco recommendations for marked values for voice and video traffic.

Cisco suggests that voice payload be marked with DSCP EF, CoS 5, and/or precedence 5; video payload with DSCP AF41, CoS 4, and/or precedence 4; and voice and video signaling with DSCP AF31, CoS 3, and/or precedence 3.

22 Describe the Cisco recommendation for queuing voice and video traffic.

Cisco recommends using LLQ, placing the voice payload into the low-latency queue, and the video payload into a separate queue. If the IOS does not support LLQ, Cisco suggests enabling IP RTP Priority.

23 Summarize the Cisco recommendations for tuning shaping parameters when shaping must be enabled on VCs carrying voice.

Cisco suggests choosing configuration settings such that Bc is equal to CIR / 100, which will make Tc equal to 10 ms. Be is equal to zero to prevent bursting over the CIR. If adaptive shaping is used, Cisco suggests that the mincir be set equal to the CIR value with no adaptive shaping being used.

Chapter 10

“Do I Know This Already?” Quiz

1 What is instantaneous buffer overrun?

Instantaneous buffer overrun occurs when a switch port TX queue fills for an instant in time causing packet loss, which can adversely affect real-time applications.

2 What is meant by the term “1p1q4t?”

This term denotes an interface that has a priority queue, a standard queue, and four drop thresholds in the standard queue.

3 What packet or frame marking(s) can a Layer 2 switch use to prioritize traffic?

CoS only.

4 What must be done for a Layer 2 switch to properly classify a packet received from a host across the WAN?

The router must map the IP precedence or IP DCSP value on the packet into a CoS value for the Layer 2 switch to properly classify the packet.

5 What is a drop threshold?

Drop thresholds define the amount of the total Layer 2 buffer use that must be reached before a specified class of traffic is dropped.

6 What is a trust boundary, and where should it be set?

A trust boundary is the first point in your network where you trust the CoS or ToS markings. Typically this point is on the IP Phones or access layer switches.

7 Explain why the first packet in a flow may not retain the proper CoS values when a Catalyst 6500 is configured with a PFC.

If this flow is not in the flow cache of the PFC or the CEF FIB of the PFC2, the first packet of the flow is forwarded to the MSFC for a routing decision. After a routing decision has been made, the fist packet in the flow is routed to the correct interface.

This rewrites the CoS value to 0. The routing decision populates the flow cache or the CEF FIB, causing subsequent packets to be switched and thereby retain the original CoS value.

8 What is the difference between Hybrid mode and Native mode in a 6500 series switch?

Hybrid mode refers to the Catalyst operating system, whereas Native mode refers to an IOS operating system on the Catalyst 6500 series switches.

9 What queue and threshold is recommended for call control traffic on a Catalyst 6500?

2q1t.

10 Which series of Ethernet line cards are preferred in the Catalyst 6500 series for QoS? Why?

The 65*xx* series of Ethernet line cards are preferred due to the increase in buffer size and the addition of a priority queue.

11 Which supervisor engine is preferred on the Catalyst 4500/4000 series? Why?

The Supervisor Engine III or the Supervisor Engine IV are preferred because of the addition of a priority queue, three standard queues, and the ability to use QoS access lists.

12 What trust state are the ports of a 4500 with a Supervisor III in when QoS is enabled?

Untrusted.

13 In which queue number does the priority queue reside on a Catalyst 4500 or 4000 with a Supervisor III Engine?

The priority queue in a Catalyst 4500 or 4000 with a Supervisor III Engine resides in Queue 3.

14 What keyword(s) is used to enable the priority queue on a Catalyst 4500 or 4000 with a Supervisor III module?

tx-queue 3

priority high

15 In which queue number does the priority queue reside on a Catalyst 3550?

The priority queue in a Catalyst 3550 resides in Queue 4.

16 What command enables QoS on the Catalyst 3550?

mls qos

17 Name the four methods a Catalyst 3550 can use to classify traffic.

CoS values

IP precedence values

IP DSCP values

IP access Lists

18 What command enables QoS on the Catalyst 3524?

QoS is enabled by default in a Catalyst 3524. No command is needed.

Q&A

LAN QoS Concepts

1 What is instantaneous buffer overrun?

Instantaneous buffer overrun occurs when a switch port TX queue fills for an instant causing packet loss, which can adversely affect real-time applications. In a VoIP conversation, for example, 40 ms of congestion causes an audible clip in the conversation.

2 What is meant by the term "1p1q4t?"

This term denotes an interface that has a priority queue, a standard queue, and four drop thresholds.

3 What packet or frame marking(s) can a Layer 2 switch use to prioritize traffic?

CoS only.

4 What packet or frame marking(s) can a Layer 3 switch use to prioritize traffic?

CoS, IP precedence, or IP DSCP.

5 What must be done for a Layer 2 switch to properly classify a packet received from a host across the WAN?

The router must map the IP precedence or IP DCSP value on the packet into a CoS value for the Layer 2 switch to properly classify the packet.

6 For a Layer 2 switch to receive a CoS value from a router, what must exist?

An 802.1Q trunk must exist between the router and switch for the CoS value to be present.

7 What type of router interface supports an 802.1Q trunk to a Layer 3 switch?

An Ethernet subinterface must exist on the router to form an 802.1Q trunk with a Layer 2 switch.

8 What is a drop threshold?

Drop thresholds define the amount of the total Layer 2 buffer use that must be reached before a specified class of traffic is dropped.

9 What is a trust boundary, and where should it be set?

A trust boundary is the first point in your network where you trust the CoS or ToS markings. Typically this point is on the IP Phones or access layer switches.

Catalyst 6500 Series of Switches

10 Name three of the five fields that the Policy Feature Card (PFC) rewrites to perform Layer 3 switching.

Layer 2 (MAC) Destination Address

Layer 2 (MAC) Source Address

Layer 3 IP Time-to-Live (TTL)

Layer 3 Checksum

Layer 2 (MAC) Checksum (also called the frame checksum or FCS)

11 Explain why the first packet in a flow may not retain the proper CoS values when a Catalyst 6500 is configured with a PFC.

If this flow is not in the flow cache of the PFC or the CEF FIB of the PFC2, the first packet of the flow is forwarded to the MSFC for a routing decision. After a routing decision has been made, the fist packet in the flow is routed to the correct interface. This rewrites the CoS value to 0. The routing decision populates the flow cache or the CEF FIB, causing subsequent packets to be switched and thereby retain the original CoS value.

12 What is the difference between Hybrid mode and Native mode?

Hybrid mode refers to the Catalyst operating system, whereas Native mode refers to an IOS operating system on the Catalyst 6500 series switches.

13 What command is used on a Catalyst 6500 running in Hybrid mode to place ports 2/1 through 2/10 in VLAN 10?

set port auxiliaryvlan 2/1-10 10

14 What command enables QoS on a 6500 running in Native mode?

mls qos

15 What queue and threshold is recommended for call control traffic on a Catalyst 6500?

2q1t

16 Which series of Ethernet line cards are preferred in the Catalyst 6500 series for QoS? Why?

The 65*xx* series of Ethernet line cards are preferred due to the increase in buffer size and the addition of a priority queue.

Catalyst 4500/4000 Series of Switches

17 Which supervisor engine is preferred on the Catalyst 4500/4000 series? Why?

The Supervisor Engine III or the Supervisor Engine IV are preferred because of the addition of a priority queue, three standard queues, and the ability to use QoS access lists.

18 In which queue number does the priority queue reside on a Catalyst 4500 or 4000 with a Supervisor II Engine?

There is no priority queue on a Catalyst 4500 or 4000 with a Supervisor II Engine.

19 In which queue number does the priority queue reside on a Catalyst 4500 or 4000 with a Supervisor III Engine?

The priority queue in a Catalyst 4500 or 4000 with a Supervisor III Engine resides in Queue 3.

20 What keyword(s) is used to enable the priority queue on a Catalyst 4500 or 4000 with a Supervisor III module?

tx-queue 3

priority high

21 Name the four methods a Catalyst 4500 with a Supervisor III uses to classify traffic.

CoS values

IP precedence values

IP DSCP values

IP ACLs

22 What command enables you to rewrite a the CoS value of a PC attached to an IP Phone with Catalyst IOS switches?

switchport priority extend cos

23 What trust state are the ports if a 4500 with a Supervisor III in when QoS is enabled?

Untrusted.

24 What is dynamic buffer limiting?

DBL is an automated algorithm that manages misbehaving traffic flows by tracking the buffering for each traffic flow and limiting the flow if excessive demands are placed on a buffer.

Catalyst 3550/3524 Series of Switches

25 In which queue number does the priority queue reside on a Catalyst 3550?

The priority queue in a Catalyst 3550 resides in Queue 4.

26 What command enables QoS on the Catalyst 3550?

mls qos

27 What trust state are the ports of a 3550 in when QoS is enabled?

Untrusted.

28 Name the four methods a Catalyst 3550 can use to classify traffic.

CoS values

IP precedence values

IP DSCP values

IP ACLs

29 What command enables QoS on the Catalyst 3524?

QoS is enabled by default in a Catalyst 3524. No command is needed.

30 When is a GigaStack GBIC module an acceptable QoS choice for cascading Catalyst 3524 switches together?

The GigaStack modules are acceptable for QoS under either of the two following conditions:

- **The Catalyst 3524 switches are cascaded using both ports of the GigaStack modules causing half-duplex connections; however, no real-time applications are present on the network.**
- **The Catalyst 3524 switches are cascaded using only a single port of the GigaStack modules, resulting in full-duplex connections. Real-time applications can be present on the network.**

APPENDIX B

Topics on the CCIP QoS Exam

The CCIP QoS exam covers some topics that are not covered on the DQOS exam. As stated in the Introduction to this book, the core chapters of this book focus on the more widely popular DQOS exam. However, some of you may be studying for the less-popular QoS exam, so this appendix covers many of the topics on the QoS exam, but not on the DQOS exam.

Keep in mind that, when the exams change, you should check both www.cisco.com and www.ciscopress.com for more information. The Cisco website is the main authoritative source for information about all Cisco exams. Additionally, the next time these exams change, the authors of this book will post information at www.ciscopress.com/1587200589 about the changes. Included there will be information about the specific changes to the exams, and how best to use your book to pass the revised exam(s).

This appendix includes two sections. One covers a classification and marking tool, and the other section covers several queuing tools. Each section includes explanations and configuration examples as appropriate. Questions are included at the end of each *section*, not at the end of the appendix, so that you can consider questions for each general topic area. The answers are included in the chapter for your convenience.

Foundation Topics

For those of you studying for both exams, you could begin by studying for, and passing, the DQOS exam. To study, you could just attack each chapter of this book in succession, and ignore this appendix. After you have passed the DQOS exam, you can then study using this appendix, and pick up the additional information you need.

For those of you just studying for the CCIP QoS exam, not only should you plan to study the topics in this chapter, you can also ignore some topics in the core chapters of this book. The Introduction to this book suggests some options for your study plan when studying just for the QoS exam. Table B-1 summarizes the major topics from Chapters 1 through 10 that you can ignore if you are preparing just for the QoS exam.

Table B-1 *Preparing for the QOS Exam*

Book Chapter	Suggested Deviation as Compared with DQOS Study Plan
1	None; read and study the entire chapter.
2	None; read and study the entire chapter.
3	Read and study all of Chapter 3; then come to this appendix and read the section "QoS Policy Propagation with BGP (QPPB)." QPPB is the one classification and marking tool that is on the QoS exam, but not on the DQOS exam.
4	Read and study Chapter 3 entirely; then come to this appendix and read the section "Congestion Management (Queuing)." This section covers several queuing tools not covered on the DQOS exam.
5	None; read and study the entire chapter.
6	None; read and study the entire chapter.
7	None; read and study the entire chapter.
8	Ignore all topics covering voice CAC, but do read all coverage of RSVP.
9	You do not need to read this chapter; the topics are not covered on the QoS exam.
10	You do not need to read this chapter; the topics are not covered on the QoS exam.

Any time you study for a Cisco exam, you should always pull a list of exam topics from www.cisco.com for the exam. Use these exam topics as your primary guide to know what to study for the exam; after all, the exam questions typically test whether you can do what the exam topic describes. As mentioned in the Introduction to this book, you can also check www.ciscopress.com/1587200589, where the authors of this book will post hints and tips when exam changes occur.

Classification and Marking

Chapter 3, “Classification and Marking,” covers classification and marking tools. QoS policy propagation with BGP (QPPB) is the one tool in this category that is on the QoS exam, but not on the DQOS exam.

QoS Policy Propagation with BGP (QPPB)

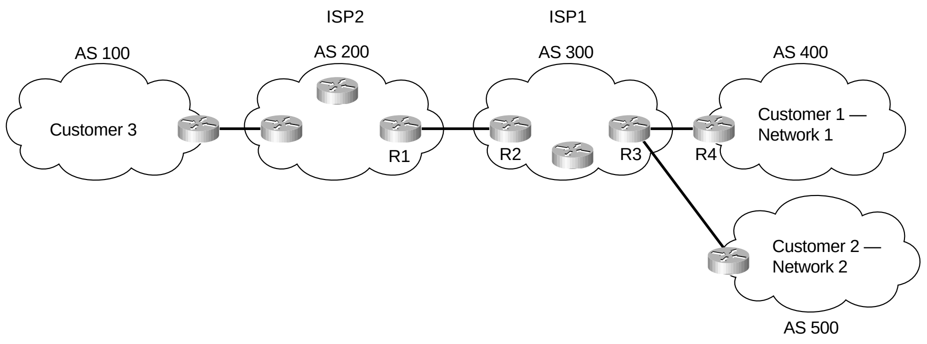
QoS policies that differentiate between different types of traffic can be most easily defined for a single enterprise network. For instance, one enterprise may want to treat important web traffic, not-important web traffic, and all other data traffic as three different classes, and use different classes for voice and video traffic. For the Internet, however, a single QoS policy would never work. Differentiated services (DiffServ), which was designed specifically to address QoS over the Internet, defines the role of ingress boundary nodes to re-mark traffic as it enters a different DiffServ domain, essentially changing the differentiated services code point (DSCP) to reflect the QoS policies of each respective DiffServ domain. This practice allows each DiffServ domain to set its own QoS policies.

QoS policies that classify traffic based on the characteristics of the flow—voice, video, different data applications, and so on—can be defined and used in enterprises and by service providers. Enterprises can afford to be more selective, because a single group can often set the QoS policies. For instance, an enterprise could classify based on the IP addresses of some mission-critical servers. QoS policies for Internet service providers (ISPs) tend to be less specific than those for an enterprise, because ISPs have many customers. However, ISPs can still implement QoS policies based on the type of traffic contained in the packet.

ISPs may want a QoS policy just to prefer one customer’s traffic over another. In Figure B-1, for instance, consider ISP 1, which has two customers. Customer 1 has agreed to pay a premium for its Internet service, in return for ISP 1 agreeing to provide better latency and delay characteristics for the traffic. Customer 2 keeps paying the same amount as always, and still gets best-effort service.

The QoS tools only need to differentiate between Customer 1 and Customer 2 traffic to support this policy. So, for packets flowing from right to left, if the source IP address is an IP address in Customer 1’s network, the packet might be marked with precedence 4, for instance. Similarly, when packets flow left to right, these same tools could examine the destination IP address, and if it’s part of Customer 1’s network, precedence 4 could be marked. Packets to or from Customer 2 could be marked with precedence 0.

Figure B-1 *QoS Policy Based on Customer—Customer 1 and Customer 2*



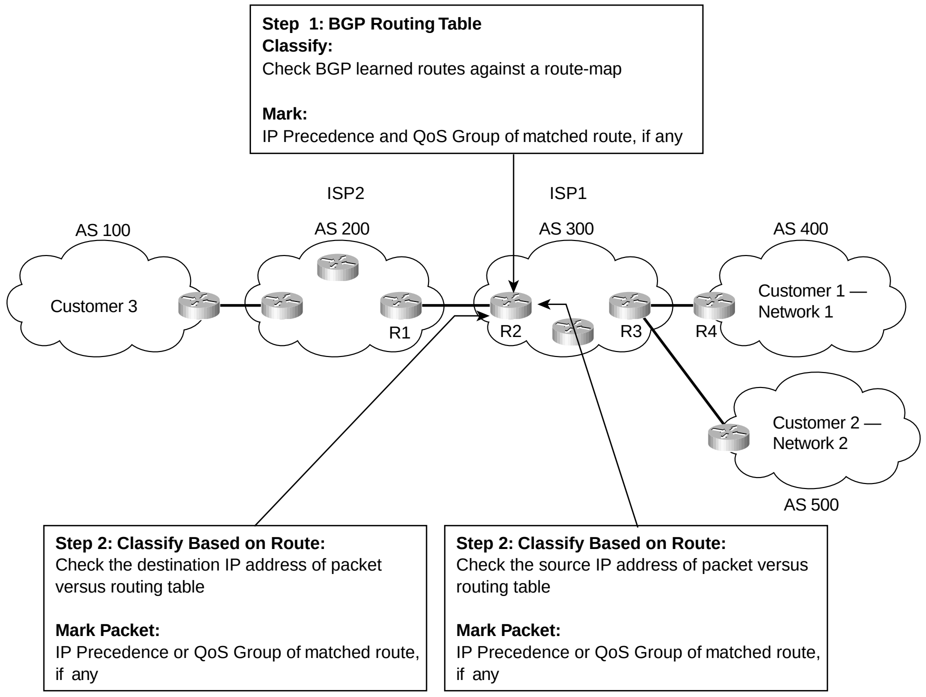
Class-based (CB) marking, policy-based routing (PBR), and committed access rate (CAR) could perform the necessary marking to support premium and best-effort customer services. However, each of these three tools has some negative side effects. For all three tools, that classification would require an IP ACL for matching the packets, for all packets. For an ISP with many customers, however, classifying and marking packets based on referencing ACLs for a large number of packets may induce too much overhead traffic. Suppose further that ISP 1 and ISP 2 agree to support each other's premium and best-effort customers in a similar manner. The two ISP's would have to continually exchange information about which networks are premium, and which are not, if they are using IP ACLs to classify the traffic. Additionally, when new customers are added, ISP 1 may be waiting on ISP 2 to update their QoS configuration before the desired level of service is offered to the new customer.

To overcome the two issues—the scalability of classifying based on ACLs, and the administrative problems of just listing the networks that need premium services—QPPB was created. QPPB allows marking of packets based on an IP precedence or QoS group value associated with a Border Gateway Protocol (BGP) route. For instance, the BGP route for Customer 1's network, Network A, could be given a BGP path attribute that both ISP 1 and ISP 2 agree should mean that this network receives better QoS service. Because BGP already advertises the routes, and the QoS policy is based on the networks described in the routes, QPPB marking can be done more efficiently than with the other classification and marking tools.

Figure B-2 shows the basic process in action. In this example, R3 is configured to use QPPB, although it would likely be used in several places around the network.

QPPB follows two steps: marking routes, and then marking packets based on the values marked on the routing entries. BGP routing information includes the network numbers used by the various customers, and other BGP path attributes. Because Cisco has worked hard over the years to streamline the process of table lookup in the routing table, to reduce per-packet processing for the forwarding process, QPPB can use this same efficient table-lookup process to reduce classification and marking overhead.

Figure B-2 *QPPB—Basic Components*


Step 1: BGP Routing Table
Classify:
Check BGP learned routes against a route-map
Mark:
IP Precedence and QoS Group of matched route, if any
ISP2
ISP1
AS 100
AS 200
AS 300
AS 400
Customer 3
R1
R2
R3
R4
Customer 1 — Network 1
Customer 2 — Network 2
AS 500
Step 2: Classify Based on Route:
Check the destination IP address of packet versus routing table
Mark Packet:
IP Precedence or QoS Group of matched route, if any
Step 2: Classify Based on Route:
Check the source IP address of packet versus routing table
Mark Packet:
IP Precedence or QoS Group of matched route, if any

For reference, Tables B-2 and B-3 summarize the QPPB configuration and exec commands, respectively.

Table B-2 *Configuration Command Reference for QPPB*

Command	Mode and Function
route-map *map-tag* [**permit** | **deny**] [*sequence-number*]	Global command; creates a route map entry
match ip address {*access-list-number* | *access-list-name*} [... *access-list-number* | ... *access-list-name*]	Route-map subcommand; used to match IP packets based on parameters matchable with an IP ACL
match length *minimum-length maximum-length*	Route-map subcommand; used to mach IP packets based on their length
set ip precedence *number* | *name*	Route-map subcommand; sets IP precedence vale using the decimal number of name.

continues

Table B-2 *Configuration Command Reference for QPPB (Continued)*

Command	Mode and Function
set ip qos-group group-id	Route-map subcommand; sets a group ID in the routing table for classification throughout the network.
table-map *map-name*	BGP subcommand; used to modify values related to BGP learned routes, including precedence and QoS group
ip community-list *community-list-number* {**permit** | **deny**} *community-number*	Global command; used to create a community list, which matches values in the BGP community string
ip as-path access-list *access-list-number* {**permit** | **deny**} *as-regexp*	Global command; used to create an autonomous system (AS) path list, which matches values in the autonomous system number (ASN) path BGP attribute
ip bgp-community new-format	BGP subcommand; used to make IOS use the *AA*:*NN* format for community values, with *AA* being the ASN, and *NN* being a user-defined value
bgp-policy ip-prec-map	Interface subcommand; enables QPPB for packets entering the interface, marking IP precedence
bgp-policy ip-qos-map	Interface subcommand; enables QPPB for packets entering the interface, marking QoS group

Table B-3 *Exec Command Reference for QPPB*

Command	Function
show ip bgp	Shows BGP routing table
show ip route *prefix*	Shows IP routing table entries, including precedence values
show ip bgp community-list *community-list-number*	Lists configuration of the community list
show ip cef *network*	Shows the Cisco Express Forwarding (CEF) Forwarding Information Base (FIB), including the marked QoS values

QPPB can be a confusing topic. The rest of this section discusses more detail about how QPPB works and how to configure it. One key to understanding QPPB, in spite of some of the detail, is to keep these two key points in mind as you read the following sections:

- QPPB classifies BGP routes based on the BGP routes' attributes, and marks BPG routes with an IP precedence or QoS group value.
- QPPB classifies packets based on the associated routing table entries, and marks the packets based on the marked values in the routing table entry.

NOTE Because QPPB involves quite a few detailed concepts and configuration, some of the true details of how QPPB works are glossed over during the initial discussions. These details are explained at the end of this section in the subsection titled "QPPB: The Hidden Details."

QPPB Route Marking: Step 1

QPPB allows routers to mark packets based on information contained in the routing table. Before packets can be marked, QPPB first must somehow associate a particular marked valued with a particular route. QPPB, as the name implies, accomplishes this task using BGP. This first step can almost be considered as a separate classification and marking step by itself, because BGP routes are classified, based on information that describes the route, and marked with some QoS value.

The classification feature of QPPB can examine many of the BGP path attributes. The two most useful BGP attributes for QPPB are the autonomous system number (ASN) sequence, referred to as the autonomous system path, and the community string. The autonomous system path contains the ordered list of ASNs, representing the ASNs between a router and the autonomous system of the network described in the route. In Figure B-2, R1 receives a BGP update for Network 1, listing ASNs 300 and 400 in the autonomous system path and a BGP update for Network 2, listing ASNs 300 and 500 in the autonomous system path. QPPB can be used to mark the route to Network 1 (Customer 1) with one precedence value, while marking the route to Network 2 (Customer 2) with another precedence value, based on the autonomous system path received for the route to each customer.

The community attribute provides a little more control than does the autonomous system path. The autonomous system path is used to avoid routing loops, and the contents of the autonomous system path changes when aggregate routes are formed. The community attribute, however, allows the engineer to essentially mark any valid value. For instance, R3 could set the community attribute to 10:200 for the route to Network 1, and advertise that route toward the left side of the network diagram. Other routers could then use QPPB to classify based on the community attribute of 10:200, and assign the appropriate precedence value to the route to Network 1. QPPB configuration would essentially create logic as follows: "If the community attribute contains 10:200, mark the route with precedence 4."

Example B-1 lists the QPPB configuration just for marking the route based on the autonomous system number. With this configuration, no packets are marked, because the QPPB configuration is not complete. (The complete configuration appears in the next section.) QPPB is a two-step process, and Example B-1 just shows the configuration for the first step.

Example B-1 *QPPB Route Marking with BGP Table Map: R2*

```
router bgp 300
 table-map mark-prec4-as400
!
route-map mark-prec4-as400 10
 match as-path 1
 set ip precedence 4
!
route-map mark-prec4-as400 20
 set ip precedence 0
!
ip as-path access-list 1 permit _400_
```

This example shows R2's configuration for QPPB. (Note that the entire BGP configuration is not shown, just the configuration pertinent to QPPB.) The **table-map** BGP router subcommand tells IOS that, before adding BGP routes to the routing table, it should examine a route map called mark-prec4-as400. Based on the **match** and **set** commands in the route map, when BGP adds routes to the routing table, it also associates either precedence 4 or precedence 0 with each route.

The route map has two clauses—one that matches routes that have autonomous system 400 anywhere in the autonomous system path sequence attribute, and a second clause that matches all routes. Clause 10 matches ASN 400 by referring to autonomous system path ACL 1, which matches any autonomous system path containing ASN 400, and sets the precedence to 4 for those routes. Clause 20 matches all packets, because no specific **match** command is configured, and sets the precedence to 0.

QPPB Per-Packet Marking: Step 2

After QPPB has marked routes with IP precedence or QoS group values, the packet marking part must be performed. After the packets have been marked, traditional QoS tools can be used to perform queuing, congestion avoidance, policing, and so on, based on the marked value.

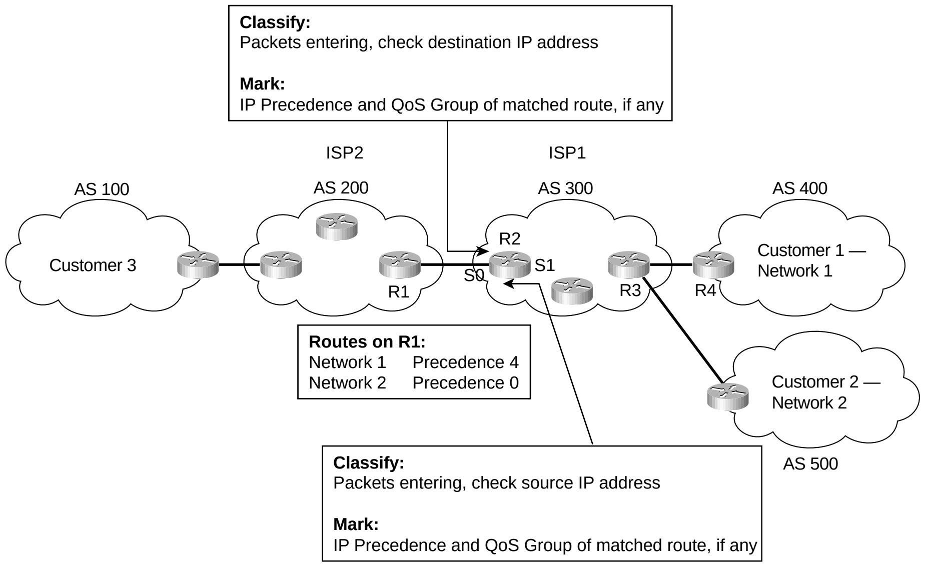
QPPB's packet-marking logic flows as follows:

1. Process packets entering an interface.
2. Match the destination or source IP address of the packet to the routing table.
3. Mark the packet with the precedence or QoS group value shown in the routing table entry.

The three-step logic for QPPB packet marking follows the same general flow as the other classification and marking tools; in this case, however, the classification options, and the marking options, are quite limited. QPPB packet classification is based on the routing table entry that matches the packet, and QPPB packet marking just marks the packet with the same value found marked in the route.

Figure B-3 shows with the same network, but with the marking logic on R2 shown.

Figure B-3 *QPPB Per-Packet Marking Logic*


Classify:
Packets entering, check destination IP address
Mark:
IP Precedence and QoS Group of matched route, if any
ISP2
ISP1
AS 100
AS 200
AS 300
AS 400
Customer 3
R2
S1
S0
R1
R3
R4
Customer 1 —
Network 1
Routes on R1:
Network 1 Precedence 4
Network 2 Precedence 0
Customer 2 —
Network 2
AS 500
Classify:
Packets entering, check source IP address
Mark:
IP Precedence and QoS Group of matched route, if any

QPPB allows for marking of packets that have been sent to Customer 1, and for marking packets that have been sent by Customer 1. For packets entering R2's S0 interface, for instance, the packet is going toward Customer 1, so the destination IP address is in Network 1. Therefore, the QPPB logic on R2's S0 should compare the packet's destination IP address to the routing table; if the appropriate QoS field has been set in the route, the packet is marked with the same value. That takes care of packets passing through R3 that are headed to Customer 1.

For packets that Customer 1 has sent, going from right to left in the figure, QPPB on R2 can still mark the packets. These packets typically enter R2's S1 interface, however, and the packets have a *source* IP addresses in Network 1. To associate these packets with Network 1, QPPB examines the routing table entry that matches the packet's source IP address. This match of the routing table is not used for packet forwarding; it is used only for finding the precedence or the QoS group value to set on the packet. In fact, the table lookup for destination addresses does not replace the normal table lookup for forwarding the packet, either. Because the routing table entry for Network 1 has IP precedence set to 4, QPPB marks these packets with precedence 4.

Example B-2 shows the completed configuration on R2, with the additional configuration for per-packet marking highlighted.

Example B-2 *QPPB: Completed Example on R2*

```
ip cef
!
Router bgp 300
 table-map mark-prec4-as400
!
route-map mark-prec4-as400 10
 match as-path 1
 set ip precedence 4
!
route-map mark-prec4-as400 20
 set ip precedence 0
!
ip as-path access-list 1 permit _400_
!
interface Serial0
 bgp-policy destination ip-prec-map
!
interface serial1
 bgp-policy source ip-prec-map
```

The **bgp-policy** interface subcommand enables QPPB for packets entering the interface. The **destination** or **source** keyword identifies whether QPPB should perform table lookup on the packets' destination or source addresses, respectively. On S0, the **destination** keyword is used, because the packets entering S0 presumably are going toward Customer 1. Conversely, on S1 the **source** keyword is used, because the packets entering S1 presumably were sent by Customer 1. Finally, the **ip-prec-map** keyword implies that the precedence should be set based on the routing table entry, and not the QoS group.

QPPB Sample Configuration

QPPB can classify based on both the autonomous system path and the community string. BGP considers the autonomous system path as a well-known mandatory path attribute; therefore, in the earlier examples, R3 could just examine the autonomous system path. Conversely, BGP considers the community string to be an optional transitive attribute—which means that the community string does not have to be set, and is not set without some additional configuration causing it to be set.

Example B-3 shows the same network, with the same goal of giving Customer 1 premium service. In this example, however, the BGP community attribute is used. The community attribute is set by R3, for routes received from Customer 1 via BGP. With the community attribute set, other routers can use it for classifying the BGP routes and marking the routes with precedence 4. The example shows R3's QPPB configuration. Example B-3 lists the configuration on R3, and Example B-4 lists the configuration on R2.

Example B-3 *QPPB Sample Based on BGP Community: R3 Configuration*

```
router bgp 300
 neighbor 192.168.1.1 remote-as 400
 neighbor 192.168.1.1 route-map set-comm in
 neighbor 192.168.2.2 remote-as 300
 neighbor 192.168.2.2 send-community
!
route-map set-comm permit 10
 set community 4:50
```

Example B-4 *QPPB Sample Based on BGP Community: R2 Configuration*

```
ip cef
!
router bgp 300
 table-map mark-prec4-comm
!
route-map mark-prec4-comm permit 10
 match community 1
 set ip precedence 4
!
route-map mark-prec4-comm permit 20
 set ip precedence 0
!
ip community-list 1 permit 4:50
!
interface Serial0
 bgp-policy destination ip-prec-map
!
interface serial1
 bgp-policy source ip-prec-map
```

In Example B-3, R3 has just set the community string to 4:50 for BGP routes learned from neighbor 192.168.1.1, which is a router at Customer 1. To set the community, BGP uses **route-map set-comm** based on the **neighbor 192.168.1.1 route-map set-comm in** command. This route map contains 1 clause, which matches all routes because there is no **match** command in clause 10, and sets the community string to 4:50. IOS BGP does not forward the community attribute by default, so the **neighbor 192.168.2.2 send-community** command is needed to make R3 send the community string to R2, whose BGP ID is 192.168.2.2. So, R3 has set all incoming routes from R4 with community 4:50, and includes the community attribute in the updates sent to R2.

Example B-4 shows the configuration for QPPB on R2. The configuration is similar to Example B-2, with the highlighted sections pointing out the added or changed configuration. The **table-map** BGP router subcommand still directs BGP to mark the routes with precedence 4, but this time using a new route map, mark-prec4-comm. This route map uses two clauses. The first clause, clause 10, matches the community set by R3 by referring to IP community list 1 using the **match community 1** command. The community list, created in the single global command **ip community-list 1 permit 4:50**, just matches all BGP routes whose community string contains 4:50. Route map mark-prec4-comm sets IP precedence 4 for BGP routes that match the community sting. The second route map clause, clause 20, matches all routes because no explicit **match** statement is configured, and sets the IP precedence to 0 for these routes.

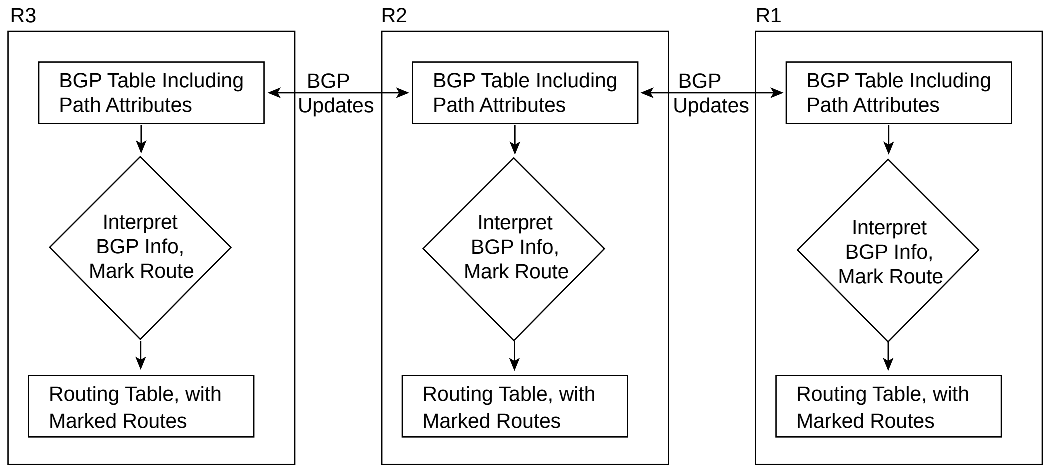
The packet-marking function, as opposed to the route-marking function, is enabled by the **bgp-policy interface** subcommands, which are exactly the same as saw in Chapter 3 in Example 3-10.

QPPB: The Hidden Details

As mentioned earlier, QPPB confuses most people the first time they learn about it. Therefore, you should understand a bit more about it. The first aspect of QPPB you should understand pertains to what BGP updates contain in support of QPPB, and the second aspect of QPPB you should understand is what really happens when QPPB marks a route.

First, BGP updates do *not* include the IP precedence or QoS group value inside the BGP update. QPPB reacts to the information in a normal BGP update to perform QoS marking of BGP routes, and then in turn performs packet marking based on the marked routes. In other words, BGP RFCs did not add any specification for adding a QoS marking field to the information inside the update. Therefore, to mark based on BGP routes, QPPB uses preexisting fields in the BGP update, such as the autonomous system path and the community attribute. In fact, the BGP-4 RFCs added the community attribute to provide a flexible field for marking BGP routes for future unforeseen purposes, such as QPPB. Figure B-4 depicts the general idea:

Figure B-4 *BGP Updates and QPPB Route Marking: No QoS-Marked Fields in BGP Update*


R3
R2
R1
BGP Table Including Path Attributes
BGP Updates
BGP Table Including Path Attributes
BGP Updates
BGP Table Including Path Attributes
Interpret BGP Info, Mark Route
Interpret BGP Info, Mark Route
Interpret BGP Info, Mark Route
Routing Table, with Marked Routes
Routing Table, with Marked Routes
Routing Table, with Marked Routes

When marking IP precedence in packets, QPPB marks the same field already covered in depth in this chapter—the first 3 bits of the ToS byte. When QPPB marks the QoS group, it actually marks a header that is added to the packet when passing through a 7500, GSR, or ESR series router. However, QPPB must mark the route first, and then mark the packet based on the route that matches the source or destination IP address in the packet. To understand what mark the route really means, you must take at least a cursory look at Cisco Express Forwarding (CEF).

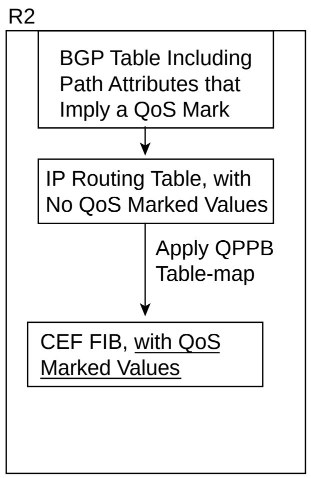
IOS provides several different processing paths in software for forwarding packets. Process switching is one of those paths, and is the most processor-intensive path. Fast switching is another switching path still in use today. CEF is yet another switching or forwarding path, and CEF has been designed to be very efficient. Other switching paths have also been added over the years, some specific to particular hardware models. The one thing all these optimized forwarding paths have in common is that they optimize for the forwarding process by streamlining two functions: the process of matching the correct route in the routing table, and the process of building and adding the new data-link header to the packet.

CEF optimizes forwarding by creating a new table that includes entries for the routes in the routing table. This table is called the *Forwarding Information Base* (FIB). The FIB optimizes the process of locating a route by performing a table lookup in the FIB rather than the less-efficient table lookup of the routing table. In other words, CEF switching crunches the routing table into the FIB, and then uses the FIB to make the forwarding decisions. (This in itself is somewhat of an oversimplification of CEF; for more detail, refer to Vijay Bollapragada's *Inside Cisco IOS Software Architecture* [Cisco Press, 2000].)

CEF optimizes the creation of new data-link headers by creating a table that contains the new data-link header associated with each next-hop IP address in the FIB. By doing so, when FIB table lookup is complete, the header can be added to the packet with little processing.

When QPPB marks a route, it actually marks either or both of the two fields inside each entry in the FIB. The FIB contains IP precedence and QoS group fields in order to support QPPB. Therefore, when CEF crunches the routing table to create FIB entries, when QPPB is configured, the appropriate FIB precedence and QoS group fields are set. Figure B-5 shows the general idea.

Figure B-5 *"Marking the Route": Marking the CEF FIB*


R2
BGP Table Including Path Attributes that Imply a QoS Mark
IP Routing Table, with No QoS Marked Values
Apply QPPB Table-map
CEF FIB, with QoS Marked Values

QPPB Summary

QPPB provides convenient classification and marking when BGP is already in use. Because QPPB bases classification decisions on BGP information, however, the classification process should consume less overhead per packet than the other generalized classification and marking tools.

Q&A for QPPB

1 When configuring QPPB to set a precedence value of 4, describe the two places in which precedence 4 is set. Specifically, note when it is marked in routing updates that flow between routers, in user packets that flow between routers, and in tables in a single router.

Answer: QPPB marks the CEF FIB internally to a router. In this case, the FIB entry for some routes is marked with precedence 4, and user packets in turn are marked precedence 4 based on the FIB entry. The BGP routing updates never include the marked value of precedence 4.

2 Imagine a router for which QPPB is enabled on interface E0. Describe the process that QPPB performs on this packet. Refer to the information that must be available to QPPB before the packet can be marked. Assume IP precedence is marked.

Answer: QPPB processes the incoming packet. Depending on the configuration, QPPB performs a table lookup on the source or destination IP address of the packet, as compared with the CEF FIB. The FIB entry may have a precedence value assigned to it. If so, QPPB marks the packet with that precedence value. QPPB must have already processed BGP routing information, and marked the FIB entry, before the packet can be marked.

3 What command enables QPPB packet marking on an interface, based on the source IP address of the packet, for packets entering the interface?

Answer: The bgp-policy source ip-prec-map interface subcommand enables QPPB for packets entering the interface.

4 What command enables QPPB packet marking on an interface, based on the destination IP address of the packet, for packets exiting the interface?

Answer: QPPB only processes packets entering an interface, so this is a trick question. No command could perform the action of enabling a QPPB packet marking for packets exiting an interface.

5 What RFC defines the BGP NLRI extensions that support QoS marking in BGP updates?

Answer: This is a trick question. The marked value is not carried in the BGP update; QPPB looks at existing BGP attributes, such as autonomous system path and community, and makes a choice to mark the routing (FIB) entry based on those existing BGP fields.

6 What BGP path attributes can QPPB use for classification?

Answer: QPPB can match the BGP autonomous system path and community fields.

7 What fields can QPPB mark?

Answer: QPPB can mark the IP Precedence or the QoS Group fields, and the IP ToS bits.

Congestion Management (Queuing)

Chapter 4, “Congestion Management,” of this book covers congestion management tools—in other words, queuing tools. The QoS exam covers several variations of Weighted Fair Queuing (WFQ) and the Modified Deficit Round-Robin (MDRR) tool that is useful on Gigabit Switch Routers (GSRs). These topics are covered next. Also included here is coverage of Priority Queuing (PQ) and Custom Queuing (CQ) configuration, which is also found only on the QoS exam.

PQ Configuration

PQ configuration resembles access-control list (ACL) configuration, except the result is to queue a packet rather than discarding it. Global commands are used to define the logic for classifying packets by matching header fields, and an interface subcommand is used to enable PQ on an interface. Example configurations for PQ follow Tables B-4 and B-5, which list the configuration and exec commands related to PQ, respectively.

Table B-4 *Configuration Command Reference for PQ*

Command	Mode and Function
priority-list *list-number* **protocol** *protocol-name* {**high** | **medium** | **normal** | **low**} *queue-keyword keyword-value*	Global configuration command; creates a priority list entry, sets classification first by protocol, and then further refines it with keywords. Also identifies the queue into which to place matched packets. (See Example B-5 for a list of keywords.)
priority-list *list-number* **interface** *interface-type interface-number* {**high** | **medium** | **normal** | **low**}	Global configuration command; creates a priority list entry, sets classification based on the incoming interface, and then identifies the queue into which to place matched packets.
priority-list *list-number* **queue-limit** [*high-limit* [*medium-limit* [*normal-limit* [*low-limit*]]]]	Global configuration command; sets queue lengths for the 4 queues, respectively.
priority-list *list-number* **default** {**high** | **medium** | **normal** | **low**}	Changes the default queue for packets that are not matched.
priority-group *list-number*	Interface subcommand; enables a priority list for packets exiting an interface.

Table B-5 *Exec Command Reference for PQ*

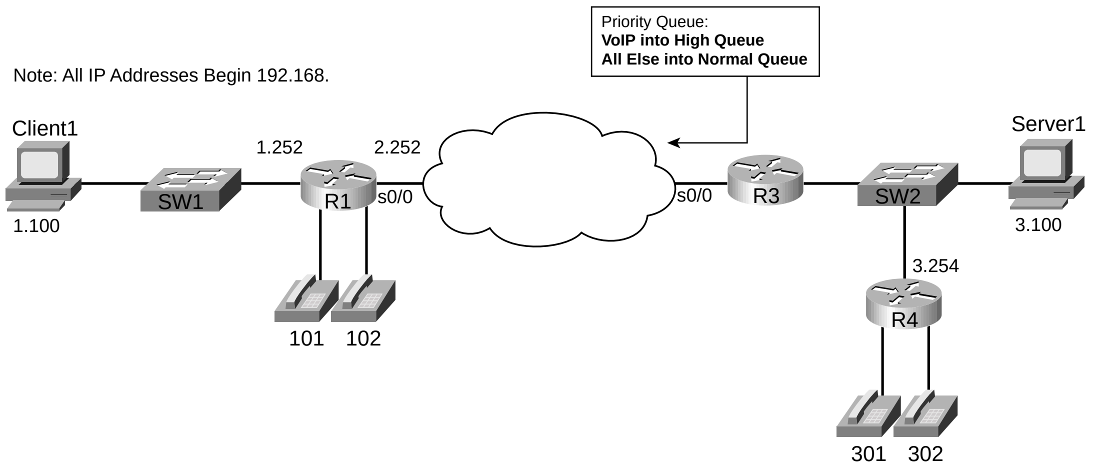
Command	Function
show queue *interface-name interface-number* [**vc** [*vpi/*] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing [**custom** | **fair** | **priority** | **random-detect** [**interface** *atm-subinterface* [**vc** [[*vpi/*] *vci*]]]]	Lists configuration and statistical information about the queuing tool on an interface

To understand the core PQ configuration commands, examine Example B-5. In this example, R3 uses PQ on its S0/0 interface. The engineer configuring R3 decided that voice traffic could benefit from being placed into the High queue, so a simple QoS policy has been devised:

- All VoIP payload traffic is placed in the High queue.
- All other traffic is placed in the Normal queue.

Figure B-6 shows the network in which the configuration is applied, and Example B-5 shows the configuration and the commands used. Note that all IP addresses in the example start with 192.168.

Figure B-6 *Network Used with PQ Configuration Examples*



Example B-5 *Priority Queuing, VoIP in High Queue, All Else in Normal Queue*

```
R3#show running-config
Building configuration...

! Portions omitted for brevity

interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 load-interval 30
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 priority-group 5
 clockrate 64000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
access-list 120 permit udp any range 16384 32767 any range 16384 32767
!
```

continues

Example B-5 *Priority Queuing, VoIP in High Queue, All Else in Normal Queue (Continued)*

```
priority-list 5 protocol ip high list 120

R3#show queue serial 0/0
Output queue for Serial0/0 is 26/60

Packet 1, linktype: ip, length: 1404, flags: 0x88
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xF560, ttl: 127,
  TOS: 0 prot: 6, source port 2831, destination port 1668
    data: 0x0B0F 0x0684 0x79EB 0x0D2A 0x05B4 0x0FF5 0x5010
          0x4510 0x5BF8 0x0000 0x6076 0xEEFD 0xFBB6 0xCC72

Packet 2, linktype: ip, length: 724, flags: 0x88
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xF561, ttl: 127,
  TOS: 0 prot: 6, source port 80, destination port 1667
    data: 0x0050 0x0683 0x79C1 0x0930 0x05B3 0xE88E 0x5010
          0x41C5 0x276E 0x0000 0xDA9B 0x48F7 0x7F64 0x7313

Packet 3, linktype: ip, length: 724, flags: 0x88
  source: 192.168.3.100, destination: 192.168.1.100, id: 0xF562, ttl: 127,
  TOS: 0 prot: 6, source port 80, destination port 1666
    data: 0x0050 0x0682 0x79BC 0xE875 0x05B3 0xE2C6 0x5010
          0x441A 0xA5A2 0x0000 0x8071 0x4239 0x5906 0xD18C

! Several lines omitted for brevity

R3#show queueing interface serial 0/0
Interface Serial0/0 queueing strategy: priority

Output queue utilization (queue/count)
    high/13593 medium/0 normal/206 low/0

R3#show queueing priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
5      high   protocol ip          list 120

R3#show int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 9/255, rxload 8/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  79, LMI stat recvd 70, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 165/2, interface broadcasts 149
  Last input 00:00:02, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:13:25
```

Example B-5 *Priority Queuing, VoIP in High Queue, All Else in Normal Queue (Continued)*

```
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 2
  Queueing strategy: priority-list 5
  Output queue (queue priority: size/max/drops):
     high: 4/20/2, medium: 0/40/0, normal: 20/60/0, low: 0/80/0

!
! Lines omitted for brevity.
```

Example B-5 uses only one **priority-list** command in this case, putting voice traffic into the High queue and letting the rest of the traffic be placed in the Normal queue by default. The **priority-list 5 protocol ip high list 120** command matches all IP packets that are permitted by ACL 120, which matches all VoIP User Datagram Protocol (UDP) ports. The **priority-group 5** interface subcommand enables PQ on S0/0, linking PQ list 5 with the interface.

Two similar commands display some limited information about PQ. First, the **show queue** command lists brief information about packets currently in the queue. Notice that some information is listed for each packet in the queue; in Example B-5, a packet that was sent by the web server (source port 80) is highlighted. Also note that none of the stanzas describing the packets in the queue shows a voice packet, with UDP ports between 16384 and 32767. Because PQ always serves the High queue first, you would need to have more voice traffic being sent into the network than the speed of the interface before packets would ever actually back up into the High queue. Therefore, it is rare to see queue entries for packets in the High queue.

The **show queueing interface** command lists configuration information about PQ and statistical information about how many packets have been placed into each queue. Note that no packets have been placed into the Medium or Low queues, because **priority-list 5** matches VoIP packets, placing them in the High queue, with all other packets defaulting to the Normal queue. The **show queueing priority** command lists information about the configuration of PQ.

Finally, the output of the **show interfaces** command states that PQ is in use, along with statistics about each queue. The current number of packets in each queue, the maximum length of each queue, and the cumulative number of tail drops is listed for each of the four queues.

Good QoS design calls for the marking of packets close to the source of the packet. The next example (Example B-6) accomplishes the same queuing goals as the preceding example, but in this case, PQ relies on the fact that the packets have been marked before reaching R3's S0/0 interface. In a real network, the packets could be marked on one of the LAN switches, or in an IP Phone, or by the computers in the network. This example shows the packets being marked upon entering R3's E0/0 interface, as shown in Example 3-1 in Chapter 3. Example B-6 shows the revised configuration based on the following criteria:

- All VoIP payload traffic has been marked with DSCP EF; place this traffic in the High queue.
- All other traffic has been marked with DSCP BE; place this traffic in the Normal queue.

Example B-6 *Priority Queuing, DSCP EF in High Queue, All Else in Normal Queue*

```
R3#show running-config
! Portions omitted for brevity
! Next several lines are CB Marking configuration, not PQ configuration
class-map match-all all-else
  match any
class-map match-all voip-rtp
  match ip rtp 16384 16383
!
policy-map voip-and-be
  class voip-rtp
   set ip dscp 46
  class class-default
   set ip dscp 0
!
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 load-interval 30
 service-policy input voip-and-be
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 priority-group 6
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
! Portions omitted for brevity
!
access-list 121 permit ip any any dscp ef
!
priority-list 6 protocol ip high list 121
!
! Portions omitted for brevity

R3#show queueing priority
Current DLCI priority queue configuration:
Current priority queue configuration:

List   Queue  Args
6      high   protocol ip          list 121
R3#show interface s 0/0
Serial0/0 is up, line protocol is up
```

Example B-6 *Priority Queuing, DSCP EF in High Queue, All Else in Normal Queue (Continued)*

```
   Hardware is PowerQUICC Serial
   Description: connected to FRS port S0. Single PVC to R1.
   MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
      reliability 255/255, txload 20/255, rxload 6/255
   Encapsulation FRAME-RELAY, loopback not set
   Keepalive set (10 sec)
   LMI enq sent  29, LMI stat recvd 29, LMI upd recvd 0, DTE LMI up
   LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
   LMI DLCI 1023  LMI type is CISCO  frame relay DTE
   Broadcast queue 0/64, broadcasts sent/dropped 68/0, interface broadcasts 63
   Last input 00:00:01, output 00:00:00, output hang never
   Last clearing of "show interface" counters 00:04:50
   Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 751
   Queueing strategy: priority-list 6
   Output queue (queue priority: size/max/drops):
      high: 0/20/0, medium: 0/40/0, normal: 4/60/751, low: 0/80/0

!
!Portions omitted for brevity.
```

In Example B-6, **priority-list 6** contains a single entry, referring to ACL 121, which matches all IP packets with DSCP EF. The **priority-group 6** command enables PQ on S0/0. Before the PQ configuration can actually do what the requirements suggest, the packets must be marked. The **policy-map voip-and-be** command matches VoIP packets based on their UDP port numbers, and marks them on ingress to interface e0/0. Note that unlike the preceding example, when an ACL matched all UDP ports between 16384 and 32767, this example matches only even-numbered ports, which are the ports actually used for VoIP payload. (The odd-numbered ports are used for voice signaling.) The **match ip rtp 16384 16383** command matches the same range of port numbers as ACL 120, except it only matches the even (payload) ports.

PQ can only support four different classifications of packets, because it only has four queues. Example B-7 shows all four queues being used. The following criteria is used for queuing packets on R3's S0/0 interface:

- VoIP payload is placed into the High queue.
- NetMeeting voice and video from Server1 to Client1 is placed in the Medium queue.
- Any HTTP traffic is placed into the Normal queue.
- All other traffic is placed into the low queue.

The same network used in the preceding example, as shown in Figure B-6, is used for this example, too.

Example B-7 *PQ Example: VoIP in High, NetMeeting in Medium, HTTP in Normal, and All Else in Low Queue*

```
access-list 150 permit udp host 192.168.1.100 range 16384 32767 192.168.3.0 0.0.0.255
range 16384 32767
!
access-list 151 permit udp any range 16384 32768 any range 16384 32768
!
access-list 152 permit tcp any eq www any
access-list 152 permit tcp any any eq www
!
priority-list 8 protocol ip medium list 150
priority-list 8 protocol ip high list 151
priority-list 8 protocol ip normal list 152
priority-list 8 default low
!
interface serial 0/0
priority-group 8
```

Note a couple of points from the example. First, there is no need to match the traffic that goes into the Low queue, because the **priority-list 8 default low** command makes the Low queue the default queue for unmatched packets. Second, note that ACL 152 matches web traffic from web servers, as well as to web servers. In this limited example, you only needed to check for packets from the web server when performing queuing on output of R3's S0/0 interface. Because one day you might have traffic going to a web server exiting R3's S0/0 interface, however, you might want to go ahead and match all web traffic as shown in the ACL.

Finally, one last example shows the equivalent configuration as shown in Example B-7, but with the assumption that the traffic had been marked before reaching R3's S0/0 interface. Again, the packets are marked upon entering R3's E0/0, using CB marking. The criteria used for the final example, whose configuration is shown in Example B-8, is as follows:

- VoIP payload is marked with DSCP EF (decimal 46); put this traffic in the High queue.
- NetMeeting voice and video from Server1 to Client1 has been marked with DSCP AF41 (decimal 34); place this traffic in the Medium queue.
- Any HTTP traffic has been marked with DSCP AF22 (decimal 20); place this traffic in the Normal queue.
- All other traffic has been marked with DSCP BE (decimal 0); place this traffic in the Low queue.

Example B-8 *PQ Example: DSCP EF in High, DSCP AF41 in Medium, DSCP AF22 in Normal, and All Else in Low Queue*

```
R3#show running-config
! Portions omitted for brevity
class-map match-all http-all
  match protocol http
class-map match-all voip-rtp
```

Example B-8 *PQ Example: DSCP EF in High, DSCP AF41 in Medium, DSCP AF22 in Normal, and All Else in Low Queue (Continued)*

```
  match ip rtp 16384 16383
class-map match-all NetMeet
  match access-group NetMeet-ACL
class-map match-all all-else
  match any
!
policy-map laundry-list
  class voip-rtp
   set ip dscp 46
  class NetMeet
   set ip dscp 34
  class http-all
   set ip dscp 20
  class class-default
   set ip dscp 0
!
interface Ethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 ip nbar protocol-discovery
 load-interval 30
 service-policy input laundry-list
!
interface Serial0/0
 description connected to FRS port S0. Single PVC to R1.
 no ip address
 encapsulation frame-relay
 load-interval 30
 priority-group 7
 clockrate 128000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
ip access-list extended NetMeet-ACL
 permit udp host 192.168.3.100 range 16384 32768192.168.1.0 0.0.0.255 range 16384 32768
!
access-list 130 permit ip any any dscp ef
access-list 131 permit ip any any dscp af41
access-list 132 permit ip any any dscp af22
access-list 133 permit ip any any dscp default
!
priority-list 7 protocol ip high list 130
priority-list 7 protocol ip medium list 131
priority-list 7 protocol ip normal list 132
priority-list 7 protocol ip low list 133
```

continues

Example B-8 *PQ Example: DSCP EF in High, DSCP AF41 in Medium, DSCP AF22 in Normal, and All Else in Low Queue (Continued)*

```
priority-list 7 default low

R3#show queueing int s 0/0
Interface Serial0/0 queueing strategy: priority

Output queue utilization (queue/count)
    high/42092 medium/1182 normal/52 low/3242
```

As seen in Example B-8, if the packets have already been marked, PQ can simply match the DSCP field. Routers that forward these packets later can also just configure PQ, and match the DSCP field. Note that PQ needs to refer to an ACL to match the DSCP field, whereas some other queuing tools can refer directly to the DSCP field without using an ACL. Another interesting point to note in this configuration is that the CB marking configuration used network-based application recognition (NBAR) with the **match protocol http** command to match all web traffic, and then set those packets with DSCP 20. Example B-7 showed why you might want to match all web traffic, but the configuration used two entries to match all web traffic (ACL 152). With NBAR, you can match all web traffic, regardless of whether it is coming from or going to a web server.

CQ Configuration

CQ configuration resembles PQ configuration. Global commands are used to define the logic for classifying packets by matching header fields, and an interface subcommand is used to enable CQ on an interface. Example configurations for CQ follow Tables B-6 and B-7, which list the configuration and exec commands related to CQ, respectively.

Table B-6 *Configuration Command Reference for CQ*

Command	Mode and Function
queue-list *list-number* **protocol** *protocol-name queue-number queue-keyword keyword-value*	Global configuration command; creates a custom queue list entry, sets classification first by protocol, and then further refines it with keywords. Also identifies the queue into which to place matched packets. (See Example B-9 for a list of keywords.)
queue-list *list-number* **interface** *interface-type interface-number queue-number*	Global configuration command; creates a custom queue list entry, sets classification based on incoming interface, and then identifies queue into which to place matched packets.
queue-list *list-number* **queue** *queue-number* **byte-count** *byte-count-number*	Global config command; sets the byte count for the specified queue, which is the number of bytes taken from this queue each pass.

Table B-6 *Configuration Command Reference for CQ (Continued)*

Command	Mode and Function
queue-list *list-number* **queue** *queue-number* **limit** *limit-number*	Global configuration command; sets queue lengths for the specified queue.
queue-list *list-number* **default** *queue-number*	Changes the default queue for packets that are not matched.
custom-queue-list [*list-number*]	Interface subcommand; enables a custom queue list for packets exiting an interface.

Table B-7 *Exec Command Reference for CQ*

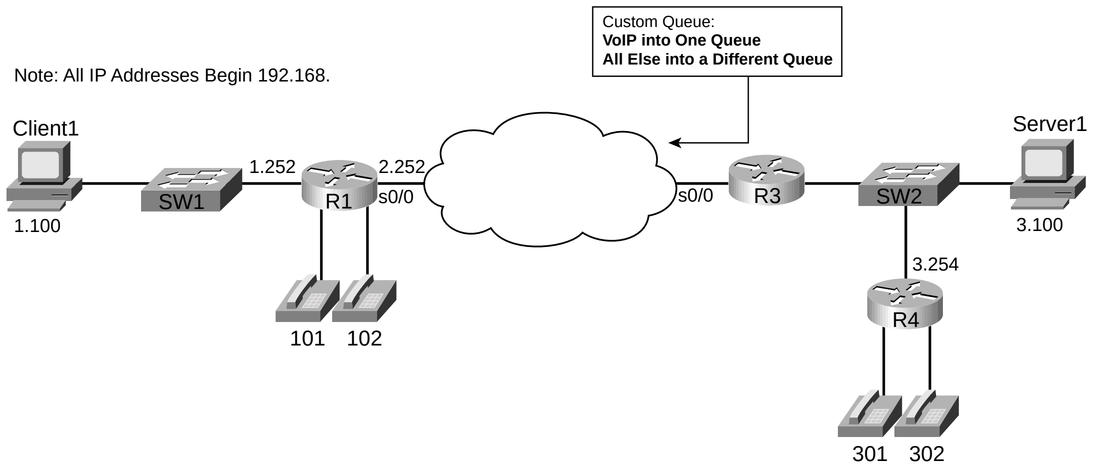
Command	Function
show queue *interface-name interface-number* [**vc** [*vpi/*] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing [**custom** | **fair** | **priority** | **random-detect** [**interface** atm-subinterface [**vc** [[*vpi/*] *vci*]]]]	Lists configuration and statistical information about the queuing tool on an interface

To understand the core CQ configuration commands, examine Example B-9. (This example is identical to Example B-5, shown in the "PQ Configuration" section earlier in this chapter, except that CQ is used rather than PQ.) In this example, R3 uses CQ on its S0/0 interface. The engineer configuring R3 decided that voice traffic could benefit from being given preferential queuing treatment, so a simple QoS policy has been devised, as noted in the following:

- All VoIP payload traffic is placed in Queue 2.
- All other traffic is placed in Queue 1.
- Assign about 2/3 of the bandwidth to the VoIP traffic.

Figure B-7 shows the network in which the configuration is applied, and Example B-9 shows the configuration and **show** commands:

Figure B-7 *Network Used with CQ Configuration Examples*



Example B-9 *Custom Queuing: VoIP in High Queue, All Else in Normal Queue*

```
R3(config)#queue-list 1 protocol ip 1 ?
  fragments  Prioritize fragmented IP packets
  gt         Classify packets greater than a specified size
  list       To specify an access list
  lt         Classify packets less than a specified size
  tcp        Prioritize TCP packets 'to' or 'from' the specified port
  udp        Prioritize UDP packets 'to' or 'from' the specified port
  <cr>
R3(config)#^Z
R3#show running-config

access-list 120 permit udp any range 16383 32767 any range 16383 32767
!
queue-list 5 protocol ip 2 list 120
queue-list 5 queue 2 byte-count 5000
queue-list 5 queue 1 byte-count 2500
queue-list 5 queue 2 limit 30
!
interface serial0/0
custom-queue-list 5
!
R3#show queueing custom
Current custom queue configuration:

List   Queue  Args
5      2      protocol ip          list 120
5      1      byte-count 2500
5      2      byte-count 5000 limit 30
```

Example B-9 *Custom Queuing: VoIP in High Queue, All Else in Normal Queue (Continued)*

```
R3#show queueing int s 0/0
Interface Serial0/0 queueing strategy: custom

Output queue utilization (queue/count)
    0/15 1/91 2/3549 3/0 4/0 5/0 6/0 7/0 8/0
    9/0 10/0 11/0 12/0 13/0 14/0 15/0 16/0
R3#show int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 9/255, rxload 8/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  1, LMI stat recvd 1, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 4/0, interface broadcasts 3
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:00:12
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 282
  Queueing strategy: custom-list 5
  Output queues: (queue #: size/max/drops)
     0: 0/20/0 1: 6/20/0 2: 28/30/282 3: 0/20/0 4: 0/20/0
     5: 0/20/0 6: 0/20/0 7: 0/20/0 8: 0/20/0 9: 0/20/0
     10: 0/20/0 11: 0/20/0 12: 0/20/0 13: 0/20/0 14: 0/20/0
     15: 0/20/0 16: 0/20/0
  5 minute input rate 52000 bits/sec, 102 packets/sec
  5 minute output rate 60000 bits/sec, 73 packets/sec
     1394 packets input, 89409 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1085 packets output, 102851 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

The configuration for this scenario only requires a few commands. The **queue-list 5 protocol ip 2 list 120** command causes CQ to match all packets permitted by ACL 120 and to place these packets into Queue 2. CQ uses Queue 1 for unmatched packets, so no other classification commands are required. The byte count for each queue defaults to 1500 bytes; in this case, the criteria specified that CQ should give about 2/3 of the bandwidth to the VoIP traffic, so queue 2 was assigned a byte count of 5000, and Queue 1 a byte count of 2500. At a 64-kbps clock rate, each complete cycle takes a little less than a second. Finally, to enable CQ list 5, the **custom-queue** interface subcommand was used.

IOS uses the same **show** commands to display information about CQ as it does for PQ. The **show queue** command lists brief information about packets in the queue at the present time (not shown in the example). The **show queueing interface** command lists configuration information about CQ and statistical information about how many packets have been placed into each queue. In the example, Queue 2 has had a total of 3549 packets pass through it. Also note that no packets have been placed into Queues 3 through 16, because CQ list 5 does not classify packets into Queues 3 through 16. The **show queueing custom** command lists information about the configuration of CQ. The output of the **show interfaces** command states that CQ is in use, along with statistics about each queue. The current number of packets in each queue, the maximum length of each queue, and the cumulative number of tail drops are listed for each of the 16 queues.

You may have noticed that Queue 0 shows up in a couple of commands. CQ uses a special queue, Queue 0, which is used for important high-priority packets generated by the router. Routing protocol updates are considered high priority, for instance, so these packets are placed into Queue 0. CQ treats Queue 0 as a priority queue—that is, when packets are waiting in Queue 0, CQ interrupts its normal scheduling logic, takes the packet from Queue 0 next, and then resumes the scheduling process. CQ does not allow packets to be explicitly classified into Queue 0.

Stepping back from the configuration for a moment, consider the time required to service 2500 bytes from Queue 1, and then 5000 bytes from Queue 2, at 64 kbps. It requires roughly .875 seconds total, with roughly 310 ms to service Queue 1, and the rest to service Queue 2. So, if Queue 1 has many packets, the voice packet in Queue 2 that is the next packet to get serviced, after Queue 1 gets to send its 2500 bytes, has to wait an extra 310 ms! So, this example, although giving voice 2/3 of the bandwidth, does not actually give voice very good performance.

Good QoS design calls for the marking of packets close to the source of the packet. The next example accomplishes the same queuing goals as the preceding example, but CQ relies on the fact that the packets have been marked before reaching R3's S0/0 interface. In a real network, the packets could be marked on one of the LAN switches; for review, however, this example shows the packets being marked upon entering R3's E0/0 interface, just as shown in Example 3-1 in Chapter 3. Example B-10 shows the revised configuration based on the following criteria:

- All VoIP payload traffic has been marked with DSCP EF; place this traffic in Queue 1.
- All other traffic has been marked with DSCP BE; place this traffic in Queue 2.
- VoIP traffic is given about 2/3 of the bandwidth.

Example B-10 *Custom Queuing: DSCP EF in Queue 1, All Else in Queue 2*

```
ip cef
!
class-map match-all class-default
  match any
```

Example B-10 *Custom Queuing: DSCP EF in Queue 1, All Else in Queue 2 (Continued)*

```
class-map match-all voip-rtp
  match ip rtp 16384 16383
!


!
policy-map voip-and-be
  class voip-rtp
   set ip dscp 46
  class all-else
   set ip dscp 0
!
interface Ethernet0/0
 service-policy input voip-and-be
!
interface Serial0/0
 custom-queue-list 6
 clockrate 64000
!
interface Serial0/0.1 point-to-point
 description point-point subint global DLCI 103, connected via PVC to DLCI 101 (R1)
 ip address 192.168.2.253 255.255.255.0
 frame-relay interface-dlci 101
!
! Portions omitted for brevity
!
access-list 121 permit ip any any dscp ef
!
queue-list 6 protocol ip 1 list 121
queue-list 6 queue 1 byte-count 5000
queue-list 6 queue 2 byte-count 2500
queue-list 6 default 2
R3#sh queueing int s 0/0
Interface Serial0/0 queueing strategy: custom

Output queue utilization (queue/count)
    0/5 1/1012 2/30 3/0 4/0 5/0 6/0 7/0 8/0
    9/0 10/0 11/0 12/0 13/0 14/0 15/0 16/0
R3#show queueing custom
Current custom queue configuration:

List   Queue  Args
6      2      default
6      1      protocol ip          list 121
6      1      byte-count 5000
6      2      byte-count 2500
```

continues

Example B-10 *Custom Queuing: DSCP EF in Queue 1, All Else in Queue 2 (Continued)*

```
R3#show int s 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Description: connected to FRS port S0. Single PVC to R1.
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 10/255, rxload 8/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent 4, LMI stat recvd 4, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent 0, LMI upd sent 0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  Broadcast queue 0/64, broadcasts sent/dropped 7/0, interface broadcasts 7
  Last input 00:00:02, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:00:33
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 718
  Queueing strategy: custom-list 6
  Output queues: (queue #: size/max/drops)
     0: 0/20/0 1: 16/20/718 2: 19/20/0 3: 0/20/0 4: 0/20/0
     5: 0/20/0 6: 0/20/0 7: 0/20/0 8: 0/20/0 9: 0/20/0
     10: 0/20/0 11: 0/20/0 12: 0/20/0 13: 0/20/0 14: 0/20/0
     15: 0/20/0 16: 0/20/0
  5 minute input rate 52000 bits/sec, 102 packets/sec
  5 minute output rate 61000 bits/sec, 81 packets/sec
     3533 packets input, 226737 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     2788 packets output, 260972 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

In Example B-10, **queue-list 6** refers to ACL 121, which matches all IP packets with DSCP EF, and places these packets into Queue 1. The default queue is now Queue 2, based on the **queue-list 6 default 2** command. The byte counts for the first two queues have been set to 5000 and 2500, respectively. The new CQ configuration is enabled on S0/0 with the **custom-queue 6** command.

Example B-11 shows CQ with four queues, with classification as follows:

- VoIP payload is placed into Queue 1.
- NetMeeting voice and video from Server1 to Client1 is placed into Queue 2.
- Any HTTP traffic is placed into Queue 3.
- All other traffic is placed into Queue 4.
- Give each of the 4 queues an equal amount of link bandwidth.

The same network used in the preceding example, as shown in Figure B-7, is used for this example as well.

Example B-11 *CQ Example: VoIP in Queue 1, NetMeeting in Queue 2, HTTP in Queue 3, and All Else in Queue 4*

```
access-list 150 permit udp host 192.168.1.100 range 16384 32767 192.168.3.0 0.0.0.255
  range 16384 32767
!
access-list 151 permit udp any range 16384 32768 any range 16384 32768
!
access-list 152 permit tcp any eq www any
access-list 152 permit tcp any any eq www

!
queue-list 8 protocol ip 2 list 150
queue-list 8 protocol ip 1 list 151
queue-list 8 protocol ip 3 list 152

queue-list 8 queue 1 byte-count 2500
queue-list 8 queue 2 byte-count 2500
queue-list 8 queue 3 byte-count 2500
queue-list 8 queue 4 byte-count 2500
queue-list 8 default 4
!
interface serial 0/0
custom-queue 8
```

Distributed Weighted Fair Queuing (dWFQ)

IOS provides three streamlined WFQ options specifically designed for the Cisco 7500 series hardware architecture: distributed WFQ (dWFQ), ToS-based WFQ, and QoS group–based WFQ. The 7500 architecture includes distributed processing on line cards called *Versatile Interface Processors* (VIPs). Many functions that the general-purpose processor on the Route Switch Processor (RSP) card would normally perform by can instead be distributed to the processor on the VIPs. Distributing the processing can prove very helpful in typical 7500 series configurations that use a large number of line cards and interfaces, relieving processing burden from the general-purpose RSP processor.

You can use WFQ on 7500s without distributing the processing to the VIPs. To do so, just configure WFQ as described in the preceding section. However, Cisco created these other three distributed WFQ options specifically for 7500s, to address some specific needs. In particular, because 7500s tend to have faster WAN interfaces, Cisco created WFQ tools that run more efficiently in a higher-speed environment. One of the key goals of queuing is to reduce delay for the appropriate traffic; these distributed queuing tools are streamlined so that, even on the higher-speed interfaces on the 7500, the queuing process does not artificially delay any packets.

The three distributed WFQ options are as follows:

- Distributed WFQ (dWFQ)
- ToS-based WFQ
- QoS group–based WFQ

All three tools require distributed CEF (dCEF) processing to be enabled globally, and on the interface on which distributed WFQ is enabled. The following three sections outline the basic similarities and differences among the three distributed WFQ options, respectively.

WFQ terminology can be confusing at best. Table B-8 lists some of the fundamental terms and their respective definitions. Note that some terms can have more than one meaning.

Table B-8 *WFQ Terminology*

Term	Possible Use for the Term
WFQ	Stands for Weighted Fair Queuing. Can specifically refer to Flow-Based WFQ, as described in Chapter 4. Can also refer to the broader family of queuing tools that have WFQ as part of their names.
Flow-Based WFQ	Specifically refers to WFQ as described in Chapter 4.
Distributed WFQ, or dWFQ	Stands for distributed WFQ. Can specifically refer to Flow-Based dWFQ, as described in the next section. This can also refer to the broader family of queuing tools that have dWFQ as part of their names. These tools only function on Cisco 7500 series routers.
Flow-Based dWFQ	Specifically refers to dWFQ as described in the next section.
ToS-Based dWFQ	One of the dWFQ tools, with classification based on the ToS byte.
QoS Group–Based dWFQ	One of the dWFQ tools, with classification based on the QoS groups.
CBWFQ	Stands for Class-Based WFQ. Refers to a queuing tool that can be applied to most Cisco routers; it uses the Modular QoS command-line interface (MQC) for configuration.

Flow-Based dWFQ

Flow-Based dWFQ classifies packets based on the flow, examining the source and destination IP address and port and the protocol type. All of the other main features of dWFQ differ slightly when compared to WFQ. Table B-9 lists several points for comparison.

Table B-9 *WFQ Functions and Features*

Tool	Classification	Drop Policy	Weighting	Max # of Queues	Can It Be Used on Shaping Queues?
WFQ	Flow based	Modified tail drop	Based on IP precedence	4096	Yes
dWFQ	Flow based	Tail drop for individual queues, plus a maximum aggregate for all queues	None	512	No

The first big difference between WFQ and dWFQ involves the tail-drop policy. dWFQ has to answer two simple questions when it determines whether to drop a packet:

- Has the aggregate packet limit been exceeded
- Has the individual queue limit been exceeded?

If either of these questions is true (the aggregate packet limit or the individual queue limit has been exceeded), the new packet is discarded. Unlike WFQ, dWFQ cannot enqueue a packet, only to discard it later because its sequence number (SN) is larger than a new packet's SN. dWFQ uses the terms "aggregate limit" and "individual limit" to describe the limit over all queues and the limit per queue, respectively.

Unlike WFQ, dWFQ ignores the IP precedence value when calculating the SN of a packet. In other words, dWFQ does not actually weight the packet, totally ignoring the contents of the ToS byte. In effect, all packets are weighted equally, so dWFQ always prefers lower-volume flows in comparison to higher-volume flows.

Another difference between the two is that dWFQ does allow for the maximum queue length to be changed, but the number of queues remains constant at 512.

Finally, one other difference between WFQ and dWFQ has to do with how dWFQ works internally. dWFQ uses a mechanism called a *calendar queue* to sort all the packets waiting in a dWFQ queue. WFQ uses a simple sorted linked list. The effect is the same—both schedulers select the packet with the lowest SN as the next packet to be forwarded to the TX Queue/TX Ring. By using a calendar queue, however, dWFQ actually performs the final scheduling function more efficiently, which is particularly useful on higher-speed interfaces. (Note that the internals of calendar queues is not important for the QoS exam; if you want to read more, however, refer to *Inside Cisco IOS Software Architecture*.)

dWFQ Configuration

dWFQ configuration, like WFQ, takes little effort. The **fair-queue** interface subcommand enables flow-based dWFQ on the interface. If you want to change the aggregate or individual queue limits, you use the **fair-queue aggregate-limit** and **fair-queue individual-limit** commands, respectively. You can use the same familiar **show** commands to examine the status of dWFQ. Tables B-10 and B-11 list the configuration commands and exec commands for Flow-Based dWFQ.

Table B-10 *Configuration Command Reference for Flow-Based dWFQ*

Command	Mode and Function
fair-queue	Interface configuration mode; enables dWFQ. There are no other parameters on this command when it's used with dWFQ.
fair-queue aggregate-limit *aggregate-packets*	Interface configuration mode; sets the aggregate limit of the number of packets held in all dWFQ queues.
fair-queue individual-limit *individual-packet*	Interface configuration subcommand; sets the maximum number of packets in a single queue.

Table B-11 *Exec Command Reference for Flow-Based dWFQ*

Command	Function
show queue *interface-name interface-number* [**vc** [*vpi*/] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing [**custom** | **fair** | **priority** | **random-detect** [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]]	Lists configuration and statistical information about the queuing tool on an interface

Coincidentally, in some cases, a valid configuration for WFQ can be identical to a valid configuration for dWFQ. For instance, the **fair-queue** subcommand is used to enable both WFQ and dWFQ on an interface. Example B-12 shows a configuration on the familiar R3, which has now been upgraded to a 7500 series router with VIP2-50 line cards:

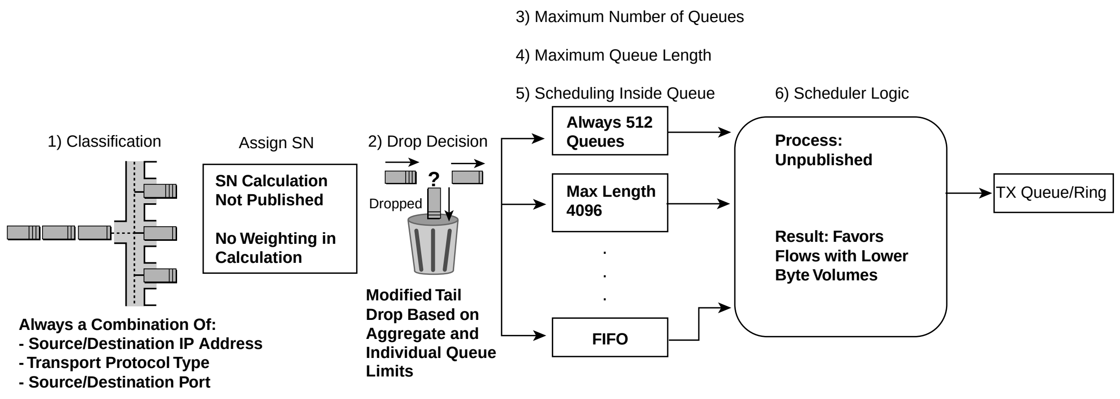
Example B-12 *dWFQ Configuration*

```
ip cef
!
interface serial 0/0/1
 encapsulation ppp
 description Serial interface on VIP2-50
 fair-queue
```

In the example, CEF has been enabled globally, and the **fair-queue** command has been added to the serial interface. If the serial interface were not on a VIP at all, WFQ would be performed. With a VIP2-50 installed, dWFQ would be performed.

Figure B-8 summarizes the sequencing and main features of dWFQ.

Figure B-8 *Flow-Based dWFQ: Summary of Main Features*


3) Maximum Number of Queues
4) Maximum Queue Length
5) Scheduling Inside Queue
6) Scheduler Logic
1) Classification
Assign SN
2) Drop Decision
SN Calculation Not Published
No Weighting in Calculation
Dropped
?
Always 512 Queues
Max Length 4096
FIFO
Process: Unpublished
Result: Favors Flows with Lower Byte Volumes
TX Queue/Ring
Always a Combination Of:
- Source/Destination IP Address
- Transport Protocol Type
- Source/Destination Port
Modified Tail Drop Based on Aggregate and Individual Queue Limits

ToS-Based dWFQ

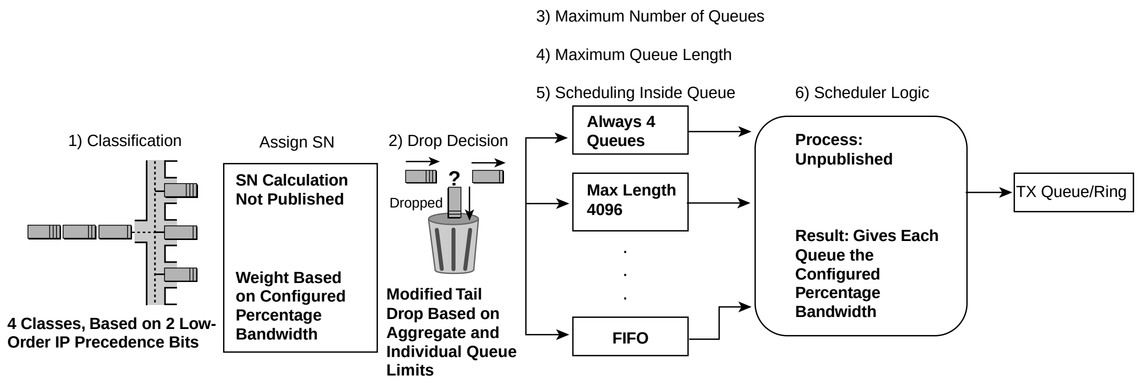
ToS-Based dWFQ uses a totally different classification method and a totally different scheduler as compared with WFQ. After reading this section, you may wonder why Cisco calls this tool WFQ at all, because it does not resemble WFQ in very many ways! ToS-Based WFQ classifies based on the 2 low-order bits in the IP Precedence field—in other words, the second and third bits of the Precedence field. It places packets into one of four queues based on these values. It does not consider the flow at all; it is a class-based tool, as opposed to a flow-based tool. Classification is based on the 2 low-order bits in the IP Precedence field, as opposed to WFQ's flow-based classification.

Although the details are not published, ToS-Based dWFQ does use a fair-queuing scheduler, with the weighting based on the bandwidth percentages configured in each case. If only 2 queues were used, and one queue were given 75 percent of the bandwidth, and the other queue were given 25 percent, for example, internally, IOS calculates weights for the 2 queues with a ratio of 3:1. That's why it's called WFQ, although ToS-Based dWFQ differs significantly from WFQ. Table B-12 lists points for comparison, which are illustrated in Figure B-9.

Table B-12 *ToS-Based WFQ Functions and Features*

Tool	Classification	Drop Policy	Weighting	Max # of Queues	Can It Be Used on Shaping Queues?
ToS-Based WFQ	Class based, predicated on low-order 2 bits of precedence	Same as dWFQ	Configured as percentage of link bandwidth per queue	4	No

Figure B-9 *ToS-Based dWFQ: Summary of Main Features*


3) Maximum Number of Queues
4) Maximum Queue Length
5) Scheduling Inside Queue
6) Scheduler Logic
1) Classification
Assign SN
2) Drop Decision
Always 4 Queues
Max Length 4096
FIFO
Process: Unpublished
Result: Gives Each Queue the Configured Percentage Bandwidth
TX Queue/Ring
SN Calculation Not Published
Weight Based on Configured Percentage Bandwidth
Dropped
?
Modified Tail Drop Based on Aggregate and Individual Queue Limits
4 Classes, Based on 2 Low-Order IP Precedence Bits

The classification function cannot be changed for ToS-Based WFQ. It always places packets into one of four queues, based on the 2 low-order bits in the IP Precedence field. In other words, precedence 0 and 4 go into Queue 0, because decimal 0 converts to 000, and decimal 4 converts to 100—so the 2 low-order bits are equal. Likewise, precedence 1 and 5 end up in Queue 1, precedence 2 and 6 in Queue 2, and precedence 3 and 7 in Queue 3.

The drop policy is just like dWFQ, using an aggregate limit and individual queue limit method to set the value.

The scheduler provides a percentage of bandwidth for each queue. Cisco does not publish the scheduler logic, but it works like WFQ, using SNs. IOS derives the weights for each queue based on the configured bandwidth percentages. Interestingly, the parameter is actually called weight, although it is interpreted as a percentage of bandwidth. The percentages default to 10 percent, 20 percent, 30 percent, and 40 percent, for Queues 0 through 3, respectively. You can configure the percentages for Queues 1 through 3, with the rest being assigned to Queue 0.

ToS-Based WFQ Configuration

The configuration for ToS-Based dWFQ is simple. Tables B-13 and B-14 list the generic commands, and an example follows these tables.

Configuring the **fair-queue tos** subcommand on an interface enables ToS-Based dWFQ. For this command to work, distributed CEF (dCEF) must be enabled, and the interface must really be on a VIP2-*xx*, where *xx* is 40 or higher. Example B-13 shows a sample configuration, along with changes to the bandwidth percentages given to Queues 1, 2, and 3. In this case, Queue 0 gets 39 percent of the link bandwidth, because 61 percent has been assigned using configuration commands.

Table B-13 *Configuration Command Reference for ToS-Based dWFQ*

Command	Mode and Function
fair-queue tos	Interface configuration mode; enables ToS-Based dWFQ. There are no other parameters on this command when it's used with dWFQ.
fair-queue aggregate-limit *aggregate-packets*	Interface configuration mode; sets the aggregate limit of the number of packets held in all dWFQ queues.
fair-queue individual-limit *individual-packet*	Interface configuration subcommand; sets the maximum number of packets in a single queue.
fair-queue {**qos-group** *number* | **tos** *number*} **limit** *class-packet*	Interface configuration subcommand; sets the individual queue limit for a single queue. Takes precedence over the **fair-queue individual-limit** command.
fair-queue {**qos-group** *number* | **tos** *number*} **weight** *weight*	Interface subcommand; sets the percentage link bandwidth assigned to the queue. The **tos** parameter can be 1, 2, or 3.

Table B-14 *Exec Command Reference for ToS-Based dWFQ*

Command	Function
show queue *interface-name interface-number* [**vc** [*vpi*/] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing [**custom** | **fair** | **priority** | **random-detect** [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]]	Lists configuration and statistical information about the queuing tool on an interface

Example B-13 *ToS-Based dWFQ Configuration*

```
ip cef
!
interface serial 0/0/1
 encapsulation ppp
 description Serial interface on VIP2-50
 fair-queue tos
 fair-queue tos 1 weight 30
 fair-queue tos 2 weight 30
 fair-queue tos 3 weight 1
```

Distributed QoS Group–Based WFQ

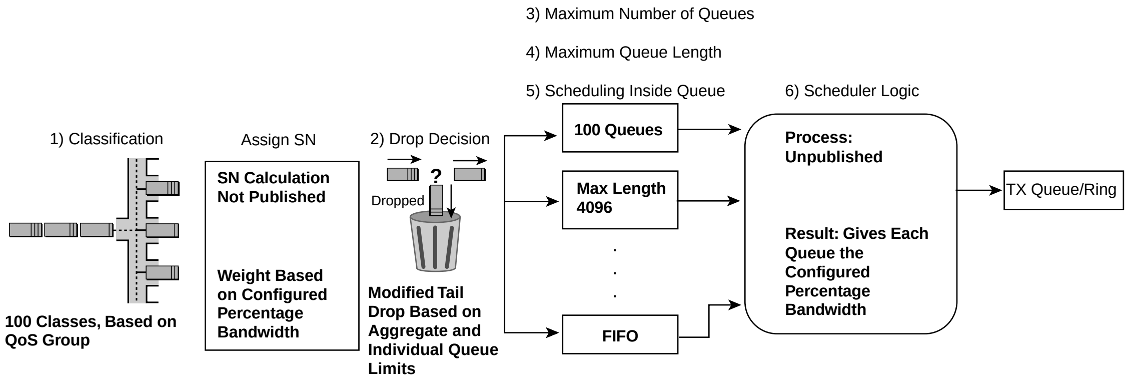
QoS Group–Based dWFQ behaves almost identically to ToS-Based WFQ. The classification feature of each is similar, but with QoS Group–Based dWFQ, the classification is based on the QoS group value assigned to the packet. The QoS group is a value between 0 and 99, inclusive, which is assigned to the packet as it passes through a single router. QoS Group–Based dWFQ classifies a packet into 1 of 100 queues, numbered 0 through 99, matching the QOS group assigned to the packet. (Queue 0 is the default queue.)

The drop policy and scheduler work just like ToS-Based dWFQ as well. The bandwidth percentages can be spread among as many as 100 queues. Table B-15 lists the familiar points for comparison, this time highlighting QoS Group–Based dWFQ, and Figure B-10 depicts the general sequence of events when using QoS Group–Based dWFQ.

Table B-15 *QoS Group–Based WFQ Functions and Features*

Tool	Classification	Drop Policy	Weighting	Max # of Queues	Can It Be Used on Shaping Queues?
QoS Group–Based WFQ	QoS group based	Same as dWFQ	Configured as percentage of link bandwidth per queue	100	No

Figure B-10 *QoS Group–Based dWFQ: Summary of Main Features*


3) Maximum Number of Queues
4) Maximum Queue Length
5) Scheduling Inside Queue
6) Scheduler Logic
1) Classification
Assign SN
2) Drop Decision
100 Classes, Based on QoS Group
SN Calculation Not Published
Weight Based on Configured Percentage Bandwidth
Dropped
?
Modified Tail Drop Based on Aggregate and Individual Queue Limits
100 Queues
Max Length 4096
FIFO
Process: Unpublished
Result: Gives Each Queue the Configured Percentage Bandwidth
TX Queue/Ring

QoS Group–Based WFQ Configuration

The configuration for QoS Group–Based dWFQ is simple. Tables B-16 and B-17 lists the generic commands.

Table B-16 *Configuration Command Reference for QoS Group–Based dWFQ*

Command	Mode and Function
fair-queue qos-group	Interface configuration mode; enables QoS Group–Based dWFQ. There are no other optional parameters on this command.
fair-queue aggregate-limit *aggregate-packets*	Interface configuration mode; sets the aggregate limit of the number of packets held in all dWFQ queues.
fair-queue individual-limit *individual-packet*	Interface configuration subcommand; sets the maximum number of packets in a single queue.
fair-queue {**qos-group** *number* | **tos** *number*} **limit** *class-packet*	Interface configuration subcommand; sets the individual queue limit for a single queue. Takes precedence over the **fair-queue individual-limit** command.
fair-queue {**qos-group** *number* | **tos** *number*} **weight** *weight*	Interface subcommand; sets the percentage link bandwidth assigned to the queue. The QoS group number can be between 1 and 99, with Queue 0 getting the remaining bandwidth.

Table B-17 *Exec Command Reference for QoS Group–Based dWFQ*

Command	Function
show queue *interface-name interface-number* [**vc** [*vpi*/] *vci*]]	Lists information about the packets that are waiting in a queue on the interface
show queueing [**custom** | **fair** | **priority** | **random-detect** [**interface** *atm-subinterface* [**vc** [[*vpi*/] *vci*]]]]	Lists configuration and statistical information about the queuing tool on an interface

Configuring the **fair-queue qos-group** subcommand on an interface enables QoS Group–Based dWFQ. For this command to work, dCEF must be enabled, and the interface must really be on a VIP2-*xx*, where *xx* is 40 or higher. Example B-14 shows a sample configuration, along with changes to the bandwidth percentages given to Queues 1, 2, and 3. In this case, queue 0 gets 39 percent of the link bandwidth, because 61 percent has been assigned using configuration commands.

Example B-14 *QoS Group–Based dWFQ Configuration*

```
ip cef
!
interface serial 0/0/1
 encapsulation ppp
```

continues

Example B-14 *QoS Group–Based dWFQ Configuration (Continued)*

```
description Serial interface on VIP2-50
fair-queue qos-group
fair-queue qos-group 1 weight 30
fair-queue qos-group 2 weight 30
fair-queue qos-group 3 weight 1
```

Example B-14 lists the same basic parameters as Example B-13, but with QoS Group–Based dWFQ in this case. Of course, more queues could be used, with different weights (bandwidth percentages) for each queue.

Summary: dWFQ Options

Of the three dWFQ options, useful on the 7500 series routers only, dWFQ (also called *Flow-Based dWFQ*) is most similar to WFQ. It is flow based, uses a similar (although slightly different) drop policy, and uses the same scheduling logic. However, the other differences are significant: dWFQ does not consider the IP precedence when assigning SNs, and it only allows 512 queues.

The other two dWFQ options, ToS-Based dWFQ and QoS Group–Based dWFQ, stand out in their lack of similarity to WFQ. Neither are flow based, the number of queues is predetermined, and the scheduler assigns weights based on bandwidth percentages, which is more like CQ or CBWFQ. Like dWFQ, these two options are available only on 7500 series routers.

ToS-Based dWFQ classifies based on the 2 low-order bits in the IP Precedence field, which may require you to rethink your QoS classification and marking strategy. For instance, Cisco recommends that you mark video payload with AF41, and the least-important data with DSCP BE. The video needs high bandwidth, low delay, and low jitter. The unimportant data can get the leftover bandwidth, with no significant delay requirements. With DSCP AF41's binary code being 1**00**010, however, and DSCP BE being 0**00**000, ToS-Based dWFQ would work poorly. (The low-order 2 bits of the IP Precedence field are in bold, and in both cases, the value is binary 00.) Similarly, DSCP EF traffic and DSCP AF11, AF12, and AF13 would all be placed in the same queue with ToS-Based dWFQ. Therefore, you would need to reconsider the QoS policy for marking before using this tool.

From an exam-preparation perspective, try to focus on the details of WFQ and the comparisons between the tools that Table B-18 outlines.

Table B-18 *Summary of dWFQ Functions and Features*

Tool	Classification	Drop Policy	Weighting	Max # of Queues	Can It Be Used on Shaping Queues?
WFQ	Flow based	Modified tail drop	Based on IP precedence	4096	Yes
dWFQ	Flow based	Tail drop for individual queues, plus a max aggregate for all queues	None	512	No
ToS-Based WFQ	Class based, predicated on low-order 2 bits of precedence	Same as dWFQ	Configured as percentage of link bandwidth per queue	4	No
QoS Group–Based WFQ	QoS group based	Same as dWFQ	Configured as percentage of link bandwidth per queue	100	No

Modified Deficit Round-Robin

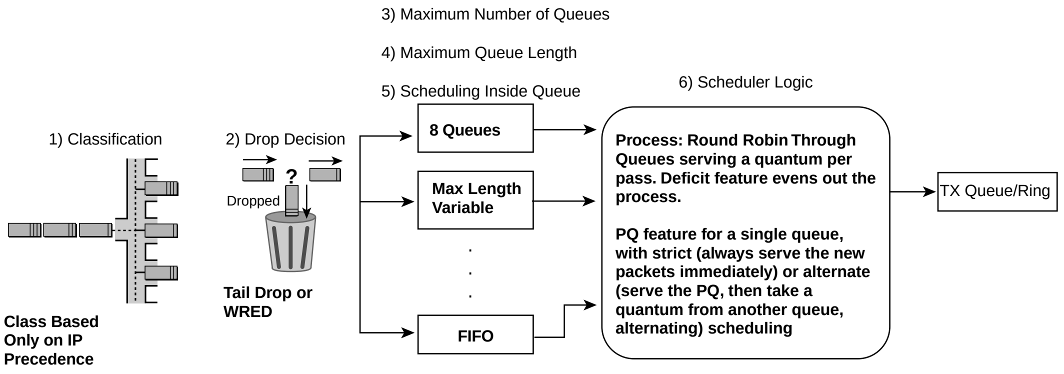
Cisco designed Modified Deficit Round-Robin (MDRR) specifically for the Gigabit Switch Router (GSR) models of Internet routers. In fact, MDRR is supported only on the GSR 12000 series routers, and the other queuing tools (WFQ, CBWFQ, PQ, CQ, and so on) are not supported on the GSRs.

Many of the features of MDRR will be familiar. Like all queuing tools, it needs to perform classification and make drop decisions, scheduling decisions, and so on. Figure B-11 depicts the major parts of the queuing puzzle of MDRR, followed by some comments about each step.

MDRR classifies packets only based on the IP Precedence field. Not surprisingly, MDRR supports eight classes, and therefore eight queues, because there are eight precedence values. The queues are numbered 0 through 7, and the precedence values (decimal) are also 0 through 7—but MDRR does map the values one to one. MDRR enables you to configure each precedence value to map to any of the queues, so more than one precedence value could map to the same queue.

Like CBWFQ, MDRR supports either tail drop or Weighted Random Early Detection (WRED) for the drop policy, per queue. Like most other tools, inside each queue, first-in, first-out (FIFO) logic is used. In addition, as expected, the most interesting part relates to the logic used by the scheduler, which is where MDRR gets its name.

Figure B-11 *MDRR: Summary of Main Features*

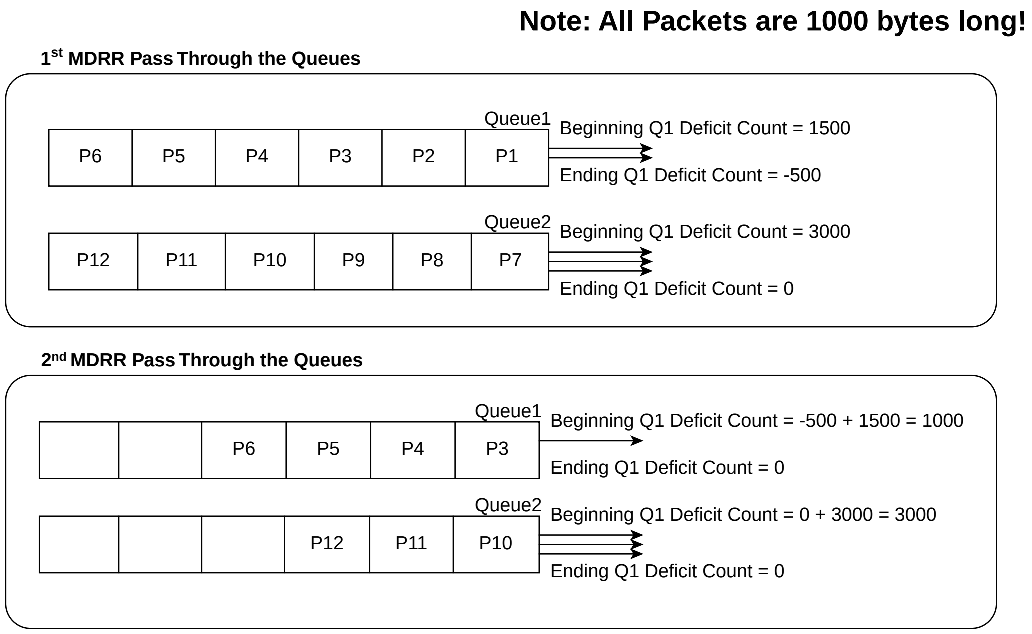

3) Maximum Number of Queues
4) Maximum Queue Length
5) Scheduling Inside Queue
6) Scheduler Logic
1) Classification
2) Drop Decision
?
Dropped
Tail Drop or WRED
Class Based Only on IP Precedence
8 Queues
Max Length Variable
FIFO
Process: Round Robin Through Queues serving a quantum per pass. Deficit feature evens out the process.
PQ feature for a single queue, with strict (always serve the new packets immediately) or alternate (serve the PQ, then take a quantum from another queue, alternating) scheduling
TX Queue/Ring

MDRR schedules traffic by making a round-robin pass through the configured queues. (For the time being, assume that the optional PQ [LLQ] feature has not been configured.) MDRR removes packets from a queue, until the quantum value (QV) for that queue has been removed. The QV quantifies a number of bytes, and is used much like the byte count is used by the CQ scheduler. MDRR repeats the process for every queue, in order from 0 through 7, and then repeats this round-robin process. The end result is that each queue gets some percentage bandwidth of the link.

The CQ scheduler has a problem with trying closely to provide a particular percentage bandwidth. MDRR overcomes this same problem, but it is useful to recall an example of the problem first. Repeating an earlier example about CQ, suppose a router uses CQ on an interface, with 3 queues, with the byte counts configured to 1500, 1500, and 1500. Now suppose that all the packets in the queues are 1500 bytes. (This is not going to happen in real life, but it is useful for making the point.) CQ takes a 1500-byte packet, notices that it has met the byte count, and moves to the next queue. In effect, CQ takes one packet from each queue, and each queue gets 1/3 of the link bandwidth. Now suppose that Queue 3 has been configured to send 1501 bytes per queue service, and all the packets in all queues are still 1500 bytes long. CQ takes 1 packet from Queue 1, 1 from Queue 2, and then 2 packets from Queue 3! CQ does not fragment the packet. In effect, Queue 3 sends 2 packets for every 1 packet sent from Queues 1 and 2, effectively giving 25 percent of the bandwidth each to Queues 1 and 2, and 50 percent of the link bandwidth to Queue 3.

MDRR deals with this problem by treating any “extra” bytes sent during a cycle as a “deficit.” Next time around through the queues, the number of “extra” bytes sent by MDRR is subtracted from the QV. In Figure B-12, MDRR is using only two queues, with QVs of 1500 and 3000, respectively, and with all packets at 1000 bytes in length.

Figure B-12 *MDRR: Making Up Deficits*


Note: All Packets are 1000 bytes long!
1st MDRR Pass Through the Queues
Queue1
P6 P5 P4 P3 P2 P1
Beginning Q1 Deficit Count = 1500
Ending Q1 Deficit Count = -500
Queue2
P12 P11 P10 P9 P8 P7
Beginning Q1 Deficit Count = 3000
Ending Q1 Deficit Count = 0
2nd MDRR Pass Through the Queues
Queue1
P6 P5 P4 P3
Beginning Q1 Deficit Count = -500 + 1500 = 1000
Ending Q1 Deficit Count = 0
Queue2
P12 P11 P10
Beginning Q1 Deficit Count = 0 + 3000 = 3000
Ending Q1 Deficit Count = 0

First, some extra information on how to interpret the figure may help. The figure shows the action during the first round-robin pass in the top half of the figure, and the action during the second pass in the lower half of the figure. The example begins with six packets (labeled P1 through P6) in Queue 1, and six packets (labeled P7 through P12) in Queue 2. Each arrowed line, attached to the right sides of the queues, and pointing to the right, represents the choice by MDRR to send a single packet.

When a queue first fills, the queue's deficit counter is set to the QV for that queue, which is 1500 for Queue 1, and 3000 for Queue 2. In the figure, MDRR begins by taking 1 packet from Queue 1, decrementing the DC to 500, and deciding that the DC has not been decremented to 0 (or less). MDRR takes a second packet from Queue 1, decrementing the DC to –500. MDRR then moves on to Queue 2, taking three packets, after which the DC for Queue 2 has decremented to 0.

That concludes the first pass through the queues. MDRR has taken 2000 bytes from Queue 1, and 3000 from Queue 2, giving the queues 40 percent and 60 percent of link bandwidth, respectively.

In the second round-robin pass, shown in the lower half of the figure, the process begins by MDRR adding the QV for each queue to the DC for each queue. Queue 1's DC becomes 1500 + –500, or 1000, to begin the second pass. During this pass, MDRR takes P3 from Queue 1, decrements DC to 0, and then moves on to Queue 2. After taking three more packets from

Queue 3, decrementing Queue 2's DC to 0, MDRR completes the second pass. Over these two round-robin passes, MDRR has taken 3000 bytes from Queue 1, and 6000 from Queue 2—which is the same ratio as the ratio between the QVs.

With the deficit feature of MDRR, over time each queue receives a guaranteed bandwidth based on the following formula:

$$\frac{\text{QV for Queue}x}{\text{Sum of all QVs}}$$

For additional examples of the operation of the MDRR deficit feature, refer to http://www.cisco.com/warp/public/63/toc_18841.html. Alternatively, you can go to www.cisco.com and search for "Understanding and Configuring MDRR and WRED on the Cisco 12000 Series Internet Router."

MDRR also allows one queue to be used as a PQ, but with two variations on how the PQ is scheduled. One scheduling option, called *strict scheduling*, acts just like PQ's treatment of the High queue, which is also how Low Latency Queuing (LLQ) and IP RTP Priority scheduling works. Whenever the scheduler decides to remove another packet from any queue, it first checks the strict PQ, and takes a packet if one is in the queue. With strict treatment, MDRR does not need to assign a QV to the low-latency queue, because there is no concept of taking a QV of bytes per pass through the queue—the low-latency queue gets strict service whenever it needs it.

MDRR offers a less strict algorithm for servicing the low-latency queue called *alternate service*. With alternate service, the low-latency queue is assigned a QV, just like the other queues. MDRR takes QV bytes from the low-latency queue, and then take bytes from a non-low-latency queue. MDRR then goes back to the low-latency queue, taking QV bytes, and then to another queue. If Queues 0 through 2 are Normal queues, and Queue 7 is the low-latency queue, for instance, MDRR serves the queues in this order: 7, 0, 7, 1, 7, 2, and repeats this sequence. Because the low-latency queue gets serviced on every alternate attempt, it can get a disproportionate amount of bandwidth. However, with alternate service, MDRR does prevent the low-latency queue from taking over the link.

Strict and alternate priority affect delay and jitter differently. With alternate priority, packets experience longer delay and more jitter, as compared with strict scheduling. Alternate priority gives the other queues a little more service, however, which improves their performance.

Table B-19 summarizes some of the key features of MDRR.

Table B-19 *MDRR Functions and Features*

MDRR Feature	Explanation
Classification	Classifies on IP precedence, mapping 1 or more values to a single queue.
Drop policy	Tail drop or WRED, configurable per queue.
Number of queues	8.

Table B-19 *MDRR Functions and Features (Continued)*

MDRR Feature	Explanation
Maximum queue length	Variable, based on the type of card.
Scheduling inside a single queue	FIFO.
Scheduling among all queues	Deficit round-robin logic is used for queues that are not priority queues, with the scheduler taking a number of bytes from each queue during each pass through the queues. The number of any extra bytes taken in one pass is deducted from the number taken in the next pass. LLQ service is either strict (like PQ), or alternate, where MDRR alternates between the low-latency queue and a non-low-latency queue, serving the low-latency queue half of the service slots.

MDRR Configuration

MDRR configuration requires you to understand GSR architecture to a small degree. MDRR configuration also introduces some new terminology. Table B-20 lists the configuration commands related to MDRR, and following the tables is an explanation of the new terminology, the formula used to calculate the QV per queue, and an explanation of the flow of the configuration commands.

Table B-20 *Command Reference for MDRR*

Command	Mode and Function
cos-queue-group *cos-queue-group-name*	Global config mode; creates a queue group template, where other parameters are added as subcommands
precedence <0-7> queue [**<0-6>** | **low-latency**]	**cos-queue** subcommand; maps a precedence value to a queue.
queue *queue-number weight*	**cos-queue** subcommand; assigns a weight to a queue.
queue low-latency {**alternate-priority** *weight* | **strict-priority**}	**cos-queue** subcommand; enables a queue as a low-latency queue, defines scheduling strategy, and assigns weight
random-detect-label *label minimum-threshold maximum-threshold mark-probability*	**cos-queue** subcommand; creates a set of WRED parameters
precedence *queue-number* **random-detect-label** *label*	**cos-queue** subcommand; enables WRED on the queue, with the WRED parameters in the **random-detect-label** specified.

After you understand the basic concepts, you just need to understand a few facts about how the configuration commands work before being able to configure basic MDRR. First, the low-latency queue is always Queue 7, and instead of being referred to as Queue 7, the low-latency queue is referred to with the **low-latency** keyword. The next thing to note is that the QoS configuration details are configured as subcommands of the **cos-queue-group** command. In other words, the classification details, choosing QVs by using the **weight** keyword, and selecting strict or alternate LLQ scheduling, are configured as subcommands of the **cos-queue-group** command. MDRR configuration does not enable you to specify the QV for each queue, but rather you configure a value called *weight*. The QV is then calculated from the weight using the following formula:

QV = MTU + Weight * 512

Here, MTU is the maximum transmission unit configured for the interface. Weight values can be configured between 1 and 2048, but Cisco recommends using low values—in fact, for the lowest-weight queue, use a value of 1.

Finally, like LAN switches, GSR routers benefit from having both input and output queuing. MDRR can be applied to an interface for input packets using the **rx-cos** command, and for output packets using the **tx-cos** command.

The following criteria is used in the upcoming example configuration:

- Precedence 0 and 1 are placed in Queue 0, with a weight of 1.
- Precedence 5 is placed in the low-latency queue, with strict priority.
- All other precedence values are placed in Queue 1, with a weight of 3.
- WRED is used on Queues 0 and 1, with default WRED parameters.
- The MDRR configuration is used for packets entering interface pos (Packet over SONET) 3/0, and for packets exiting pos 3/1.

Example B-15 shows the configuration and **show** commands.

Example B-15 *MDRR: VoIP in High Queue, All Else in Normal Queue*

```
interface pos 3/0
 Rx-cos my-sample
Interface pos 3/1
 Tx-cos my-sample
!
cos-queue-group my-sample
 precedence 0 queue 0
 precedence 0 random-detect-label 0
 precedence 1 queue 0
 precedence 1 random-detect-label 0
 precedence 2 queue 1
 precedence 2 random-detect-label 0
 precedence 3 queue 1
 precedence 3 random-detect-label 0
```

Example B-15 *MDRR: VoIP in High Queue, All Else in Normal Queue (Continued)*

```
 precedence 4 queue 1
 precedence 4 random-detect-label 0
 precedence 5 queue low-latency
 precedence 6 queue 1
 precedence 6 random-detect-label 0
 precedence 7 queue 1
 precedence 7 random-detect-label 0
!
 precedence 0 random-detect-label 0
!
 queue 0 1
 queue 1 3
 queue low-latency strict
!
 random-detect-label 0
```

Most of the commands in the sample are configuration subcommands under the **cos-queue-group my-sample** command. For instance, the **precedence 0 queue 0** command maps precedence 0 packets to Queue 0; a similar command maps the appropriate precedence values to queues, as stated in the policy before the example. Similarly, WRED parameters apply per precedence, so the **precedence 0 random-detect-label 0** command links precedence 0 packets to a set of WRED parameters with a WRED label of 0. At the end of the example, the **random-detect-label 0** command creates a WRED label 0, with no specific parameters—even if you want to take default WRED parameters, you still need to create the label, and refer to it with a command such as **precedence 0 random-detect-label 0** to enable WRED for those packets.

The weights are configured with the **queue** command seen toward the end of the configuration. The **queue 0 1** command assigns a weight of 1 to Queue 0, and the **queue 1 3** command assigns a weight of 3 to Queue 1. The **queue low-latency strict** command defines that Queue 7, the low-latency queue, should use strict scheduling.

Finally, at the top of the configuration, the **rx-cos my-sample** command enables **cos-queue-list my-sample** for packets entering interface pos 3/0. Similarly, the **tx-cos my-sample** interface subcommand enables MDRR for packets exiting pos 3/1.

MDRR provides queuing services on GSR 12000 series routers. It assigns a percentage bandwidth through the use of the **weight** keyword, with the percentages being much more exact than CQ through the deficit feature of the MDRR scheduler. MDRR allows two options for treatment of the low-latency queue, strict and alternate, which gives the engineer more flexibility in how to deal with delay- and jitter-sensitive traffic.

The classification feature of MDRR only supports IP precedence, which can be seen as a negative. Because most networks deploy GSRs in the core of the network, however, a good QoS design, with classification and marking at the edge of the network, can work very well with the precedence-based classification of MDRR.

Q&A for Queuing Tools

1 What command enables you to configure Priority Queuing to classify based on ACL 101, placing the traffic in the Medium queue?

Answer: priority-list 5 protocol ip medium list 101

2 What command enables you to configure Custom Queuing to classify based on ACL 101, placing the traffic into queue 4?

Answer: queue-list 5 protocol ip 4 list 101

3 Which interface subcommands enable Priority Queuing list 5 on interface s 0/0? Custom Queuing?

Answer: priority-group 5 and custom-queue-list 5

4 Compare and contrast WFQ and dWFQ. Specifically mention features relating to scheduling, classification, and drop policy.

Answer: WFQ and dWFQ both classify based on the flow. The dWFQ drop policy drops a packet if the individual queue limit or aggregate queue limit is reached. WFQ can also dequeue and discard a previously enqueued packet, if a new packet has a better sequence number. The WFQ scheduler weights packets based on length and IP precedence, whereas dWFQ only weights based on length.

5 Compare and contrast WFQ and ToS-Based dWFQ. Specifically mention features relating to scheduling, classification, and drop policy.

Answer: WFQ classifies based on the flow, whereas ToS-Based dWFQ classifies based on the 2 low-order bits of the IP Precedence field. The ToS-Based dWFQ Drop policy drops a packet if the individual queue limit or aggregate queue limit is reached. WFQ can also dequeue and discard a previously enqueued packet, if a new packet has a better sequence number. The WFQ scheduler weights packets based on length and IP precedence, whereas ToS-Based dWFQ guarantees a minimum amount of bandwidth to each queue.

6 Compare and contrast WFQ and QoS Group–Based dWFQ. Specifically mention features relating to scheduling, classification, and drop policy.

Answer: WFQ classifies based on the flow, whereas QoS Group–Based dWFQ classifies based on the QoS group. The QoS Group–Based dWFQ drop policy drops a packet if the individual queue limit or aggregate queue limit is reached. WFQ can also dequeue and discard a previously enqueued packet, if a new packet has a better sequence number. The WFQ scheduler weights packets based on length and IP precedence, whereas QoS Group–Based dWFQ guarantees a minimum amount of bandwidth to each queue.

7 Compare and contrast WFQ, dWFQ, ToS-Based dWFQ, and QoS Group–Based dWFQ, in relation to the maximum number of queues. Also discuss why the wide range of values is meaningful.

Answer: These four tools allow a maximum of 4096, 512, 4, and 100 queues, respectively. WFQ and dWFQ have a much larger number of queues, because they are flow based. ToS-Based and QoS Group–Based dWFQ are class based, meaning many flows end up in a single queue, therefore requiring fewer queues.

8 Compare and contrast WFQ, dWFQ, ToS-Based dWFQ, and QoS Group–Based dWFQ, in relation to which tools can be enabled on 2600 and 7500 series platforms. Also identify the requirements for a 7500 to support these tools.

Answer: These four tools can be enabled on 7500s, but only WFQ can be deployed on 2600 routers. The other three tools are specifically designed for use with 7500 series routers. The router must be running distributed CEF, and have VIP2-40 or better line cards installed on the interfaces on which the queuing tool is deployed.

9 You are migrating from WFQ on a 7200 series router to dWFQ on a new 7500 series with VIP2-50s installed. You will be migrating the old 7200 configuration over to the 7500. What changes are needed to support dWFQ, other than changing the numbers of the interfaces?

Answer: The ip cef command is required, in case the 7200 did not run CEF. Because the fair-queue command already exists on the interface, the router automatically uses dWFQ.

10 What commands define the percentage bandwidth to be used for Queue 2 for ToS-Based dWFQ? For QoS Group–Based dWFQ?

Answer: The fair-queue tos 2 weight 30 command defines Queue 2 to have 30 percent of guaranteed bandwidth. The fair-queue qos 2 weight 30 command defines the same, but for QoS Group–Based dWFQ.

11 Describe the classification options available to Modified Deficit Round-Robin (MDRR), and how these options relate to the number of available queues.

Answer: MDRR classifies based on IP precedence, with eight queues. Conceivably, each precedence value could be placed into a different queue, but MDRR enables you to map each precedence value to a queue, with multiple precedence values into the same queue.

12 Describe how the Modified Deficit Round-Robin scheduler works, and specifically why the word "deficit" refers to part of the scheduler logic.

Answer: DRR schedules some number of bytes per pass through the queues. MDRR takes packets from the queue, which means it may take more than the allotted number of bytes; this excess is called the *deficit*. The deficit is subtracted from the number of bytes taken from that queue in the next round. As a result, MDRR can accurately predict the percentage bandwidth assigned to a queue.

13 Describe the names and logic behind the two options for low-latency queues with Modified Deficit Round-Robin (MDRR).

Answer: MDRR allows one queue to be used as a low-latency queue. With strict scheduling, the low-latency queue is serviced like PQ's High queue—if a packet is waiting in that queue, it always goes next. With alternate scheduling, the low-latency queue is serviced, taking a configured number of bytes. Then another non-low-latency queue is serviced. Then the low-latency queue is served again, and so on, so every other queue service serves the low-latency queue.

14 Describe the drop policy options for Modified Deficit Round-Robin (MDRR).

Answer: MDRR can use tail drop or WRED inside each queue.

15 List the command syntax of the command that configures the type of queue service to use with the MDRR low-latency queue.

Answer: queue low-latency {alternate-priority *weight* **| strict-priority}**

16 List the command syntax of the command that maps an IP precedence to a particular queue when using Modified Deficit Round-Robin (MDRR).

Answer: precedence <0-7> queue [<0-6> | low-latency]

INDEX