IRIS® Token Ring Administrator's Guide

Document Number 007-1561-040

CONTRIBUTORS

Written by Kim Simmons, Carlin Otto, and Susan Thomas
Production by Kirsten Johnson
Engineering contributions by Jay Lan, Jong Kim, and Yu-Lin Lu

IRIS Token Ring Administrator's Guide
Document Number 007-1561-040
Silicon Graphics, Inc.
Mountain View, California

IRIS® Token Ring Administrator's Guide
Document Number 007-1561-040

Contents

Figures

Tables

Introduction

The *Token Ring Administrator's Guide* describes the Silicon Graphics® device drivers for the 802.5 Token Ring communications protocol. The guide includes basic Token Ring concepts as well as configuration issues for the IRIS® Token Ring product. The IRIS Token Ring product provides network access for Silicon Graphics applications, including TCP/IP-based network applications, IBM® emulation and connectivity products, and NetVisualyzer™.

IRIS Token Ring can be a station's only network connection, or it can be one of many network connections operating simultaneously (for example, with Ethernet and/or FDDI).

Using This Guide

This guide provides the information needed to configure, verify, and maintain your token ring connections. This guide also contains basic troubleshooting tips.

The information in this guide will help you understand how the various network applications (TCP/IP-based and IBM emulation and connectivity products) can fit into the token ring environment.

Audience

To use the information in this guide, you should have experience in the following areas:

- understanding the UNIX® file system structure
- using UNIX-based editors (for example, *jot* or *vi*)

Style Conventions

This guide uses the following stylistic conventions:

`screen display`
: Indicates system output, such as responses to commands that you see on the screen. Code samples, screen displays, and file contents also appear in this font.

`user input`
: Indicates what you must enter at a command line, such as commands, options, and arguments to commands.

variable
: Indicates generic, place-holding names (for example, *filename*) that are replaced in examples with specific names (for example, MyDoc).

`<xx>`
: Indicates keys on the keyboard that you press; for example, press **`<Enter>`** means press only the key labeled Enter.

command
: Designates command and utility names.

file name
: Indicates file names and file name suffixes.

glossary item
: Designates a term that is defined in the Glossary.

. . .
: Denotes omitted material or indicates that the preceding optional items may appear more than once in succession.

Product Support

Silicon Graphics, Inc., provides a comprehensive product support and maintenance program for its products. If you are in North America and would like support for your Silicon Graphics-supported products, contact the Technical Assistance Center at 1-800-800-4SGI. If you are outside North America, contact the Silicon Graphics subsidiary or authorized distributor in your country.

Chapter 1

Understanding IRIS Token Ring

This chapter describes the Token Ring environment. It provides a brief overview of the token ring architecture, including a summary of the 802.5 Token Ring Standard (protocol), and a discussion of the software portion of Silicon Graphics' Token Ring implementation.

Overview of Token Ring

The 802.5 Token Ring Standard is a local area network communications protocol designed around the basic token ring architecture[1]. In this chapter, one section describes the basic token ring architecture, and another section describes the specifics of the 802.5 Token Ring Standard.

Note: The term "token ring" is frequently used to refer to the 802.5 Token Ring Standard and is not used for other protocols based on the token ring architecture. Within this document, the basic token ring architecture is referred to in lowercase letters (token ring) while the 802.5 Token Ring is represented in uppercase (Token Ring).

Basic Token Ring Architecture

This section describes the basic terms and concepts for communication protocols based on the token ring architecture.

The Ring and the Token

The basic token ring architecture is designed around the two elements in its name: ring and token. The *ring* is a collection of hardware devices called *repeater*s (or *attaching device*s) that are connected by cable in a manner that causes the signal within the cable to

[1] The basic token ring architecture is used in a number of protocols, for example, FDDI Token Ring.

travel in a circle or closed loop (ring), as illustrated in Figure 1-1. Each repeater has two signal paths (usually contained within a single cable connection): an input and an output. Repeaters read information from the cable on their input *port* and repeat what they read onto the cable on the output port. Each computer connects to the ring through a repeater. The computer's network adapter / controller board contains the repeater. When a computer is attached to the ring, it is referred to as a *node* or *station*.

Communication occurs when one station transmits a signal onto the cable through its output line, the signal passes along the cable to the next *downstream* station, and that station reads the signal through its input line. The signal travels in only one direction on the ring. Each signal goes all the way around the ring, returning to the originating station.

The *token* is a particular pattern (a special *frame*) that circulates the ring. It is an electrical or optical signal pattern when on the cable and is represented by a ones-and-zeros pattern. Each protocol based on the token ring architecture specifies a specific pattern for its token. The token is initially placed (transmitted) onto the ring by one specially designated station, and, as it moves around the ring, the stations read and retransmit it (one-by-one). This scheme ensures that only one token will exist on the ring at any point in time, as illustrated in Figure 1-1.

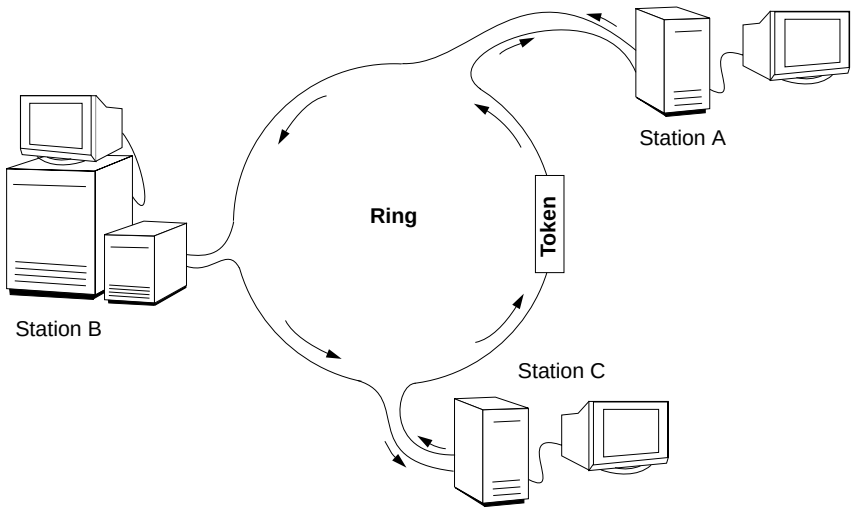


Figure 1-1 Token Ring—Listening Stations and an Idle Ring

Transmission

The token controls transmission onto the ring. If numerous stations were to transmit simultaneously, the signals would affect each other (meaning the data would be corrupted), and no communication would be possible. So, only one station, the station that is currently holding the token, is allowed to transmit its data. When a station has data to transmit (either user-application data in *data frame*s or management data in *MAC frames*), it must *capture* or *claim* the token first. To capture the token, a station does not *repeat* the token pattern when it reads it from the cable; instead, the station puts its first data frame or MAC frame onto the cable. While the transmitting station holds the token, other stations can only read from the ring and retransmit what they read. When the transmitting station has no more data to send (or when its *token holding timer* expires), the station places the token back onto the ring, giving the next downstream station an opportunity to capture the token and transmit data. If the next station does not have any data to transmit, it passes the token along.

States

At any given time, a station on the ring can be in one of two basic states: sending or listening. In the sending state, the station has captured the token and is transmitting its own data or *fill* (or idle) patterns. (Fill is explained later in this section.) In the listening state, the station is reading information (the signal) from the ring and transmitting it back onto the ring with minor changes. This activity is referred to as repeating and is performed by the station's repeater. The repeated information may be the token, MAC frames, or data frames. When a station is in the listening state, it is not transmitting any data of its own.

The ring can also be in two basic states: *idle* and *active*. In the idle state, only the token (and fill) are on the ring and are being read and retransmitted by the stations on the ring, as illustrated in Figure 1-1. In the active state, data or MAC frames have been placed (or are being placed) onto the ring and are being read and retransmitted by the stations on the ring. Some token ring protocols allow only one data or MAC frame on the ring at a time; other protocols allow numerous frames to be on the ring simultaneously. Figure 1-2 describes the activities of an active 802.5 Token Ring.

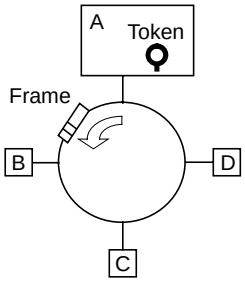
Frames

When a data or MAC frame reaches the destination station, that station copies the frame and marks it as "seen" (the *A bit*) and "copied" (the *C bit*) before it repeats the frame back onto the ring. Note that the data frame is copied, not removed, by the destination station. The frame continues along the ring until it reaches the originating station, which recognizes the frame as one of its own, notices that the frame has been copied by the targeted receiver, and *strip*s (removes) it from the ring. (See Figure 1-2 for a summary of this sequence for an 802.5 Token Ring.) This design ensures that undeliverable frames do not circle endlessly around the ring. When the transmitting station finishes (that is, has no more data to transmit or runs out of time), it places fill or idle patterns onto the ring until it *regenerate*s (releases) the token, at which time the token is available for use by the next downstream station wishing to transmit data.

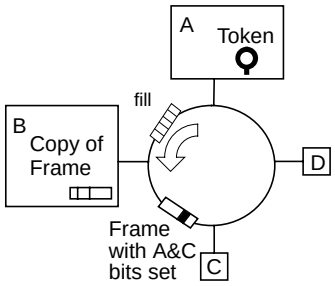
Management

Each token ring station has a management module that detects and recovers from error conditions. Each protocol based on the token ring architecture specifies a behavior for its management entities. Some protocols assign one management entity to be the master or active one, while all other management entities are passive; 802.5 Token Ring uses this scheme. Other protocols, such as FDDI, are designed so that all the stations participate equally in ring management.

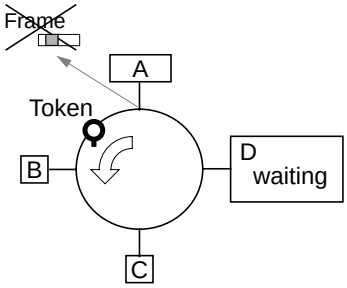
Each communication protocol based on the basic token ring architecture (for example, 802.5, FDDI) must specify an algorithm for determining the length of its token holding timer and its own rules for governing the release of the token, the use of fill, and the stripping of frames. Figure 1-2 illustrates the algorithm for 802.5 Token Ring when not using the early token release optimization.



1. Station **A** captures token.
2. **A** transmits first data or MAC frames (not shown).
3. **A** transmits final frame onto ring. Intended receiver is Station **B**.



4. Station **A** transmits fill, awaiting return of final frame. (Early token release is not in use.)
5. Station **B**, the destination station, copies data in final frame.
6. **B** marks frame as seen and copied and places back onto ring.
7. Frame continues to circulate around entire ring.



8. When frame returns to station **A**, **A** strips frame from ring.
9. **A** places token onto the ring.
10. Next downstream station waiting to transmit (**D**) will capture token.
11. Process is repeated (not shown).

Figure 1-2 Token Ring—a Sending Station and an Active Ring

802.5 Token Ring

This section describes design features and specifications that are specific to the 802.5 Token Ring (as compared to the basic token ring architecture described in the previous section). Detailed subsections cover the following topics: token release and frame stripping, ring maintenance, data prioritization, and early token release.

Table 1-1 lists some of the items that are defined by the 802.5 Token Ring Standard. These items are specific to 802.5 and not the same for other communications protocols (such as FDDI) that are based on the token ring architecture.

Table 1-1 802.5 Token Ring Specifics

Item	Specification for 802.5 Token Ring
pattern for token	see Figure Gl-6
format for data frame	see Figure Gl-1
types of management frames	see Figure Gl-4 and Table 1-2
data transmission speeds	4 and 16 megabits per second (Mbps)
physical transmission media	shielded and unshielded twisted pair and (in the future[a]) optical fiber
maximum frame size (MTU)	4500 bytes for 4 Mbps ring 18000 for 16 Mbps ring
method for connecting stations to ring	*trunk coupling unit* for example, multistation access unit (MAU), wiring concentrator (WC), or lobe attachment module (LAM)
station states required for implementing the protocol	defined in the *802.5 Token Ring Standard*
rules for releasing token and stripping frames	see “802.5 Rules for Token Release and Frame Stripping” on page 17

Table 1-1 (continued) 802.5 Token Ring Specifics

Item	Specification for 802.5 Token Ring
method for managing and maintaining the ring	see “802.5 Ring Maintenance” on page 18
design enhancements	early token release (see “802.5 Early Token Release” on page 23) mechanism for prioritizing data (see “802.5 Mechanism for Prioritizing Data” on page 22)

a. Some vendors already offer proprietary optical fiber for Token Ring, but the standard for this transmission medium has not been approved officially by IEEE.

802.5 Rules for Token Release and Frame Stripping

The 802.5 Token Ring Standard specifies that the transmitting station must release the token when both of the following conditions have been met:

- The station cannot complete transmitting another data or MAC frame before the token-release timer expires or the station has no more data to transmit, whichever condition occurs first.
- All of the station’s transmitted frames have returned, and the station has stripped them from the ring.

Note: The early token release feature, described in the “802.5 Early Token Release” on page 23 section changes these rules slightly.

These rules ensure that each frame circles the ring at least once, thus giving every station the opportunity to copy frames sent to it, and make every station responsible for removing its own frames from the ring. Until the above two conditions are met, the transmitting station does not regenerate the token. As long as the token-holding timer has not expired, a station awaiting the return of its frames simply transmits fill. Once the token-holding timer expires, if the token cannot be regenerated, there must be a problem somewhere on the ring that will be handled by the Token Ring maintenance procedures.

The *active monitor* (explained in the next section) removes (*purge*s) any frames that are not stripped by their creators.

802.5 Ring Maintenance

The 802.5 Standard details how a ring must be maintained. Ring maintenance is needed to maintain order, keep track of control parameters, and overcome error conditions. Common error conditions include the following: the transmitting station does not regenerate the token (a condition called *lost token*), or a station stops receiving the signal from its upstream neighbor (a condition referred to as a *break in the ring*).

Each Token Ring station has one or more software modules that handle the 802.5 network management (NMT) and *monitor* activities. The NMT and monitor oversee low-level (MAC and PHY) behavior (such as insertion into the ring) and recover the ring when a problem occurs. Both NMT and monitor activities obtain information by sending and receiving special management frames (*MAC frames*).

The remaining paragraphs in this section describe how stations participate in ring maintenance during these stages:

- start-up
- normal operation
- as active monitor
- as standby monitor
- when the station suspects problems on the ring

When a station starts, it *insert*s itself onto the ring by starting to listen to the passing signal. It then performs the following sequence of initialization steps:

- Verifies that its *MAC address* is unique (using the *duplicate address test frame*).
- Sets a special timer.
- Obtains its upstream neighbor's MAC address (*UNA*) from a passing *standby* or *active monitor present* frame.
- Watches the ring for evidence that there is an *active monitor*. Each ring must have one active monitor. (Active monitor duties are explained later in this section.)

If it identifies an active monitor, the station puts its own monitor into the standby state and begins functioning as a listening (repeating) station. This is the normal sequence leading to normal operation.

If the timer (set during initialization) expires without the station seeing evidence of an active monitor, there is a problem. The station begins the *token claiming* process (using *claim token* frames), during which it bids to be the active monitor. It is possible that other stations also have noticed the problem and also will be token claiming. The station that wins the bidding becomes the active monitor and places a token onto the ring. Only one station should be the active monitor; all the other monitors are in standby.

If a station becomes aware (during its initialization and insertion process) that it, or the ring, is dysfunctional or that its MAC address is already in use on the ring, the station's management module removes the station from the ring by transitioning to a *bypass* state where it is not inserted onto the ring at all.

During normal operation, each station that is inserted onto the ring has the following responsibilities:

- Repeat all frames it sees and, when appropriate, modify special bits.
- Copy frames for which it is the destination.
- Perform appropriate monitor duties (either as the active monitor or as a standby monitor, as explained in paragraphs below).
- Manage the *priority* mechanism.
- If the station transmits frames, strip its own frames from the ring and regenerate the token.

The active monitor's responsibilities, over and above its normal operational duties, include the following:

- Set the ring's timing.
- Set the ring's latency buffer.
- *Purge* orphaned (*persistently circulating*) frames or tokens.
- Regenerate the token, if it is lost.
- Advertise its continuing presence as the active monitor by generating *active monitor present* (AMP) frames at regular intervals.
- Verify that it is the only active monitor on the ring. If it isn't, take steps to rectify this condition by becoming a standby station and token claiming.
- Verify that the ring is a complete loop by monitoring the return of its own frames. If the ring is broken, take action to remedy the condition by becoming a standby monitor and *beaconing*.

Each standby monitor has the following responsibilities, over and above its regular station duties:

- Advertise its continuing presence as an active station / standby monitor, using *standby monitor present* (SMP) frames.
- Verify that an active monitor exists on the ring, and if not, take action to remedy the condition by token claiming (generating *claim token frame*s).
- Verify that the ring is a complete loop by monitoring the return of its own frames. If the ring is broken, take action to remedy the condition by *beaconing* (generating *beacon frames*).

When a station suspects a problem on the ring, it initiates token claiming and / or beaconing. These activities are explained in the following paragraphs.

In the token claiming process, one or more stations offer to become the active monitor. Token claiming starts when there is no active monitor. Token claiming ends when one of the stations becomes the active monitor. Whenever a standby-monitor station fails to see the token or an AMP frame within a specified period of time, the station initiates the token claiming process by generating a claim token frame (also referred to as a claim).

It is common for a number of stations to issue claims more or less simultaneously because they all noticed the problem. As the claim(s) circulates the ring, data transmission activity on the ring stops. Claiming continues until the ring is *stable*. The first claiming station to receive its own claim token frame and two consecutively matching upstream neighbor addresses becomes the active monitor. Its first activity is to purge all information from the ring, then issue a new token.

If a station detects an upstream *break in the ring*, it issues a *beacon* frame to notify all downstream stations that a serious problem has occurred. The problem may be in the reporting station's own reception *port* or somewhere upstream (for example, the ring cable, the *lobe* cabling, a trunk coupling unit, or the neighbor station's transmission system).

As with the claim frame, when a beacon circulates the ring, data transmission activity on the ring stops, and all stations participate in the ring recovery process. As each station sees the beacon frame, it goes into the listening state. Each station continues issuing beacons until it receives a beacon frame (either its own or another station's). Reception of any beacon indicates that the portion of the ring immediately upstream from this station is again intact; reception of its own beacon indicates that the entire ring is intact.

If the beaconing station sees its own beacon, it begins token claiming; if it sees another station's beacon, it stops beaconing but does not token claim.

Token Ring maintenance is accomplished using a number of special frames, referred to as management or *MAC frames*. Some of the more important MAC frames are summarized in Table 1-2.

Table 1-2 802.5 Token Ring Management (or MAC) Frames

Frame	Acronym	Description
Active Monitor Present	AMP	Issued by the active monitor to notify all the other stations of its presence on the ring.
Standby Monitor Present	SMP	Issued by each active station on the ring, except the active monitor, to notify the downstream station of its presence and its address. Each station obtains its UNA information from this frame.
Duplicate Address Test	DAT	Issued by a station during its initialization process to verify that no other station is using the same MAC address.
Claim Token	CL_TK	Issued by any station that detects the loss of the active monitor. Used to determine which station will be the new active monitor on the ring (a process called token claiming).
Beacon	BCN	Issued by any station that detects a break in the network (for example, no signal seen on the ring for a period of time).
Ring Purge	PRG	Used by the active monitor to clear the ring prior to issuing a new token.
Lobe Media Test	LMT	Used by each station during its initialization phase in order to verify the section of cable (*lobe*) that connects the station to the ring.

802.5 Mechanism for Prioritizing Data

The 802.5 Token Ring Standard contains a priority mechanism that results in higher priority data, from any station, being transmitted before any lower priority data, as explained below.

Each Token Ring frame (data, token, or management) contains two special fields for implementing the priority mechanism: *priority* field and *reservation* field. The priority field contains the ring's current priority setting; the reservation field contains a requested priority setting.

The priority field supports eight levels of priority. Any time a station captures the token, it is allowed to transmit data whose priority setting is equal to or higher than the token's priority setting. If a station has data with only a lower priority level, it must wait until the ring's priority setting is lowered before it can transmit its data.

When a waiting station has data with a higher priority level than the current setting, the station is allowed to increase the ring's priority level, thus locking out stations with lower priority data and increasing its own access to the ring. Raising the priority level is accomplished in the following manner:

- First, the station indicates its desired priority by writing the level into the reservation field of any frame that it repeats. This action makes the statement "I want to transmit data at this priority level." Stations can raise, but not lower, this requested value.
- Second, when the station captures the token, it transmits as much data as it can in its allowed time (assuming that its waiting data is of a priority equal to or higher than the ring's current priority). If the station still has high-priority data left, it regenerates the token with a new higher priority level, equal to its data's priority level.

Raising the ring's priority in this way is called *stacking*. The new token cannot be captured by stations with lower-priority data, only by stations with equal or higher priority; therefore, it is likely that this station will have an opportunity to recapture the token much sooner than if every station on the ring were allowed to transmit.

When the station recaptures the token and finishes transmitting its high-priority data, it generates a token with a new ring priority. The new setting must be whichever of the following settings is higher: the priority level that was in effect before the station stacked the priority or the highest requested level seen in any reservation field since the station stacked the ring's priority.

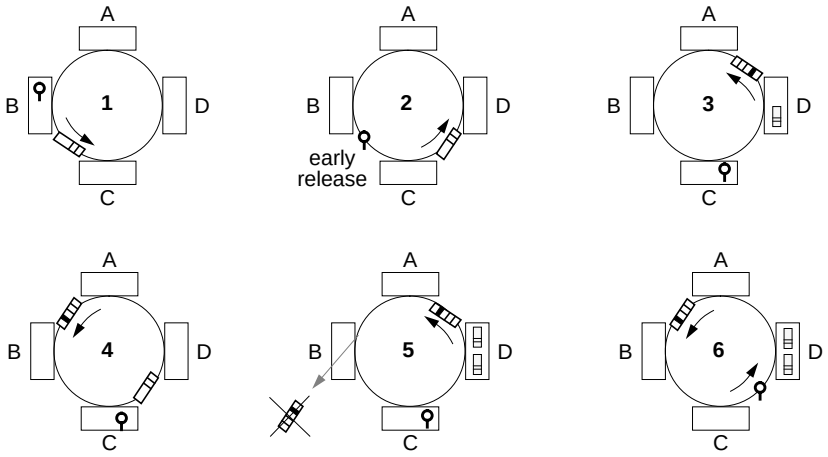
After a station has increased the ring's priority and before it recaptures the token, other stations with equal or higher priority can capture the token and transmit data. They can even raise the priority level such that the original stacking station is out-prioritized and must wait for the level to be decreased. The reservation mechanism guarantees that the level will return to each station's requested level as soon as all waiting higher-level priority data has been transmitted.

802.5 Early Token Release

The 802.5 Token Ring Standard provides an optional enhancement to the basic token ring design: *early token release*. This feature is available only for rings operating at 16 megabits per second (Mbps). Early token release allows multiple frames on the ring at the same time, thus increasing ring efficiency and *use of bandwidth*. On a 16-Mbps ring, stations that use early token release can coexist with stations that do not.

Note: The PCI Token Ring board does not support the early token release feature.

Stations that support the early token release feature generate the token immediately after transmitting their final frame, without waiting for the frame to return and be stripped. Since the transmitting station does not have to wait and the next station can transmit sooner (before all frames have been stripped from the ring), this feature can make the ring more efficient. Figure 1-3 illustrates early token release. In this example, notice that at position 4, two data frames exist on the ring simultaneously, and at positions 2 and 6 the token is on the ring while data frames are still circling.



1. B has captured token, so it transmits a frame to D.
2. C repeats B's frame.
   B releases token (early release).
3. C captures token.
   D copies B's frame and retransmits it (marked).
4. C transmits a frame to D.
   A repeats B's returning frame.
5. B purges its frame from ring.
   D copies C's frame and retransmits it (marked).
6. C releases token (early release).

Figure 1-3 Early Token Release

Token Ring Compared to Ethernet

Token Ring and Ethernet are similar and yet different. The design of both protocols supports the following:

- use of shielded and unshielded twisted pair copper cable
- multiple hosts transmitting and receiving over shared cable

There are also a great many differences between the two protocols. One of the main differences is the order and manner in which hosts access the cable. This difference affects network *latency* (waiting for access to the cable), *use of bandwidth* (data carrying capacity of the cable), and network management.

Ethernet hosts transmit more or less at will, without regard to any order, refraining only if they detect another station transmitting. Ethernet stations that have gained access to the cable hold onto their access only for the time necessary to transmit their data, unlike Token Ring, where stations generate fill (that wastes bandwidth) during certain periods of their access time. In addition, an Ethernet cable never has to be idle, as the Token Ring cable is when stations are passing the token.

The Ethernet scheme minimizes network latency and optimizes the *use of bandwidth*. When many Ethernet stations want to transmit simultaneously, two or more stations' signals can "collide" in their access attempts. No data is lost and no user-application retransmission is required, since the hardware automatically makes access attempts until it succeeds; however, a very small amount of lost (unused) bandwidth is associated with each collision.

Token Ring hosts transmit only when the circulating token arrives at their reception port and in the order they are attached to the cable (sometimes referred to as *round-robin*). Since no data can be transmitted while the token is on the ring, this scheme carries with it a set minimum latency (*guaranteed latency*), which results in some loss of bandwidth. This latency increases with the network's cable length and the number of attached hosts. On the other hand, no Token Ring bandwidth is lost due to unsuccessful access attempts.

Under light to medium loads (up to 50% of the cable's bandwidth), Ethernet performance benefits from its lower latency; under extremely heavy loads (over 85% of bandwidth) Token Ring performance benefits from its contention-free design.

Ethernet stations do not need to know about their ordering because they do not pass anything to the "next" station. In order to function properly, a Token Ring network requires that all attached stations be ordered sequentially. Because of this difference, Token Ring network management is more complex than Ethernet management. In addition, a small portion of the Token Ring's bandwidth is occupied by management frames.

Other differences include how fast the signal travels along the cable (the data rate or transmission speed) and how large each frame can be, as summarized in Table 1-3.

Table 1-3 802.5 Token Ring Versus Ethernet

Protocol	Data Rate (megabits per second)	Maximum Frame Size	Access Order	Method for Obtaining Permission to Transmit	Method to Acknowledge Reception
Token Ring	4 or 16	4,500 or 18,000	round-robin	capture token	C bit set in original data frame
Ethernet	10	1,500	first-come first-serve, if cable is idle; random, if there is contention	listen to verify cable is not being used (CSMA / CD)	none at Ethernet layers

Common Cabling and Connectors in Token Ring Environments

Token Ring sites can be equipped with a variety of cables and connectors; this section describes some of the more common ones. Table 1-4 summarizes common combinations of cabling and connectors, while Figure 1-4 illustrates them.

Hardware components exist for converting one type of connector to another (for example, an RJ-11 to an RJ-45) or for splicing one type of cabling to another (for example, type 2 to type 3). Consult an authority before installing any of these converters; the compatibility issues can be complex (for example, when converting an RJ-11 to an RJ-45 you must know whether the connectors are wired straight-through or cross-pinned).

Note: Common telephone cabling, sometimes referred to as “flat cable,” “silver satin,” or “flat gray cable,” should not be used for adapter cables, trunk ring cabling, or lobe cabling.

Table 1-4 Cables and Connectors in Token Ring Environments

Cable	Max. Mbps	Commonly Used Connectors	Description
Type 1	155	Data Connector at trunk coupling unit (TCU); DB-9 at adapter board.	Shielded twisted pair. 2 shielded pairs of copper wire; 22 AWG.
Type 2	155	Data Connector at TCU; DB-9 at adapter board.	Shielded and unshielded twisted pair. 2 shielded pairs of copper wire and 4 unshielded pairs of copper wire; 22 AWG.
Type 3, Category 3	10	RJ-11 (only for 4&6 wire cable) or RJ-45 at TCU and at adapter board.	Unshielded twisted pair. 2, 3, or 4 unshielded pairs of copper wire; 22-24 AWG.
Type 3, Category 4	20	RJ-11 (only for 4&6 wire cable) or RJ-45 at TCU and at adapter board.	Unshielded twisted pair. 2, 3, or 4 unshielded pairs of copper wire; 22-24 AWG.
Type 3, Category 5	100	RJ-11 (only for 4&6 wire cable) or RJ-45 at TCU and at adapter board.	Unshielded twisted pair. 2, 3, or 4 unshielded pairs of copper wire; 22-24 AWG.

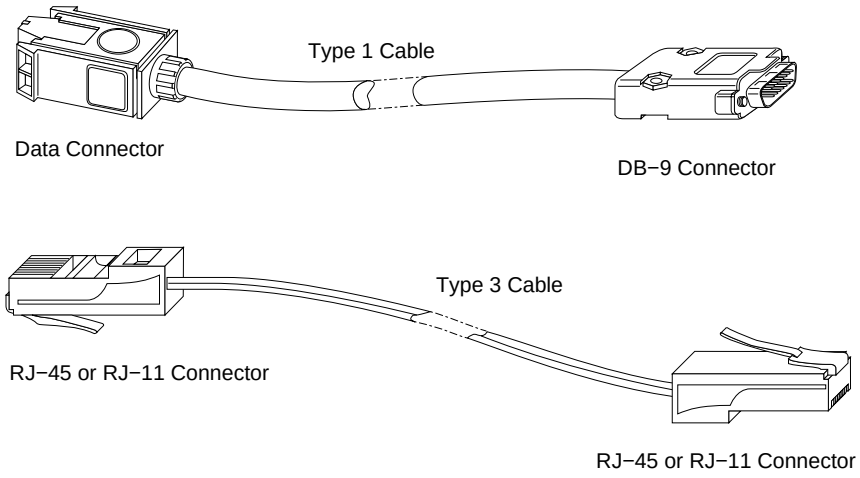


Figure 1-4 Token Ring Connectors and Cabling

Standardization

Token Ring is a popular local area network (LAN) technology. The protocol has been standardized by the Institute of Electrical and Electronics Engineers (IEEE) and approved by the American National Standards Institute (ANSI) and the International Standards Organization (ISO).

802.5 Token Ring interoperability is provided through standard interfaces at the physical layer (PHY), *medium access control sublayer* (MAC), and *logical link control sublayer* (LLC) of the Open Systems Interconnection (OSI) reference model. Figure 1-5 illustrates the protocol services provided by the IRIS Token Ring product compared to the OSI and *Systems Network Architecture* (SNA) layers.

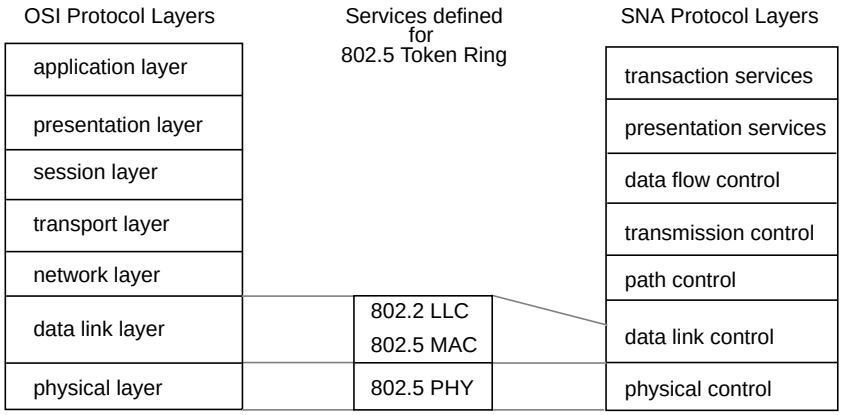


Figure 1-5 802.5 Token Ring Protocol Services Compared to OSI and SNA

Additional Information

Additional sources of information about 802.5 Token Ring are listed in Table 1-5, with a brief description of the type of information each document provides.

Table 1-5 Additional Sources of Information

Document	Description
802.5 Local Area Network Standard. Published by Institute of Electrical and Electronics Engineers, Inc., Service Center, P.O. Box 1331, Piscataway, New Jersey, 08855, USA. 908-981-0060	Official IEEE 802.5 Token Ring documentation specifying the protocol. Useful or engineers and communication designers, implementors, and writers.
8802-5 Token Ring Protocol. Published by Phillips Publishing, 7811 Montrose Road, Potomac, MD, 20854, USA.	Official ISO 8802-5 Token Ring documentation specifying the protocol. Useful for engineers and communication designers, implementors, and writers. The technical content of this document is the same as for the IEEE document listed above.
Handbook of Computer-Communications Standards, Local Network Standards, Vol. 2. Written by William Stallings. Published by Macmillan Publishing Company, New York.	Detailed descriptions, in layman's English, of many local area network protocols and technologies, including 802.2 and 802.5. Valuable for people planning, managing, and/or troubleshooting any LAN, including Token Ring.

IRIS Token Ring Software

The IRIS Token Ring driver provides the communication path between the upper layers of the protocol stack (TCP/IP or SNA) and the IRIS Token Ring board, as illustrated in Figure 1-6. This section provides an overview of the IRIS Token Ring driver.

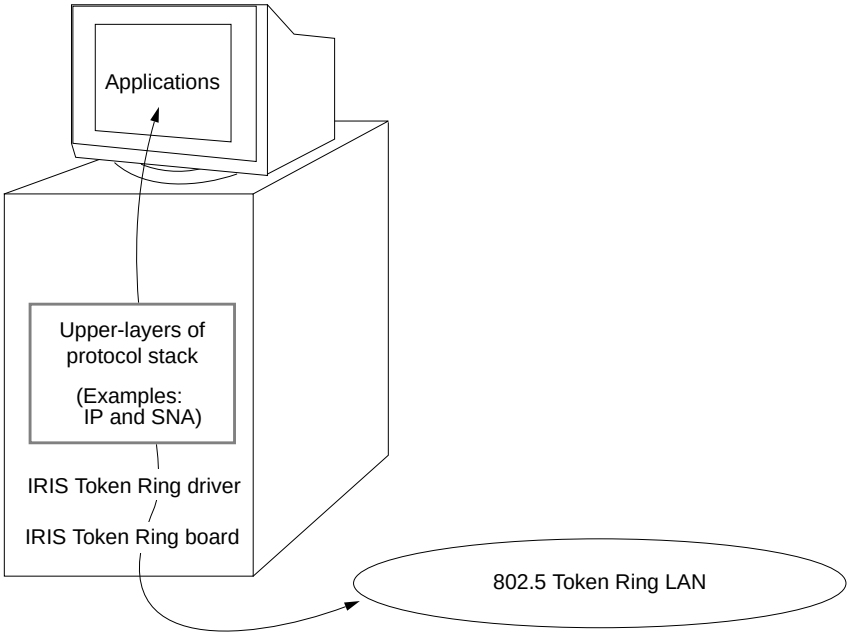


Figure 1-6 Token Ring Board within Protocol Stack

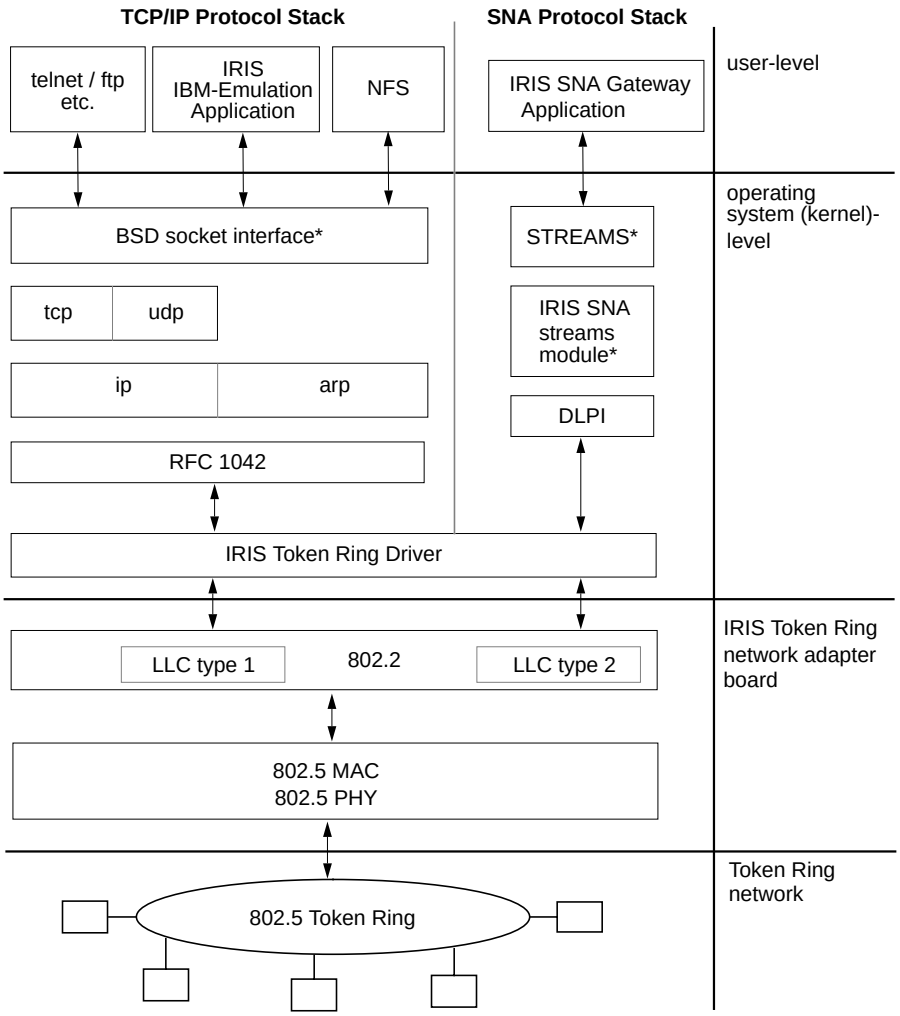
The IRIS Token Ring Driver and Protocol Stacks

The services of the IRIS Token Ring network connection are used by the higher layers of the protocol stack, as illustrated in Figure 1-7. A Silicon Graphics workstation or server can be configured with one protocol stack (TCP/IP only) or more, for example, TCP/IP and SNA. The IRIS Token Ring driver provides the logical interface between the board and the network protocols.

The higher layers of the protocol stack can be from a number of different protocol families, for example, TCP/IP, Snoop Protocol, SNA, IBM's NetBIOS™, or Novell®'s NetWare®. Silicon Graphics' TCP/IP, Snoop Protocol, and SNA stacks are described in separate subsections. Figure 1-7 summarizes the driver's role as it relates to user-level applications, other operating system-level processes, and the IRIS Token Ring board.

The IRIS Token Ring driver has two main functions:

- to inspect and pass incoming frames to the correct upper layer
- to accept outgoing frames from the higher layers of the protocol stack(s) and queue them to the board



* These components do not represent network layering.

Figure 1-7 Token Ring Driver and Protocol Stacks

TCP/IP

The IRIS TCP / IP protocol stack is the default, and required, protocol stack for a Silicon Graphics system. The IRIS Token Ring driver depends on a properly configured and functioning TCP / IP protocol stack. The TCP / IP stack (or IP suite) includes the following protocols, among others: Internet Protocol (IP), Address Resolution Protocol (ARP), Internet Control Message System protocol (ICMP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). The TCP / IP stack uses type 1 service at the data link layer (that is, LLC type 1, unacknowledged connectionless service).

Communication between the IRIS user applications (including IBM emulation products) and the IRIS Token Ring driver is through the BSD socket interface, as illustrated in Figure 1-6 and Figure 1-7.

The TCP / IP stack supports the following user applications (among many others):

- The Network File System (NFS)™, which allows transparent file access between remote nodes throughout the network. NFS is discussed in detail in the *ONC3/NFS Administrator's Guide.*
- The Network Information Service (NIS), which is a name and address lookup service (directory service) that provides a centralized network database to ease network administration. NIS is discussed in detail in the *NIS Administration Guide.*
- Internet applications / utilities, including those listed below, all of which are part of the standard IRIX operating system. To obtain more information about these applications, use the *man* command.
  - telnet(1C), the user interface to the TELNET protocol that allows users to log on to a system within a remote network
  - ftp(1C), the Internet file transfer program that allows users to transfer files to and from remote networks
  - rcp(1C), the remote copy utility that allows users to copy files to and from systems throughout a site's networks
  - rlogin(1C), the remote log on utility that allows users to log on to remote systems located within a site's networks

- IRIS products that emulate IBM SNA applications, such as IRIS 3270 Terminal Emulator and SGI 3770 SNA Terminal Emulator. These products rely on an SNA gateway (for example, the IRIS SNA Server product) to handle their connection to the SNA world (for example, their link to an IBM host).
- IRIS user applications such as Showcase™ and IRIS Insight™.
- User applications that have been ported to the IRIX™ operating system. Examples include FrameMaker®, Wingz™, and Z-Mail™.

SNA

The IRIS SNA protocol stack is an optional protocol stack. It includes modules to handle the Data Link Provider Interface (DLPI), System Network Architecture (SNA) streams, and UNIX STREAMS, as illustrated in Figure 1-7. The SNA protocol stack requires LLC type 2 (connection mode service). IRIS SNA Server is an example of a product that requires the SNA protocol stack.

Communication between the IRIS Token Ring driver and the IRIS SNA protocol stack is achieved through UNIX STREAMS and a Data Link Provider Interface (DLPI), as illustrated in Figure 1-7. The DLPI module provides SNA *source routing* support.

Figure 1-8 illustrates some of the configurations that can be achieved in an SNA environment. A tag of “IRIS IBM Ap” indicates the presence of any of the products that emulate IBM SNA-based applications.

Raw Protocol Stack and Capture All Frames

The IRIS Token Ring board supports promiscuous (or capture-all-frames, CAF) functionality. This functionality is enabled by running an application that utilizes a raw protocol stack (for example, the BSD socket-based Snoop Protocol of the Raw Network Protocol family). Applications (for example, NetVisualyzer) that use the Snoop Protocol automatically download the CAF firmware to the IRIS Token Ring board when the application is started.

When operating in promiscuous mode, the IRIS Token Ring board does not filter the frames it sees on the cable; rather, it copies all the frames it sees and passes them to the upper layer applications. This functionality is useful for diagnosing problems on a ring. When CAF firmware is running on the IRIS Token Ring board, only LLC type 1 is available.

For more information, see the man pages for raw(7F) and snoop(7P).

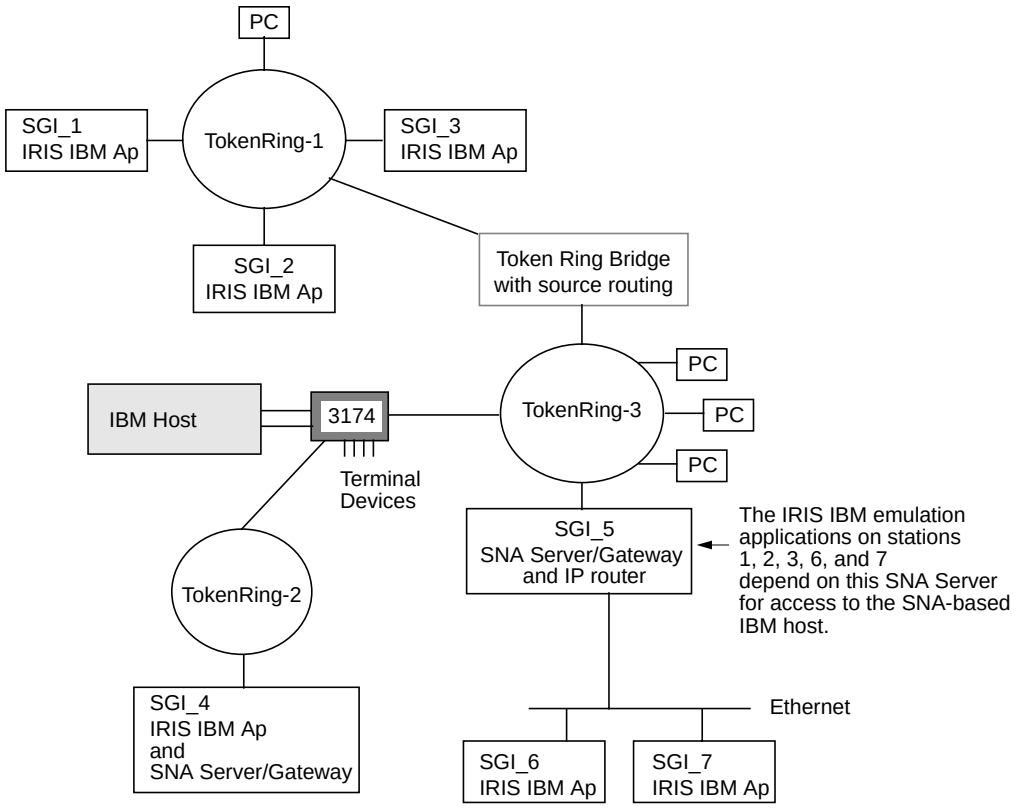


Figure 1-8 SNA over Token Ring Configurations

Note: The IBM host connections (for example, 3174 Local SNA Cluster controllers, 3172 Interconnect Controllers, 3725/3745 Communication controllers) that are tested and supported for use with Silicon Graphics' products are specified in the *IRIS Token Ring Release Notes.*

Maximum Number of Token Ring Boards

The IRIS Token Ring driver can support up to eight Token Ring boards.

Note: A particular system may have further limitations on the number of Token Ring boards, depending on the system's configuration. For example, the IRIS SNA Server product supports a maximum of four devices, which can be any combination of Token Ring boards and SDLC adapters.

Link Station Support

The number of supported link stations, service access points (SAPs), and memory buffers varies from board to board. For example, a board with expanded memory provides more support than one without this expansion. All IRIS Token Ring boards provide sufficient (not necessarily optimal) support for IRIS applications and can accommodate both workstation and server applications. See the *IRIS Token Ring Release Notes* for recommended configurations.

Chapter 2

Configuring IRIS Token Ring

This chapter describes how to configure the IRIS Token Ring product. For an initial IRIS Token Ring installation, the configuration must be done after installation of the software and before installation of the board.

Note: The IRIS Token Ring software must be already installed. Information on software installation is provided in the *IRIS Token Ring Release Notes*.

The configuration tasks that you may want to perform are divided into these areas:

- the physical (MAC) address for the IRIS Token Ring board
- the data transmission speed for the IRIS Token Ring board
- the IRIS Token Ring driver (including its transmission speed)
- all network interfaces on this station
- the protocol stack(s)

The procedures you use to accomplish these configuration tasks are dependent on the type of Token Ring board that you have. To summarize:

- With the PCI Token Ring board, you use the *mtrconfig* utility (see mtrconfig(1M)) and its various options to modify the default configuration.
- For all other types of Token Ring boards, you either use the *trconfig* utility (see trconfig(1M)) or manually edit the */etc/config/trconfig.options* file to modify the default configuration.

Configuring the Board's Physical (MAC) Address

This section describes how to assign a locally defined physical address (*MAC address*) to the IRIS Token Ring board. If your site uses universally assigned 48-bit MAC addresses, skip this section.

Note: The IRIS Token Ring product requires the 6-byte (48-bit) MAC address format. The product does not support networks that use the 2-byte (16-bit) format.

MAC Address Description

Each Token Ring board must have its own unique MAC address. MAC addresses are either locally assigned (LAA, assigned at the site and guaranteed to be unique only to that site) or universally assigned (UAA, assigned by a standards organization and guaranteed to be unique across the world). Silicon Graphics' Token Ring boards are shipped with IEEE universally assigned 48-bit MAC addresses that are permanently stored within a component on the board. Each board's address is read by the software during reset (startup).

If your Token Ring local area network uses locally assigned MAC addresses, you need to manually override the factory-assigned address for each board. Do this by updating the Token Ring configuration to specify the locally assigned addresses, as described in "Updating the MAC Address."

Note: A Silicon Graphics system running the IRIS SNA Server product or the IRIS 5080 Gateway product is a probable candidate for configuration of a locally assigned MAC address.

Updating the MAC Address

The Token Ring board configuration file overrides the IRIS Token Ring board's universally assigned MAC address. The entry in the configuration file becomes the board's functional physical address.

To override the board's MAC address, do the following:

1. From the appropriate network administrator, obtain a site-assigned MAC address for each IRIS Token Ring board. The address must be a 48-bit address in hexadecimal format (for example, 40.00.70.0c.00.04, which can also be represented as 40.0.70.c.0.4).
2. Become the superuser with the following command.

   ```
   % /sbin/su
   Password: thepassword
   ```

3. If you have a PCI Token Ring board, complete this step; otherwise, go to the next step. At the shell prompt, type the command:

   mtrconfig *interface* **-M** *LLA_address*

   where *interface* is the board you want to change (for example, mtr0 for the first board) and *LLA_address* is the site-assigned MAC address you obtained in step 1.

   This update is now complete. Follow the instructions in "Displaying the Current MAC Address" on page 44 to verify that the new address is being used.

   If the board is not installed, continue installing and configuring the IRIS Token Ring product.

4. Use Table 2-1 to identify the interface name for each of your station's Token Ring boards. If your platform is not listed in the table, refer to the *IRIS Token Ring Release Notes*.

Table 2-1 Network Interface Names for IRIS Token Ring

Platform	Token Ring Interface Name
VMEbus-based systems	fv0 for first board encountered fv1 for second board encountered fv2 for third board encountered fv3 for fourth board encountered
IRIS Indigo™ systems	gtr0

5. Use any text editor to modify the *trconfig.options* file. The commands below open the file for editing with *jot*:

   # **/usr/sbin/jot /etc/config/trconfig.options**

   An unaltered *trconfig.options* file is illustrated in Figure 2-1.

```
# trconfig.options:
#
# This file allows you to change your token ring MAC address.
# field 1 - device name (gtr0, fv0, fv1, fv2, or fv3)
# field 2 - '-M' to indicate MAC address following
# field 3 - a 6-byte MAC address in hexadecimal representation
#           where consecutive bytes are separated by a period (.)
#
#gtr0 -M 40.0.0.b.6c.5
#fv0 -M 40.0.70.0.0.4
#fv1 -M 40.0.70.0.0.5
```

Figure 2-1 Unaltered */etc/config/trconfig.options* File

6. For each Token Ring board that needs a locally assigned MAC address, do the following:

   - Copy the appropriate line (for example, `#fv0 -M 40.0.70.0.0.4` or `#gtr0 -M 40.0.0.b.6c.5`) to the last line of the file.
   - On the new line, remove the "ignore this line" (#) sign from the beginning of the line.
   - Replace the MAC address (for example, `40.0.70.0.0.4`) with the valid MAC address you received from your system administrator. Do not remove the leading `-M` nor the space between the `M` and the address.

   The altered file should look something like Figure 2-2, where a *gtr* Token Ring board has been assigned an address of c0.00.70.ff.f1.0b.

```
# trconfig.options:
#
# This file allows you to change your token ring MAC address.
# field 1 - device name (gtr0, fv0, fv1, fv2, or fv3)
# field 2 - '-M' to indicate MAC address following
# field 3 - a 6-byte MAC address in hexadecimal representation
#           where consecutive bytes are separated by a period (.)
#
#gtr0 -M 40.0.0.b.6c.5
#fv0 -M 40.0.70.0.0.4
#fv1 -M 40.0.70.0.0.5

gtr0 -M c0.0.70.ff.f1.b
```

Figure 2-2 Example of Altered */etc/config/trconfig.options* File

7. Save and close the file.
8. If the board is already installed, reboot the system to start using the new MAC address(es). Follow the instructions in the next section to verify that the new address is being used.

   If the board is not installed, continue installing and configuring the IRIS Token Ring product.

Displaying the Current MAC Address

To display an installed device's current MAC address, use one of these commands, depending on the type of board you have:

- For the PCI board, use the *mtrconfig* command as shown below:

  `mtrconfig` *interface* **`-v`**

  where *interface* specifies the interface to report (for example, mtr0 is the first interface). The output should be similar to the following display:

```
TokenRing controller: mtr0
    Burn-in MAC address 00:00:f6:54:02:25
    Current MAC address 00:00:f6:54:10:55
    ring speed 16Mbit
    Current broadcast address ff:ff:ff:ff:ff:ff
    Current max packet size 4472
```

- For all other boards, use the *netstat* command as shown below (see netstat(1)):

  # **`/usr/etc/netstat -ia`**

  The output for the Token Ring board (for example, *fv0*) should look similar to Figure 2-3. The MAC address displayed is the one currently being used by the system to identify that board.

  Note: If you altered the */etc/config/trconfig.options* but the display does not show that your locally assigned address is being used, reboot the system.

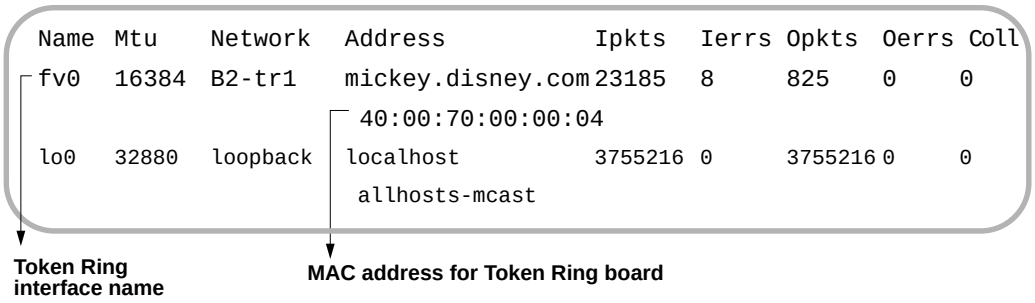


Figure 2-3 Displaying Current MAC Address

Configuring the Board's Data Transmission Speed

The IRIS Token Ring board's data transmission speed must be set to match the speed at which the ring is operating: either 4 or 16 megabits per second (Mbps). If the IRIS Token Ring board is configured to operate at one setting and the ring to which it is attached operates at a different speed, the IRIS Token Ring board's attempt to insert itself onto the ring fails and the driver disables itself.

Note: IRIS Token Ring boards are shipped already configured for 16 Mbps.

Some IRIS Token Ring boards have a transmission speed component with a jumper that can be moved to change the speed from one setting to another. This configuration method results in fast system reboots.

Note: To set the jumper on an IRIS Indigo workstation, refer to the *IRIS Token Ring for Indigo Installation Guide*. To set the jumper on the IRIS Token Ring board for any other Silicon Graphics system, refer to the installation instructions.

Some IRIS Token Ring boards (for example, for the IRIS Indigo platform) allow the board's transmission speed to be set from the console, using either the *mtrconfig* or the *trconfig* command. This method causes the new setting to be written into a memory chip on the board where the driver reads it each time the driver is enabled. The setting in the board's memory takes precedence over the board's jumper setting.

To change the IRIS Token Ring board's data transmission speed from the console, follow the steps below:

1. Become superuser (root):

   ```
   % /sbin/su
   Password: thepassword
   ```

2. Disable the Token Ring network interface, if necessary.

   ```
   # /usr/etc/ifconfig interfacename down
   ```

3. Change the transmission speed setting. Use the command appropriate for your board.

   - For PCI boards, use the following command:

     mtrconfig *interface* **-s** *newspeed*

     where *interface* is the board you want to change (for example, mtr0) and *newspeed* is either 4 or 16.

   - For all other boards, use the following command:

     ```
     # /usr/etc/trconfig interfacename -s newspeed
     Warning: Writing interfacename EEPROM.
     Do not reset the system until finished.
     Finished.
     ```

     where *interface* is the board you want to change and*newspeed* is either 4 or 16.

4. Use the following command to verify that the network interface is enabled. In the display, an UP flag indicates that the interface is enabled.

   ```
   % /usr/etc/ifconfig interfacename
   interfacename:flags=c63<UP,BROADCAST,RUNNING,MULTICAST>
   inet x.x.x.x netmask 0xffffff00 broadcast x.x.x.255
   ```

   Note: If the interface is not UP, use the command below to enable it:

   ```
   % /usr/etc/ifconfig interfacename up
   ```

5. Verify that the new transmission speed is set with the command below:

   ```
   % /sbin/hinv
   . . .
   IRIS Token Ring controller interfacename: speed Mbit
   ```

Configuring the IRIS Token Ring Driver

The configurable items for IRIS network interface drivers change from release to release and, therefore, are documented in each driver's configuration file, located in the */usr/var/sysgen/master.d* directory. Some examples of IRIS Token Ring driver configuration files are */usr/var/sysgen/master.d/if_fv* (for the *fv* drivers), */usr/var/sysgen/master.d/if_gtr* (for the *gtr* drivers), and *usr/var/sysgen/master.d/if_mtr* (for the *mtr* drivers). Configuration for all of these parameters is optional.

To determine the name of your station's IRIS Token Ring driver, refer to the *IRIS Token Ring Release Notes*, or, if the Token Ring connection is already functional, use this command:

```
% /usr/etc/netstat -i
```

Some common configurable driver parameters may include the following:

- frame size (also referred to as maximum transmission unit)
- values for timers in the logical link control, type 2 service
- location for checksum calculation

Note: Promiscuous mode (also referred to as capture-all-frames or CAF) is supported, but is not a configurable item. For more information about promiscuous mode, see "Raw Protocol Stack and Capture All Frames" in Chapter 1.

Configuring the Network Interfaces

The IRIS Token Ring network interface (or interfaces) must be configured before the Token Ring connection will function. During configuration of the Token Ring interface, the configurations of other network interfaces may need to be changed.

Note: If you do not perform any configuration tasks, the automatic network startup procedure fails during system startup and the Token Ring interface does not function. The failure occurs because the startup script (*/etc/init.d/network*) searches for, but fails to find, a default name and address entry of tr-*hostname* in the local */etc/hosts* file. The Token Ring network interface cannot be started unless there is an entry in the */etc/hosts* file. See Table 2-2 for a summary of the *network* script's assignments and the default names it expects.

Easy step-by-step configuration instructions for a basic configuration are provided below in “Quick and Easy Configuration Instructions for Token Ring” on page 48.

If your site requires special operational parameters (for example, a network mask for subnetworking) or if your station has a combination of network connections not covered by the step-by-step instructions, follow the more complete instructions outlined in “Network Interface Configuration Outline” on page 54.

Quick and Easy Configuration Instructions for Token Ring

To configure a Token Ring station with a basic configuration, select one of the configurations listed below, then follow the step-by-step instructions in the corresponding subsection. If the configuration you want is not listed, see “Network Interface Configuration Outline” on page 54.

- Token Ring as the second interface with Ethernet as primary
  (This is the required configuration for a diskless station or any system that boots from the network.)
- Token Ring as the primary interface with Ethernet as second
- Token Ring as the only network interface

Token Ring As the Second Interface and Ethernet As the Primary

The configuration described in this section contains no special configuration items, just the most basic functionality. These steps assume that the station is already connected to an Ethernet. If the station does not currently have a functional Ethernet connection, follow the instructions in the section in the *Personal System Administration Guide* or the *IRIX Admin: Networking and Mail* guide for setting up a networked station. Return to these instructions when the station’s built-in (or primary) Ethernet connection is operational.

Note: If your site uses an NIS service, the changes described in this section must *also* be made to the NIS server.

1. Open a shell window.
2. Become superuser (root):

   ```
   % /sbin/su
   Password: thepassword
   #
   ```

3. Determine your station's host name:

   ```
   # /usr/bsd/hostname
   ```

   Note: The host name should not be IRIS. (IRIS is the default name shipped on every system.) If the name is IRIS, follow the instructions in the *Personal System Administration Guide* for setting up a networked station, or follow the instructions in the *IRIX Admin: Networking and Mail* guide for assigning a host name to a system.

4. Open the */etc/hosts* file with your favorite editor. The command line below opens the file for editing with *jot*:

   ```
   # /usr/sbin/jot  /etc/hosts
   ```

5. Find the line containing your station's host name.

   Note: If the file does not contain a line for your host name, follow the instructions in the *Personal System Administration Guide* for setting up a networked station or the instructions in the *IRIX Admin: Networking and Mail* guide for configuring a network's */etc/hosts* file.

6. Copy the line and place the copy immediately below the original.
7. On this new line, change the *IP address* (all the numbers on the left) to the IP address that has been assigned to the Token Ring connection on this station. This line configures the Token Ring network interface.

   Note: The IP address must be in *dotted decimal notation* (for example, 191.51.69.237).

8. Also on the new line, change each instance of the host name to tr-*hostname*.

   For example, the lines for a station with a host name of *mickey* residing in a domain of *disney.com*, would look like this:

   ```
   x.x.x.x   mickey.disney.com   mickey  #Ether primary
   x.x.x.x   tr-mickey.disney.com   tr-mickey  #TokenRing sec
   ```

   where each *x* represents one to three decimal digits, and the text after the pound (#) sign is your comments.

9. Save and close the file.
10. You are now ready to install the Token Ring board. Follow the instructions in the board's installation guide or installation instructions.

    If the board is already installed, use these commands to finish the configuration task:

    ```
    # /etc/init.d/autoconfig
    Automatically rebuild the operating system (y/n)? y
    # /etc/reboot
    ```

Token Ring As the Primary Interface and Ethernet As the Second

The configuration described in this section contains no special configuration items, just the most basic functionality. These steps assume that the station is already connected to an Ethernet. If the station does not currently have a functional Ethernet connection, follow the instructions in the section in the *Personal System Administration Guide* or the *IRIX Admin: Networking and Mail* guide on setting up a networked station.

Note: If your site uses an NIS service, some of the changes described in this section must *also* be made to the NIS server.

1. Open a shell window.
2. Become superuser (root):

    ```
    % /sbin/su
    Password: thepassword
    #
    ```

3. Determine your station's host name:

    ```
    # /usr/bsd/hostname
    ```

    Note: The host name should not be IRIS. (IRIS is the default name shipped on every system.) If the name is IRIS, follow the instructions in the *Personal System Administration Guide* for setting up a networked station or the instructions in the *IRIX Admin: Networking and Mail* guide for assigning a host name to a system.

4. Open the */etc/hosts* file with your favorite editor. The command line below opens the file for editing with *jot*:

    ```
    # /usr/sbin/jot  /etc/hosts
    ```

5. Find the line containing your station's host name.

   Note: If the file does not contain a line for your host name, follow the instructions in the *Personal System Administration Guide* for setting up a networked station or the instructions in the *IRIX Admin: Networking and Mail* guide for configuring a network's */etc/hosts* file.

6. Copy the line and place the copy immediately below the original.

7. Return to the original line and change the address (numbers on the left) to the *IP address* that has been assigned to the Token Ring connection on this station. This line configures the Token Ring network interface.

   Note: The IP address must be in *dotted decimal notation* (for example, 223.4.69.7).

8. Go to the new line and change each instance of the host name to gate-*hostname*. This line now configures the Ethernet connection.

   For example, the lines for a station with a host name of *mickey* residing in a domain of *disney.com*, would look like this:

   ```
   x.x.x.x   mickey.disney.com  mickey  #TokRng primary
   x.x.x.x   gate-mickey.disney.com  gate-mickey  #Ether second
   ```

   where each *x* represents one to three decimal digits, and the text after the pound (#) sign is your comments.

9. Save and close the file.

10. Determine the name of your Ethernet interface with the command shown below. Some common examples include ec0, et0, enp0, mtr0, and fxp0.

    ```
    # /usr/etc/netstat -i
    ```

11. Determine the name of your IRIS Token Ring interface. This information is listed in the *IRIS Token Ring Release Notes*. Examples of this name include mtr0, gtr0, and fv0.

12. Open the */etc/config/netif.options* file with your favorite editor. The command line below opens the file for editing with *jot*:

    ```
    # /usr/sbin/jot  /etc/config/netif.options
    ```

13. Find the line shown below:

    ```
    : if1name=
    ```

14. Change the line as shown below. Be sure to remove the colon and the leading space.

    if1name=*TokenRinginterfacename*

15. Find the line shown below:

    ```
    : if2name=
    ```

16. Change the line as shown below. Be sure to remove the colon and the leading space.

    ```
    if2name=Ethernetinterfacename
    ```

17. Find the line shown below:

    ```
    : if2addr=gate-$HOSTNAME
    ```

18. Change the line as shown below. Be sure to remove the colon and the leading space.

    ```
    if2addr=gate-$HOSTNAME
    ```

19. Save and close the file.

20. You are now ready to install the IRIS Token Ring board. Follow the instructions in the board's installation guide or installation instructions.

    If the board is already installed, use these commands to finish the configuration task:

    ```
    # /etc/init.d/autoconfig
    Automatically rebuild the operating system (y/n)? y
    # /etc/reboot
    ```

Token Ring as the Only Network Interface

The configuration described in this section contains no special configuration items, just the most basic functionality for a station with one network connection.

Note: If your site uses an NIS service, some of the changes described in this section must *also* be made to the NIS server.

1. Open a shell window.

2. Log on as superuser:

    ```
    % /sbin/su
    Password: thepassword
    #
    ```

3. Determine your station's host name:

   # **/usr/bsd/hostname**

   Note: The host name should not be IRIS. (IRIS is the default name shipped on every system.) If the name is IRIS, follow the instructions in the *Personal System Administration Guide* for setting up a networked station or the instructions in the *IRIX Admin: Networking and Mail* guide for assigning a host name to a system.

4. Open the */etc/hosts* file with your favorite editor. The command line below opens the file for editing with *jot*:

   # **/usr/sbin/jot /etc/hosts**

5. Find the line containing your station's host name.

   Note: If the file does not contain a line for your host name, follow the instructions in the *Personal System Administration Guide* for setting up a networked station or the instructions in the *IRIX Admin: Networking and Mail* guide for configuring a network's */etc/hosts* file.

6. Change the address (all the numbers on the left) to the *IP address* that has been assigned to the Token Ring connection on this station. This line configures the Token Ring network interface.

   Note: The IP address must be in *dotted decimal notation* (for example, 223.4.69.7). The name used must be the station's host name.

   For example, the line for a station with a host name of *mickey* residing in a domain of *disney.com*, would look like this:

   x.x.x.x `mickey.disney.com mickey #TokRng primary`

   where each *x* represents one to three decimal digits, and the text after the # sign is your comments.

7. Save and close the file.

8. Determine the name of your IRIS Token Ring interface. This information is listed in the *IRIS Token Ring Release Notes*. Examples include mtr0, gtr0, and fv0.

9. Open the */etc/config/netif.options* file with your favorite editor. The command line below opens the file for editing with *jot*:

   # **/usr/sbin/jot /etc/config/netif.options**

10. Find the line shown below:

    `: if1name=`

11. Change the line as shown below. Be sure to remove the colon and the leading space.

    `if1name=`*`TokenRinginterfacename`*

    For example:

    `if1name=gtr0`

12. Save and close the file.
13. If the system has a built-in Ethernet interface, follow the instructions in "Network Interface Configuration Outline" on page 54 to prevent the Ethernet from being automatically started with each reboot. Then continue to the next step below.
14. You are now ready to install the IRIS Token Ring board. Follow the instructions in the board's installation guide or installation instructions.

    If the board is already installed, use these commands to finish the configuration task:

    ```
    # /etc/init.d/autoconfig
    Automatically rebuild the operating system (y/n)? y
    # /etc/reboot
    ```

Network Interface Configuration Outline

This section summarizes the tasks that must be performed to configure the IRIS Token Ring network interface(s). Details on how to perform each step are provided in the *IRIX Admin: Networking and Mail* guide.

Note: Installing the IRIS Token Ring software loads new copies of the */etc/init.d/network*. The older copy is stored as */etc/init.d/network.O*. A new */etc/config/netif.options.N* file is also installed, for your convenience.

The network startup script (*/etc/init.d/network*) runs each time the system is rebooted. It attempts to configure all drivers for which software is installed and a hardware device is found. If you add the names, indicated in the Names column of Table 2-2, to the station's */etc/hosts* file, the script configures the network connections in the order shown in the table. The script does not handle other configurations unless the */etc/config/netif.options* file is edited, as explained in the instructions that follow.

Note: The *network* script handles only one Token Ring board. If your system has two or more Token Ring boards, you must edit the *netif.options* file as explained below.

Table 2-2 Default Configurations Handled by *network* Startup Script

Station's Network Connections	Name searched for in */etc/hosts* file
Primary (only one is selected)[a]	
FDDI	*hostname*
Built-in Ethernet	*hostname*
Add-on Ethernet	*hostname*
Second (only one is selected):	
Built-in Ethernet	gate-*hostname*
Add-on Ethernet	gate-*hostname*
Token Ring	tr-*hostname*
Third (only one is selected):	
Add-on Ethernet	gate1-*hostname*
Fourth (only one is selected):	
Add-on Ethernet	gate2-*hostname*

a. At startup, the *network* script locates all the network interfaces (driver software) that are installed. The script then looks for each interface listed in this table in the order shown. As soon as it locates and assigns a primary interface, it stops looking for the interfaces listed under primary and starts looking for those listed under second. This same procedure is followed for second, third, and fourth.

1. Add an entry (name and IP address) to the local */etc/hosts* file for each network connection. (Existing network connections may already have entries. Verify that these are correct.)

   Note: For each network interface, the startup script (*/etc/init.d/network*) searches for a name in the format shown in Table 2-2. If you do not want to use these default name formats, you must do step 3 to tell the script which names to use.

2. If the site uses an NIS service, add to the NIS hosts database the same entries described in step 1.

3. If the station has any combination of network connections not listed in Table 2-2, or if you want the ordering or naming to differ from the configurations shown in Table 2-2, edit the */etc/config/netif.options* file to specify the ordering and naming of the network interfaces.

   Note: If the *netif.options* file is not edited, the startup script configures a single Token Ring interface as the second interface on the system, as summarized in Table 2-2.

4. For any network connection that uses site-specific non-default operational parameters (including netmasks, broadcast addresses, route metrics, ARP usage, and unused built-in Ethernet connections), edit the */etc/config/ifconfig-#.options* file (where the pound sign represents the network interface's order).

   The default settings for operational parameters are listed in Table 2-3.

Table 2-3 Default Network Interface Configuration: Operational Parameters

Operational Parameter	Default Setting
Netmask (*/etc/config/ifconfig-#.options* file)	none
Broadcast address (*/etc/config/ifconfig-#.options* file)	interface's IP address with host portion set to binary 1s (255 decimal for each byte of the host portion)
ARP (*/etc/config/ifconfig-#.options* file)	on
Routing metric (*/etc/config/ifconfig-#.options* file)	0
Routing (*/etc/config/routed* file)	on
Routing behavior (*/etc/config/routed.options* file)	makes routing information public (-s option)
IP Forwarding (in the operating system, configured by */usr/var/sysgen/master.d/bsd* file)	on
Built-in Ethernet is automatically started with each restart (*/etc/config/ifconfig-#.options* file)	up (even when file does not exist)

5. If non-default routing behavior is desired, edit the following files to configure the routing behavior between the networks: */etc/config/routed*, */etc/config/routed.options*, and / or */usr/var/sysgen/master.d/bsd*. (The man pages for *routed* and *chkconfig* and the comments in the *bsd* file provide information on these settings.)
6. Install the IRIS Token Ring board. Follow the instructions in the board's installation guide.

   If the board is already installed, use these commands to finish the configuration task.

   ```
   # /etc/init.d/autoconfig
   Automatically rebuild the operating system (y/n)? y
   # /etc/reboot
   ```

Configuring the Protocol Stack

The services of the IRIS Token Ring network connection are used by the higher layers of the protocol stack, as illustrated in Figure 1-6. Common communication protocols on Token Ring LANs include the Internet Protocol (IP or TCP / IP) suite and the Systems Network Architecture (SNA) protocol suite. This section discusses configuration issues for the IP and SNA protocol stacks.

Note: For configuration information about NetBIOS, SPX / IPX™ (NetWare), or another protocol, refer to that product's documentation.

Configuring IP over Token Ring

The IP protocol stack (also referred to as TCP / IP) is the Silicon Graphics system's default and required communications protocol. Proper configuration of the IP protocol is required for the system to operate. (For example, the IRIS Token Ring driver does not start running if the IP configuration fails.) When a Silicon Graphics system has the SNA protocol stack configured, it must also have the IP stack configured.

Note: Configuring the IP protocol stack does not mean that an IBM emulation application will activate a "TCP link type." The "link type" identifies the protocol spoken by the destination system (port). All IBM emulation products or IRIS SNA applications for Silicon Graphics systems (regardless of the configured "link type") require configuration of the local IP protocol stack.

To set up the station's IP over Token Ring protocol stack, follow the instructions in "Configuring the Network Interfaces" on page 47.

Configuring SNA over Token Ring

This section discusses how the Systems Network Architecture (SNA) protocol fits into the configuration of an IRIS Token Ring station.

Note: You need to configure the SNA protocol stack only when an installed application requires the SNA protocol stack (for example, when the IRIS SNA Server product is installed). Most IBM emulation products developed for Silicon Graphics systems do *not* require an SNA protocol stack on the local system.

SNA is IBM's preferred protocol and is used by many IBM products to communicate among each other. TCP/IP is Silicon Graphics' (and UNIX') preferred protocol. An IBM SNA product uses the SNA protocol to communicate with other products and systems, while a Silicon Graphics' IBM emulation product uses TCP/IP (not SNA). When an IBM emulation product running on a Silicon Graphics system needs to communicate with a system that speaks SNA (for example, an IBM host), the IRIS product uses TCP/IP to request the services of a translator (that is, an SNA server/gateway). The SNA gateway must speak both TCP/IP and SNA; therefore, configuration of the SNA protocol stack is required when the IRIS SNA Server product is installed, but is not required for most IBM emulation products.

IBM emulation products for Silicon Graphics systems (referred to as IRIS SNA applications) use the TCP/IP protocols to transfer their data and depend on a translator to handle the SNA protocol issues. IRIS SNA applications running on an IRIS Token Ring workstation use the BSD UNIX socket interface and TCP/IP protocol stack to access a translator (server), which then handles both the SNA protocol and the procedure for accessing the IBM mainframe. The translator can be the IRIS SNA Server product (running on the local system or on a remote system) or an IBM system, as illustrated in Figure 2-4.

To configure the SNA protocol stack, follow the instructions in the *IRIS SNA Server Administrator's Guide*. Any IRIS SNA applications that depend on this server will not operate properly until the server is operational.

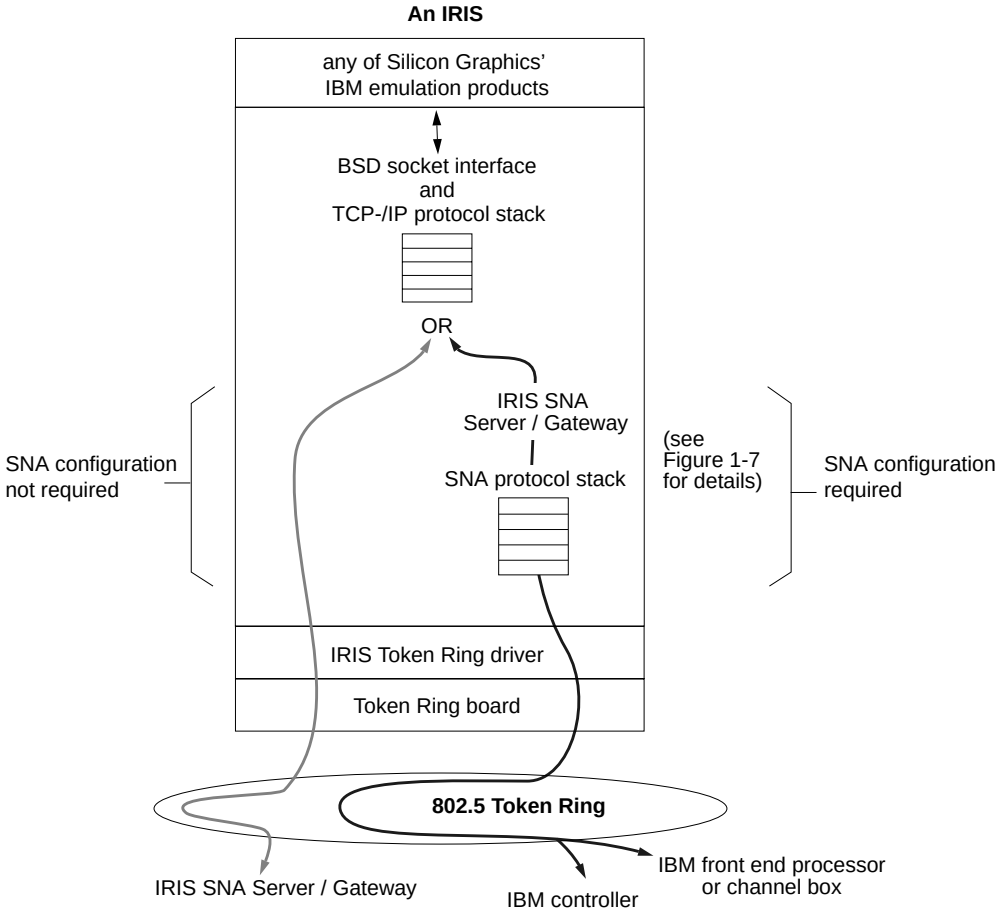


Figure 2-4 When to Configure the SNA Protocol Stack

Verifying the Network Connection

This section describes methods for verifying that the station's applications can use the Token Ring connection(s).

Verifying IP over Token Ring

To verify the Token Ring connection through the required IP protocol stack, you can use the standard IP-based software commands (for example, *ifconfig* and *ping*).

Step-by-Step Verification Instructions

Use the steps in this section to verify that a Token Ring connection is functional. This example uses the *ifconfig* and *ping* commands (see ifconfig(1M) and ping(1M)).

1. Verify that the Token Ring board(s) is recognized by the operating system:

   ```
   %/sbin/hinv
   ```

   If the board is installed correctly and recognized by the operating system, the *hinv* screen display should be similar to the following:

   ```
   1 36 MHZ IP12 Processor
   . . .
   IRIS Token Ring controller interfacename#: 16Mbit
   . . .
   ```

   There should be one line for each IRIS Token Ring board installed. The display indicates the name of the IRIS Token Ring driver as well as the data transmission speed the board is currently using.

   Note: If an expected board is not listed, follow the instructions in "When hinv Does Not List a Token Ring Board or Interface" on page 68.

2. Verify that the network configuration information for each Token Ring connection is correct. Use the following command for each Token Ring connection listed by the *hinv* display. For *interfacename#*, use the network interface name from the *hinv* display (for example, gtr0, fv1).

   # /usr/etc/ifconfig *interfacename#*

   When the interface is configured correctly, you see output similar to the following:

   ```
   fv0: flags=843<UP,BROADCAST,RUNNING>
    inet 223.48.50.4 netmask 0xffffff00 broadcast 223.48.50.255
   ```

   Note: If the display does not list the Token Ring interface as UP, the interface is disabled. Follow the instructions in "When Token Ring Interface Is Disabled" in Chapter 3 to enable the interface before proceeding.

3. For each Token Ring network connection, obtain the name of at least one other station.

   One method for obtaining this name is to use one of the commands below. The variable *netaddress* is the network portion of the Token Ring interface's IP address, which can be discovered with the */usr/etc/netstat -ina* command.

   If your site uses an NIS server, use this command:

   % **/usr/bin/ypcat hosts | grep** *netaddress* **> stations.***intrfacenam#*

   For sites without NIS, use this command:

   % **/sbin/grep** *netaddress* **/etc/hosts > stations.***intrfacenam#*

   For example:

   % /sbin/grep 223.48.50 /etc/hosts > stations.gtr0

   From each *stations.intrfacenam#* file, select one station (*host*).

4. Use *ping* to test communication between the newly configured station and the other stations:

   # **/usr/etc/ping -r** *host*

   When communication between the two stations is working correctly, you see output similar to the following:

```
PING tr-mickey (223.48.50.4): 56 data bytes
64 bytes from 223.48.50.4: icmp_seq=0 ttl=255 time=10 ms
64 bytes from 223.48.50.4: icmp_seq=1 ttl=255 time=0 ms
64 bytes from 223.48.50.4: icmp_seq=2 ttl=255 time=2 ms

----tr-mickey PING Statistics----

3 packets transmitted, 3 packets received, 0% packet loss
round-trip (ms) min/avg/max = 3/3/4
```

   Note: If the other station cannot be reached, select a different station from the *stations.interfacename#* file. If the *ping* still fails, follow the troubleshooting instructions in Chapter 3, “Troubleshooting and Error Messages.”

5. Use this command to display the activity on the Token Ring network connection. The number of input packets should increment steadily.

   % **/usr/etc/netstat -I** *interfacename#* **2**

When you successfully complete this procedure, the Token Ring connection is functioning and ready to provide network communication services. Successful completion of these steps also verifies the TCP/IP stack; the communication channel (protocol stack) is functional for IP-based applications (including IBM emulation products running on Silicon Graphics systems).

TCP/IP Management Tools

Following is a brief list and summary of some of the IP network management tools available. The on-line manual pages (also referred to as man pages) provide detailed descriptions and instructions on how and when to use these and other management tools. These tools reside in the */usr/etc* directory.

Note: The online manual pages that describe each of these commands can be viewed with the command:

% **/usr/bin/man** *commandname*

arp	Use this tool to display and / or modify the Internet-to-Ethernet (IP-to-MAC) address translation tables. The address translation tables are used by the ARP program and are based on the address resolution protocol (ARP).
ifconfig	Use this utility to assign (or display) the IP network address to a network interface and to configure (or display) network interface parameters.
netstat	Use this utility to display the status and various network data structures for network connections.
netstat -i	Use this command to list all the network interfaces currently configured on this system.
netstat -ina	Use this command to list the IP and MAC addresses (among other information) for all the network interfaces currently configured on this system.
ping	Use this tool for network testing, measurement, and management. Using ICMP request and response datagrams, *ping* sends round-trip packets between nodes on a network. It gathers and displays statistics regarding number of packets transmitted, number of packets displayed, percentage of packet loss, and round-trip time.
route	Use this tool to manually manipulate the network routing tables.
trconfig	Use this tool to configure Token Ring network interface parameters.
mtrconfig	Use this tool to configure PCI Token Ring network interface parameters.

Verifying SNA over Token Ring

To verify that the SNA protocol stack is able to communicate through the Token Ring network connection, set up a link as described in the *IRIS SNA Server Administrator's Guide*.

Chapter 3

Troubleshooting and Error Messages

This chapter contains advice for troubleshooting, solutions for commonly encountered symptoms, and descriptions of error messages.

General Advice

When you experience difficulty with the Token Ring network connection at a particular station, do the following:

1. Check the physical connections at the station as detailed in “Checking Physical Connections” on page 66.
2. If the Token Ring network connection has never been functional (for example, you have just installed the product), use */sbin/hinv* and */usr/etc/netstat -ina* to check the following configuration parameters:
   - IRIS Token Ring controller must be listed by *hinv*.
   - Transmission speed shown by *hinv* must match the attached ring’s speed.
   - IRIS Token Ring network interface (for example, mtr0, fv0, or gtr0) must be listed by *netstat -ina*.
   - IP address shown by *netstat* must be correct for the attached local area network.
   - Token Ring network interface must be enabled (for example, the *netstat* listing does not have an asterisk next to it).

   If any of these parameters are incorrect, follow the installation and configuration instructions to correct the problem.

3. If the symptom is one of those covered in "Symptoms" on page 68, follow the instructions provided.
4. Search or read the */usr/var/adm/SYSLOG* file and console window (or startup screen) for error messages. If you find any IRIS Token Ring driver messages, read about them in "Error Messages" on page 72.

   Note: Messages from an IRIS Token Ring driver contain the name of the driver (for example, fv0, gtr0, fv1, mtr0).
5. Ask your network administrator to troubleshoot the ring by following your site's fault-isolation procedure.

Checking Physical Connections

To check the physical connection between the IRIS and the Token Ring network, perform these steps:

1. At the I/O panel for the IRIS Token Ring board, verify that the *adapter cable*'s connector is seated firmly into the board's connector.

   Caution: For I/O panels that have two connectors (RJ-45 and DB-9), only one of the connectors must be used. The Token Ring board for the IRIS Indigo system is a board with this design. Connecting both connectors can cause irreparable damage to the board.
2. Follow the adapter cable from the Token Ring board's I/O panel to the cable's other end. Verify that the connector at this end is firmly seated into the connection. The connection at this end may be to a faceplate in the wall or to a *trunk coupling unit* (for example, a MAU).

   Note: Physical connections past this point are handled by your site's fault-isolation or problem determination procedures.
3. If you found any of the connections unplugged or loose, or if you unplugged them during the verification process, shutdown the system, turn off the power, then restart the system.

   Note: A loose connection causes the IRIS Token Ring board to fail when it attempts to insert itself onto the ring. When the driver notices these failures, it disables the Token Ring interface and the board. If the board is disabled during startup, the network interface is not configured. To reenable the board, you must turn off the power and restart the system; a shut down or reboot does not work.

Monitoring Token Ring Performance

The NetVisualyzer product is useful for collecting and displaying Token Ring statistics. NetVisualyzer collects and graphically displays statistics about all data on the ring.

Silicon Graphics' Token Ring implementation supports promiscuous mode or capture-all-frames (CAF) for use with NetSnoop, one of the tools available with NetVisualyzer. Promiscuous mode is enabled automatically whenever NetVisualyzer is started.

NetSnoop is useful in diagnosing network problems; it copies all frames on the physical medium (cable), regardless of destination, and then analyzes and displays the contents. This information can be used to pinpoint an overloaded gateway, monitor packet errors, identify remote users, collect network statistics, and so on. This is useful for network planning and performance monitoring.

Note: The SNA protocol stack is incompatible with promiscuous mode. For example, both NetSnoop and the IRIS SNA Server can be installed onto a system running the IRIS Token Ring driver and each product can be used, but not at the same time. While NetSnoop is running, the IRIS SNA Server application is not operational. See your *IRIS Token Ring Release Notes* for details.

For more information about NetVisualyzer and NetSnoop, see the *NetVisualyzer User's Guide* or contact your local Silicon Graphics sales representative.

Symptoms

This section describes some common problems and provides step-by-step instructions for resolving them.

When hinv Does Not List a Token Ring Board or Interface

If an expected board or network interface is not listed by */sbin/hinv*, the operating system did not recognize the board during the last reboot. If this occurs, follow these steps:

1. Did you install the IRIS Token Ring software? If you did, proceed.

   If not, follow the instructions in the *IRIS Token Ring Release Notes* to install the software.

2. Did you rebuild the operating system (by restarting or rebooting the system or by invoking the */etc/init.d/autoconfig* command) and then did you reboot? If you rebooted twice or invoked *autoconfig* then rebooted, proceed.

   If not, use these commands to rebuild the operating system:

   ```
   # /etc/init.d/autoconfig
   Automatically rebuild the operating system (y/n)? y
   # /etc/reboot
   ```

3. Verify that your network configuration (*/etc/hosts* file and */etc/config/netif.options* file) has been set up correctly. For example, the */etc/hosts* file must contain one line for each network connection (Token Ring, Ethernet, FDDI, and so on); the */etc/config/netif.options* file must contain uncommented lines if the network connection names and ordering do not conform to the default configuration (summarized in Table 2-2 and Table 2-3). If the configuration is correct, proceed.

   If there is a problem with the configuration, correct it, reboot the system, and retry the verification process.

4. If the configuration information is correct, it is possible that the missing Token Ring board is not seated firmly into its connection to the system. Follow the instructions in the board's installation guide to reinstall the board. Take extra precautions in seating the board into its connector.

5. If you have done all of the above, and the missing Token Ring board still does not appear, it is possible that the board is defective. Contact Silicon Graphics' Technical Assistance Center.

When Token Ring Interface Is Disabled

Follow the instructions in this section to enable (bring up) the IRIS Token Ring network interface.

1. Use *ifconfig* or *netstat* to verify that the Token Ring network interface is really disabled:

   # **/usr/etc/ifconfig** *interfacename*

   Note: If the display shows an `UP` flag, the interface is enabled and there is no problem. If the display does not list `UP`, the interface is disabled.

   # **/usr/etc/netstat -in**

   Note: If the display shows an asterisk next to the Token Ring interface (for example, `gtr0*`), the interface is disabled.

2. If the Token Ring network interface is enabled, but the network connection is not functional, follow the instructions in "When Token Ring Connection Is Not Responsive" on page 71. Otherwise proceed to the next step.

3. If the Token Ring network interface is disabled, use the command below to verify that the configured data transmission speed matches the ring's speed:

   # **/sbin/hinv**

   The speed indicated for the IRIS Token Ring board must match the speed at which your Token Ring local area network operates. If the speed indicated in the listing is not correct, follow the instructions in "Configuring the Board's Data Transmission Speed" on page 45, to make the board's speed match the ring's.

   If the speed is correct, proceed to the next step.

4. Try enabling the Token Ring network interface with this command:

   # **/usr/etc/ifconfig** *interfacename* **up**

5. Repeat step 1. If the interface is still disabled, proceed to the next step.

6. Use the command below to display the configuration:

   # **/usr/etc/netstat -in**

7. Is the value shown in the `Address` column zero (`0.0.0.0`)? If so, follow the instructions in "Checking Physical Connections" on page 66, to resolve the problem. If the problem persists after verifying the physical connections, continue to the next step.

8. Is the value shown in the `Address` column correct for the Token Ring local area network? If not, follow the instructions in "Configuring the Network Interfaces" on page 47, to correct the incorrect configuration.

   Configuration areas that commonly cause problems include the following:

   - The */etc/hosts* file does not contain the correct address. For example, if the word none appears in the Address column, the */etc/init.d/network* startup script did not find an entry in the */etc/hosts* file for the name it was expecting (for example, tr-*hostname*).
   - The lines in the edited (non-default) */etc/config/netif.options* file are not correct (for example, they contain extra spaces or colons or network connection names that do not exist in the */etc/hosts* file or incorrect network interface names).

When Station Does Not Load Miniroot or Boot from the Network

Silicon Graphics computers are capable of loading (installing) a small-sized version of the operating system (the miniroot) and booting themselves over the network; however, they do this only over Ethernet local area networks. They cannot boot over Token Ring networks. Since Silicon Graphics systems always use their primary network interface for booting over the network, the miniroot can be loaded only when the primary network interface is an Ethernet connection.

If your system is unable to load the miniroot (or boot over the network), verify that its primary network interface is an Ethernet connection by following these instructions:

1. Restart the system from the System Maintenance menu. Do not rebuild the operating system during this restart.
2. Log on and open a shell window.
3. Use this command to display the ordering of the network interfaces:

   ```
   % /usr/etc/netstat -i
   <primary interface>
   <secondary interface>
   ...
   ```

4. If the primary interface is an Ethernet (for example, ec0, et0, enp#), the Ethernet network connection may be dysfunctional. See the *IRIX Admin: Networking and Mail* guide for information about Ethernet network connections.

   If the primary interface is not an Ethernet, proceed to the next step.

5. Configure an Ethernet connection as the primary interface, following the instructions in “Configuring the Network Interfaces” on page 47.
6. Reboot the system. When the system is up and running, it should be capable of loading the miniroot over the Ethernet network and booting from it.

When Token Ring Connection Is Not Responsive

When the IRIS Token Ring connection is not servicing requests for its services, it is possible that the board has transitioned into a state intended to prevent dysfunctional equipment from disrupting the ring with infinite insertion attempts. This state can be caused by a broken cable between the trunk coupling unit and the computer or by disconnecting the adapter cable from either the wallplate or the computer’s I/O panel. Simply reconnecting the cable while the computer is powered up does not change the board’s state.

Note: Once the computer is started, do not disconnect the adapter cable from the computer’s I/O panel or the wallplate. This action causes the IRIS Token Ring adapter board to transition into a non-responsive state.

The following characteristics describe this non-responsive state:

- The Token Ring network connection fails to pass the *ping* verification test described in “Verifying IP over Token Ring” on page 60.
- The */usr/etc/netstat -i* command lists the IRIS Token Ring interface as enabled.
- The */usr/etc/ifconfig* command lists the IRIS Token Ring interface as UP.
- Invoking the */usr/etc/ifconfig* command to stop and then start the Token Ring driver does not fix the problem.
- The */sbin/hinv* command lists the IRIS Token Ring board, and the displayed transmission speed matches the speed currently being used on the attached ring.
- Invoking the */etc/shutdown* command, then restarting the system does not remedy the problem.
- Shutting down the system, turning off the power, then turning on the power, and restarting the system fixes the problem.

Take the following steps to remedy an unresponsive network interface:

1. Use the command below to stop and then start the Token Ring network interface:

   # **`/usr/etc/ifconfig`** *interfacename#* **`down up`**

2. Try to use the network connection. One method for doing this is described in "Verifying IP over Token Ring" on page 60.

3. If this does not fix the problem, use the following command to shut down the system:

   # **`/etc/shutdown`**

4. When you are informed that it is safe to do so, turn off the power switch, count to 10, then turn on the power switch.

5. Restart the operating system.

6. Try to use the network connection. If the problem is not resolved, troubleshoot the Token Ring adapter board as explained in the board's installation instructions.

Error Messages

This section contains an alphabetical list of all error messages specific to the IRIS Token Ring drivers. Each entry has a discussion of the problems the message may indicate and suggested actions.

Note: The list of error messages in this section covers only those unique to the IRIS Token Ring drivers. Standard system error messages, even when caused by the driver code, are not covered in this section.

Messages are organized according to the following rules:

- Messages are alphabetized. (Figure 3-1 illustrates the text of an error message.)
- Nonletters (for example, - or %) and blank spaces are shown in the text of the message, but are ignored in alphabetization. For example, the message `tr_open` appears between `trnet and tp`.
- When an error message includes a variable that the software specifies differently (fills in) for each instance of the message, the variable is given with a generic name (for example, *filename*) and represented in italic font. Generic names such as *filename* or *hostname* are not alphabetized; the variable is skipped and the message is alphabetized using the next word. For example, the error message `goofy not responding` is listed as *hostname* `not responding`, and it is alphabetized among the "n" listings; *hostname* is skipped and the word "not" is used for alphabetizing the message. Common names for variables in this listing include *#*, *digit*, and *hexnumeral*.

  Note: If you cannot find an error message in the alphabetical listing, identify potential variables, then look up the message without those words.
- Capitalization is not considered in alphabetization.
- The creator of each message is listed, in angled brackets, below the text of the message: (<*creator*>).

Driver error messages and IRIS Token Ring commands are written into the file */usr/var/adm/SYSLOG* or displayed at the terminal during startup or the Console during normal operation; some messages appear in more than one location. Within the *SYSLOG* file, each message is preceded by the date, time, host name, the name of the process that created the message, and its process ID number, as illustrated below. Only the text of the error message is included in the alphabetic list that follows.

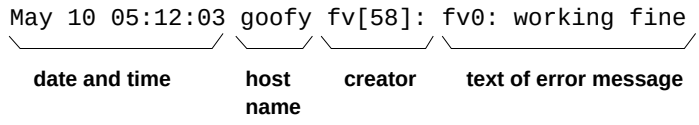


Figure 3-1 Error Message Format in */usr/var/adm/SYSLOG* File

The fv Driver Error Messages

This section describes error messages displayed by the *fv* driver.

`fv#: Address verification phase: <`*message from Table 3-1*`>`

During startup, a failure occurred when the IRIS Token Ring board was attempting to verify the uniqueness of its MAC address. The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-1.

Table 3-1 *fv* Startup Phase Failures

Text of Message	Description
Duplicate node address	This station's MAC address is being used by another station on the ring.
Function failure	The station is unable to transmit to itself when its transmit and receive lines are wrapped at the wiring concentrator.
Remove received	A "remove adapter" MAC frame was received.
Request initialization	This station was unable to obtain ring parameters from the ring's parameter server. If the ring does not have a parameter server, this message is not displayed and the system uses internal default parameters.
Ring beaconing	A beacon MAC frame was seen.
Ring failure	The station became the active monitor and began purging the ring, but subsequently did not see its own MAC frames return to it.
Signal loss	This station cannot see any signal on the physical medium at its input port.
Timeout	The station's insertion timer for this phase expired before the station could complete the startup phase.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`fv#: bad burnin address`

While attempting to start, the driver discovered that the MAC address burned into a component on the IRIS Token Ring board is invalid.

Contact Silicon Graphics' Technical Assistance Center.

`fv#: bad EDT entry`

While attempting to start, the driver encountered invalid initialization (EDT) parameters. This indicates a problem with the software.

If this message displays during an installation, reinstall the software, taking extra care to install the correct version for the installed IRIS Token Ring board; otherwise, contact Silicon Graphics' Technical Assistance Center.

`fv#: broadcast address set to 0xC000FFFFFFFF`

During startup, the driver has set the broadcast address to the value indicated. There is no problem.

You do not need to do anything; however, if you want to change this broadcast address to a different one, use the commands below:

```
% su
Password: thepassword
# /usr/etc/ifconfig fv# broadcast 0xnewaddress
```

where *newaddress* is a 6-byte value in hexadecimal format (for example, FFFFFFFFFFFF) and *fv#* is the specific *fv* network interface.

`fv#: died cmd=`*hexnumeral*`, rc=`*hexnumeral*

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The *rc* value indicates the reason for the interrupt, as described by the error bits in the Adapter Check Status register (described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.)

Contact Silicon Graphics' Technical Assistance Center.

`fv#: early interrupt`

The driver received an interrupt from the board before it was ready to process that interrupt.

When this message is intermittent, no action is needed. If the message is persistent, shutdown and restart the system. If the message continues to be displayed, contact Silicon Graphics' Technical Assistance Center.

`fv#: executing CAF`

The driver enabled the collect-all-frames (CAF) / promiscuous mode. This message does not indicate a problem.

You do not need to do anything.

`fv#: firmware failed to start`

While attempting to start, the driver was unable to start the firmware on the board. The board is dysfunctional; it may be incorrectly configured, or the board's firmware version may not match the board.

If this message displays during an initial installation, verify that the installation steps have been followed correctly; otherwise, contact Silicon Graphics' Technical Assistance Center.

`fv#: invalid buffer size`

During startup, the driver encountered an invalid buffer. This indicates an incompatibility between the driver and other IRIX software modules.

Contact Silicon Graphics' Technical Assistance Center.

`fv#: invalid list size`

During startup, the driver encountered an invalid parameter. This indicates an incompatibility between the driver and other IRIX software modules.

Contact Silicon Graphics' Technical Assistance Center.

`fv#: invalid LLC options`

During startup, the driver was passed logical link control (LLC) layer options that it does not support. This indicates an incompatibility between the driver and other IRIX software modules.

Contact Silicon Graphics' Technical Assistance Center.

`fv#: invalid node address`

During startup, the driver found its own MAC address to be invalid. This usually indicates an improper entry in the */etc/config/trconfig.options* file.

Verify the entries in the */etc/config/trconfig.options* file.

`fv#: IP open: <any of the messages below>`

```
a parameter has exceeded max limit
group has too many members
invalid options
invalid SAP value
nonexistent group
resources not available
unauthorized access priority
```

While attempting to process a service request, the driver was unable to open a communication path to the IP protocol stack because the service access point (SAP) into the IP protocol stack failed to open. The reason for the failure is indicated in the final segment of the message. All of these messages indicate incompatibilities between the driver and a user-level application; the user-level application has used invalid / unsupported parameters.

If this message occurs with a user-level application that is sold or distributed by Silicon Graphics, contact Silicon Graphics' Technical Assistance Center; otherwise, contact the vendor of the user-level product.

`fv#: Lobe media test phase: <`*message from Table 3-1*`>`

During startup, a failure occurred when the IRIS Token Ring board was performing its lobe media test. This test is used to verify the connection to the trunk coupling unit (MAU or CAU). The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-1.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`fv#: missing`

When the *fv* driver started, it did not find the IRIS Token Ring board. This indicates that the driver is built into the operating system (kernel) and starts with each reboot or restart of the system, but the board is unreachable.

If the board is not installed and you want the Token Ring connection to function, contact your Silicon Graphics' field service engineer to install the board.

If the board has been removed, this message is not a problem, although it does mean that the Token Ring connection is not functional. If you do not plan to reinstall the board, you can rebuild the operating system (at startup or with *autoconfig*) to remove this driver so that this message does not display. No harm occurs if you do not remove the driver from the operating system.

If the board is installed, it is possible that it has become loose from its connection or that it is dysfunctional. Have your Silicon Graphics' field service engineer reinstall the board, or contact Silicon Graphics' Technical Assistance Center.

`fv#: packet too small(`*digit*`)`

While receiving an incoming frame (from the board), the driver found that it was too small to be legal. The packet is thrown away automatically.

You do not need to do anything.

`fv#: Participation in ring poll phase: <`*message from Table 3-1*`>`

During startup, a failure occurred when the IRIS Token Ring board attempted to participate in the ring's poll process. During this startup phase, the board attempts to determine if it should be a standby monitor or become the active monitor. The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-1.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`fv#: Physical insertion phase: <`*message from Table 3-1*`>`

During startup, the IRIS Token Ring board's physical insertion phase failed. During this phase, the board attempts to synchronize itself with the received signal and the ring's signal clock. The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-1.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`fv#: Request Initialization phase: <`*message from Table 3-1*`>`

During startup, the IRIS Token Ring board's attempt to contact the ring's parameter server failed. The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-1.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`fv#: reset FAILED`

The driver attempted to reset (restart) the IRIS Token Ring board and failed to get a response. The board may be loose from its connections or it may be dysfunctional.

If this message displays during an initial or routine installation, reinstall the board, taking extra care to seat the board firmly into its connections; otherwise, contact Silicon Graphics' Technical Assistance Center.

`fv#: spurious interrupt =` *hexnumeral*

The driver received an unrecognized and unrecoverable interrupt from the board. The value indicated by *hexnumeral* indicates the unrecognized STATUS interrupt.

When this message is intermittent, nothing need be done. If the message is persistent, shutdown and restart the system. If the message continues to be displayed, contact Silicon Graphics' Technical Assistance Center.

`fv#: tokenring address` *hexnumerals*

The MAC address for the IRIS Token Ring board is displayed by *hexnumerals*. This message is informational; it does not indicate a problem.

You do not need to do anything.

`fv#: too few recv buffers`

During startup, the driver was unable to obtain enough memory. This indicates an incompatibility between the driver and other IRIX software modules.

Contact Silicon Graphics' Technical Assistance Center.

The gtr Driver Error Messages

Many of the *gtr* error messages indicate problems that occur during different phases of the IRIS Token Ring board startup procedure. These phases happen in the following order: power-up diagnostics (PUD), bring-up diagnostics (BUD), initialization, open, and insert.

`gtr#: ADAPTER checksum error`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was an invalid or undecipherable operation code, as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: ADAPTER DIO parity error`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was bad parity, as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: ADAPTER DMA read error`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was an abort during a direct memory access (DMA) read, as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: ADAPTER DMA underrun`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was an underrun during a direct memory access (DMA), as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: ADAPTER DMA write error`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was an abort during a direct memory access (DMA) write, as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: ADAPTER internal bus error`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was a parity error on the adapter's internal bus, as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: ADAPTER invalid error interrupt`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was an unrecognized error interrupt, as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: ADAPTER invalid interrupt`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was an unrecognized or invalid interrupt, as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: ADAPTER invalid xop`

The driver received an interrupt from the board (of type ADAPTER.CHECK), indicating that the board has experienced an unrecoverable error. The reason for the interrupt was an unrecognized XOP request, as described in Figure 4-19 and Table 4-15 of the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: bad EDT entry`

When starting, the driver found that the unit number (EDT entry) for the IRIS Token Ring board is invalid. This indicates a software problem.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: bd not responding`

When the driver attempted to issue a command to the board, the board did not respond. This indicates a hardware problem. The board may be loose from its connection or it may be dysfunctional.

First reboot the system. If the problem persists, follow the instructions in the *Token Ring for IRIS Indigo Installation Guide* to reinstall the board, taking extra care to seat the board firmly into its connections. If the message continues to display, contact Silicon Graphics' Technical Assistance Center.

`gtr#: BUD: message (`*hexnumeral*`)`

During the board's bring up diagnostic (BUD) phase, a problem was encountered with the Texas Instrument components. The cause of the problem is indicated by the text of the *message* and the error code (*hexnumeral*). The messages and error codes are documented in the section "Bring-up Diagnostics" of Texas Instruments' *TMS380 Second-Generation Token Ring User's Guide*. The IRIS Token Ring board is dysfunctional.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: CLOSE failed(`*hexnumerals*`)`

When the driver attempted to close the adapter on the IRIS Token Ring board, an error occurred, indicating that the adapter component is dysfunctional. The value displayed by *hexnumerals* corresponds to the return codes documented in the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: died and restarted(`*hexnumeral*`)`

The driver was unable to contact the IRIS Token Ring board, so it reset the board. Subsequent error messages will indicate the success or failure of the reset.

If the reset succeeds, nothing is wrong and nothing needs to be done. If the reset fails, follow the instructions for that failure message.

`gtr#: duplicate EDT entry`

When starting, the driver found that the unit number (EDT entry) for this IRIS Token Ring board is already known to the system. This indicates a software problem.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: EPROM checksum error(`*hexnumeral*`)`

During the board's power up diagnostic (PUD) phase, the data within a memory component of the IRIS Token Ring board was found to be corrupted. The board is dysfunctional.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: failure`

The IRIS Token Ring board is dysfunctional.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: INIT:` *message* `(`*hexnumerals*`)`

During initialization of the board's adapter, an error occurred, indicating that the adapter component is dysfunctional. The reason for the failure is described in the text of the *message* and the error code (*hexnumerals*). (The messages and error codes are documented in the section "Adapter Initialization" of Texas Instruments' *TMS380 Second-Generation Token Ring User's Guide*.) The IRIS Token Ring board is dysfunctional.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: init time out`

The if_init call for initializing the board, timed out. This indicates a hardware problem. The board may be loose from its connection or it may be dysfunctional.

Follow the instructions in the *Token Ring for IRIS Indigo Installation Guide* to reinstall the board, taking extra care to seat the board firmly into its connections. If the message continues to display, contact Silicon Graphics' Technical Assistance Center.

`gtr#: invalid MOD options(`*hexnumeral*`)`

When the driver attempted to open the IRIS Token Ring board, an error occurred, indicating that the adapter component is dysfunctional, is not responding, or the options requested by the MAC OPEN command are not supported. Follow the instructions in "When Token Ring Interface Is Disabled" on page 69 to troubleshoot the problem.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: missing`

When the *gtr* driver started, it did not find the IRIS Token Ring board. This indicates that the driver is built into the operating system (kernel) and starts with each reboot / restart of the system, but the board is unreachable.

Follow the instructions in one of the bullets below:

If the board is not installed and you want the Token Ring connection to function, follow the instructions in the *Token Ring for IRIS Indigo Installation Guide* to install the board.

If the board has been removed, this message is not a problem, although it does mean that the Token Ring connection is not functional. If you do not plan to reinstall the board, you can restart the system and answer **`yes`** to rebuild the operating system and remove this driver.

If the board is installed, it is possible that it has become loose from its connection or that it is dysfunctional. Follow the instructions in the *Token Ring for IRIS Indigo Installation Guide* to reinstall the board, taking extra care to seat the board firmly into its connections. If the message continues to display, contact Silicon Graphics' Technical Assistance Center.

`gtr#: no memory for dma`

When starting, the driver was unable to obtain system memory for itself. This indicates a software problem.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: no memory for frame filter`

When starting, the driver could not (or did not) allocate memory for a multicast filter table. This indicates a software problem.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: OPEN: Address verification phase` *<message from Table 3-2>*

During startup, an OPEN_ERROR occurred when the IRIS Token Ring board was attempting to insert itself into the ring and verify the uniqueness of its MAC address. The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-2.

Table 3-2 *gtr* Startup Phase Failures

Text of Message	Description
Duplicate node address	This station's MAC address is being used by another station on the ring.
Function failure	The station is unable to transmit to itself when its transmit and receive lines are wrapped at the wiring concentrator.
Remove received	A "remove adapter" MAC frame was received.
Request initialization	This station was unable to obtain ring parameters from the ring's parameter server. If the ring does not have a parameter server, this message is not displayed and the system uses internal default parameters.
Ring beaconing	A beacon MAC frame was seen.
Ring failure	This station became the active monitor and began purging the ring, but subsequently did not see its own MAC frames return to it.
Signal loss	This station cannot see any signal on the physical medium at its input port.
Timeout	This station's insertion timer for this phase expired before the station could complete the startup phase.

`gtr#: OPEN: invalid buffer size`

When the driver attempted to open the IRIS Token Ring board, an error occurred, indicating that the adapter component is dysfunctional. The failure occurred because the buffer size provided by the OPEN command is invalid.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: OPEN: invalid list size`

When the driver attempted to open the IRIS Token Ring board, an error occurred, indicating that the adapter component is dysfunctional. The failure occurred because the receive / transmit list size provided by the OPEN command is invalid.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: OPEN: invalid LLC options`

When the driver attempted to open the IRIS Token Ring board, an error occurred, indicating that the adapter component is dysfunctional. The failure occurred because the options requested by the LLC OPEN command are not supported.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: OPEN: invalid node address`

When the driver attempted to open the IRIS Token Ring board, an error occurred, indicating that the adapter component is dysfunctional. The failure occurred because the MAC address is either all zeros or unreadable.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: OPEN: Lobe media test phase` *<message from Table 3-2>*

During startup, a failure occurred when the IRIS Token Ring board was performing its lobe media test. This test verifies the connection to the trunk coupling unit (MAU or CAU). The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-2.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`gtr#: OPEN: open failed` *(hexnumerals)*

The driver cannot complete the open phase for the board. The IRIS Token Ring board is dysfunctional.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: OPEN: Participation in ring poll phase` *<msg from Table 3-2>*

During startup, an OPEN_ERROR occurred when the IRIS Token Ring board was attempting to insert itself into the ring and to participate in the ring's poll process. During this startup phase, the board attempts to determine if it should be a standby monitor or become the active monitor. The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-2.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`gtr#: OPEN: Physical insertion phase` *<message from Table 3-2>*

During startup, the IRIS Token Ring board's physical insertion phase failed. During this phase, the board attempts to synchronize itself with the received signal and the ring's signal clock. The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-2.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`gtr#: OPEN: Request Initialization phase` *<msg from Table 3-2>*

During startup, an OPEN_ERROR occurred when the IRIS Token Ring board was attempting to insert itself into the ring and to contact the ring's parameter server. The reason for the failure is provided in the final segment of the error message, as summarized in Table 3-2.

Ask your network administrator to troubleshoot the ring, using your site's fault-isolation or problem determination procedures.

`gtr#: OPEN: too few recv buffers`

When the driver attempted to open the IRIS Token Ring board, an error occurred, indicating that the adapter component is dysfunctional. The failure occurred because the request for transmit buffers provided by the OPEN command specified so many buffers that not enough were left for reception.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: packet too small`*(digit)*

When receiving a packet from the board, the driver discovered that the packet was too small to be a legal packet. The packet is discarded. This indicates that a system somewhere is generating illegal packets.

You do not need to do anything with this system; however, you may wish to use a network management tool, such as NetSnoop, to locate the faulty system.

`gtr#: read BIA failed`

During initialization, the driver was unable to read the burned-in MAC address (BIA) from the board, or the checksum validation of that address failed after the address was read. This indicates a problem with the board.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: reset failed`

The driver attempted to reset (start) the IRIS Token Ring board and failed to get a response. The board may be loose from its connections or it may be dysfunctional.

If this message displays during an installation, immediately after a hardware maintenance procedure has been performed, or after the computer has been moved, reinstall the board, taking extra care to seat the board firmly into its connections; otherwise, contact Silicon Graphics' Technical Assistance Center.

`gtr#: RING INSERT failed`

During initialization, the board failed to insert itself onto the ring. This indicates a problem with the ring or with the board.

First, shutdown the system, turn off the power, then restart the system. If the message displays again, have your network administrator verify that the ring is functional. If the problem persists, contact Silicon Graphics' Technical Assistance Center.

`gtr#: RUN: died(`*hexnumeral*`)`

The driver cannot insert the board into the ring. The IRIS Token Ring board is dysfunctional.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: set address failed`

During initialization, the driver was unable to set the functional MAC address. This indicates a problem with the locally assigned MAC address value controlled by the trconfig.options file.

Locate and correct the invalid entry in the trconfig.options file. Instructions for editing this file are provided in "Updating the MAC Address" on page 41.

`gtr#: tokenring address` *hexnumerals*

The MAC address being used for the IRIS Token Ring board (either the burned-in universally assigned one or the locally assigned one) is displayed by *hexnumerals*. This message is informational; it does not indicate a problem.

You do not need to do anything.

`gtr#: tokenring address set to` *hexnumerals*

The MAC address for the IRIS Token Ring board is displayed by *hexnumerals*. This message is informational; it does not indicate a problem.

You do not need to do anything.

`gtr#: Unknown error` *(hexnumeral)*

The driver received an interrupt from the board, indicating that the board has experienced an unrecoverable and unrecognized error. The value displayed by *hexnumerals* indicates which bits of the Status Register (documented in the Texas Instrument, *TMS380 Second-Generation Token Ring User's Guide*) were set at the time of the error. The IRIS Token Ring board is dysfunctional.

Contact Silicon Graphics' Technical Assistance Center.

`gtr#: present`

The *gtr* driver has located its IRIS Token Ring board. This message is informational; it does not indicate a problem.

You do not need to do anything.

The mtr Driver Error Messages

The error messages for the *mtr* driver fall into two categories: the Alert and Warning messages and the Notice messages.

Alert and Warning messages occur due to a failure in the adapter, such as when it cannot initialize or function. Alert and Warning messages cannot be corrected without help from Silicon Graphics Technical Support. If you receive one of these messages, call the Silicon Graphics Technical Assistance Center.

Notice messages are not serious, but inform you of a condition that may need to be monitored. The adapter is initialized and working, but a problem has occurred that you should be aware of.

Alert and Warning Messages

This section lists the Alert and Warning messages for the PCI Token Ring Board. Should you receive any of these messages, call the Technical Assistance Center and report the message.

`mtr#: again for (`*hexnumerals*`): (`*hexnumerals*`)!`

The driver has detected that the board has already been initialized.

```
mtr#: attach(): kmem_zalloc (#) !
mtr#: attach(): kmem_zalloc (#) 1!
mtr#: attach(): kmem_zalloc (#) 2!
mtr#: attach(): dev_desc!
```

The driver failed to allocate enough memory for its internal structure.

`mtr#: bad burnin address: `*hexnumerals*`!`

The driver has detected an incorrect burn-in address (BIA) on the board.

`mtr# BUD: #, `*hexnumerals!*

The PCI Token Ring board failed during bring up diagnostics (BUD).

```
mtr# DATA_32 no match #.#:
```

The driver could not correctly download the board firmware.

```
mtr_detach (hexnumerals) not found!
mtr_detach (hexnumerals): (hexnumerals) not found!
```

A driver or system internal error has occurred.

```
mtr#: dmamap_addr(#) for RX!
mtr#: dmamap_addr(#) for TX!
```

The driver has failed to allocate resources for the board.

```
mtr#: dmamap_alloc(#) for SSB & SCB!
```

The driver has failed to allocate resources for testing the bus master.

```
mtr#: dmamap_alloc(#) for TX & RX!
```

The driver has failed to allocate resources for the board.

`mtr#: driver_prepare_adapter() kmem_zalloc(#) init_block!`

The driver failed to allocate # bytes for the initialization block.

`mtr#: fmplus_buffer_size (#) < FMPLUS_MIN_TXRX_BUFF!`
`mtr#: fmplus_buffer_size: # < #!`

An internal error has occurred within the driver that prevents it from correctly preparing or initializing the PCI Token Ring board.

`mtr#: hwi_at24_wait_ack(): no ack:` *hexnumerals* `(`*hexnumerals*`)!`

The PCI Token Ring board has failed to respond to the driver's request.

`mtr#: hwi_download_code(): # != #!`
`mtr#: hwi_download_code():` *hexnumerals* `!= 0!`

The driver detected an error in the PCI Token Ring board firmware.

`mtr#: IFF_PROMISC but driver_prepare_adapter: #!`
`mtr#: IFF_PROMISC but driver_start_adapter: #!`

The driver did not enable or disable the Snoop Protocol when restarting the PCI Token Ring board.

`mtr#: IFF_UP but driver_prepare_adapter: #!`

The driver did not prepare the PCI Token Ring board to restart correctly.

`mtr#: IFF_UP but driver_start_adapter: #!`

The driver did not restart the PCI Token Ring board.

`mtr*: if_mtrstart(): no recognized adapter!`

The *mtr* driver does not detect a PCI Token Ring board in the system.

`mtr#: INIT failed: (`*hexnumerals*`).`

The PCI Token Ring board failed during initialization.

`mtr#: kmem_zalloc(#) for SSB & SCB!`

The driver has failed to allocate resources for testing the bus master.

`mtr#: kvpalloc(#) for TX & RX!`

The driver has failed to allocate resources for the board.

`mtr#: mtr_output (`*hexnumerals*`)!`
`mtr#: mtr_watchdog (#)!`

A driver or system internal error has occurred.

`mtr#: no memory or io base register!`

The driver failed to obtain the board's PCI base register.

`mtr#: not support 4.31:` *hexnumerals*`!`
`mtr#: not support bus master:` *hexnumerals*`!`

The driver has detected the wrong version of the PCI Token Ring board.

`mtr#: pciio_dmamap_addr(`*hexnumerals*`, `*hexnumerals*`, #) for SCB!`
`mtr#: pciio_dmamap_addr(#) for SSB`

The driver has failed to allocate resources for testing the bus master.

`mtr#: pciio_intr_alloc (#) failed!`
`mtr#: pciio_intr_connect () failed!`

The driver failed to allocate or initialize the structure for handling an interrupt.

`mtr#: pciio_piomap_alloc (#)`

The driver failed to obtain the address to access the PCI Token Ring board.

`mtr#: POLLING_SIFINT: (`*hexnumerals*`)!`

The driver failed to receive the correct status from the PCI Token Ring board.

`mtr#: possible lockup: #, (`*hexnumerals*`)!`

A bus master operation has not completed correctly.

`mtr#: RX slot idx: # >= #!`

An internal error has occurred within the driver that prevents it from correctly preparing or initializing the PCI Token Ring board.

`mtr#: scb_buff[#]: (`*hexnumerals*`) != scb_test_pattern[#] (`*hexnumerals*`).`
`mtr#: ssb_buff[#]: (`*hexnumerals*`) != ssb_test_pattern[#] (`*hexnumerals*`).`

The driver could not correctly perform the bus master test.

`mtr#: SIFINT_ADAPTER_CHECK (`*hexnumerals*`)!`

The PCI Token Ring board reports an unrecoverable error.

`mtr#: SIOC...MULTI: srb_used: (`*hexnumerals*`).`

The driver has an internal error that prevents the PCI Token Ring board from being correctly set up for IP multicast.

`mtr#: too many adapters -> ignored!`

Your system cannot support the number of PCI Token Ring boards you have installed. A maximum of eight PCI Token Ring boards are supported.

`mtr#: TX slot idx: # >= #!`

An internal error has occurred within the driver that prevents it from correctly preparing or initializing the PCI Token Ring board.

Notice Messages

The following Notice messages provide information about an error condition.

`mtr#: enter PROMISC mode.`
`mtr#: exit PROMISC mode.`

Promiscuous mode has been enabled or disabled in the driver.

`mtr#: mtr_intr(): sifsts(`*hexnumerals*`) `*hexnumerals*`!`

The driver has detected a null interrupt from the PCI Token Ring board. This may be due to heavy traffic on the ring.

`mtr#: mtr_output(): sa_family (`*hexnumerals*`) not supported.`

The driver has received a packet from an unsupported upper-layer protocol.

`mtr#: mtr_snoop_input: m_vget(#).`

The driver could not allocate the memory that is necessary to enable snooping. This may be due to heavy traffic.

`mtr#: open_address.byte[#]:` *hexnumerals* `!= mtr_enaddr[#]` *hexnumerals*`!`

The node address (or open address) was not correctly set. This occurs when the MAC address has not been correctly configured. Repeat the procedure to configure the MAC address. See “Configuring the Board’s Physical (MAC) Address” on page 40.

`mtr#: open error(#,` *hexnumerals*`.`*hexnumerals*`)`

The PCI Token Ring board could not be accessed. This could be due to a broken cable connection. Check cable connections between the board and the MAU and then restart the board.

`mtr#: read bad frame:` *hexnumerals*

The PCI Token Ring board has received a bad frame or the receiving operation itself has failed.

`mtr#: ring_speed:` *hexnumerals* `but mtr_s16Mb (`*hexnumerals*`)!`

The ring speed and the ring speed configuration for the board are not the same. Reconfigure the board with the correct ring speed. See “Configuring the Board’s Data Transmission Speed” on page 45.

`mtr# RX: m_get(#)!`
`mtr# RX: m_vget(#).`

The driver has failed to allocate memory for the received packet. The packet has been discarded.

`mtr#: RX rx_len: #.#:# [#..#]!`

The length of the packet just received is incorrect. The packet has been discarded.

`mtr#: SIFINT_ARB (`*hexnumerals*`): arb:` *hexnumerals*`.`*hexnumerals* `open:` *hexnumerals*`.`*hexnumerals*

The driver has reported a change in the ring status.

`mtr#: `*hexnumerals*`(SIFINT_SRB) but #!`

The driver has reported a harmless internal race condition.

`mtr#: SIFINT_SSB (`*hexnumerals.hexnumerals*`)?`

The driver has received an unexpected SSB from the PCI Token Ring board.

```
mtr#: SIOCSIFADDR: halt: #!
mtr#: SIOCSIFADDR: start: #!
```

The driver has failed to configure the new MAC address for the board. Repeat the MAC configuration process. See “Configuring the Board’s Physical (MAC) Address” on page 40.

`mtr#: SRB(`*hexnumerals*`) result: `*hexnumerals*`.`

The PCI Token Ring board failed to execute the command it received from the driver.

`mtr#: TX pkt: # > #!`

The size of an outbound packet is too large. The packet has been discarded.

Glossary

A bit

The bit in Token Ring *data frames* that is set to zero when the frame is initially transmitted and set to one by the destination station when it has seen the frame. The original transmitting station uses the setting of this bit to decide whether or not to make additional transmissions to the destination station. For example, when the A bit on a returning data frame is zero this may indicate the destination station is not *insert*ed into the ring or that it is not functional.

active monitor

The one *monitor* on the ring that is currently responsible for maintaining the ring's health. The duties of the active monitor include detecting and correcting error conditions, such as a *lost token* or a *persistently circulating* frame. The assignment of the active monitor is dynamically determined with the *token claiming* process each time the ring recovers from a fault.

active monitor present frame (AMP)

A type of *MAC frames* (management frame) used to indicate the continuing presence of an *active monitor* on the ring. Only the active monitor issues these frames and it issues one about every 7 seconds. For the active monitor, this frame communicates the station's address to the next station (that is, the frame communites the *upstream neighbor address*, UNA).

active state

A ring condition in which *data frames* have been or are being transmitted onto the ring. This state is opposed to the ring's *idle* state.

adapter

Also called a network adapter board or controller. A hardware device that is capable of communicating over a communications protocol medium; the adapter "speaks" the specific protocol (for example, 802.5 Token Ring, Ethernet, FDDI). Each station on a Token Ring must have an adapter through which it sends and receives data. The IRIS Token Ring board is an adapter. Two of the important functions performed by an adapter board are *repeater* and *medium access control sublayer*.

adapter cable

The cable that connects an *adapter* to the network. The connection to the network is usually through a wall faceplate or directly to a *trunk coupling unit* (for example, a MAU or LAM). Two types of adapter cables are available: shielded twisted pair cable ending in a DB-9 connector for the adapter and unshielded twisted pair cable ending in an RJ-45 connector for the adapter.

AMP

See active monitor present frame.

attaching device

A generic term referring to any intelligent hardware and software system that can be inserted into a Token Ring (through an *adapter* and a *trunk coupling unit*). Some examples include printers, networked fax machines, main frames, and stations (such as PCs and IRISes).

beacon frame (or beacon)

A type of MAC frame (management frame) used to indicate a *break in the ring*. Only *standby monitor*s issue these frames. The presence of this type of frame on a ring indicates two things: a station thinks it has detected a break in the ring, and the *beaconing* procedure is in progress.

beaconing

A ring recovery process that starts when a station fails to see a *token* after a set period of time while *token claiming*. (The length of this time period is defined by the 802.5 Token Ring Standard.) During the beaconing process, one or more stations transmit *beacon frame*s until the ring is again intact (that is, the *break in the ring* is fixed). All stations refrain from transmitting data during beaconing. When a beaconing station sees another station's beacon frame, it stops beaconing. When a beaconing station sees its own beacon frame, it starts *token claiming*.

break in the ring

A ring error condition where the signal cannot complete the loop from a *repeater*'s output *port* around the ring to the same repeater's input port. Whenever a station fails to see the *token* or an *active monitor present frame* for a defined period of time, a break in the ring is assumed and *beaconing* starts.

bypass state

A state in which a station is not *insert*ed onto the ring. The station is not listening, *repeating*, or transmitting when in the bypass state. If a station is attached to a *trunk coupling unit* (TCU), the TCU may bypass the station if it determines that the station is dysfunctional. This action should cause the station to transition into its bypass state. Other events that may cause a station to transition into bypass state are the discovery that its *MAC address* is already in use on the ring or reception of a *beacon* or *claim token frame* while trying to add itself to the ring.

capture

The act of receiving the *token* but not *repeating* it onto the ring. The purpose of capturing the token is to transmit data. On a Token Ring, only the station that has captured the token has permission to transmit data.

C bit

The bit in Token Ring *data frames* that is set to zero when the frame is initially transmitted and set to one by the destination station when it has copied the frame. The original transmitting station uses the setting of this bit to decide whether or not to retransmit the data within that frame. For example, whenever the C bit in a returning frame is still zero, the destination station has not been able to copy the frame, so the data has not been received (even though the frame may have been seen).

claim

As a verb, see *capture*. As a noun, see *claim token frame*.

claim token frame (or claim)

A type of MAC frame (management frame) used to manage the *token claiming* process. Any *standby monitor* may use this frame.

CAU

See controlled access unit.

concentrator

A hardware device that acts as a *trunk coupling unit* and that allows multiple *station*s to attach to it so that they can participate in the ring without being directly attached to the *trunk* ring. *Multistation access unit*s and *lobe attachment module*s are examples of concentrators.

configuration report server

Also called network manager. An application-level software module, residing within one station on the ring, for managing ring parameters and statistics. The presence of a configuration report server on a ring is optional. (Silicon Graphics does not currently provide this software module.)

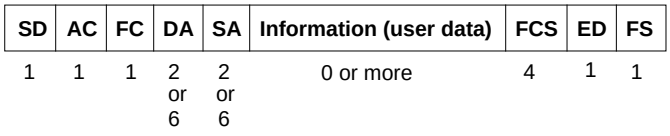
The configuration report server maintains ring configuration parameters and updates other stations on the ring in the event any of the parameters change. As stations insert themselves onto the ring, they request the current parameters from the configuration report server. This server communicates (via *MAC frames*) with each station's *medium access control sublayer*. For example, in IRIS Token Ring products, this functionality is provided by the Texas Instrument TMS38016C chip on the adapter board.

controlled access unit (CAU)

An intelligent hardware device that provides diagnostic support for the ring. A CAU is always associated with a *lobe attachment module*. CAUs reside in a wiring closet.

data frames

Token Ring *frames* that contain user data in the "Information" segment, as illustrated in Figure Gl-1.



SD = starting delimiter
AC = access control
FC = frame control
DA = destination MAC address
SA = source MAC address
FCS = frame check sequence (checksum)
ED = ending delimiter
FS = frame status

Figure Gl-1 Data Frame Format

data link layer

Network layer 2 for both the Open Systems Interconnect and the *Systems Network Architecture* environments. The data link layer is responsible for data transfer across a single physical connection. For Token Ring, the services of this layer are divided into two sublayers: *medium access control* and *logical link control*.

data link provider interface (DLPI)

In an SNA environment, the interface (language for talking) to the SNA *data link layer*.

DLPI

See *data link provider interface*.

dotted decimal notation

A method of representing a 4-byte (32-bit) Internet address (IP address) in ASCII characters (digits 0-9). Each byte of the address is represented as a decimal number ranging in value from 0 to 255. Bytes are separated by a dot (.), for example, 126.4.71.254. See *Internet address*.

downstream

A term indicating the relative positioning of stations on the ring. Downstream stations are those that see the signal after the station in question.

duplicate address

An error condition in which two stations on a ring use the same physical address (*MAC address*).

duplicate address test frame (DAT)

A type of MAC frame (management frame) used by a station during initialization to verify that its physical address (*MAC address*) is unique on the ring. All stations use this type of frame when they *insert* themselves onto the ring.

early token release

An optional set of rules for governing Token Ring data transmission, frame *strip*ping, and token release. This enhancement is available only for 16 megabit per second Token Rings. The rules allow a transmitting station to *regenerate* (release) the token before *strip*ping all of its *data frames* and allow the next station to start transmitting. Early token release makes it possible for more than one data frame to be on the ring at a time, thus increasing the ring's efficiency (or use of bandwidth).

efficient use of bandwidth

A condition where a high percentage of the theoretical data-carrying capacity of a transmission medium (cable) is used. For example, a 16 megabit per second Token Ring theoretically can carry 16 million bits of information every second. If every second (on the average), the stations on a ring were able to fill the cable with data so that 95% of the 16 million available bits were actually bits of data (as opposed to *fill* or garbage), this would be very efficient use of bandwidth. If, however, only 20% of the bits carried data, this would be inefficient use. Items that affect how efficient the use of bandwidth is on a network include the following:

- Station states required by the communications protocol that prohibit transmitting (for example, waiting for the final transmitted frame to return before releasing the token).
- Station activities that make all transmission on the ring impossible (for example, token claiming).
- Station software that does not process fast enough to keep the transmission medium full to capacity (for example, a poorly designed driver).

fill

A signal pattern placed on the transmission medium (cable) when a station is waiting (for example, when a station that has captured the token is not transmitting data and is waiting for its *data frames* to return).

frame

The method for organizing information for communication among Token Ring *station*s and *medium access control sublayer*s. There are three types of Token Ring frames: *MAC frames*, *data frames*, and the *token*.

guaranteed latency

The amount of time it takes for a signal to make a complete loop around the ring from the output port of one station around to the same station's input port. A ring with a small guaranteed latency is perceived as fast to its human users. The guaranteed latency of a ring increases as the *trunk* cable lengthens and as the number of stations increases. For example, the guaranteed latency for a 10 meter ring is less than for a 1000 meter ring, and the guaranteed latency of a ring with 5 stations is significantly less than a ring with 200 stations. The ring's data speed also affects its guaranteed latency; the higher the data rate, the smaller the guaranteed latency.

idle pattern

See *fill*.

idle state

A state of the ring where no *data frames* are being transmitted or *repeat*ed. Only the *token* and *MAC frames* are circling. This state is opposed to the ring's *active state*.

insert

The action taken by an *attaching device* by which it synchronizes its receive clock with the signal on the transmission medium (cable), verifies the uniqueness of its *MAC address*, and begins *repeating* the signal.

Internet address

Also called IP address. A 4-byte (32-bit) number used by the Internet Protocol (IP or TCP / IP) software to identify computers (or more accurately, computers' network connections). As the commonly quoted dictum says: "In the IP world, hosts do not have addresses, network interfaces do."
One computer (host) can have one or more IP addresses; each physical network connection for a host must have at least one unique IP address.

Internet addresses come in a number of classes; the major classes are A, B, and C. All IP addresses have three parts: class identifier, network identifier, and host identifier. The number of bits used to represent each part depends on the address' class, as described below:.

- Class A addresses use 1 bit (bit 0) for class identification, 7 bits for network identification, and 24 bits for host identification.
- Class B addresses use 2 bits (bits 0 and 1) for class identification, 14 bits for network, and 16 for host.
- Class C addresses use 3 bits (bits 0 to 2) for class identification, 21 for network, and 8 for host.

The class plus network identification parts are commonly referred to as the “network address,” while the class, network, and host identification parts are referred to as the “host address.” For example, the network address for a device with an IP (or host) addresss of 206.2.71.198 is “net 206.2.71.” IP addresses are usually represented in ASCII digits 0 to 9 in *dotted decimal notation* (for example, 126.13.69.237). Table Gl-2 summarizes the ranges of valid addresses within these three classes.

Table Gl-2 Internet Address Ranges

Class (Number of Networks Possible)	Bit Usage within Address (each X represents one byte)	Maximum Number of Hosts Possible for Each Network Address	Smallest Non-broadcast Valid Address	Largest Valid Non-broadcast Address
A (128)	X. (classid + netid) X.X.X (hostid)	16,777,215	1.0.0.0	126.255.255.254 (127.x.x.x is reserved)
B (16,384)	X.X. (classid + netid) X.X (hostid)	65,535	128.0.0.0	191.255.255.254
C (2,097,152)	X.X.X (classid + netid) X (hostid)	255	192.0.0.0	223.255.255.254

In order to ensure global uniqueness, network addresses (or blocks of them) are assigned by the *Network Information Center* to requesting organizations. The network administrator for each organization allocates the individual addresses (host addresses within the assigned blocks) to specific devices. Local network administrators are responsible for ensuring that two devices at the same site do not use the same address.

IP address

See *Internet address.*

LAM

See *lobe attachment module.*

latency

A measurement of time during which a transmission medium (cable) is not available for use. For example, there is latency associated with a token traveling from one station to the next; the longer the cable between the stations, the higher the latency; the slower the data rate used on that cable, the higher the latency. See also *guaranteed latency.*

LLC

See *logical link control sublayer.*

lobe

The section (consisting of one or more segments) of cabling that connects an *adapter* to a *trunk coupling unit* (TCU), as illustrated in Figure Gl-2. For example, a lobe can consist of a simple *adapter cable* connected directly to a TCU, or a lobe can be a drop (a segment of trunk cable) from a TCU to a wall faceplate connection and an adapter cable running from the wall connection to the adapter.

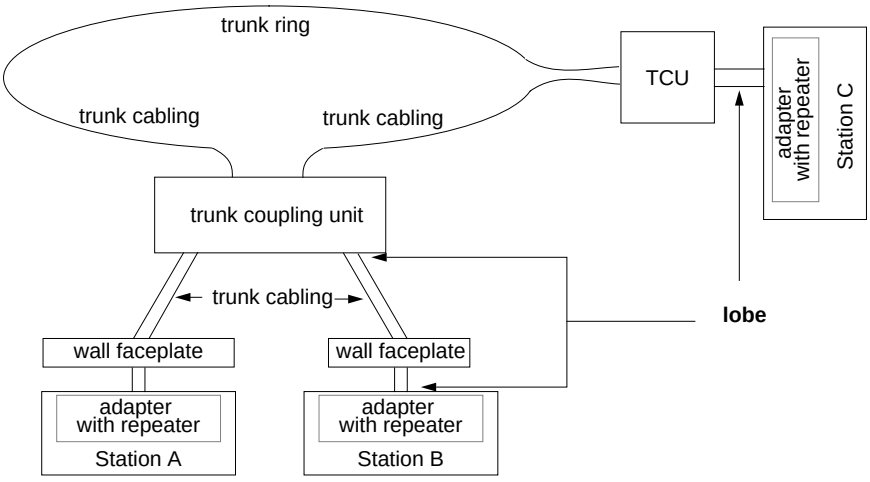


Figure Gl-2 Illustration of Lobe

lobe attachment module (LAM)

A hardware device that acts as a *concentrator* (and *trunk coupling unit*), allowing one or more *adapter*s to access the main (*trunk*) ring through it. LAMs may reside in a wiring closet and have *lobe* cabling that extends out to wall faceplate connections. See also *lobe* and *concentrator.*

logical link control sublayer (LLC)

Local area network protocol functions (services) handled by the upper portion of the OSI data link layer (layer 2), as illustrated in Figure Gl-3. IEEE 802.2 (also known as ISO 8802-2 and upon which Token Ring is based) is an example of a protocol standard for this sublayer. LLC 802.2 / 8802-2 provides three types of service: type 1 is unacknowledged connectionless, type 2 is connection oriented, and type 3 is acknowledged connectionless.

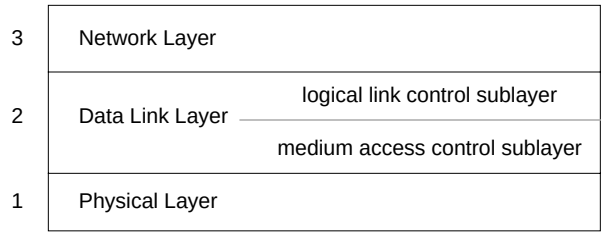


Figure Gl-3 Sublayers of Data Link Layer

lost token

A ring error condition. The condition exists when no station is transmitting and yet there is no token on the ring. The *active monitor* is responsible for correcting this error condition by regenerating the token.

MAC

See *medium access control sublayer.*

MAC address

Also called the physical address. An address that uniquely identifies the *medium access control* module of a station. In 802.5 environments, this is a 2-byte (16-bit) or 6-byte (48-bit) address. In FDDI environments, this is a 6-byte address. There is one MAC address for each network *adapter* on the ring. All the adapters on a ring must use the same address format: either the 2-byte format or the 6-byte one. MAC addresses can be universally assigned (UAA), in which case they are guaranteed to be globally unique, or they can be locally assigned (LAA), in which case they are unique only to their particular ring (or site). LAA MAC addresses (both 2-byte and 6-byte formats) are usually assigned and administered by a site's network manager. Blocks of UAA MAC addresses (6-byte format only) are assigned by the Institute of Electrical and Electronics Engineers (IEEE). (The IRIS Token Ring product supports only the 6-byte format for MAC addresses.)

MAC frames

Token Ring *frames* that contain management information in the “Information” segment, as shown in Figure Gl-4. MAC frames are used to monitor and manage the ring. The frames are generated and interpreted by the *medium access control sublayer* (MAC) on each network interface board. The information exchanged with these frames is used by each station’s *monitor* module. Table 1-2 describes the types of MAC frames.

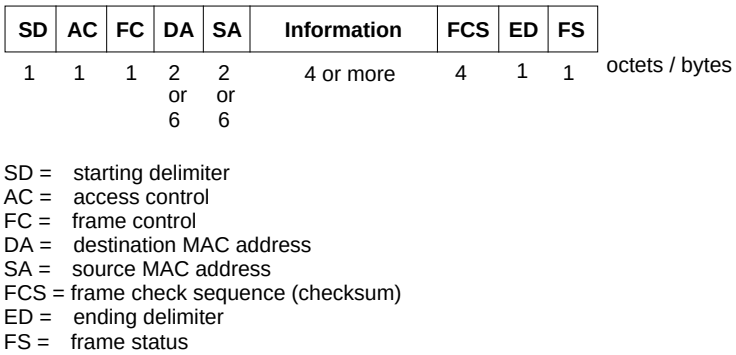


Figure Gl-4 MAC Frame Format

management frames

See *MAC frames*.

MAU

See *multistation access unit*.

medium access control sublayer (MAC)

Local area network protocol functions (services) handled by the lower portion of the OSI *data link layer* (layer 2), as illustrated in Figure Gl-3. The MAC interfaces to the *logical link control sublayer*. Each instance of a MAC must have an address that is unique on that ring. A MAC controls and mediates a station’s access to the ring as well as assembles *frames* (on transmission), disassembles them (on reception), and handles physical addressing procedures. ISO 8802-5 Token Ring and IEEE 802.5 Token Ring are examples of a protocol standard for this sublayer. See also *MAC address*.

monitor

A software module within each Token Ring station that maintains and manages the ring and recovers from various error situations. Only one monitor on a ring is *active*; all other monitors are *standby*.

multistation access unit (MAU)

A hardware device that acts as a *concentrator* (and *trunk coupling unit*), allowing one or more *adapter*s to access the main (*trunk*) ring through it. MAUs may reside in a wiring closet and have *lobe* cabling that extends out to wall faceplate connections. See also *lobe* and *concentrator*.

Network Information Center

The central authority that assigns blocks of Internet Protocol (IP) network addresses to worldwide public and private organizations. The current address for this organization is Government Systems, Inc., Attn: Network Information Center, 14200 Park Meadow Drive, Suite 200, Chantilly, VA 22021 (at telephone 1-800-365-3642).

node

A generic term referring to any device attached to the ring. All nodes must have a physical layer address (*MAC address*). However, not all nodes have network layer addresses; user data cannot be sent to nodes that do not have network addresses. A bridge is an example of a node without a network address; a *station* is a node with a network layer address.

persistently circulating

A condition whereby a frame circles the ring endlessly. For example, the station that put the frame onto the ring may fail before it can strip the frame.

port

The point at which the ring's signal passes into or out of a *node*. Each *adapter* has one input port and one output port.

priority

A method for indicating the relative urgency or importance of data. Token Ring provides eight levels of priority and a set of rules for ensuring that higher-priority data be transmitted before lower-level data, as explained in the section "802.5 Mechanism for Prioritizing Data" in Chapter 1.

priority field

A sequence of bits (a field) within the access control portion of each *token* that is used to indicate the ring's current *priority* level. Stations can transmit data that has the same or a higher priority level than the level indicated in the token's priority field. Data with a lower priority must wait for the ring's priority level to decrease.

purge

To clear the ring of all signals. This is one of the first actions performed by the *active monitor* upon taking on its role.

regenerate the token

Also called release the token. The act of placing the *token frame* on the transmission medium (cable) whenever this action is not *repeating*. Only a station in either of the following states is allowed to regenerate the token: a station that has *capture*d the token or a station that is the *active monitor*. In the former case, the station releases the token when it has finished transmitting data. In the latter case, a token is regenerated when the ring does not currently have a token (for example, a *lost token* condition or when the active monitor first assumes its duties).

release

See *regenerate the token*

repeater

A hardware device that repeats back onto the ring each signal it receives (reads) from it. Each *attaching device* on a Token Ring must be connected to its own repeater; the repeater is usually located on the network *adapter* board.

(In some texts, but not this one, this term is used to refer to a special node used only for boosting the ring's signal, which enables lengthening the geographical distance covered by a network.)

repeating

The act of reading a signal from the transmission medium (cable) and retransmitting it. Every token ring station is required to constantly do this, except when it is in the *bypass* state or when it is transmitting data. When repeating a frame, a station may alter the settings on some of the bits (for example, the *reservation field*, the *A bit*, the *C bit*). See also *repeater*.

reservation field

A sequence of bits (a field) within the access control field of each frame (data, MAC, or token) that is used to indicate a desired *priority* level. When a station has higher-priority data to transmit, it indicates the desired priority level in the reservation field of any frame that it is *repeating*. Sometime later (governed by the priority mechanism described in "802.5 Mechanism for Prioritizing Data" in Chapter 1), a token is generated with the requested priority level.

ring

A set of *repeater*s connected by transmission media (cabling) in a manner such that the signal travels from repeater to repeater in a closed loop. A number of cable layouts can be used to construct a ring. Two examples are illustrated in Figure Gl-5.

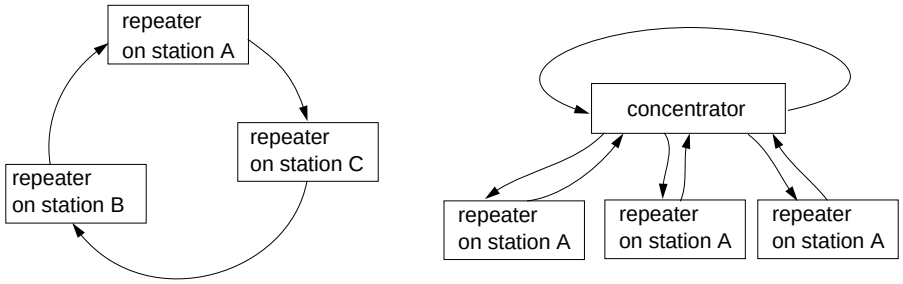


Figure Gl-5 Cable Layouts for a Ring

round-robin

A scheme for controlling action in which all the devices in a circle take turns sequentially. In Token Ring, the controlled action is permission to transmit data onto the ring. At any point in time, only one *station* has the permission (the turn). When the turn-taker finishes transmitting, it passes the turn (*token*) to the next *downstream* entity. This system guarantees that only one station at a time is transmitting and that each station is given a turn.

route

Also referred to as a link. A group of addresses that define a path from one station (the source address) to another station (the destination address). Just as the stepping stones of a garden path provide the sequential steps that move you along the path, each address within a route provides the next step along that route. And, using this same analogy, as stepping stones provide places to stop and transfer your body from one foot to the other, each address for an intermediate station is a location where the packet stops (is picked up) and is transferred to another local area network. A complete route from source to destination may consist of one or numerous addresses. In Token Ring, routes are limited to a maximum of 7 addresses.

router

Also referred to as intermediate system. A computer, with connections to two or more networks, that performs *routing* services for those networks.

routing

The process of discovering and assigning a *route* from a sending computer (the source address) to an intended receiving computer (the destination address). The routing method used for any particular packet / datagram can be either *source routing* or *transparent routing*. The exact manner in which routing is done differs from protocol to protocol.

In Token Ring environments, source routing at the *medium access control sublayer* is a major routing method. Each station maintains the routing information for all the stations with which it wishes to communicate. Each route (or link) consists of one to seven addresses that, when followed sequentially, lead to the destination. The number of routes that a particular station (or *adapter*) can handle depends on the amount of memory available.

In Ethernet environments, routing is handled by the upper layers of the protocol stack (for example, the Exterior Gateway Protocol, the Routing Information Protocol, the source routing option in the Internet Protocol). For BSD UNIX environments (including IRIX) using the TCP / IP protocol, both source and transparent routing are available. Software modules acting as *router*s and gateways maintain the routing information and handle transparent routing. There is virtually no limit to the number of addresses that can be specified in a source route; however, there is a practical limit set by available memory.

SMP

See *standby monitor present frame*.

SNA

See *Systems Network Architecture*.

source routing

A method of *routing* whereby the originator (sender) dictates the complete *route* that must be followed by the message in order to reach its destination. Each router / gateway along the route honors the route specified by the sender instead of following any of its usual or known routes. Some types of source routing (for example, Token Ring) are handled by the *medium access control sublayer*. Other types (for example, IP) are managed by the network layer. For a different routing method, see *transparent routing*. See also *routing*.

stable ring

A ring condition where no stations are being added or removed. When a ring is stable, the *upstream neighbor address*es (UNAs) that stations receive in circulating frames match the UNAs that are stored in their memories.

stacking

An act whereby a station raises the ring's *priority* level by generating a *token* whose *priority field* contains a value that is higher than the value that was in the *captur*ed token.

standby monitor

A *monitor* that watches the ring to verify that one *active monitor* exists and that there is no *break in the ring*. If an active monitor does not exist, standby monitors take action to remedy the problem by *token claiming*. If a break in the ring occurs, standby monitors starting *beaconing*. All of the stations on the ring, except one, are standby monitors.

standby monitor present frame (SMP)

A MAC frame (management frame) generated by each station whose *monitor* is in the *standby monitor* state. This frame is used to communicate the *upstream neighbor address* (UNA) that is essential for monitoring and maintaining ring integrity.

station

A system (hardware and software) with a unique *MAC address* and a network layer address (for example, an IP address or an LU address). Each station is attached to the ring through an *adapter*.

strip

Remove a *data frame* from the ring. Each station is responsible for *strip*ping the data frames that it transmits.

Systems Network Architecture (SNA)

A family of protocols developed by IBM that make it possible for SNA-speaking computer systems to exchange data and interoperate. SNA has seven layers, as illustrated in Figure 1-5; each layer handles a specific set of communication functions and provides these services to the adjacent layers. The services defined for each SNA layer do not correspond necessarily to those in the OSI layer at the same relative position.

token frame (or token)

A control signal that consists of a unique signaling sequence (pattern), shown in Figure Gl-6. At any point in time, a ring can have zero or one token circulating, but no more. The token is *capture*d (not repeated onto the ring) by a *station* that wants to transmit data. It is regenerated (put onto the ring) when the transmitting station finishes transmitting.

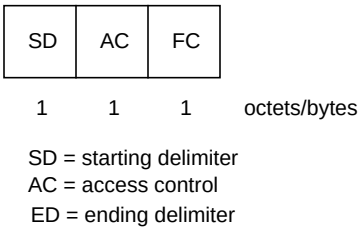


Figure Gl-6 Token Format

token claiming

A ring management process whose goal is to make one station into the *active monitor* for the ring. The absence of an active monitor causes token claiming to start. During the token claiming process, one or more stations transmit *claim token frame*s. All stations refrain from transmitting data during token claiming. When a token claiming station sees its own claim token frame return, and when that frame's *upstream neighbor address* (UNA) matches the station's previously received (stored) UNA value, the station becomes the active monitor. The event that causes a station to start token claiming is failure to see an *active monitor present* or *token frame* for a period of time. (The length of this time period is defined by the 802.5 Token Ring Standard.)

token holding timer

A timer that controls the maximum amount of time a station can use (occupy) the transmission medium (cable). The transmission medium is "occupied" as long as the station holds the *token*. When this timer expires, the station must place the token back onto the ring. The length for this timer is defined by the 802.5 Token Ring Standard.

transparent routing

A method of *routing* whereby the originator (sender) of a packet / datagram specifies only the destination address, and a *router* / gateway on the sender's LAN does the *routing* to either the destination system or to the next stop along the way to the destination. The sender does not specify, and does not need to know, the complete route. Each *router* (intermediate system) that receives the packet looks at the destination address, decides which of its known routes is best for reaching the destination, and sends the packet to the next-step along that route, until the destination is reached.

trunk

An adjective denoting the main ring or something attached to the main ring (as opposed to an entity indirectly attached to the ring through a *trunk coupling unit*). Figure Gl-7 illustrates a trunk ring.

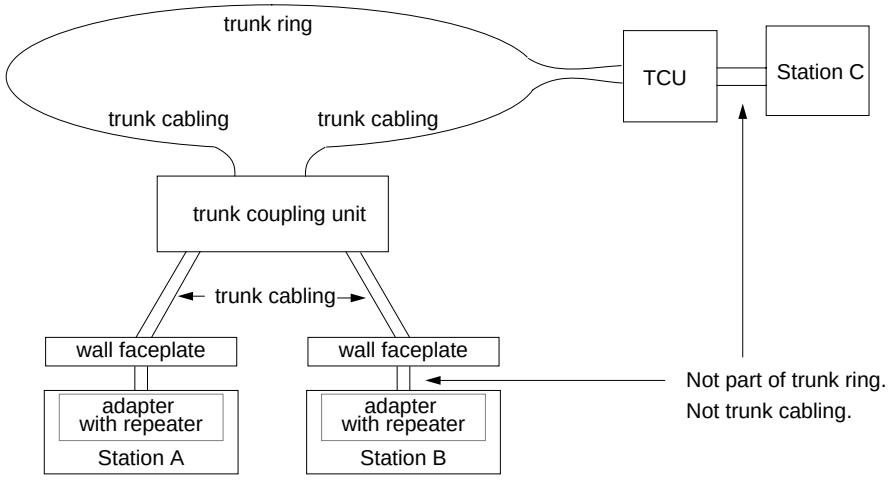


Figure Gl-7 Definition of Trunk

trunk coupling unit (TCU)

A hardware device that enables one or more stations to connect to the *trunk* cable, as illustrated in Figure Gl-7. The TCU can insert the station onto the ring as well as bypass the connection to a station that is missing or dysfunctional. Examples of TCUs include the following: *multistation access unit*s, *lobe attachment module*s, and other *concentrator*s.

upstream

A term describing the relative positioning of stations on the ring. Upstream stations are those that see and repeat the signal before the station in question. A station can become aware of instability in the ring only as a change in its upstream neighbor or the failure to receive a signal from the upstream neighbor, hence the importance of the *upstream neighbor address* (UNA).

upstream neighbor address (UNA)

The physical address (*MAC address*) of the station that transmits / repeats immediately before the station in question. The *upstream* neighbor's address (UNA) is one of the most important items for monitoring the health of a ring. Each station on a Token Ring keeps track of its UNA. As long as a station continues to receive frames indicating the same UNA, the ring is *stable*. When the UNA in the current frame does not match the previously received (stored) UNA, the ring configuration is changing. For example, the old upstream neighbor may have failed or may have been removed.

use of bandwidth

See *efficient use of bandwidth*.

wiring concentrator (WC)

See *concentrator*.

Index

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

Y

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-1561-040.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 415-965-0964
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389