PART 2

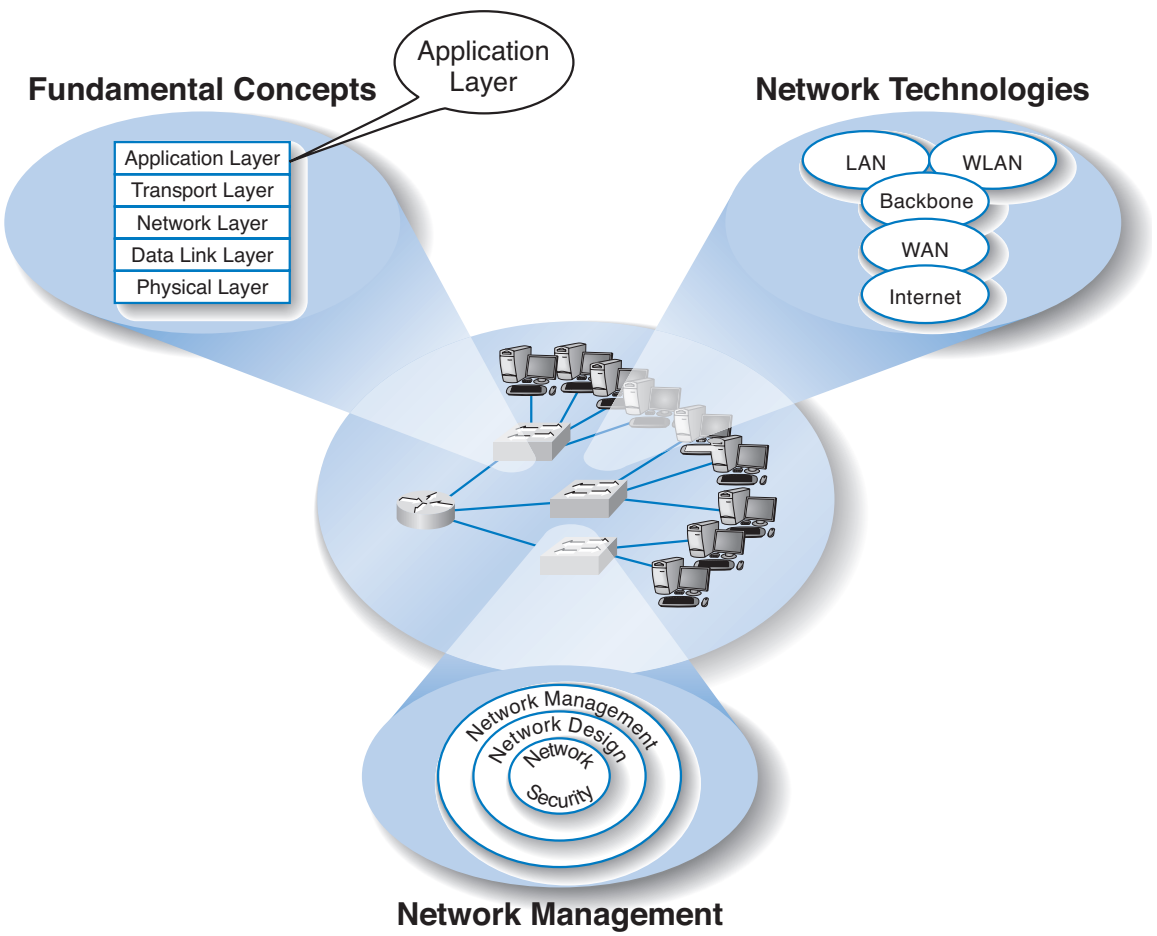
FUNDAMENTAL CONCEPTS

Courtesy Cisco Systems, Inc.

Network switches from Cisco Systems, Inc.

CHAPTER 2

APPLICATION LAYER



The Three Faces of Networking

THE APPLICATION layer (also called layer 5) is the software that enables the user to perform useful work. The software at the application layer is the reason for having the network because it is this software that provides the business value. This chapter examines the three fundamental types of application architectures used at the application layer (host-based, client-based, client-server). It then looks at the Internet and the primary software application packages it enables: the Web, e-mail, Telnet, FTP, and Instant Messaging.

OBJECTIVES

- Understand host-based, client-based, and client–server application architectures
- Understand how the Web works
- Understand how e-mail works
- Be aware of how FTP, Telnet, and instant messaging works

CHAPTER OUTLINE

INTRODUCTION

Network applications are the software packages that run in the application layer. You should be quite familiar with many types of network software, because it is these application packages that you use when you use the network. In many respects, the only reason for having a network is to enable these applications.

In this chapter, we first discuss three basic architectures for network applications and how each of those architectures affects the design of networks. Because you probably have a good understanding of applications such as the Web and word processing, we will use those as examples of different application architectures. We then examine several common applications used on the Internet (e.g., Web, e-mail) and use those to explain how application software interacts with the networks. By the end of this chapter, you should have a much better understanding of the application layer in the network model and what exactly we meant when we used the term *packet* in Chapter 1.

APPLICATION ARCHITECTURES

In Chapter 1, we discussed how the three basic components of a network (client computer, server computer, and circuit) worked together. In this section, we will get a bit more specific about how the client computer and the server computer can work together to provide application software to the users. An *application architecture* is the way in which the functions of the application layer software are spread among the clients and servers in the network.

The work done by any application program can be divided into four general functions. The first is *data storage.* Most application programs require data to be stored and retrieved, whether it is a small file such as a memo produced by a word processor or a large database such as an organization's accounting records. The second function is *data access logic,* the processing required to access data, which often means database queries in SQL (structured query language). The third function is the *application logic* (sometimes called business logic), which also can be simple or complex, depending on the application. The fourth function is the *presentation logic,* the presentation of information to the user and the acceptance of the user's commands. These four functions, data storage,

TECHNICAL FOCUS

2-1 CLIENTS AND SERVERS

There are many different types of clients and servers that can be part of a network, and the distinctions between them have become a bit more complex over time. Generally speaking, there are four types of computers that are commonly used as servers:

- A *mainframe* is a very large general-purpose computer (usually costing millions of dollars) that is capable of performing *very* many simultaneous functions, supporting *very* many simultaneous users, and storing *huge* amounts of data.
- A *minicomputer* is a large general-purpose computer (usually costing hundreds of thousands of dollars) that is capable of performing many simultaneous functions, supporting many simultaneous users, and storing large amounts of data. Minicomputers are sometimes used as database servers in client-server networks.
- A *microcomputer* is the type of computer you use. Microcomputers used as servers can range from a small microcomputer, similar to a desktop one you might use, to one costing $50,000 or more.
- A *cluster* is a group of computers (often microcomputers or workstations) linked together so that they act as one computer. Requests arrive at the cluster (e.g., Web requests) and are distributed among the computers so that no one computer is overloaded. Each computer is separate, so that if one fails, the cluster simply bypasses it. Clusters are more complex than single servers because work must be quickly coordinated and shared among the individual computers. Clusters are very scalable because one can always add one more computer to the cluster.

There are six commonly used types of clients:

- A *microcomputer* is the most common type of client today. This includes desktop and portable computers, as well as Tablet PCs that enable the user to write with a pen-like stylus instead of typing on a keyboard.
- A *terminal* is a device with a monitor and keyboard but no central processing unit (CPU). *Dumb terminals,* so named because they do not participate in the processing of the data they display, have the bare minimum required to operate as input and output devices (a TV screen and a keyboard). In most cases when a character is typed on a dumb terminal, it transmits the character through the circuit to the server for processing. Every keystroke is processed by the server, even simple activities such as the up arrow. *Intelligent terminals* were developed to reduce the processing demands on the server and have some small internal memory and a built-in, programmable microprocessor chip. Many simple functions, such as moving the cursor or displaying words in different colors, are done by the terminal, thus saving processing time on the server.
- A *workstation* is a more powerful microcomputer designed for use in technical applications such as mathematical modeling, computer-assisted design (CAD), and intensive programming. As microcomputers become more powerful, the difference between a microcomputer and a workstation is blurring.
- A *network computer* is designed primarily to communicate using Internet-based standards (e.g., HTTP, Java) but has no hard disk. It has only limited functionality.
- A *transaction terminal* is designed to support specific business transactions, such as the automated teller machines (ATM) used by banks. Other examples of transaction terminals are point-of-sale terminals in a supermarket.
- A handheld computer, Personal Digital Assistant (PDA), or mobile phone can also be used as a network client.

data access logic, application logic, and presentation logic, are the basic building blocks of any application.

There are many ways in which these four functions can be allocated between the client computers and the servers in a network. There are three fundamental application architectures in use today. In *host-based* architectures, the server (or host computer) performs virtually all of the work. In *client-based* architectures, the client computers perform most of the work. In *client-server* architectures, the work is shared between the servers and clients. The client-server architecture is becoming the dominant application architecture.

Host-Based Architectures

The very first data communications networks developed in the 1960s were host-based, with the server (usually a large mainframe computer) performing all four functions. The clients (usually terminals) enabled users to send and receive messages to and from the host computer. The clients merely captured keystrokes, sent them to the server for processing, and accepted instructions from the server on what to display (Figure 2.1).

This very simple architecture often works very well. Application software is developed and stored on the one server along with all data. If you've ever used a terminal (or a microcomputer with Telnet software), you've used a host-based application. There is one point of control, because all messages flow through the one central server. In theory, there are economies of scale, because all computer resources are centralized (but more on cost later).

There are two fundamental problems with host-based networks. First, the server must process all messages. As the demands for more and more network applications grow, many servers become overloaded and unable to quickly process all the users' demands. Prioritizing users' access becomes difficult. Response time becomes slower, and network managers are required to spend increasingly more money to upgrade the server. Unfortunately, upgrades to the mainframes that usually are the servers in this architecture are "lumpy." That is, upgrades come in large increments and are expensive (e.g., $500,000); it is difficult to upgrade "a little."

Client-Based Architectures

In the late 1980s, there was an explosion in the use of microcomputers and microcomputer-based LANs. Today, more than 90 percent of most organizations' total computer

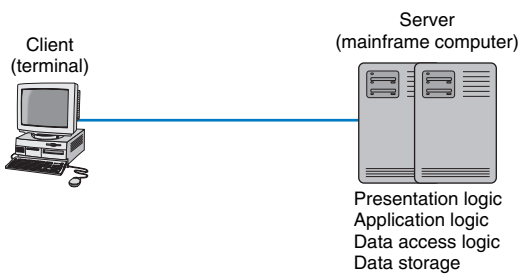


FIGURE 2.1 Host-based architecture.

processing power now resides on microcomputer-based LANs, not in centralized mainframe computers. Part of this expansion was fueled by a number of low-cost, highly popular applications such as word processors, spreadsheets, and presentation graphics programs. It was also fueled in part by managers' frustrations with application software on host mainframe computers. Most mainframe software is not as easy to use as microcomputer software, is far more expensive, and can take years to develop. In the late 1980s, many large organizations had application development backlogs of 2 to 3 years; that is, getting any new mainframe application program written would take years. New York City, for example, had a 6-year backlog. In contrast, managers could buy microcomputer packages or develop microcomputer-based applications in a few months.

With client-based architectures, the clients are microcomputers on a LAN, and the server is usually another microcomputer on the same network. The application software on the client computers is responsible for the presentation logic, the application logic, and the data access logic; the server simply stores the data (Figure 2.2).

This simple architecture often works very well. If you've ever used a word processor and stored your document file on a server (or written a program in Visual Basic or C that runs on your computer but stores data on a server), you've used a client-based architecture.

The fundamental problem in client-based networks is that all data on the server must travel to the client for processing. For example, suppose the user wishes to display a list of all employees with company life insurance. All the data in the database (or all the indices) must travel from the server where the database is stored over the network circuit to the client, which then examines each record to see if it matches the data requested by the user. This can overload the network circuits because far more data is transmitted from the server to the client than the client actually needs.

Client-Server Architectures

Most organizations today are moving to client-server architectures. Client-server architectures attempt to balance the processing between the client and the server by having both do some of the logic. In these networks, the client is responsible for the presentation logic, whereas the server is responsible for the data access logic and data storage. The application logic may either reside on the client, reside on the server, or be split between both.

Figure 2.3 shows the simplest case, with the presentation logic and application logic on the client and the data access logic and data storage on the server. In this case, the client software accepts user requests and performs the application logic that produces

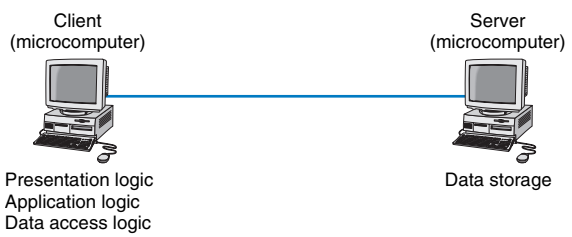


FIGURE 2.2 Client-based architecture.

database requests that are transmitted to the server. The server software accepts the database requests, performs the data access logic, and transmits the results to the client. The client software accepts the results and presents them to the user. When you used a Web browser to get pages from a Web server, you used a client-server architecture. Likewise, if you've ever written a program that uses SQL to talk to a database on a server, you've used a client-server architecture.

For example, if the user requests a list of all employees with company life insurance, the client would accept the request, format it so that it could be understood by the server, and transmit it to the server. On receiving the request, the server searches the database for all requested records and then transmits only the matching records to the client, which would then present them to the user. The same would be true for database updates; the client accepts the request and sends it to the server. The server processes the update and responds (either accepting the update or explaining why not) to the client, which displays it to the user.

One of the strengths of client-server networks is that they enable software and hardware from different vendors to be used together. But this is also one of their disadvantages, because it can be difficult to get software from different vendors to work together. One solution to this problem is *middleware,* software that sits between the application software on the client and the application software on the server. Middleware does two things. First, it provides a standard way of communicating that can translate between software from different vendors. Many middleware tools began as translation utilities that enabled messages sent from a specific client tool to be translated into a form understood by a specific server tool.

The second function of middleware is to manage the message transfer from clients to servers (and vice versa) so that clients need not know the specific server that contains the application's data. The application software on the client sends all messages to the middleware, which forwards them to the correct server. The application software on the client is therefore protected from any changes in the physical network. If the network layout changes (e.g., a new server is added), only the middleware must be updated.

There are literally dozens of standards for middleware, each of which is supported by different vendors and each of which provides different functions. Two of the most important standards are Distributed Computing Environment (DCE) and Common Object Request Broker Architecture (CORBA). Both of these standards cover virtually all aspects of the client-server architecture but are quite different. Any client or server software that conforms to one of these standards can communicate with any other software that conforms to the same standard. Another important standard is Open Database Connectivity (ODBC), which provides a standard for data access logic.

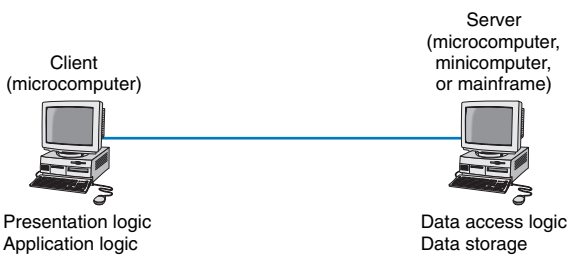


FIGURE 2.3 Two-tier client-server architecture.

MANAGEMENT FOCUS **2-1 A MONSTER CLIENT-SERVER ARCHITECTURE**

Every spring, Monster.com, one of the largest job sites in the United States, with an average of more than 40 million visits per month, experiences a large increase in traffic. Aaron Braham, vice president of operations, attributes the spike to college students who increase their job search activities as they approach graduation.

Monster.com has 1,000 Web servers, e-mail servers, and database servers at its sites in Indianapolis and Maynard, Massachusetts. The main Web site has a set of load-balancing devices that forward Web requests to the different servers depending on how busy they are.

Braham says the major challenge is that 90 percent of the traffic is not simple requests for Web pages but rather search requests (e.g., what network jobs are available in New Mexico), which require more processing and access to the database servers. Monster.com has more than 1 million job postings and more than 20 million résumés on file, spread across its database servers. Several copies of each posting and résumé are kept on several database servers to improve access speed and provide redundancy in case a server crashes, so just keeping the database servers in sync so that they contain correct data is a challenge.

SOURCE: monster.com case study,www.Dell.com, 2004

Two-Tier, Three-Tier, and n-Tier Architectures There are many ways in which the application logic can be partitioned between the client and the server. The example in Figure 2.3 is one of the most common. In this case, the server is responsible for the data and the client, the application and presentation. This is called a *two-tier architecture,* because it uses only two sets of computers, one set of clients and one set of servers.

A *three-tier architecture* uses three sets of computers, as shown in Figure 2.4. In this case, the software on the client computer is responsible for presentation logic, an application server is responsible for the application logic, and a separate database server is responsible for the data access logic and data storage.

An *n-tier architecture* uses more than three sets of computers. In this case, the client is responsible for presentation logic, a database server is responsible for the data access logic and data storage, and the application logic is spread across two or more dif-

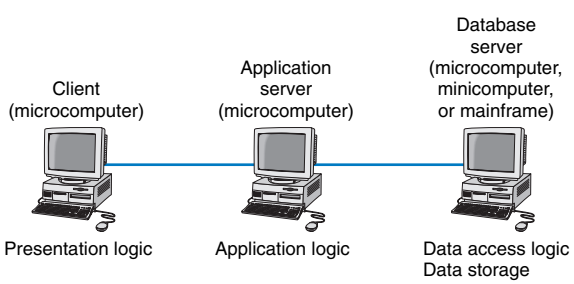


FIGURE 2.4 Three-tier client-server architecture.

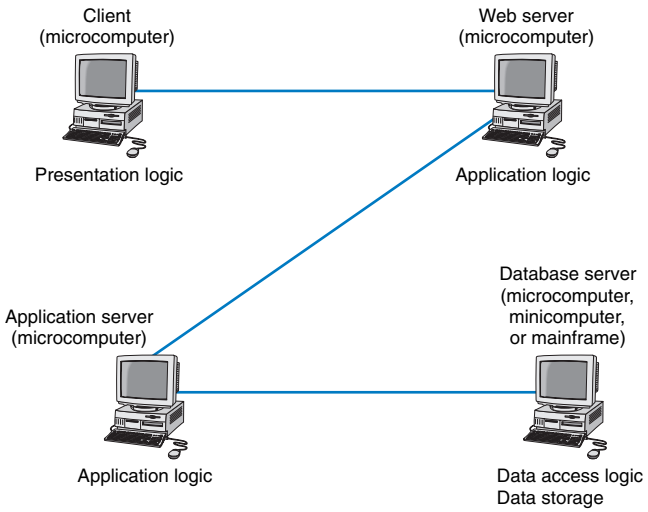


FIGURE 2.5 The *n*-tier client-server architecture.

ferent sets of servers. Figure 2.5 shows an example of an *n*-tier architecture of a groupware product called TCB-Works developed at the University of Georgia. TCB Works has four major components. The first is the Web browser on the client computer that a user uses to access the system and enter commands (presentation logic). The second component is a Web server that responds to the user's requests, either by providing HTML pages and graphics (application logic) or by sending the request to the third component, a set of 28 C programs that perform various functions such as adding comments or voting (application logic). The fourth component is a database server that stores all the data (data access logic and data storage). Each of these four components is separate, making it easy to spread the different components on different servers and to partition the application logic on two different servers.

The primary advantage of an *n*-tier client-server architecture compared with a two-tier architecture (or a three-tier with a two-tier) is that it separates out the processing that occurs to better balance the load on the different servers; it is more scalable. In Figure 2.5, we have three separate servers, which provides more power than if we had used a two-tier architecture with only one server. If we discover that the application server is too heavily loaded, we can simply replace it with a more powerful server, or even put in two application servers. Conversely, if we discover the database server is underused, we could put data from another application on it.

There are two primary disadvantages to an *n*-tier architecture compared with a two-tier architecture (or a three-tier with a two-tier). First, it puts a greater load on the network. If you compare Figures 2.3, 2.4, and 2.5, you will see that the *n*-tier model requires more communication among the servers; it generates more network traffic so you need a higher capacity network. Second, it is much more difficult to program and test software in *n*-tier architectures than in two-tier architectures because more devices have to communicate to complete a user's transaction.

Thin Clients versus Thick Clients Another way of classifying client-server architectures is by examining how much of the application logic is placed on the client computer. A *thin-client* approach places little or no application logic on the client (e.g., Figure 2.5), whereas a *thick-client* (also called *fat-client*) approach places all or almost all of the application logic on the client (e.g., Figure 2.3). There is no direct relationship between thin and fat client and two-, three- and *n*-tier architectures. For example, Figure 2.6 shows a typical Web architecture: a two-tier architecture with a thin client. One of the biggest forces favoring thin clients is the Web.

Thin clients are much easier to manage. If an application changes, only the server with the application logic needs to be updated. With a thick client, the software on all of the clients would need to be updated. Conceptually, this is a simple task; one simply copies the new files to the hundreds of affected client computers. In practice, in can be a very difficult task.

Thin-client architectures are the wave of the future. More and more application systems are being written to use a Web browser as the client software, with Java applets (containing some of the application logic) downloaded as needed. This application architecture is sometimes called the *distributed computing model.*

Choosing Architectures

Each of the preceding architectures has certain costs and benefits, so how do you choose the "right" architecture? In many cases, the architecture is simply a given; the organization has a certain architecture, and one simply has to use it. In other cases, the organization is acquiring new equipment and writing new software and has the opportunity to develop a new architecture, at least in some part of the organization. There are at least three major sets of factors to consider (Figure 2.7).

Cost of Infrastructure One of the strongest forces driving companies toward client-server architectures is cost of infrastructure (the hardware, software, and networks that will support the application system). Simply put, microcomputers are more than 1,000 times cheaper than mainframes for the same amount of computing power. The microcomputers on our desks today have more processing power, memory, and hard disk space than a mainframe of the early 1990s, and the cost of the microcomputers is a frac-

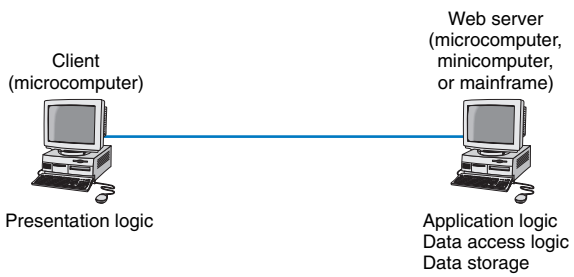


FIGURE 2.6 The typical two-tier thin-client architecture of the Web.

	Host-Based	Client-Based	Client-Server
Cost of infrastructure	High	Medium	Low
Cost of development	Low	Medium	Medium
Scalability	Low	Medium	High

FIGURE 2.7 Factors involved in choosing architectures.

tion of the cost of the mainframe. Therefore, the cost of client-server architectures is lower than that of server-based architectures, which rely on mainframes. Client-server architectures also tend to be cheaper than client-based architectures because they place less of a load on networks and thus require less network capacity.

Cost of Development The cost of developing application systems is an important factor when considering the financial benefits of client-server architectures. Developing application software for client-server architectures can be complex. Developing application software for host-based architectures is usually cheaper. The cost differential may change as companies gain experience with client-server applications, as new client-server products are developed and refined, and as client-server standards mature. However, given the inherent complexity of client-server software and the need to coordinate the interactions of software on different computers, there is likely to remain a cost difference.

Even updating the network with a new version of the software is more complicated. In a host-based network, there is one place in which application software is stored; to update the software, you simply replace it there. With client-server networks, you must update all clients and all servers. For example, suppose you want to add a new server and move some existing applications from the old server to the new one. All application software on all fat clients that send messages to the application on the old server must now be changed to send to the new server. Although this is not conceptually difficult, it can be an administrative nightmare.

Scalability *Scalability* refers to the ability to increase or decrease the capacity of the computing infrastructure in response to changing capacity needs. The most scalable architecture is client-server computing because servers can be added to (or removed from) the architecture when processing needs change. For example, in a four-tier client-server architecture, one might have 10 Web servers, four application servers, and three database servers. If the application servers begin to get overloaded, it is simple to add another two or three application servers.

Also, the types of hardware that are used in client-server settings (e.g., minicomputers) typically can be upgraded at a pace that most closely matches the growth of the application. In contrast, host-based architectures rely primarily on mainframe hardware that needs to be scaled up in large, expensive increments, and client-based architectures have ceilings above which the application cannot grow because increases in use and data can result in increased network traffic to the extent that performance is unacceptable.

WORLD WIDE WEB

The Web was first conceived in 1989 by Sir Tim Berners-Lee at the European Particle Physics Laboratory (CERN) in Geneva. His original idea was to develop a database of information on physics research, but he found it difficult to fit the information into a traditional data-base. Instead, he decided to use a *hypertext* network of information. With hypertext, any document can contain a link to any other document.

CERN's first Web browser was created in 1990, but it was 1991 before it was available on the Internet for other organizations to use. By the end of 1992, several browsers had been created for UNIX computers by CERN and several other European and American universities, and there were about 30 Web servers in the entire world. In 1993, Marc Andreessen, a student at the University of Illinois, led a team of students that wrote Mosaic, the first graphical Web browser, as part of a project for the university's National Center for Supercomputing Applications (NCSA). By the end of 1993, the Mosaic browser was available for UNIX, Windows, and Macintosh computers, and there were about 200 Web servers in the world. In 1994, Andreessen and some colleagues left NCSA to form Netscape, and a half a dozen other startup companies introduced commercial Web browsers. Within a year, it had become clear that the Web had changed the face of computing forever. NCSA stopped development of the Mosaic browser in 1996, as Netscape and Microsoft began to invest millions to improve their browsers.

How the Web Works

The Web is a good example of a two-tier client-server architecture (Figure 2.8). Each client computer needs an application layer software package called a *Web browser.* There are many different browsers, such as Microsoft's Internet Explorer. Each server on the network that will act as a Web server needs an application layer software package called a *Web server.* There are many different Web servers, such as those produced by Microsoft and Apache.

To get a page from the Web, the user must type the Internet *uniform resource locator (URL)* for the page he or she wants (e.g., www.yahoo.com) or click on a link that provides the URL. The URL specifies the Internet address of the Web server and the directory and name of the specific page wanted. If no directory and page are specified, the Web server will provide whatever page has been defined as the site's home page. If no server name is specified, the Web browser will presume the address is on the same server and directory as the page containing the URL.

For the requests from the Web browser to be understood by the Web server, they must use the same standard *protocol* or language. If there were no standard and each Web browser used a different protocol to request pages, then it would be impossible for a Microsoft Web browser to communicate with an Apache Web server, for example.

The standard protocol for communication between a Web browser and a Web server is *Hypertext Transfer Protocol (HTTP).*[1] To get a page from a Web server, the Web

[1]The formal specification for HTTP version 1.1 is provided in RFC 2616 on the IETF's Web site. The URL is www.ietf.org/rfc/rfc2616.txt.

browser issues a special packet called an *HTTP request* that contains the URL and other information about the Web page requested (see Figure 2.8). Once the server receives the HTTP request, it processes it and sends back an *HTTP response,* which will be the requested page or an error message (see Figure 2.8).

This request-response dialogue occurs for every file transferred between the client and the server. For example, suppose the client requests a Web page that has two graphic images. Graphics are stored in separate files from the Web page itself using a different file format than the HTML used for the Web page (in JPEG [Joint Photographic Experts Group] format, for example). In this case, there would be three request-response pairs. First, the browser would issue a request for the Web page, and the server would send the response. Then, the browser would begin displaying the Web page and notice the two graphic files. The browser would then send a request for the first graphic and a request for the second graphic, and the server would reply with two separate HTTP responses, one for each request.

Inside an HTTP Request

The HTTP request and HTTP response are examples of the packets we introduced in Chapter 1 that are produced by the application layer and sent down to the transport, network, data link, and physical layers for transmission through the network. The HTTP response and HTTP request are simple text files that take the information provided by the application (e.g., the URL to get) and format it in a structured way so that the receiver of the message can clearly understand it.

An HTTP request from a Web browser to a Web server has three parts. The first two parts are required; the last is optional. The parts are:

- The *request line,* which starts with a command (e.g., get), provides the Web page and ends with the HTTP version number that the browser understands; the version number ensures that the Web server does not attempt to use a more advanced or newer version of the HTTP standard that the browser does not understand.
- The *request header,* which contains a variety of optional information such as the Web browser being used (e.g., Internet Explorer) and the date.
- The *request body,* which contains information sent to the server, such as information that the user has typed into a form.

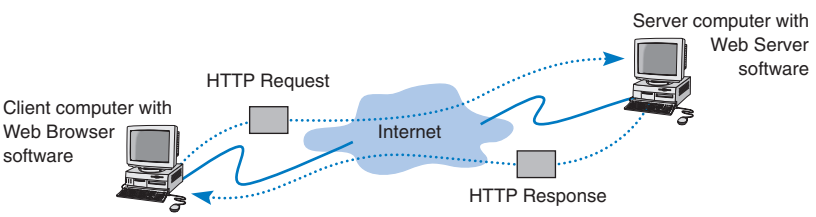


FIGURE 2.8 How the Web works.

Figure 2.9 shows an example of an HTTP request for a page on our Web server, formatted using version 1.1 of the HTTP standard. This request has only the request line and the request header, because no request body is needed for this request. This request includes the date and time of the request (expressed in Greenwich Mean Time [GMT], the time zone that runs through London) and name of the browser used (Mozilla is the code name for the browser). The "Referrer" field means that the user obtained the URL for this Web page by clicking on a link on another page, which in this case is a list of faculty at Indiana University (i.e., www.indiana.edu/~isdept/faculty.htm). If the referrer field is blank, then it means the user typed the URL him- or herself. You can see inside HTTP headers yourself at www.rexswain.com/httpview.html.

Inside an HTTP Response

The format of an HTTP response from the server to the browser is very similar to the HTTP request. It, too, has three parts, with the first two required and the last optional:

- The *response status,* which contains the HTTP version number the server has used, a status code (e.g., *200* means "okay"; *404* means "not found"), and a reason phrase (a text description of the status code).
- The *response header,* which contains a variety of optional information, such as the Web server being used (e.g., Apache), the date, and the exact URL of the page in the response.
- The *response body,* which is the Web page itself.

Figure 2.10 shows an example of a response from our Web server to the request in Figure 2.9. This example has all three parts. The response status reports "OK," which means the requested URL was found and is included in the response body. The response header provides the date, the type of Web server software used, the actual URL included

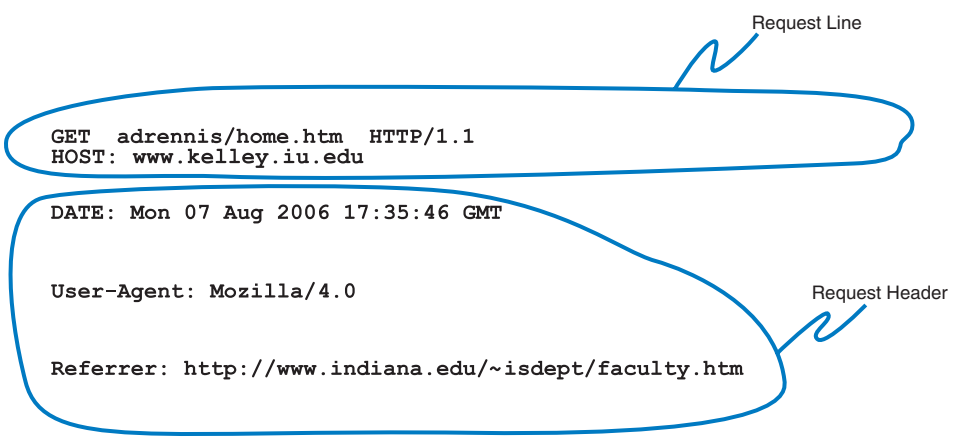


FIGURE 2.9 An example of a request from a Web browser to a Web server using the HTTP (Hypertext Transfer Protocol) standard.

in the response body, and the type of file. In most cases, the actual URL and the requested URL are the same, but not always. For example, if you request an URL but do not specify a file name (e.g., www.indiana.edu), you will receive whatever file is defined as the home page for that server, so the actual URL will be different from the requested URL.

The response body in this example shows a Web page in *Hypertext Markup Language (HTML).* The response body can be in any format, such as text, Microsoft Word, Adobe PDF, or a host of other formats, but the most commonly used format is HTML. HTML was developed by CERN at the same time as the first Web browser and has

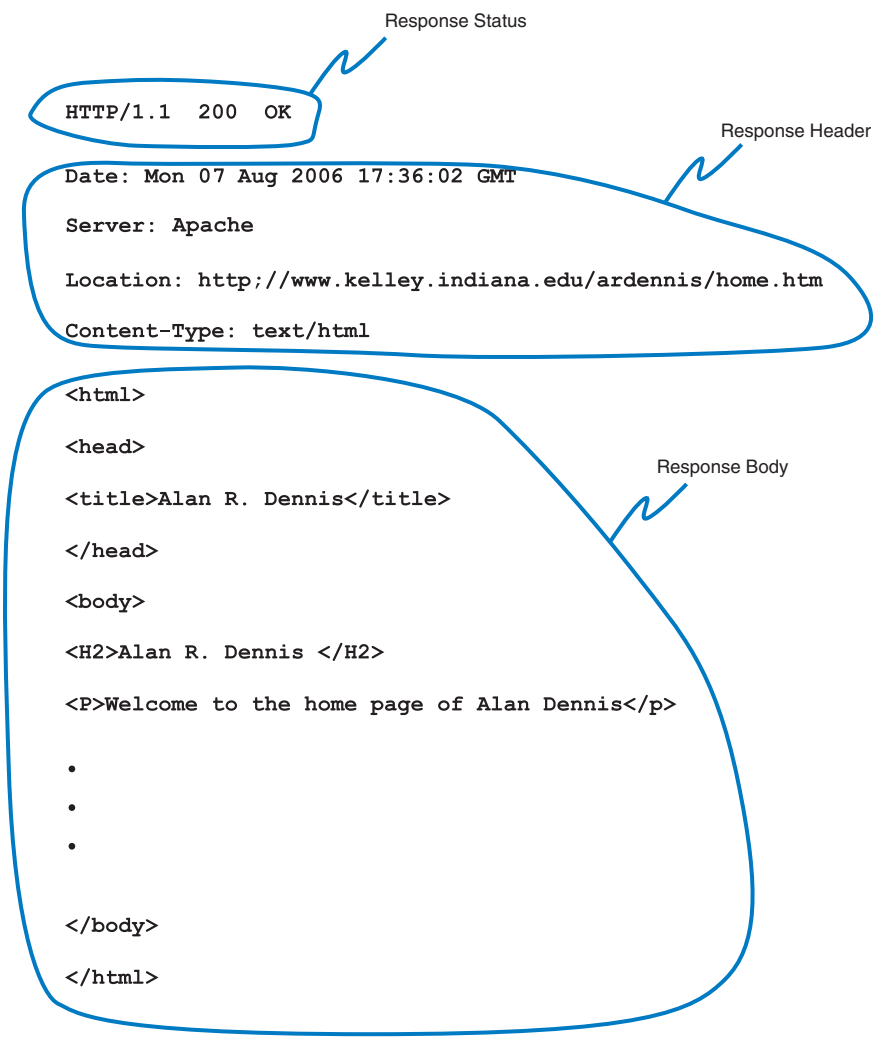


FIGURE 2.10 An example of a response from a Web server to a Web browser using the HTTP standard.

MANAGEMENT FOCUS **2-2 FREE SPEECH REIGNS ON THE INTERNET . . . OR DOES IT?**

In a landmark decision in 1997, the U.S. Supreme Court ruled that the sections of the 1996 Telecommunications Act restricting the publication of indecent material on the Web and the sending of indecent e-mail were unconstitutional. This means that anyone can do anything on the Internet, right?

Well, not really. The court decision affects only Internet servers located in the United States. Each country in the world has different laws that govern what may and may not be placed on servers in their country. For example, British law restricts the publication of pornography, whether on paper or on Internet servers.

Many countries such as Singapore, Saudi Arabia, and China prohibit the publication of certain political information. Because much of this "subversive" information is published outside of their countries, they actively restrict access to servers in other countries.

Other countries are very concerned about their individual cultures. In 1997, a French court convicted Georgia Institute of Technology of violating French language law. Georgia Tech operates a small campus in France that offers summer programs for American students. The information on the campus Web server was primarily in English because classes are conducted in English. This violated the law requiring French to be the predominant language on all Internet servers in France.

The most likely source of problems for North Americans lies in copyright law. Free speech does not give permission to copy from others. It is against the law to copy and republish on the Web any copyrighted material or any material produced by someone else without explicit permission. So don't copy graphics from someone else's Web site or post your favorite cartoon on your Web site, unless you want to face a lawsuit.

evolved rapidly ever since. HTML is covered by standards produced by the IETF, but Microsoft keeps making new additions to the HTML standard with every release of its browser, so the HTML standard keeps changing.

ELECTRONIC MAIL

Electronic mail (or *e-mail*) was one of the earliest applications on the Internet and is still among the most heavily used today. With e-mail, users create and send messages to one user, several users, or all users on a *distribution list.* Most e-mail software enables users to send text messages and attach files from word processors, spreadsheets, graphics programs, and so on. Many e-mail packages also permit you to filter or organize messages by priority.

Several standards have been developed to ensure compatibility between different e-mail software packages. Any software package that conforms to a certain standard can send messages that are formatted using its rules. Any other package that understands that particular standard can then relay the message to its correct destination; however, if an e-mail package receives a mail message in a different format, it may be unable to process it correctly. Many e-mail packages send using one standard but can understand messages sent in several different standards. The most commonly used standard is SMTP (Simple Mail Transfer Protocol). Other common standards are *X.400* and *CMC (Common Messag-*

ing Calls). In this book, we will discuss only SMTP, but CMC and X.400 both work essentially the same way. SMTP, X.400, and CMC are different from one another (in the same way that English differs from French or Spanish), but several software packages are available that translate between them, so that companies that use one standard (e.g., CMC) can translate messages they receive that use a different standard (e.g., SMTP) into their usual standard as they first enter the company and then treat them as "normal" e-mail messages after that.

How E-Mail Works

The *Simple Mail Transfer Protocol (SMTP)* is the most commonly used e-mail standard simply because it is the e-mail standard used on the Internet.[2] E-mail works similarly to how the Web works, but it is a bit more complex. SMTP e-mail is usually implemented as a two-tier client-server application, but not always. We first explain how the normal two-tier architecture works and then quickly contrast that with two alternate architectures.

Two-Tier E-Mail Architecture With a two-tier client-server architecture, each client computer runs an application layer software package called a *user agent,* which is usually more commonly called an e-mail client (Figure 2.11). There are many common e-mail client software packages such as Eudora and Outlook. The user creates the e-mail message using one of these e-mail clients, which formats the message into an SMTP packet that includes information such as the sender's address and the destination address.

The user agent then sends the SMTP packet to a mail server that runs a special application layer software package called a *message transfer agent,* which is more commonly called mail server software (see Figure 2.11).

This e-mail server reads the SMTP packet to find the destination address and then sends the packet on its way through the network—often over the Internet—from mail server to mail server, until it reaches the mail server specified in the destination address (see Figure 2.11). The mail transfer agent on the destination server then stores the message in the receiver's mailbox on that server. The message sits in the mailbox assigned to the user who is to receive the message until he or she checks for new mail.

The SMTP standard covers message transmission between mail servers (i.e., mail server to mail server) and between the originating e-mail client and its mail server. A different standard is used to communicate between the receiver's e-mail client and his or her mail server. Two commonly used standards for communication between e-mail client and mail server are *Post Office Protocol (POP)* and *Internet Message Access Protocol (IMAP).* Although there are several important technical differences between POP and IMAP, the most noticeable difference is that before a user can read a mail message with a POP (version 3) e-mail client, the e-mail message must be copied to the client computer's hard disk and deleted from the mail server. With IMAP, e-mail messages can remain stored on the mail server after they are read. IMAP therefore offers considerable benefits to users who

[2]The formal specification for SMTP is provided in RFC 822 on the IETF's Web site: www.ietf.org/rfc/rfc0821.txt

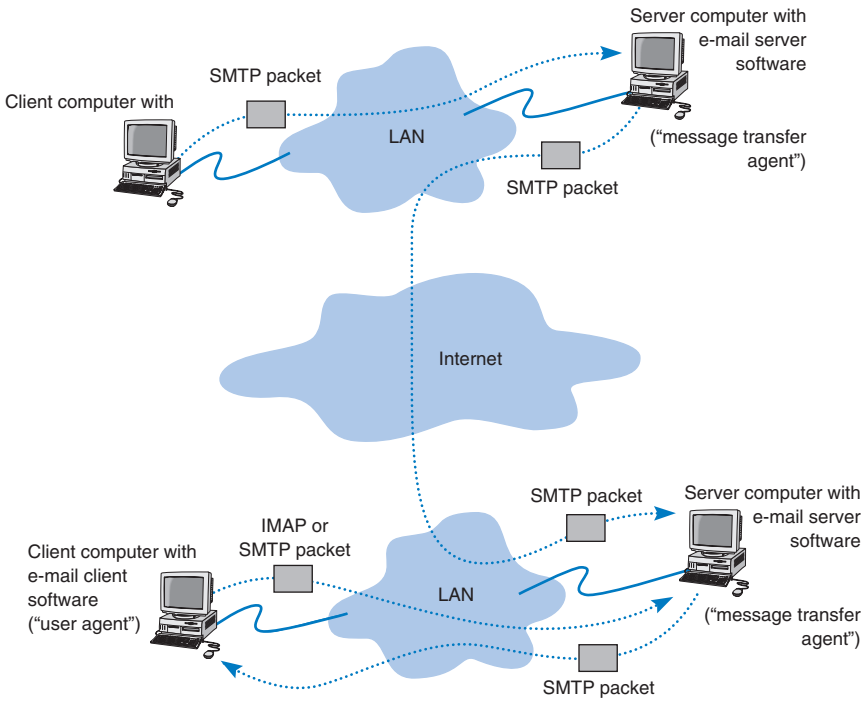


FIGURE 2.11 How SMTP (Simple Mail Transfer Protocol) e-mail works. IMAP = Internet Message Access Protocol; LAN = local area network.

read their e-mail from many different computers (e.g., home, office, computer labs) because they no longer need to worry about having old e-mail messages scattered across several client computers; all e-mail is stored on the server until it is deleted.

In our example in Figure 2.11, when the receiver next accesses his or her e-mail, the e-mail client on his or her computer contacts the mail server by sending an IMAP or POP packet that asks for the contents of the user's mailbox. In Figure 2.11, we show this as an IMAP packet, but it could just as easily be a POP packet. When the mail server receives the IMAP or POP request, it sends the original SMTP packet created by the message sender to the client computer, which the user reads with the e-mail client. Therefore, any e-mail client using POP or IMAP must also understand SMTP to create messages and to read messages it receives. Both POP and IMAP provide a host of functions that enable the user to manage his or her e-mail, such as creating mail folders, deleting mail, creating address books, and so on. If the user sends a POP or IMAP request for one of these functions, the mail server will perform the function and send back a POP or IMAP response packet that is much like an HTTP response packet.

Host-Based E-Mail Architectures When SMTP was first developed, host-based architectures were the rule, so SMTP was first designed to run on mainframe computers. If you use a text-based version of Linux or UNIX, chances are you are using a host-based architecture for your e-mail.

With this architecture, the client computer in Figure 2.11 would be replaced by a terminal that would send all of the user's keystrokes to the server for processing. The server would then send characters back to the terminal to display. All software would reside on the server. This software would take the user's keystrokes, create the SMTP packet, and then send it on its way to the next mail server.

Likewise, the receiver would use a terminal that would send keystrokes to the server and receive letters back to display. The server itself would be responsible for understanding the user's commands to read a mail message and sending the appropriate characters to the user's terminal so he or she could read the e-mail message. If you had been wondering why the SMTP standard does not include the delivery of the message to the receiver's client computer, you should now understand. Because no software existed on the receiver's terminal, the SMTP standard did not include any specification about how the receiver's mail server software should display messages. Communication between the mail server and the receiver's terminal was left to the e-mail software package running on the server. Because each package and each terminal was different, no standards were developed to cover communication between the terminal and the server.

Three-Tier Client-Server Architecture The three-tier client-server e-mail architecture uses a Web server and Web browser to provide access to your e-mail. With this architecture, you do not need an e-mail client on your client computer. Instead, you use your Web browser. This type of e-mail is sometimes called Web-based e-mail and is provided by a variety of companies such as Hotmail and Yahoo.

You use your browser to connect to a page on a Web server that lets you write the e-mail message by filling in a form. When you click the send button, your Web browser sends the form information to the Web server inside an HTTP request (Figure 2.12). The Web server runs a program (written in C or Perl, for example) that takes the information from the HTTP request and builds an SMTP packet that contains the e-mail message. Although not important to our example, it also sends an HTTP response back to the client. The Web server then sends the SMTP packet to the mail server, which processes the SMTP packet as though it came from a client computer. The SMTP packet flows through the network in the same manner as before. When it arrives at the destination mail server, it is placed in the receiver's mailbox.

When the receiver wants to check his or her mail, he or she uses a Web browser to send an HTTP request to a Web server (see Figure 2.12). A program on the Web server (in C or Perl, for example) processes the request and sends the appropriate IMAP (or POP) request to the mail server. The mail server responds with an IMAP (or POP) packet, which a program on the Web server converts into an HTTP response and sends to the client. The client then displays the e-mail message in the Web browser.

A simple comparison of Figures 2.11 and 2.12 will quickly show that the three-tier approach using a Web browser is much more complicated than the normal two-tier approach. So why do it? Well, it is simpler to have just a Web browser on the client computer rather than to require the user to install a special e-mail client on his or her computer and then set up the special e-mail client to connect to the correct mail server using either POP or IMAP. It is simpler for the user to just type the URL of the Web server providing the mail services into his or her browser and begin using mail.

It is also important to note that the sender and receiver do not have to use the same architecture for their e-mail. The sender could use a two-tier client-server architecture,

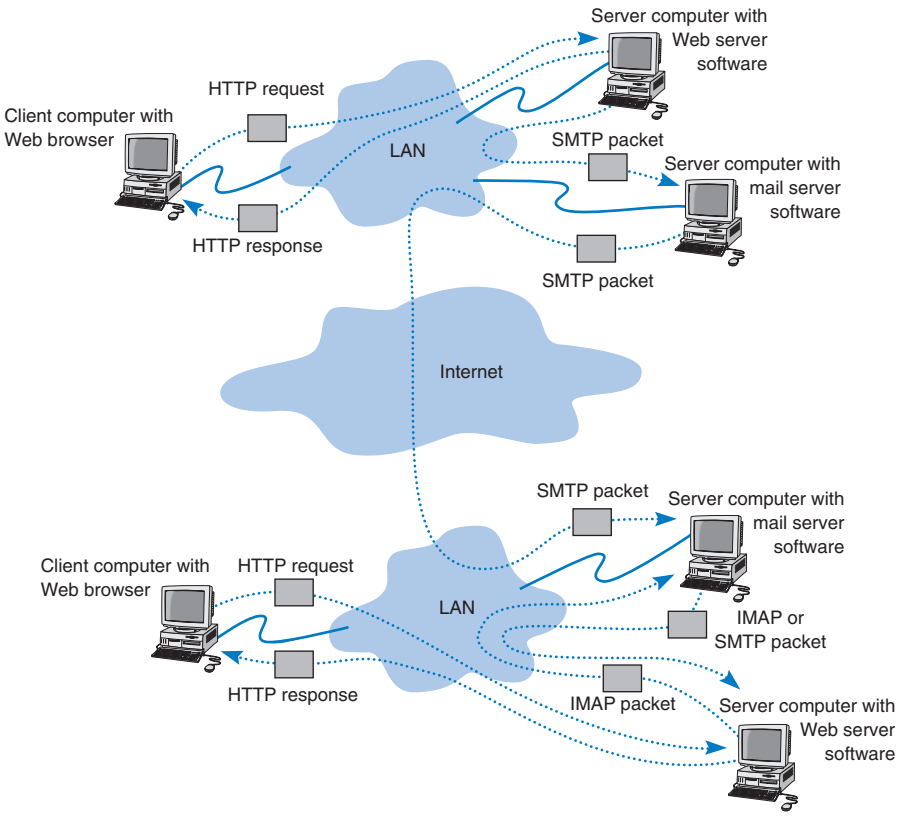


FIGURE 2.12 Inside the Web. HTTP = Hypertext Transfer Protocol; IMAP = Internet Message Access Protocol; LAN = local area network; SMTP = Simple Mail Transfer Protocol.

and the receiver, a host-based or three-tier client-server architecture. Because all communication is standardized using SMTP between the different mail servers, how the users interact with their mail servers is unimportant. Each organization can use a different approach.

In fact, there is nothing to prevent one organization from using all three architectures simultaneously. At Indiana University, we usually access our e-mail through an e-mail client (e.g., Eudora), but we also access it over the Web because many of us travel internationally and find it easier to borrow a Web browser with Internet access than to borrow an e-mail client and set it up to use the Indiana mail server.

Inside an SMTP Packet

SMTP defines how message transfer agents operate and how they format messages sent to other message transfer agents. An SMTP packet has two parts:

TECHNICAL FOCUS

2-2 SMTP TRANSMISSION

SMTP (Simple Mail Transfer Protocol) is an older protocol, and transmission using it is rather complicated. If we were going to design it again, we would likely find a simpler transmission method. Conceptually, we think of an SMTP packet as one packet. However, SMTP mail transfer agents transmit each element within the SMTP packet as a separate packet and wait for the receiver to respond with an "OK" before sending the next element.

For example, in Figure 2.13, the sending mail transfer agent would send the *from* address and wait for an OK from the receiver. Then it would send the *to* address and wait for an OK. Then it would send the date, and so on, with the last item being the entire message sent as one element.

- The *header,* which lists source and destination e-mail addresses (possibly in text form [e.g., "Pat Smith"]) as well as the address itself (e.g., psmith@somewhere .com), date, subject, and so on.
- The *body,* which is the word *DATA,* followed by the message itself.

Figure 2.13 shows a simple e-mail message formatted using SMTP. The header of an SMTP message has a series of fields that provide specific information, such as the sender's e-mail address, the receiver's address, date, and so on. The information in quotes on the *from* and *to* lines is ignored by SMTP; only the information in the angle brackets is used in e-mail addresses. The *message ID* field is used to provide a unique identification code so that the message can be tracked. The message body contains the actual text of the message itself.

Listserv Discussion Groups

A list server (or *Listserv*) group is simply a mailing list of users who have joined together to discuss some topic. Listserv groups are formed around just about every topic imaginable, in-

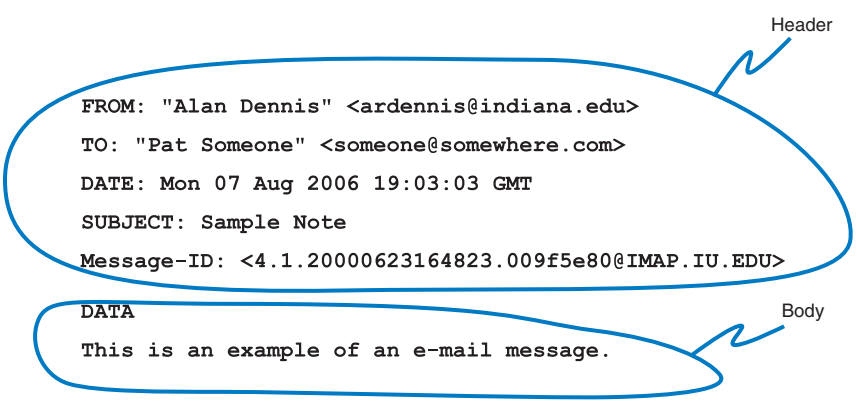


FIGURE 2.13 An example of an e-mail message using the SMTP (Simple Mail Transfer Protocol) standard.

cluding cooking, skydiving, politics, education, and British comedy. Some are short lived, whereas others continue indefinitely. Some permit any member to post messages; others permit only certain members to post messages. Most businesses have Listservs organized around job functions, so that it is easy to reach everyone in a particular department.

There are two parts to every Listserv. The first part, the *Listserv processor,* processes commands such as requests to subscribe, unsubscribe, or to provide more information about the Listserv. The second part is the *Listserv mailer.* Any message sent to the Listserv mailer is resent to everyone on the mailing list. To use a Listserv, you need to know the addresses of both the processor and the mailer.

To subscribe to a Listserv, you send an e-mail message to the Listserv processor, which adds your name to the list (see "Listserv Commands" for the message format). It is important that you send this message to the processor, not the mailer; otherwise, your subscription message will be sent to everyone on the mailing list, which might be embarrassing.

For example, suppose you want to join a Listserv on widgets that has a processor address of listerv @abc.com, and the mailer address is widget-1 @abc.com. To subscribe, you send an e-mail message to listerv @abc.com containing the text: *subscribe widget-1 your name.* To send a message to everyone on this Listserv, you would e-mail your message to widget-1 @abc.com.

Attachments in Multipurpose Internet Mail Extension

As the name suggests, SMTP is a simple standard that permits only the transfer of text messages. It was developed in the early days of computing, when no one had even thought about using e-mail to transfer nontext files such as graphics or word processing documents. Several standards for nontext files have been developed that can operate together with SMTP, such as *Multipurpose Internet Mail Extension (MIME),* uuencode, and binhex.

TECHNICAL FOCUS

2-3 LISTSERV COMMANDS

There are many different commands that can be sent to the Listserv processor to perform a variety of functions. These commands are included as lines of text in the e-mail message sent to the processor. Each command must be placed on a separate line. Some useful commands include

- SUBSCRIBE listserv-mailer-name your-name: Subscribes you to a mailing list (e.g., *subscribe maps-1 robin jones*)
- UNSUBSCRIBE listserv-mailer-name your-name: Unsubscribes you from the mailing list (e.g., *unsubscribe maps-1 robin jones*)
- HELP: Requests the Listserv to e-mail you a list of its commands
- LIST: Requests the Listserv to e-mail you a list of all Listserv groups that are available on this Listserv processor
- LIST DETAILED: Requests the Listserv to e-mail you a detailed description of all Listserv groups that are available on this Listserv processor and are public

Each of the standards is different, but all work in the same general way. The MIME software, which exists as part of the e-mail client, takes the nontext file such as a PowerPoint graphic file, and translates each byte in the file into a special code that looks like regular text. This encoded section of "text" is then labeled with a series of special fields understood by SMTP as identifying a MIME-encoded attachment and specifying information about the attachment (e.g., name of file, type of file). When the receiver's e-mail client receives the SMTP message with the MIME attachment, it recognizes the MIME "text" and uses its MIME software (that is part of the e-mail client) to translate the file from MIME "text" back into its original format.

OTHER APPLICATIONS

There are literally thousands of applications that run on the Internet and on other networks. Most application software that we develop today, whether for sale or for private internal use, runs on a network. We could spend years talking about different network applications and still cover only a small number.

Fortunately, most network application software works in much the same way as the Web or e-mail. In this section, we will briefly discuss only three commonly used applications: File Transfer Protocol (FTP), Telnet, and instant messaging (IM).

File Transfer Protocol

File Transfer Protocol (FTP) enables you to send and receive files over the Internet. FTP works in a similar manner as HTTP. FTP requires an application layer program on the client computer and a FTP server application program on a server. There are many software packages that use the FTP standard, such as WS-FTP. The user uses his or her client to send FTP requests to the FTP server. The FTP server processes these requests and sends back FTP packets containing the requested file.[3]

Most FTP sites require users to have permission before they can connect and gain access to the files. Access is granted by providing an account name with a password. For example, a network manager or Web-master would write a Web page using software on his or her client computer and then use FTP to send it to a specific account on the Web server.

Many files and documents available via FTP have been compressed to reduce the amount of disk space they require. Because there are many types of data compression programs, it is possible that a file you want has been compressed by a program you lack, so you won't be able to access the file until you find the decompression program it uses. That's one of the "advantages" of the decentralized, no-rules structure of the Internet.

Telnet

Telnet enables users to log in to servers (or other clients). Telnet requires an application layer program on the client computer and an application layer program on the server or

[3]The formal specification for FTP is provided in RFC 2640 on the IETF's Web site: www.ietf.org/rfc/rfc2640.txt

A DAY IN THE LIFE: NETWORK MANAGER

It was a typical day for a network manager. It began with the setup and troubleshooting for a videoconference. Videoconferencing is fairly routine activity but this one was a little different; we were trying to videoconference with a different company who used different standards than we did. We attempted to use our usual web-based videoconferencing but could not connect. We fell back to ISDN-based videoconferencing over telephone lines, which required bringing in our videoconferencing services group. It took two hours but we finally had the technology working.

The next activity was building a Windows database server. This involved installing software, adding a server into our ADS domain, and setting up the user accounts. Once the server was on the network, it was critical to install all the security patches for both the operating system and database server. We receive so many security attacks that it is our policy to install all security patches on the same day that new software or servers are placed on the network or the patches are released.

After lunch, the next two hours was spent in a boring policy meeting. These meetings are a necessary evil to ensure that the network is well-managed. It is critical that users understand what the network can and can't be used for, and our ability to respond to users' demands. Managing users' expectations about support and use rules helps ensure high user satisfaction.

The rest of the day was spent refining the tool we use to track network utilization. We have a simple intrusion detection system to detect hackers, but we wanted to provide more detailed information on network errors and network utilization to better assist us in network planning.

With thanks to Jared Beard

host computer. There are many programs that conform to the Telnet standard, such as EWAN. Once Telnet makes the connection from the client to the server, you must use the account name and password of an authorized user to login.

Because Telnet was designed in the very early days of the Internet, it assumes your client is a dumb terminal. Therefore, when you use Telnet, you are using a host-based architecture. All keystrokes you type in the Telnet client are transferred one by one to the server for processing. The server processes those commands—including simple keystrokes such as up arrow or down arrow—and transfers the results back to the client computer, which displays the letters and moves the cursor as directed by the server.[4]

Telnet can be useful because it enables you to access your server or host computer without sitting at its keyboard. Most network managers use Telnet to work on servers, rather than physically sitting in front of them and using their keyboards. Telnet also poses a great security threat, because it means that anyone on the Internet can attempt to log in to your account and use it as he or she wishes. Two commonly used security precautions are to prohibit remote logins via Telnet unless a user specifically asks for his or her account to be authorized for it and to permit remote logins only from a specific set of Internet addresses. For example, the Web server for this book will accept Telnet logins only from computers located in the same building. Chapter 11 discusses network security.

[4]The formal specification for Telnet is provided in RFC 854 and RFC 2355 on the IETF's Web site. The URLs are www.ietf.org/rfc/rfc0854.txt and www.ietf.org/rfc/rfc2355.txt, respectively.

MANAGEMENT FOCUS **2-3 TAGGING PEOPLE**

Joseph Krull has a chip on his shoulder—well, in his shoulder to be specific. Krull is one of a small but growing number of people who have a Radio Frequency Identification (RFID) chip implanted in their bodies.

RFID technology has been used to identify pets, so that lost pets can be easily reunited with their owners. Now, the technology is being used for humans.

Krull has a blown left pupil from a skiing accident. If he were injured in an accident and unable to communicate, an emergency room doctor might misinterpret his blown pupil as a sign of a major head injury and begin drilling holes to relieve pressure. Now doctors can use the RFID chip to identify Krull and quickly locate his complete medical records on the Internet.

Critics say such RFID chips pose huge privacy risks because they enable any firms using RFID to track users such as Krull. Retailers, for example, can track when he enters and leaves their stores.

Krull doesn't care. He believes the advantages of having his complete medical records available to any doctor greatly outweighs the privacy concerns.

SOURCE: "RFID is really getting under people's skin," *NetworkWorld*, April 4, 2005, p. 1.

Instant Messaging

One of the fastest growing Internet applications is *instant messaging (IM)*. With IM, you can exchange real-time typed messages or chat with your friends. Some IM software also enables you to verbally talk with your friends in the same way as you might use the telephone or to use cameras to exchange real-time video in the same way you might use a videoconferencing system. Several types of IM currently exist, including ICQ and AOL Instant Messenger.

IM works in much the same way as the Web. The client computer needs an IM client software package, which communicates with an IM server software package that runs on a server. When the user connects to the Internet, the IM client software package sends an IM request packet to the IM server informing it that the user is now online. The IM client software package continues to communicate with the IM server to monitor what other users have connected to the IM server. When one of your friends connects to the IM server, the IM server sends an IM packet to your client computer so that you now know that your friend is connected to the Internet. The server also sends a packet to your friend's client computer so that he or she knows that you are on the Internet.

With the click of a button, you can both begin chatting. When you type text, your IM client creates an IM packet that is sent to the IM server (Figure 2.14). The server then retransmits the packet to your friend. Several people may be part of the same chat session, in which case the server sends a copy of the packet to all of the client computers. IM also provides a way for different servers to communicate with one another, and for the client computers to communicate directly with each other.

Videoconferencing

Videoconferencing provides real-time transmission of video and audio signals to enable people in two or more locations to have a meeting. In some cases, videoconferences are held in

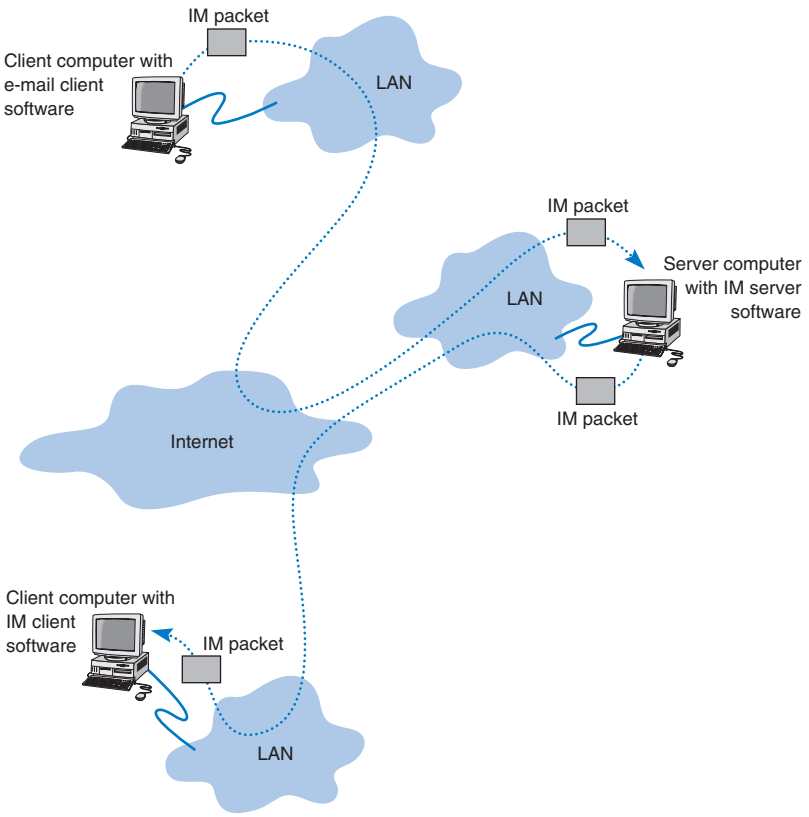


FIGURE 2.14 How instant messaging (IM) works. LAN = local area network.

special-purpose meeting rooms with one or more cameras and several video display monitors to capture and display the video signals (Figure 2.15). Special audio microphones and speakers are used to capture and play audio signals. The audio and video signals are combined into one signal that is transmitted though a MAN or WAN to people at the other location. Most of this type of videoconferencing involves two teams in two separate meeting rooms, but some systems can support conferences of up to eight separate meeting rooms.

The fastest growing form of videoconferencing is *desktop videoconferencing*. Small cameras installed on top of each computer permit meetings to take place from individual offices (Figure 2.16). Special application software (e.g., Yahoo IM, Net Meeting) is installed on the client computer and transmits the images across a network to application software on a videoconferencing server. The server then sends the signals to the other client computers that want to participate in the videoconference. In some cases, the clients can communicate with one another without using the server. The cost of desktop videoconferencing ranges from less than $20 per computer for inexpensive systems to more than $1,000 for high-quality systems. Some systems have integrated conferencing software with desktop videoconferencing, enabling participants to communicate verbally and,

PhotoDisc, Inc./Getty Images

FIGURE 2.15 Room-based videoconferencing.

by using applications such as white boards, to attend the same meeting while they are sitting at the computers in their offices.

The transmission of video requires a lot of network capacity. Most videoconferencing uses data compression to reduce the amount of data transmitted. Surprisingly, the most common complaint is not the quality of the video image but the quality of the voice transmissions. Special care needs to be taken in the design and placement of microphones and speakers to ensure quality sound and minimal feedback.

Most videoconferencing systems were originally developed by vendors using different formats, so many products were incompatible. The best solution was to ensure that all hardware and software used within an organization was supplied by the same vendor and to hope that any other organizations with whom you wanted to communicate used the same equipment. Today, three standards are in common use: H.320, H.323, and MPEG-2 (also called ISO 13818-2). Each of these standards was developed by different organizations and is supported by different products. They are not compatible, although some application software packages understand more than one standard. H.320 is designed for room-to-room videoconferencing over high-speed telephone lines. H.323 is a family of standards designed for desktop videoconferencing and just simple audio conferencing over the Internet. MPEG-2 is designed for faster connections, such as a LAN or specially designed, privately operated WAN.

Tom Gulfer/iStockphoto

FIGURE 2.16 Desktop videoconferencing.

Webcasting is a special type of one-directional videoconferencing in which content is sent from the server to the user. The developer creates content that is downloaded as needed by the users and played by a plug-in to a Web browser. At present, there are no standards for Webcast technologies, but the products by RealNetworks.com are the de facto standards.

IMPLICATIONS FOR MANAGEMENT

The first implication for management from this chapter is that the primary purpose of a network is to provide a worry-free environment in which applications can run. The network itself does not change the way an organization operates; it is the applications that the network enables that have the potential to change organizations. If the network does not easily enable a wide variety of applications, this can severely limit the ability of the organization to compete in its environment.

The second implication is that over the past few years there has been a dramatic increase in the number and type of applications that run across networks. In the early 1990s, networks primarily delivered e-mail and organization-specific application traffic (e.g. accounting transactions, database inquiries, inventory data). Today's traffic contains large amounts of e-mail, Web packets, videoconferencing, telephone calls, instant messaging, music, and organization-specific application traffic. Traffic has been growing much more rapidly than expected and each type of traffic has

MANAGEMENT FOCUS

2-4 VIDEOCONFERENCING AT THE ALABAMA DEPARTMENT OF REHABILITATION SERVICES

The mission of the Alabama Department of Rehabilitation Services (ADRS) is to assist the state's children and adults with disabilities in realizing their full potential. ADRS offers a comprehensive array of medical, educational, psychological, vocational, technological, and independent living services for people of all ages. More than 800 agency employees serve approximately 80,000 Alabamians each year through two dozen community-based field offices. Travel expenses related to these activities were considerable, and the state's existing telephone-based video system was too expensive to be a practical alternative. ADRS needed a cost-effective way to enhance collaboration and education for its staff.

ADRS upgraded its existing WAN to accommodate video traffic and then deployed an Internet-based videoconferencing system at nearly all of its sites around the state based on industry-standard H.323 video technology.

"Videoconferencing is now becoming part of the norm," says Denise Murray, coordinator of staff development and training. "For example, we have a 90-minute weekly meeting with our case management programming team. One of our team members is from Birmingham, which is about 100 miles away. Now, instead of driving 90 minutes each way to attend our meeting every Monday morning, he attends via videoconference, and I know he loves it."

"We really did this on a shoestring budget," says Buck Jordan, director of field services. "The state's [previous] ISDN-based video system costs approximately US $80,000 per site, whereas we're spending approximately $58,000 total during this first year and already have more than 10 of our sites running. With the Cisco technology, we can run data and video across the same circuit, so we are saving a lot of money."

ADRS is continuing to deploy its new video capability to all of its desktops statewide.

SOURCE: "IP Videoconferencing Solution Helps Alabama Department of Rehabilitation Services Improve Staff Training and Client Care," www.cisco.com, 2004.

different implications for the best network design, making the job of the network manager much more complicated. Most organizations have seen their network operating costs grow significantly even though the cost per packet (i.e., the cost divided by the amount of traffic) has dropped significantly over the last 10 years.

SUMMARY

Application Architectures There are three fundamental application architectures. In host-based networks, the server performs virtually all of the work. In client-based networks, the client computer does most of the work; the server is used only for data storage. In client-server networks, the work is shared between the servers and clients. The client performs all presentation logic, the server handles all data storage and data access logic, and one or both perform the application logic. Client-server networks can be cheaper to install and often better balance the network loads but are far more complex and costly to develop and manage.

World Wide Web One of the fastest growing Internet applications is the Web, which was first developed in 1990. The Web enables the display of rich graphical images, pictures, full-motion video, and sound. The Web is the most common way for businesses to establish a presence on the Internet.

The Web has two application software packages, a Web browser on the client and a Web server on the server. Web browsers and servers communicate with one another using a standard called HTTP. Most Web pages are written in HTML, but many also use other formats. The Web contains information on just about every topic under the sun, but finding it and making sure the information is reliable are major problems.

Electronic Mail With e-mail, users create and send messages using an application-layer software package on client computers called user agents. The user agent sends the mail to a server running an application-layer software package called a mail transfer agent, which then forwards the message through a series of mail transfer agents to the mail transfer agent on the receiver's server. E-mail is faster and cheaper than regular mail and can substitute for telephone conversations in some cases. Several standards have been developed to ensure compatibility between different user agents and mail transfer agents. SMTP, POP, and IMAP are used on the Internet. X.400 and CMC are other commonly used standards.

KEY TERMS

anonymous FTP
application architecture
application logic
client-server architecture
cluster
Common Messaging Calls (CMC)
data access logic
data storage
desktop videoconferencing
distributed computing
distribution list
domain
dumb terminal
e-mail
File Transfer Protocol (FTP)
H.320
H.323
host-based architecture
HTTP request
HTTP response
Hypertext Markup Language (HTML)
Hypertext Transfer Protocol (HTTP)
instant messaging (IM)
intelligent terminal
Internet
Internet Mail Access Protocol (IMAP)
Listserv
mainframe
message transfer agent
microcomputer
minicomputer
MPEG
MPEG-2
Multipurpose Internet Mail Extension (MIME)
Netscape
network computer
NSFNET
n-tier architecture
Post Office Protocol (POP)
presentation logic
protocol
request body
request header
request line
response body
response header
response status
server-based architecture
Simple Mail Transfer Protocol (SMTP)
Telnet
terminal
thick client
thin client
three-tier architecture
transaction terminal
two-tier architecture
uniform resource locator (URL)
user agent
videoconferencing
World Wide Web
Web browser
Web server
workstation
X.400

QUESTIONS

1. What are the different types of application architectures?
2. Describe the four basic functions of an application software package.
3. What are the advantages and disadvantages of host-based networks versus client-server networks?
4. What is middleware, and what does it do?
5. Suppose your organization was contemplating switching from a host-based architecture to client-server. What problems would you foresee?
6. Which is less expensive: host-based networks or client-server networks? Explain.

7. Compare and contrast two-tier, three-tier, and *n*-tier client-server architectures. What are the technical differences, and what advantages and disadvantages does each offer?
8. How does a thin client differ from a fat client?
9. What is a network computer?
10. What do the following tools enable you to do: the Web, e-mail, FTP, Telnet?
11. For what is HTTP used? What are its major parts?
12. For what is HTML used?
13. Describe how a Web browser and Web server work together to send a Web page to a user.
14. Can a mail sender use a 2-tier architecture to send mail to a receiver using a 3-tier architecture? Explain.
15. Describe how mail user agents and message transfer agents work together to transfer mail messages.
16. What roles do SMTP, POP, and IMAP play in sending and receiving e-mail on the Internet?
17. What are the major parts of an e-mail message?
18. What are X.400 and CMC?
19. What is FTP, and why is it useful?
20. What is Telnet, and why is it useful?
21. What is a Listserv and how could you use it to get information?
22. Explain how instant messaging works.
23. Compare and contrast the application architecture for videoconferencing and the architecture for e-mail.
24. Which of the three application architectures for e-mail (two-tier client server, Web-based, and host-based) is "best"? Explain.
25. Some experts argue that thin-client client-server architectures are really host-based architectures in disguise and suffer from the same old problems. Do you agree? Explain.
26. You can use a Web browser to access an FTP server simply by putting *ftp://* in front of the URL (e.g., *ftp://xyz.abc.com*). If that server has FTP server software installed, then the FTP server will respond instead of the Web server. What is your browser doing differently to access the FTP server? Hint: This question is more difficult than it seems, because we haven't explained how the server knows to pass certain types of packets to the right software (i.e., HTTP requests to the Web server software and SMTP packets to the e-mail software). At this point, don't worry about it. Linking the network to the application layer is the job of the transport layer, which is explained in Chapter 5.
27. Will the Internet become an essential business tool like the telephone or will it go the way of the dinosaurs? Discuss.

EXERCISES

2–1. Investigate the use of the three major architectures by a local organization (e.g., your university). Which architecture(s) does it use most often and what does it see itself doing in the future? Why?

2–2. What are the costs of client-server versus host-based architectures? Search the Web for at least two different studies and be sure to report your sources. What are the likely reasons for the differences between the two?

2–3. Investigate the costs of dumb terminals, intelligent terminals, network computers, minimally equipped microcomputers, and top-of-the-line microcomputers. Many equipment manufacturers and resellers are on the Web, so it's a good place to start looking.

2–4. What application architecture does your university use for e-mail? Explain.

MINI-CASES

I. Deals-R-Us Brokers (Part 1)

Fred Jones, a distant relative of yours and president of Deals-R-Us Brokers (DRUB), has come to you for advice. DRUB is a small brokerage house that enables its clients to buy and sell stocks over the Internet, as well as place traditional orders by phone or fax. DRUB has just decided to offer a set of stock analysis tools that will help its clients more easily pick winning stocks, or so Fred tells you. Fred's information systems department has presented him with two alternatives for developing the new tools. The first alternative will have a special tool developed in C++ that clients will download onto their computers to run. The tool will communicate with the DRUB server to select data to analyze. The second alternative will have the C++ program running on the server, the client will use his or her browser to interact with the server.

a. Classify the two alternatives in terms of what type of application architecture they use.
b. Outline the pros and cons of the two alternatives and make a recommendation to Fred about which is better.

II. Deals-R-Us Brokers (Part 2)

Fred Jones, a distant relative of yours and president of Deals-R-Us Brokers (DRUB), has come to you for advice. DRUB is a small brokerage house that enables its clients to buy and sell stocks over the Internet, as well as place traditional orders by phone or fax. DRUB has just decided to install a new e-mail package. One vendor is offering an SMTP-based two-tier client-server architecture. The second vendor is offering a Web-based e-mail architecture. Fred doesn't understand either one but thinks the Web-based one should be better because, in his words, "the Web is the future."

a. *Briefly* explain to Fred, in layperson's terms, the differences between the two.
b. Outline the pros and cons of the two alternatives and make a recommendation to Fred about which is better.

III. Accurate Accounting

Diego Lopez is the managing partner of Accurate Accounting, a small accounting firm that operates a dozen offices in California. Accurate Accounting provides audit and consulting services to a growing number of small- and medium-sized firms, many of which are high technology firms. Accurate Accounting staff typically spend many days on-site with clients during their consulting and audit projects, but has increasingly been using e-mail and Instant Messenger (IM) to work with clients. Now, many firms are pushing Accurate Accounting to adopt videoconferencing. Diego is concerned about what videoconferencing software and hardware to install. While Accurate Accounting's e-mail system enables it to exchange e-mail with any client, using IM has proved difficult because Accurate Accounting has had to use one IM software package with some companies and different IM software with others. Diego is concerned that videoconferencing may prove to be as difficult to manage as IM. "Why can't IM work as simply as e-mail?" he asks. "Will my new videoconferencing software and hardware work as simply as e-mail, or will it be IM all over again?" Prepare a response to his questions.

IV. Ling Galleries

Howard Ling is a famous artist with two galleries in Hawaii. Many of his paintings and prints are sold to tourists who visit Hawaii from Hong Kong and Japan. He paints 6–10 new paintings a year, which sell for $50,000 each. The real money comes from the sales of prints; a popular painting will sell 1,000 prints at a retail price of $1,500 each. Some prints sell very quickly, while others do not. As an artist, Howard paints what he wants to paint. As a businessman, Howard also wants to create art that sells well. Howard visits each gallery once a month to talk with clients, but enjoys talking with the gallery staff on a weekly basis to learn what visitors say about his work and to get ideas for future work. Howard has decided to open two new galleries, one in Hong Kong and one in Tokyo. How can the Internet help Howard with the two new galleries?

CASE STUDY

NEXT-DAY AIR SERVICE

See the Web site

HANDS-ON ACTIVITY

Looking Inside Your HTTP Packets

Figures 2.9 and 2.10 show you inside one HTTP request and one HTTP response that we captured. The objective of this Activity is for you to see inside HTTP packets that you create.

1. Use your browser to connect to www.rexswain.com/httpview.html. You will see the screen in Figure 2.17.
2. In box labeled URL, type any URL you like and click Submit. You will then see something like the screen in Figure 2.18. In the middle of the screen, under the label "Sending Request:" you will see the exact HTTP packet that your browser generated.
3. If you scroll this screen down, you'll see the exact HTTP response packet that the server sent back to

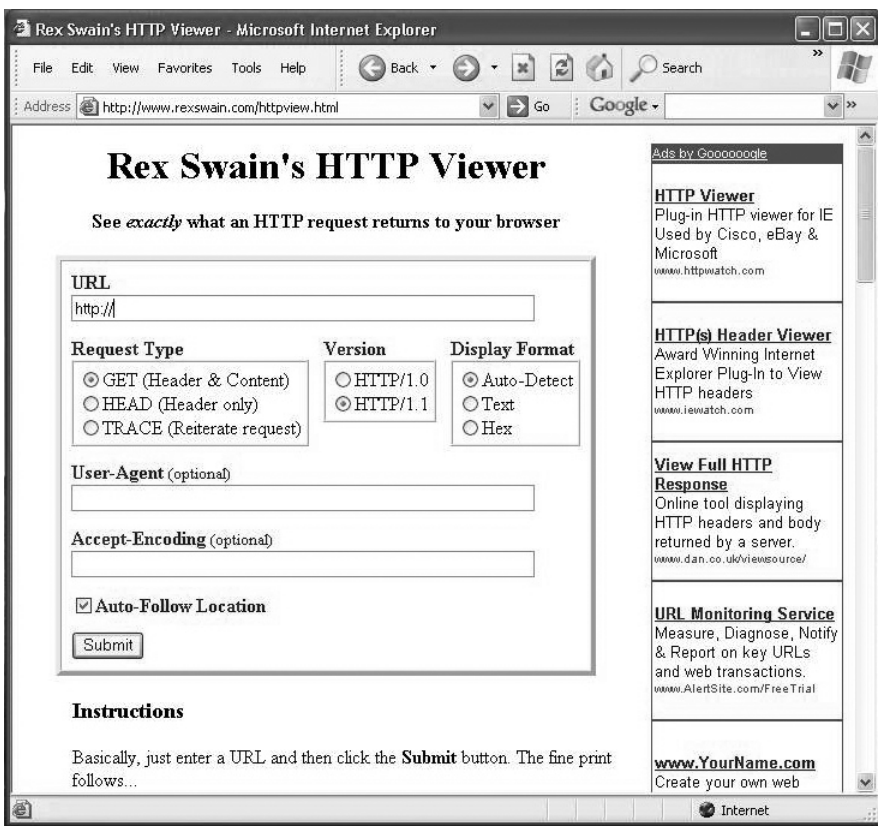


FIGURE 2.17 The HTTP Viewer.

you. In Figure 2.19, you'll see the response from the Indiana University Web server. You'll notice that at the time we did this, Indiana University was using the Apache Web server.

4. Try this on several sites around the Web to see what Web server they use. For example, Microsoft uses the Microsoft IIS Web server, while Cisco uses Apache. Some companies set their Web servers not to release this information.

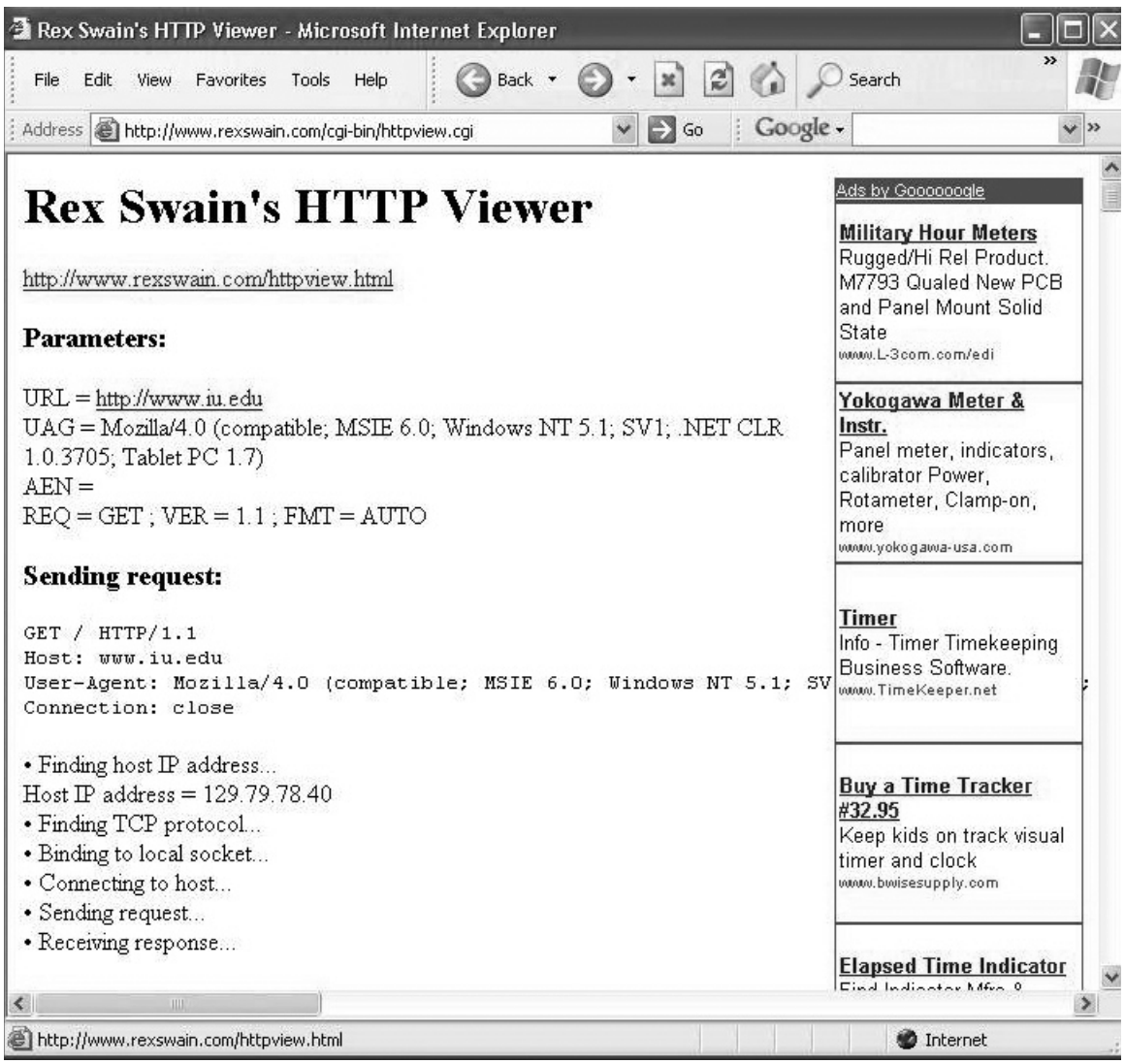


FIGURE 2.18 Looking inside an HTTP request.

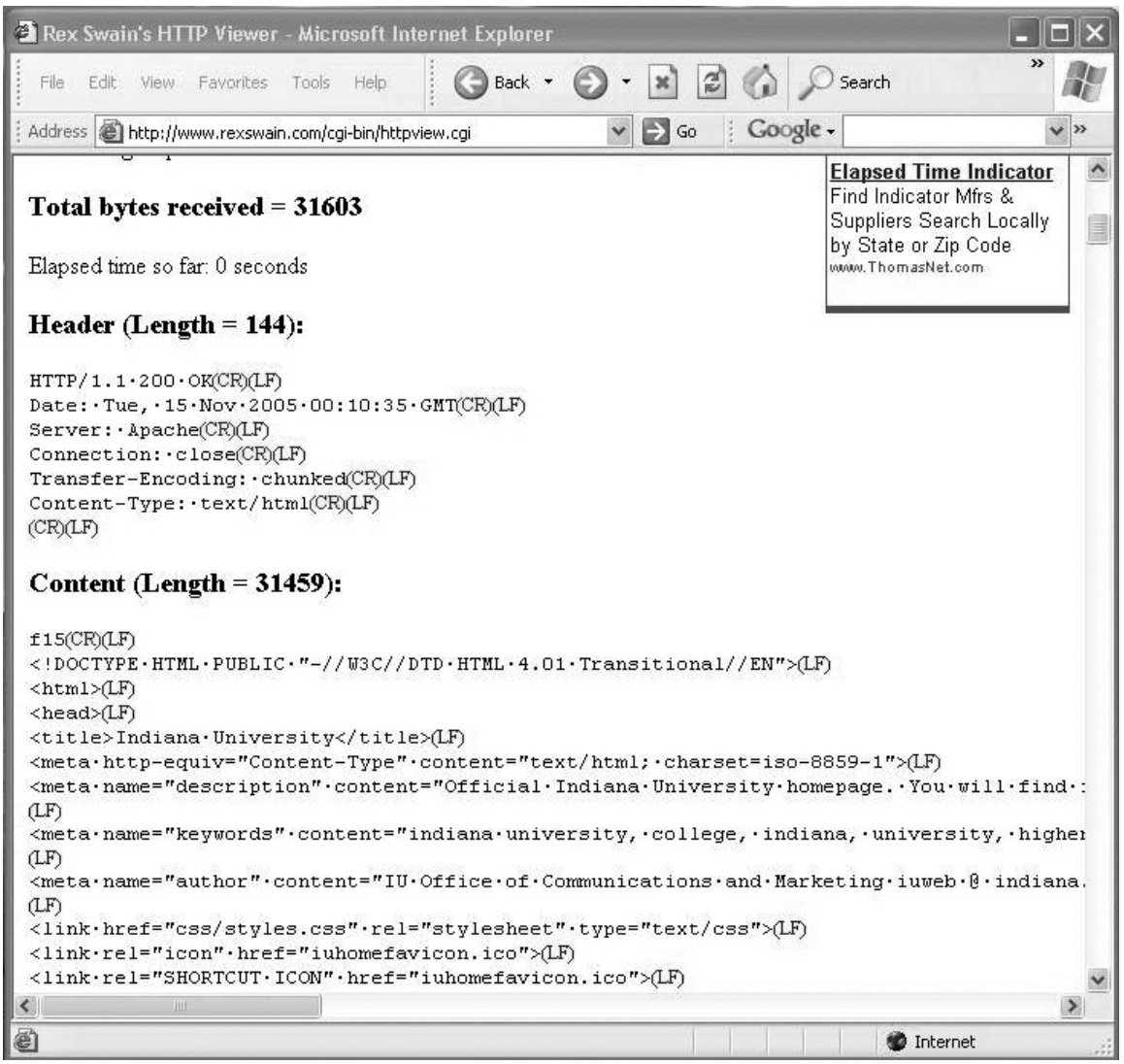


FIGURE 2.19 Looking inside an HTTP response.