CHAPTER 21

Configuring Transparent Bridging

Our router/bridge combines the advantages of a spanning-tree bridge and a full multiprotocol router. This combination provides the speed and protocol transparency of an adaptive spanning-tree bridge, along with the functionality, reliability, and security of a router.

This chapter discusses how to configure transparent bridging and source-route transparent (SRT) bridging. Configuration examples are provided at the end of the chapter. For a complete description of the commands mentioned in this chapter, refer to the *Router Products Command Reference* publication. For historical background and a technical overview of transparent bridging, see the *Internetworking Technology Overview* publication.

Cisco's Implementation of Transparent and Source-Route Transparent Bridging

Cisco supports transparent bridging for Ethernet, Fiber Distributed Data Interface (FDDI), and serial media, and supports SRT bridging for Token Ring media. In addition, Cisco supports all of the mandatory Management Information Base (MIB) variables specified for transparent bridging in RFC 1286.

Transparent Bridging Features

Cisco's transparent bridging software implementation provides the following features:

- Complies with the IEEE 802.1D standard.
- Provides two spanning-tree protocols—an older bridge protocol data unit (BPDU) format that is compatible with Digital and other local-area network (LAN) bridges for backward compatibility and the IEEE standard BPDU format. In addition to features standard with these spanning-tree protocols, Cisco's proprietary software provides for multiple domains for spanning trees. The spanning-tree parameters are configurable.
- Allows configuration of filters to effectively filter frames based on MAC address, protocol type, or the vendor code. Additionally, the bridging software can be configured to selectively filter Local Area Transport (LAT) multicast service announcements.
- Provides deterministic load distribution while maintaining a loop-free spanning tree.
- Provides the ability to bridge over X.25, Frame Relay, Switched Multimegabit Data Service (SMDS), and Point-to-Point (PPP) networks.
- Provides for compression of LAT frames to reduce LAT traffic through the network.

Routers can be configured to serve as both multiprotocol routers and Media Access Control (MAC)-level bridges, bridging any traffic that cannot otherwise be routed. For example, a router/bridge routing the Internet Protocol (IP) can also bridge Digital's LAT protocol or NetBIOS traffic.

Remote bridging over synchronous serial lines is also supported between our routers. As with frames received on all other media types, the dynamic learning and any filtering is applied to frames received on serial lines.

Transit bridging of Ethernet frames across FDDI media is also supported. The term *transit* refers to the fact that the source or destination of the frame cannot be on the FDDI media itself. This allows FDDI to act as a highly efficient backbone for the interconnection of many bridged networks. The configuration of FDDI transit bridging is identical to the configuration of transparent bridging on all other media types.

Source-Route Transparent Bridging Features

Our router/bridges also support transparent bridging on Token Ring interfaces that are capable of supporting SRT bridging.

Note Both transparent and SRT bridging are supported on all Token Ring interface cards that can be adjusted by the user for either 4- or 16-Mb transmission speeds.

As with all other media types, all the features that use **bridge-group** commands can be used on Token Ring interfaces. As with other interface types, the bridge group can be configured to run either the IEEE or Digital spanning-tree protocols. When using the IEEE spanning-tree protocol, the bridge cooperates with other bridges complying to the draft SRT bridging specification and constructs a loop-free topology across the entire extended LAN.

You can run the Digital spanning-tree protocol over Token Ring as well. Use it when you have other non-IEEE bridges on other media and do not have any SRT bridges on Token Ring. Note that in this configuration, all of your Token Ring transparent bridges must be Cisco routers. This is because the Digital spanning-tree protocol has not been standardized on Token Ring.

As specified by the SRT bridging specification, only packets without a routing information field (RIF) (RII = 0 in the SA field) will be transparently bridged. Packets with a RIF (RII = 1) are passed to the source-route bridging module for handling. Note that an SRT-capable Token Ring interface can have both source-route bridging and transparent bridging enabled at the same time. However, when running SRT bridging, frames that did not have a RIF when they were produced by their generating host will never gain a RIF, and frames that did have a RIF when they were produced will never lose that RIF.

Note Because bridges running only SRT bridging never add or remove RIFs from frames, they do not really integrate source-route bridging with transparent bridging. Rather, the two are kept separate but equal. A host that sits behind a source-route bridge that expects RIFs can *never* communicate to a device across a bridge that does not understand RIFs. Because of this fact, SRT bridging cannot be used to tie in existing source-route bridges to a transparent bridged network. If you want to tie them in, you must use source-route translational bridging (SR/TLB) instead. SR/TLB is described in the chapter "Configuring Source-Route Bridging."

When bridging between Token Ring and other media, certain packet transformations must occur. In all cases, the MAC addresses are bit-swapped, because the bit ordering on Token Ring is different from that on other media. In addition, Token Ring supports one higher-layer packet format, logical link control (LLC), while Ethernet supports two (LLC and Ethernet). The transformation of LLC frames between media is quite simple; a length field is either created (when going to non-Token Ring) or removed (when going to Token Ring). When an Ethernet format frame must go to Token Ring, it is translated into a LLC-1 "SNAP" packet ; the destination service access point (DSAP) value is AA, the source service access point (SSAP) value is AA, with the organizational unique identifier (OUI) value 0000F8. Likewise, when such a packet in LLC-1 format is going to be bridged onto Ethernet media, it is translated back into Ethernet format. You can determine the OUI type used when transporting Ethernet Type II frames over other media. Graphic illustrations of frame formats are given in the *Internetworking Technology Overview* publication.

Caution Bridging between dissimilar media presents several problems that can prevent communication from occurring. These problems include bit order translation (or usage of MAC addresses as data), maximum transmission unit (MTU) differences, frame status differences, and multicast address usage. Some or all of these problems might be present in a multimedia bridged LAN. Because of differences in the way end nodes implement Token Ring, these problems are most prevalent when bridging between Token Ring and Ethernet or between Ethernet and FDDI LANs.

We currently know that problems occur with the following protocols when bridged between Token Ring and other media: Novell IPX, DECnet Phase IV, AppleTalk, VINES, XNS and IP. Further, problems can occur with the Novell IPX and XNS protocols when bridged between FDDI and other media. We recommend that these protocols be routed whenever possible.

Transparent and SRT Bridging Configuration Task List

Perform one or more of the tasks in the following sections to configure transparent bridging or SRT bridging on your router/bridge:

- Configure Transparent Bridging and SRT Bridging
- Configure Transparent Bridging over WANs
- Configure Transparent Bridging Options
- Filter Transparently Bridged Packets
- Adjust Spanning-Tree Parameters
- Tune the Transparently Bridged Network
- Monitor and Maintain the Transparent Bridge Network

See the end of this chapter for configuration examples.

Configure Transparent Bridging and SRT Bridging

To configure transparent and SRT bridging, you must perform the tasks in the following sections:

- Assign a Bridge Group Number and Define the Spanning-Tree Protocol
- Assign Each Network Interface to a Bridge Group
- Choose the OUI for Ethernet Type II Frames

Assign a Bridge Group Number and Define the Spanning-Tree Protocol

The first step in setting up your transparent bridging network is to define a spanning-tree protocol and assign a bridge group number. You can choose either the IEEE 802.1D spanning-tree protocol or the earlier Digital protocol upon which this IEEE standard is based.

To assign a bridge group number and define a spanning-tree protocol, perform the following task in global configuration mode:

Task	Command
Assign a bridge group number and define a spanning-tree protocol as either IEEE 802.1D standard or Digital.	**bridge** *bridge-group* **protocol** {**ieee** | **dec**}

The IEEE 802.1D spanning-tree protocol is the preferred way of running the bridge. Use the Digital spanning-tree protocol only for backward compatibility.

Assign Each Network Interface to a Bridge Group

A bridge group is an internal organization of network interfaces on a router. Bridge groups cannot be used outside the router on which it is defined to identify traffic switched within the bridge group. Bridge groups within the same router function as distinct bridges; that is, bridged traffic and BPDUs cannot be exchanged between different bridge groups on a router. Furthermore, bridge groups cannot be used to multiplex or demultiplex different streams of bridged traffic on a LAN. An interface can only be a member of one bridge group. Use a bridge group for each separately bridged (topologically distinct) network connected to the router. Typically, only one such network exists in a configuration.

The purpose of placing network interfaces into a bridge group is twofold:

- To bridge all nonrouted traffic among the network interfaces making up the bridge group. If the packet's destination address is known in the bridge table, it is forwarded on a single interface in the bridge group. If the packet's destination is unknown in the bridge table, it is flooded on all forwarding interfaces in the bridge group. The bridge places source addresses in the bridge table as it learns them during the process of bridging.
- To participate in the spanning-tree algorithm by receiving, and in some cases transmitting, BPDUs on the LANs to which they are attached. A separate spanning process runs for each configured bridge group. Each bridge group participates in a separate spanning tree. A bridge group establishes a spanning tree based on the BPDUs it receives on only its member interfaces.

For SRT bridging, if the Token Ring and serial interfaces are in the same bridge group, changing the serial encapsulation method causes the state of the corresponding Token Ring interface to be reinitialized. Its state will change from "up" to "initializing" to "up" again within a few seconds.

After you assign a bridge group number and define a spanning-tree protocol, assign each network interface to a bridge group by performing the following task in interface configuration mode:

Task	Command
Assign a network interface to a bridge group.	**bridge-group** *bridge-group*

Choose the OUI for Ethernet Type II Frames

For SRT bridging networks, you must choose the OUI code that will be used in the encapsulation of Ethernet Type II frames across Token Ring backbone networks. To choose the OUI, perform the following task in interface configuration mode:

Task	Command
Select the Ethernet Type II OUI encapsulation code.	**ethernet-transit-oui** [**90-compatible** | **standard** | **cisco**]

Configure Transparent Bridging over WANs

We support transparent bridging over the following types of networks:

- X.25
- Frame Relay
- SMDS
- PPP

This section describes how to configure bridging on these networks.

Note There is no specific task you must perform to configure transparent bridging over a PPP network.

Configure X.25 Transparent Bridging

The transparent bridging software supports bridging of packets in X.25 frames. This ability is useful for such tasks as transmitting packets from proprietary protocols across an X.25 network.

The X.25 bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated in X.25 frames and transmitted across X.25 media. You specify the Internet-to-X.121 address mapping, and the system maintains a table of both the Ethernet and X.121 addresses. To configure X.25 transparent bridging, perform the following task in interface configuration mode:

Task	Command
Specify IP-to-X.121 mapping.	**x25 map bridge** *x.121-address* **broadcast** [*options-keywords*]

Configuring X.25 is discussed in more detail in the chapter “Configuring X.25 and LAPB.”

Configure Frame Relay Transparent Bridging

The transparent bridging software supports bridging of packets over Frame Relay networks. This ability is useful for such tasks as transmitting packets from proprietary protocols across a Frame Relay network. Bridging over a Frame Relay network is supported both on networks that support a multicast facility and those that do not. Both cases are described in this section.

Bridging in a Frame Relay Network with No Multicasts

The Frame Relay bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated for transmission across a Frame Relay network. You specify IP-to-DLCI (data-link connection identifier) address mapping, and the system maintains a table of both the Ethernet address and the DLCIs.

To configure bridging in a network not supporting a multicast facility, define the mapping between an address and the DLCI used to connect to the address. To bridge with no multicasts, perform the following task in interface configuration mode:

Task	Command
Define the mapping between an address and the DLCI used to connect to the address.	**frame-relay map bridge** *dlci* **broadcast**

An example configuration is provided in the section "Frame-Relay Transparent Bridging Examples" at the end of this chapter. Frame Relay is discussed in more detail in the chapter "Configuring Frame Relay."

Bridging in a Frame Relay Network with Multicasts

The multicast facility is used to learn about the other bridges on the network, eliminating the need for you to specify any mappings with the **frame-relay map bridge broadcast** command. An example configuration is provided in the section "Frame-Relay Transparent Bridging Examples" at the end of the chapter for use as a configuration guide. Frame Relay is discussed in more detail in the chapter "Configuring Frame Relay."

Configure SMDS Transparent Bridging

We support transparent bridging over SMDS networks. Standard bridging commands are used to enable bridging on an SMDS interface. To enable transparent bridging over SMDS, perform the following task in global configuration mode:

Task	Command
Enable transparent bridging of packets across an SMDS network.	**smds multicast bridge** *smds-address*[1]

1. This command is documented in the "SMDS Commands" chapter of the *Router Products Command Reference* publication.

When transparently bridging over an SMDS network, the broadcast ARP packets are treated differently from other encapsulation methods. For SMDS, two packets are sent to the multicast address. One is sent using a standard (SMDS) ARP encapsulation; the other is sent with the ARP packet encapsulated in an 802.3 MAC header. The native ARP is sent as a regular ARP broadcast.

In addition, our implementation of 802.6 bridging only supports the transmission and reception of 802.3 encapsulated bridge packets. However, other encapsulations will be supported in a future release. Software Release 9.21 does not support bridging over multiple logical IP subnets (MultiLIS) because bridging of IP packets in a MultiLIS environment is unpredictable. It also does not support bridging from SMDS to SMDS; that is, packets from one SMDS network will not be forwarded to another SMDS network interface through the bridging mechanism. SMDS is discussed in more detail in the chapter "Configuring SMDS."

Configure Transparent Bridging Options

You can configure one or more transparent bridging options. To configure transparent bridging options, perform one or more of the tasks in the following sections:

- Disable IP Routing
- Enable Autonomous Bridging
- Configure LAT Compression
- Establish Multiple Spanning-Tree Domains
- Prevent the Forwarding of Dynamically Determined Stations
- Forward Multicast Addresses
-

Disable IP Routing

If you want to bridge IP, you must disable IP routing because IP routing is enabled by default on all routers/bridges. You can enable IP routing when you decide to route IP packets. To disable or enable IP routing, perform one of the following tasks in global configuration mode:

Task	Command
Disable IP routing.	**no ip routing**
Enable IP routing.	**ip routing**

All interfaces in the bridge group that are bridging IP should have the same IP address. However, if you have more than one bridge group, each bridge group should have its own IP address.

Enable Autonomous Bridging

Normally, bridging takes place on the processor card at the interrupt level. When autonomous bridging is enabled, bridging takes place entirely on the ciscoBus II controller, significantly improving performance. Autonomous bridging is a high-speed switching feature that allows bridged traffic to be forwarded and flooded on the ciscoBus II controller between resident interfaces. If you are using the ciscoBus II controller, you can maximize performance by enabling autonomous bridging on the following ciscoBus II interfaces:

- MEC
- FCIT transparent
- HSSI HDLC

Although performance improvements will be seen most in the resident interfaces, the autonomous bridging feature can also be used in bridge groups that include interfaces that are not on the ciscoBus II controller. These interfaces include the CTR, FCI with encapsulation bridging, High-Speed Serial Interface (HSSI) with other than High-Level Data Link Control (HDLC) encapsulation such as X.25, Frame Relay, or SMDS, MCI, STR, or SBE16.

If you enable autonomous bridging for a bridge group that includes a combination of interfaces that are resident on the ciscoBus II controller and some that are not, the ciscoBus II controller forwards only packets between resident interfaces. Forwarding between nonresident and resident interfaces is done in either the fast or process paths. Flooding between resident interfaces is done by the ciscoBus II controller. Flooding between nonresident interfaces is done conventionally. If a packet

is forwarded from a nonresident to a resident interface, the packet is conventionally forwarded. If packets are flooded from a nonresident interface to a resident interface, the packet is autonomously flooded.

To enable autonomous bridging on a per-interface basis, perform the following task in interface configuration mode:

Task	Command
Enable autonomous bridging (if using the ciscoBus II controller).	**bridge-group** *bridge-group* **cbus-bridging**

Note You can only filter by MAC-level address on an interface when autonomous bridging is enabled on that interface. If any filters or priority queuing is configured, autonomous bridging is automatically disabled.

Configure LAT Compression

The Local Area Transport (LAT) protocol used by Digital and Digital-compatible terminal servers is one of the common protocols that lacks a well-defined network layer (Layer 3) and so always must be bridged.

To reduce the amount of bandwidth that LAT traffic consumes on serial interfaces, you can specify a LAT-specific form of compression. Doing so applies compression to LAT frames being sent out the router/bridge through the interface in question. To configure LAT compression, perform the following task in interface configuration mode:

Task	Command
Reduce the amount of bandwidth that LAT traffic consumes on a serial interface.	**bridge-group** *bridge-group* **lat-compression**

LAT compression can be specified only for serial interfaces. For the most common LAT operations (user keystrokes and acknowledgment packets), LAT compression reduces LAT's bandwidth requirements by nearly a factor of two.

Establish Multiple Spanning-Tree Domains

The Cisco IEEE 802.1D bridging software supports spanning-tree domains of bridge groups. Domains are a feature specific to Cisco. This feature is only available if you have specified IEEE as the spanning-tree protocol. A domain establishes an external identification of the BPDUs sent from a bridge group. The purpose of this identification is as follows:

- Bridge groups defined within the domain can recognize that BPDU as belonging to them.
- Two bridged subnetworks in different domains that are sharing a common connection can use the domain identifier to identify and then ignore the BPDUs that belong to another domain. Each bridged subnetwork establishes its own spanning tree based on the BPDUs that it receives. The BPDUs it receives must contain the domain number to which the bridged subnetwork belongs. Bridged traffic is not domain identified.

Note Domains do not constrain the propagation of bridged traffic. A bridge bridges nonrouted traffic received on its interfaces regardless of domain.

You can place any number of router/bridges within the domain. The devices in the domain, and only those devices, will then share spanning-tree information.

When multiple routers share the same cable and you want to use only certain discrete subsets of those routers to share spanning-tree information with each other, establish spanning-tree domains. This function is most useful when running other router applications, such as IP User Datagram Protocol (UDP) flooding, that use the IEEE spanning tree. You also can use this feature to reduce the number of global reconfigurations in large bridged networks.

To establish multiple spanning-tree domains, perform the following task in global configuration mode:

Task	Command
Establish a multiple spanning-tree domain.	**bridge** *bridge-group* **domain** *domain-number*

For an example of how to configure domains, see the "Complex Transparent Bridging Network Topology Example" section later in this chapter.

Prevent the Forwarding of Dynamically Determined Stations

Normally, the system forwards any frames for stations that it has learned about dynamically. By disabling this activity, the bridge will only forward frames whose address have been statically configured into the forwarding cache. To prevent or allow forwarding of dynamically determined stations, perform one of the following task in global configuration mode:

Task	Command
Filter out all frames except those whose addresses have been statically configured into the forwarding cache.	**no bridge** *bridge-group* **acquire**
Remove the ability to filter out all frames except those whose addresses have been statically configured into the forwarding cache.	**bridge** *bridge-group* **acquire**

Forward Multicast Addresses

A packet with a RIF, indicated by a source address with the multicast bit turned on, is not usually forwarded. However, you can configure bridging support to allow the forwarding of frames that would otherwise be discarded because they have a RIF. Although you can forward these frames, the bridge table will not be updated to include the source addresses of these frames.

To forward frames with multicast addresses, perform the following task in global configuration mode:

Task	Command
Allow the forwarding of frames with multicast source addresses.	**bridge** *bridge-group* **multicast-source**

Configure Bridge Table Aging Time

A bridge forwards, floods, or drops packets based on the bridge table. The bridge table maintains both static entries and dynamic entries. Static entries are entered by the network manager or by the bridge itself. Dynamic entries are entered by the bridge learning process. A dynamic entry is automatically removed after a specified length of time, known as *aging time*, from the time the entry was created or last updated.

If hosts on a bridged network are likely to move, decrease the aging-time to enable the bridge to adapt to the change quickly. If hosts do not transmit continuously, increase the aging time to record the dynamic entries for a longer time and thus reduce the possibility of flooding when the hosts transmit again.

To set the aging time, perform the following task in global configuration mode:

Task	Command
Set the bridge table aging time.	**bridge-group** *bridge-group* **aging-time** *seconds*

Filter Transparently Bridged Packets

A bridge examines frames and transmits them through the internetwork according to the destination address; a bridge will not forward a frame back to its originating network segment. The bridge software allows you to configure specific administrative filters that filter frames based upon information other than paths to their destinations. You can perform administrative filtering by performing one of the tasks in the following sections:

- Filter by MAC-Level Address
- Filter LAT Service Announcements

Note When setting up administrative filtering, remember that there is virtually no performance penalty in filtering by MAC address or vendor code, but there can be a significant performance penalty when filtering by protocol type.

When configuring transparent bridging access control, keep the following points in mind:

- You can assign only one access list to an interface.
- The conditions in the access list are applied to all outgoing packets not sourced by the router.
- Access lists are scanned in the order you enter them; the first match is used.
- An implicit deny everything entry is automatically defined at the end of an access list unless you include an explicit permit everything entry at the end of the list.
- All new entries to an existing list are placed at the end of the list. You cannot add an entry to the middle of a list. This means that if you have previously included an explicit permit everything entry, new entries will never be scanned. The solution is to delete the access list and retype it with the new entries.
- You can create extended access lists to specify more detailed filters, such as address match only.
- You should not use extended access lists on FDDI interfaces doing transit bridging as opposed to translational bridging.

Filter by MAC-Level Address

You can filter transmission of frames based on the MAC-level address various ways by performing one of the following tasks:

- Filter by specific MAC address
- Filter by vendor code
- Filter by protocol type

When filtering by a MAC-level address, you can use two kinds of access lists: standard access lists that specify a simple address, and extended access lists that specify two addresses. You can also further restrict access by creating filters for these lists. After you have completed one of the preceding tasks, perform the following task:

- Define and apply extended access lists

Note MAC addresses on Ethernets are "bit swapped" when compared with MAC addresses on Token Ring and FDDI. For example, address 0110.2222.3333 on Ethernet is 8008.4444.CCCC on Token Ring and FDDI. Access lists always use the canonical Ethernet representation. When using different media and building access lists to filter on MAC addresses, keep this point in mind. Note that when a bridged packet traverses a serial link, it has an Ethernet-style address.

Filter by Specific MAC Address

You can filter frames with a particular MAC-level station source or destination address. Any number of addresses can be configured into the system without a performance penalty. To filter by the MAC-level address, perform the following task in global configuration mode:

Task	Command
Filter particular MAC-level station addresses.	**bridge** *bridge-group* **address** *mac-address* {**forward** | **discard**} [*interface*]

When filtering specific MAC destination addresses, allow for multicast or broadcast packets that are required by the bridged network protocols. Refer to the example in the section "Multicast or Broadcast Packets Bridging Example" later in this chapter to guide you in building your configuration to allow for multicast or broadcast packets.

Filter by Vendor Code

The bridging software allows you to create access lists to administratively filter MAC addresses. These access lists can filter groups of MAC addresses, including those with particular vendor codes. There is no noticeable performance loss in using these access lists, and the lists can be of indefinite length. You can filter groups of MAC addresses with particular vendor codes by performing the first task and one or both of the other tasks that follow:

- Establish a vendor code access list
- Filter source addresses
- Filter destination addresses

To establish a vendor code access list, perform the following task in global configuration mode:

Task	Command
Prepare access control information for filtering of frames by canonical (Ethernet-ordered) MAC address.	**access-list** *access-list-number* {**permit** | **deny**} *address mask*

The vendor code is the first three bytes of the MAC address (left to right).

Note Remember that, as with any access list using MAC addresses, Ethernets swap their MAC address bit ordering, and Token Rings and FDDI do not. As such, an access list that works for one media might not work for others.

For an example of how to filter by vendor code, see "Multicast or Broadcast Packets Bridging Example" later in this chapter.

Once you have defined an access list to filter by a particular vendor code, you can assign an access list to a particular interface for filtering on the MAC *source* addresses of packets *received* on that interface or the MAC *destination* addresses of packets that would ordinarily be *forwarded* out that interface. To filter by source or destination addresses, perform one of the following tasks in interface configuration mode:

Task	Command
Assign an access list to an interface for filtering by MAC source addresses.	**bridge-group** *bridge-group* **input-address-list** *access-list-number*
Assign an access list to an interface for filtering by the MAC destination addresses.	**bridge-group** *bridge-group* **output-address-list** *access-list-number*

Filter by Protocol Type

You can filter by protocol type by using the access list mechanism and specifying a protocol type code. To filter by protocol type, perform the first task and one or more of the other tasks that follow:

- Establish a protocol type access list
- Filter Ethernet- and SNAP-encapsulated packets on input
- Filter Ethernet- and SNAP-encapsulated packets on output
- Filter IEEE 802.2-encapsulated packets on input
- Filter IEEE 802.2-encapsulated packets on output

Note It is not a good idea to have both input and output type code filtering on the same interface.

The order in which you enter **access-list** commands affects the order in which the access conditions are checked. Each condition is tested in succession. A matching condition is then used to execute a permit or deny decision. If no conditions match, a "deny" decision is reached.

Note Type-code access lists can have an impact on system performance; therefore, keep the lists as short as possible and use wildcard bit masks whenever possible.

Access lists for Ethernet- and IEEE 802.2-encapsulated packets affect only bridging functions. It is not possible to use such access lists to block frames with protocols that are being routed.

You can establish type-code access lists. Specify either an Ethernet type code for Ethernet-encapsulated packets or a DSAP/SSAP pair for 802.3 or 802.5-encapsulated packets. Ethernet type codes are listed in the "Ethernet Type Codes" appendix of the *Router Products Command Reference* publication.

To establish type-code access lists, perform the following task in global configuration mode:

Task	Command
Prepare access control information for filtering frames by protocol type.	**access-list** *access-list-number* {**permit** | **deny**} *type-code wild-mask*

You can filter Ethernet- and SNAP-encapsulated packets on input. For SNAP-encapsulated frames, the access list you create is applied against the two-byte TYPE field given after the DSAP/SSAP/OUI fields in the frame. The access list is applied to all Ethernet and SNAP frames received on that interface prior to the bridge learning process. SNAP frames also must pass any applicable IEEE 802.2 DSAP/SSAP access lists.

You can also filter Ethernet- and SNAP-encapsulated packets on output. The access list you create is applied just before sending out a frame to an interface.

To filter these packets on input or output, perform either or both of the following tasks in interface configuration mode:

Task	Command
Add a filter for Ethernet- and SNAP-encapsulated packets on input.	**bridge-group** *bridge-group* **input-type-list** *access-list-number*
Add a filter for Ethernet- and SNAP-encapsulated packets on output.	**bridge-group** *bridge-group* **output-type-list** *access-list-number*

You can filter IEEE 802-encapsulated packets on input. The access list you create is applied to all IEEE 802 frames received on that interface prior to the bridge-learning process. SNAP frames also must pass any applicable Ethernet type-code access list.

You can also filter IEEE 802-encapsulated packets on output. SNAP frames also must pass any applicable Ethernet type-code access list. The access list you create is applied just before sending out a frame to an interface.

To filter these packets on input or output, perform one or both of the following tasks in interface configuration mode:

Task	Command
Add a filter for IEEE 802-encapsulated packets on input.	**bridge-group** *bridge-group* **input-lsap-list** *access-list-number*
Add a filter for IEEE 802-encapsulated packets on output.	**bridge-group** *bridge-group* **output-lsap-list** *access-list-number*

Access lists for Ethernet- and IEEE 802-encapsulated packets affect only bridging functions. It is not possible to use such access lists to block frames with protocols that are being routed.

Define and Apply Extended Access Lists

If you are filtering by the MAC-level address, whether it is by a specific MAC address, vendor code, or protocol type, you can define and apply extended access lists. Extended access lists allow finer granularity of control. They allow you to specify both source and destination addresses and arbitrary bytes in the packet.

To define an extended access list, perform the following task in global configuration mode:

Task	Command
Define an extended access list for finer control of bridged traffic.	**access-list** *access-list-number* {**permit** | **deny**} *source source-mask destination destination-mask offset size operator operand*

To apply an extended access list to an interface, perform one or both of the following tasks in interface configuration mode:

Task	Command
Apply an extended access list to the packets being received by an interface.	**bridge-group** *bridge-group* **input-pattern** *access-list-number*
Apply an extended access list to the packet being sent by an interface.	**bridge-group** *bridge-group* **output-pattern-list** *access-list-number*

After an access list is created initially, any subsequent additions (possibly entered from the terminal) are placed at the *end* of the list. In other words, you cannot selectively add or remove access list command lines from a specific access list.

Caution Because of their complexity, only use extended access lists if you are very familiar with the router. Further, do not specify an offset value that is greater than the size of the packet.

Filter LAT Service Announcements

The bridging software allows you to filter LAT frames. LAT bridge filtering allows the selective inclusion or exclusion of LAT multicast service announcements on a per-interface basis.

Note The LAT filtering commands are not implemented for Token Ring interfaces.

In the LAT protocol, a *group code* is defined as a decimal number in the range 0 to 255. Some of the LAT configuration commands take a list of group codes; this is referred to as a *group code list*. The rules for entering numbers in a group code list follow:

- Entries can be individual group code numbers separated with a space. (The Digital LAT implementation specifies that a list of numbers be separated by commas; however, our implementation expects the numbers to be separated by spaces.)
- Entries can also specify a range of numbers. This is done by separating an ascending order range of group numbers with hyphens.
- Any number of group codes or group code ranges can be listed in one command; just separate each with a space.

In LAT, each node transmits a periodic service advertisement message that announces its existence and availability for connections. Within the message is a group code list; this is a mask of up to 256 bits. Each bit represents a group number. In the traditional use of LAT group codes, a terminal server only will connect to a host system when there is an overlap between the group code list of the user on the terminal server and the group code list in the service advertisement message. In an environment with many bridges and many LAT hosts, the number of multicast messages that each system has to deal with becomes unreasonable. The 256 group codes may not be enough to allocate local assignment policies, such as giving each DECserver 200 device its own group code in large bridged networks. LAT group code filtering allows you to have very fine control over which multicast messages actually get bridged. Through a combination of input and output permit and deny lists, you can implement many different LAT control policies.

You can filter LAT service advertisements by performing any of the tasks in the following sections:

- Enable LAT Group Code Service Filtering
- Specify Deny or Permit Conditions for LAT Group Codes on Input
- Specify Deny or Permit Conditions for LAT Group Codes on Output

Enable LAT Group Code Service Filtering

You can specify LAT group-code filtering to inform the system that LAT service advertisements require special processing. To enable LAT group code filtering, perform the following task in global configuration mode:

Task	Command
Enable LAT service filtering.	**bridge** *bridge-group* **lat-service-filtering**

Specify Deny or Permit Conditions for LAT Group Codes on Input

You can specify the group codes by which to deny or permit access upon input. Specifying deny conditions causes the system to not bridge any LAT service advertisement that contain any of the specified groups. Specifying permit conditions causes the system to bridge only those service advertisements that match at least one group in the specified group-list.

To specify deny or permit conditions for LAT groups on input, perform one of the following tasks in interface configuration mode:

Task	Command
Specify the group codes by which to deny access upon input.	**bridge-group** *bridge-group* **input-lat-service-deny** *group-list*
Specify the group codes with which to permit access upon input.	**bridge-group** *bridge-group* **input-lat-service-permit** *group-list*

If a message specifies group codes in both the deny and permit list, the message is not bridged.

Specify Deny or Permit Conditions for LAT Group Codes on Output

You can specify the group codes by which to deny or permit access upon output. Specifying deny contitions causes the system to not bridge onto the output interface any LAT service advertisements that contain any of the specified groups. Specifying permit conditions causes the system to bridge onto the output interface only those service advertisements that match at least one group in the specified group-list.

To specify deny or permit conditions for LAT groups on output, perform one of the following tasks in interface configuration mode:

Task	Command
Specify the group codes with which to deny access upon output.	**bridge-group** *bridge-group* **output-lat-service-deny** *group-list*
Specify the group codes with which to permit access upon output.	**bridge-group** *bridge-group* **output-lat-service-permit** *group-list*

If a message matches both a deny and a permit condition, it will not be bridged.

Adjust Spanning-Tree Parameters

You might need to adjust certain spanning-tree parameters if the default values are not suitable for your bridge configuration. Parameters affecting the entire spanning tree are configured with variations of the **bridge** global configuration command. Interface-specific parameters are configured with variations of the **bridge-group** interface configuration command.

You can adjust spanning-tree parameters by performing any of the tasks in the following sections:

- Set the Bridge Priority
- Set an Interface Priority
- Assign Path Costs
- Adjust BPDU Intervals
- Disable the Spanning Tree on an Interface

Note Only network administrators with a good understanding of how bridges and the spanning-tree protocol work should make adjustments to spanning-tree parameters. Poorly planned adjustments to these parameters can have a negative impact on performance. A good source on bridging is the IEEE 802.1d specification; see the "References and Recommended Reading" appendix in the *Router Products Command Reference* publication for other references.

Set the Bridge Priority

You can globally configure the priority of an individual bridge when two bridges tie for position as the root bridge, or you can configure the likelihood that a bridge will be selected as the root bridge. This priority is determined by default; however, you can change it. To set the bridge priority, perform the following task in global configuration mode:

Task	Command
Set the bridge priority.	**bridge** *bridge-group* **priority** *number*

Set an Interface Priority

You can set a priority for an interface. When two bridges tie for position as the root bridge, you configure an interface priority to break the tie. The bridge with the lowest interface value is elected. To set an interface priority, perform the following task in interface configuration mode:

Task	Command
Establish a priority for a specified interface.	**bridge-group** *bridge-group* **priority** *number*

Assign Path Costs

Each interface has a path cost associated with it. By convention, the path cost is 1000/data rate of the attached LAN, in Mbps. You can set different path costs. Refer to the entry for this command in the *Router Products Command Reference* publication for the various media defaults. To assign path costs, perform the following task in interface configuration mode:

Task	Command
Set a different path cost other than the defaults.	**bridge-group** *bridge-group* **path-cost** *cost*

Adjust BPDU Intervals

You can adjust the following hello bridge protocol data unit (BPDU) intervals:

- The interval between hello BPDUs
- The forward delay interval
- The maximum idle interval

Note Each bridge in a spanning tree adopts the interval between hello BPDUs, the forward delay interval, and the maximum idle interval parameters of the root bridge, regardless of what its individual configuration might be.

Adjust the Interval between Hello BPDUs

You can specify the interval between hello BPDUs. To adjust this interval, perform the following task in global configuration mode:

Task	Command
Specify the interval between hello BPDUs.	**bridge** *bridge-group* **hello-time** *seconds*

Define the Forward Delay Interval

The forward delay interval is the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. To change the default interval setting, perform the following task in global configuration mode:

Task	Command
Set the default of the forward delay interval.	**bridge** *bridge-group* **forward-time** *seconds*

Define the Maximum Idle Interval

If a bridge does not hear BPDUs from the root bridge within a specified interval, it assumes that the network has changed and recomputes the spanning-tree topology. To change the default interval setting, performing the following task in global configuration mode:

Task	Command
Change the amount of time a bridge will wait to hear BPDUs from the root bridge.	**bridge** *bridge-group* **max-age** *seconds*

Disable the Spanning Tree on an Interface

When a *loop-free* path exists between any two bridged subnetworks, you can prevent BPDUs generated in one transparent bridging subnetwork from impacting nodes in the other transparent bridging subnetwork, yet still permit bridging throughout the bridged network as a whole. For example, when transparently bridged LAN subnetworks are separated by a WAN, BPDUs can be prevented from traveling across the WAN link.

To disable the spanning tree on an interface, perform the following task in interface configuration mode:

Task	Command
Disable the spanning tree on an interface.	**bridge-group** *bridge-group* **spanning-disabled**

Tune the Transparently Bridged Network

In the process of loop elimination, the spanning-tree algorithm always blocks all but one of a group of parallel network segments between two bridges. When those segments are of limited bandwidth, it might be preferable to augment the aggregate bandwidth between two bridges by forwarding across multiple parallel network segments. Circuit groups can be used to group multiple parallel network segments between two bridges to distribute the load while still maintaining a loop-free spanning tree.

Deterministic load distribution distributes traffic between two bridges across multiple parallel network segments grouped together into a single circuit group. This process guarantees packet ordering between source-destination pairs, and always forwards traffic for a source-destination pair on the same segment in a circuit group for a given circuit-group configuration.

Note You should configure all parallel network segments between two bridges into a single circuit group. Deterministic load distribution across a circuit group adjusts dynamically to the addition or deletion of network segments, and to interface state changes.

To tune the transparently bridged network, perform the following tasks:

Step 1 Define a circuit group.

Step 2 Optionally, configure a transmission pause interval.

Step 3 Modify the load distribution strategy.

To define a circuit group, perform the following task in interface configuration mode:

Task	Command
Add a serial interface to a circuit group.	**bridge-group** *bridge-group* **circuit-group** *circuit-group*

For circuit groups of mixed-bandwidth serial interfaces, it might be necessary to configure a pause interval during which transmission is suspended to avoid misordering packets following changes in the composition of a circuit group. Changes in the composition of a circuit group include the addition or deletion of an interface and interface state changes. To configure a transmission pause interval, perform the following task in global configuration mode:

Task	Command
Configure a transmission pause interval.	**bridge** *bridge-group* **circuit-group** *circuit-group* **pause** *milliseconds*

For applications that depend on the ordering of mixed unicast and multicast traffic from a given source, load distribution must be based upon the source MAC address only. To modify the load distribution strategy to accommodate such applications, perform the following task in global configuration mode:

Task	Command
Base load distribution on the source MAC address only.	**bridge** *bridge-group* **circuit-group** *circuit-group* **source-based**

For an example of how to configure a circuit group, see the "Complex Transparent Bridging Network Topology Example" section later in this chapter.

Monitor and Maintain the Transparent Bridge Network

This section describes how to monitor and maintain activity on the bridged network. You can perform one or more of the following tasks in privileged EXEC mode:

Task	Command
Remove any learned entries from the forwarding database and clear the transmit and receive counts for any statically configured forwarding entries.	**clear bridge** *bridge-group*
Reinitialize the Silicon Switch Processor (SSP) on the Cisco 7000 series.	**clear sse**
Display classes of entries in the bridge forwarding database.	**show bridge** [*bridge-group*] [*interface*] [*address*] [*mask*]
Display the interfaces configured in each circuit group and show whether they are participating in load distribution.	**show bridge** [*bridge-group*] **circuit-group** [*circuit-group*] [*src-mac-address*] [*dst-mac-address*]
Display the spanning-tree topology known to the router/bridge, including whether or not filtering is in effect.	**show span**
Display a summary of SSP statistics.	**show sse summary**

Transparent and SRT Bridging Configuration Examples

The following sections provide example configurations that you can use as a guide to configuring your bridging environment:

- Basic Bridging Example
- Transparent Bridging Example
- Ethernet Bridging Example
- SRT Bridging Example
- Multicast or Broadcast Packets Bridging Example
- X.25 Transparent Bridging Example
- Frame-Relay Transparent Bridging Examples
- Complex Transparent Bridging Network Topology Example

Basic Bridging Example

Figure 21-1 is an example of a basic bridging configuration. The system has two Ethernets, one Token Ring, one FDDI port, and one serial line. The IP is being routed, and everything else is being bridged. The Digital-compatible bridging algorithm with default parameters is being used.

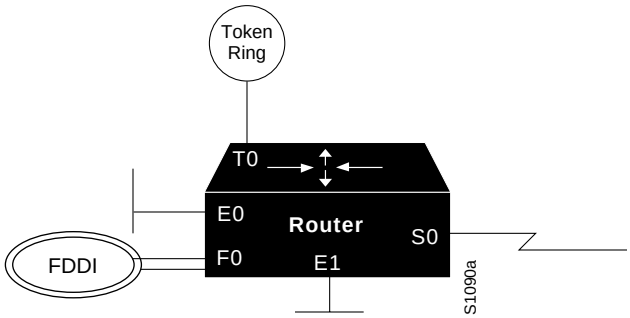


Figure 21-1 Example of Basic Bridging

The configuration file for the router/bridge depicted in Figure 21-1would be as follows:

```
interface tokenring 0
ip address 131.108.1.1 255.255.255.0
bridge-group 1
!
interface fddi 0
ip address 131.108.2.1 255.255.255.0
bridge-group 1
!
interface ethernet 0
ip address 192.31.7.26 255.255.255.240
bridge-group 1
!
interface serial 0
ip address 192.31.7.34 255.255.255.240
bridge-group 1
!
interface ethernet 1
ip address 192.31.7.65 255.255.255.240
bridge-group 1
!
bridge 1 protocol dec
```

Transparent Bridging Example

The following configuration example shows the configuration commands that enable transparent bridging between Ethernet and FDDI interfaces. Transparent bridging on an FDDI interface is allowed only on the CSC-C2FCIT interface card.

```
hostname tester
!
buffers small min-free 20
buffers middle min-free 10
buffers big min-free 5
!
no ip routing
!
interface Ethernet 0
ip address 131.108.7.207 255.255.255.0
no ip route-cache
bridge-group 1
!
interface Ethernet 2
ip address 131.108.7.208 255.255.255.0
no ip route-cache
bridge-group 1
!
interface Fddi 0
ip address 131.108.7.209 255.255.255.0
no ip route-cache
no keepalive
bridge-group 1
!
bridge 1 protocol ieee
```

If the other side of the FDDI ring were an FDDI interface running in encapsulation mode rather than in transparent mode, the following additional configuration commands would be needed:

```
interface fddi 0
fddi encapsulate
```

Ethernet Bridging Example

In the following example, two buildings have networks that must be connected via a T1 link. For the most part, the systems in each building use either IP or DECnet, and therefore, should be routed. There are some systems in each building that must communicate, but they can use only a proprietary protocol.

The example places two Ethernets in each building. One of the Ethernets is attached to the hosts that use a proprietary protocol, and the other is used to attach to the rest of the building network running IP and DECnet. The Ethernet attached to the hosts using a proprietary protocol is enabled for bridging to the serial line and to the other building.

Figure 21-2 shows an example configuration. The interfaces marked with an asterisk (*) are configured as part of spanning tree 1. The routers are configured to route IP and DECnet. This configuration permits hosts on any Ethernet to communicate with hosts on any other Ethernet using IP or DECnet. In addition, hosts on Ethernet 1 in either building can communicate using protocols not supported for routing.

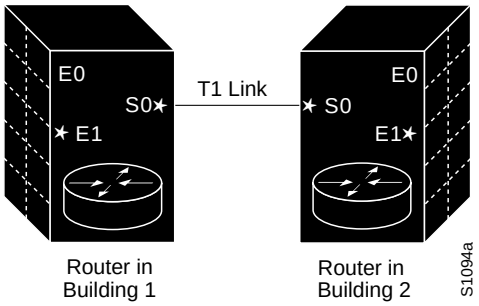


Figure 21-2 Ethernet Bridging Configuration Example

The configuration file for the router/bridge in Building 1 would be as follows. Note that no bridging takes place over Ethernet 0. Both IP and DECnet routing are enabled on all interfaces.

```
decnet address 3.34
interface ethernet 0
ip address 128.88.1.6 255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.1 255.255.255.0
bridge-group 1
decnet cost 10
!
interface ethernet 1
ip address 128.88.3.1 255.255.255.0
bridge-group 1
decnet cost 10
!
bridge 1 protocol dec
```

The configuration file for the router/bridge in Building 2 is similar:

```
decnet address 3.56
!
interface ethernet 0
ip address 128.88.11.9 255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.2 255.255.255.0
bridge-group 1
decnet cost 10
!
interface ethernet 1
ip address 128.88.16.8 255.255.255.0
bridge-group 1
decnet cost 10
!
bridge 1 protocol dec
```

SRT Bridging Example

In Figure 21-3, a Token Ring and an Ethernet at a remote sales site in New York City must be configured to pass unroutable bridged traffic across a satellite link to the backbone Token Ring at the corporate headquarters in Thule, Greenland. IP is the only routed protocol. They are running the IEEE spanning-tree protocol to comply with the SRT bridging standard.

If there were source-routed traffic to bridge, the **source-bridge** command would also be used to configure source routing.

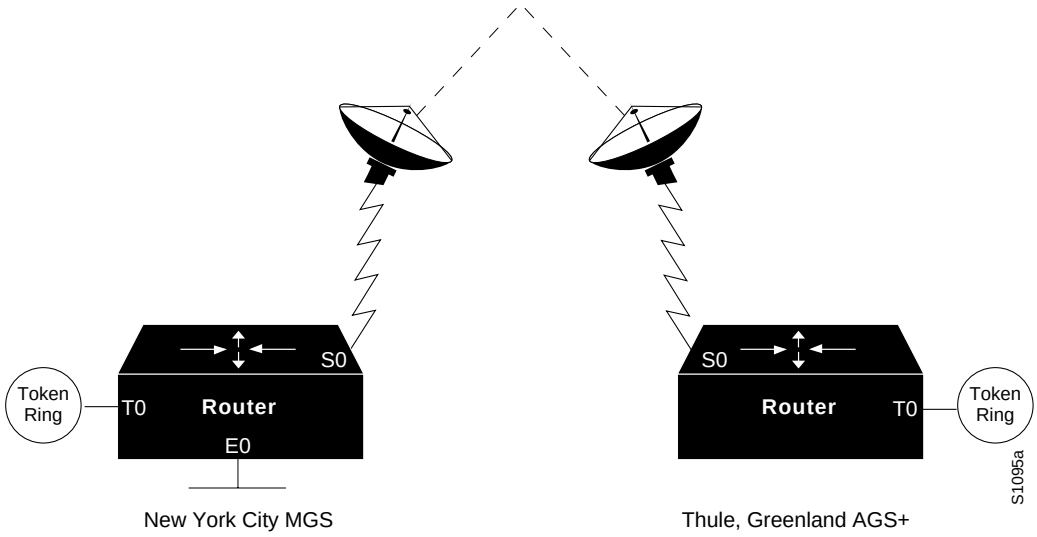


Figure 21-3 Example Network Configuration

Configuration for the New York City Router

```
interface tokenring 0
ip address 150.136.1.1 255.255.255.128
bridge-group 1
!
interface ethernet 0
ip address 150.136.2.1 255.255.255.128
bridge-group 1
!
interface serial 0
ip address 150.136.3.1 255.255.255.128
bridge-group 1
!
bridge 1 protocol ieee
```

Configuration for the Thule, Greenland Router

```
interface tokenring 0
ip address 150.136.10.1 255.255.255.128
bridge-group 1
!
interface serial 0
ip address 150.136.11.1 255.255.255.128
bridge-group 1
!
bridge 1 protocol ieee
```

Multicast or Broadcast Packets Bridging Example

When filtering specific MAC destination addresses, allow for multicast or broadcast packets that are required by the bridged network protocols.

Assume you are bridging IP in your network as illustrated in Figure 21-4.

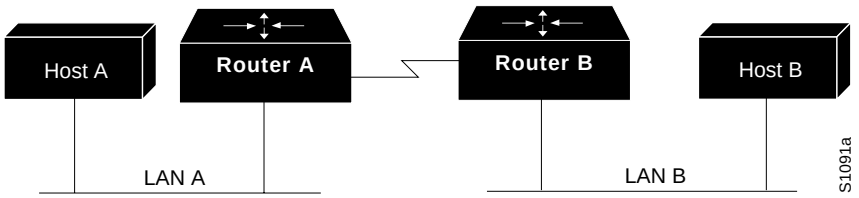


Figure 21-4 Network Demonstrating Output Address List Filtering

The MAC address of Host A is 0800.0907.0207, and the MAC address of Host B is 0260.8c34.0864. The following configuration would work as expected, because input addresses work on the source address on the incoming interface:

```
access-list 700 permit 0260.8c34.0864 0000.0000.0000
access-list 700 deny 0000.0000.0000 FFFF.FFFF.FFFF
interface ethernet 0
bridge-group 1 input-address-list 700
```

However, the following configuration might work initially but will eventually fail. The failure occurs because the configuration does not allow for an ARP broadcast with a destination address of FFFF.FFFF.FFFF, even though the destination address on the output interface is correct:

```
access-list 700 permit 0260.8c34.0864 0000.0000.0000
access-list 700 deny 0000.0000.0000 FFFF.FFFF.FFFF
interface ethernet 0
bridge-group 1 output-address-list 700
```

The correct access list would be as follows:

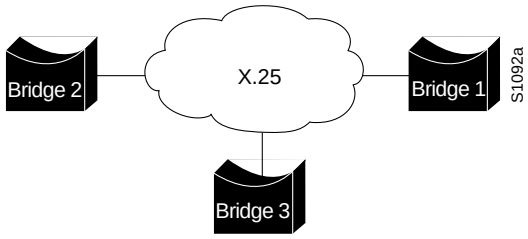
```
access-list 700 permit 0260.8c34.0864 0000.0000.0000
access-list 700 permit FFFF.FFFF.FFFF 0000.0000.0000
access-list 700 deny 0000.0000.0000 FFFF.FFFF.FFFF
interface ethernet 0
bridge-group 1 output-address-list 700
```

X.25 Transparent Bridging Example

Figure 21-5 is an example configuration illustrating three bridges connected to each other through an X.25 network.

Figure 21-5 X.25 Bridging Example

Following are the configuration commands for each of the bridges depicted in Figure 21-5.



Configuration for Bridge 1

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9 255.255.255.0
!
interface serial 0
encapsulation x25
x25 address 31370019027
bridge-group 5
x25 map bridge 31370019134 broadcast
x25 map bridge 31370019565 broadcast
!
bridge 5 protocol ieee
```

Configuration for Bridge 2

```
interface serial 1
encapsulation x25
x25 address 31370019134
bridge-group 5
x25 map bridge 31370019027 broadcast
x25 map bridge 31370019565 broadcast
!
bridge 5 protocol ieee
```

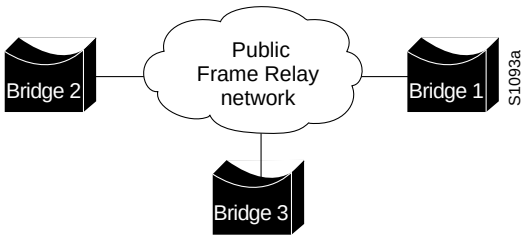
Configuration for Bridge 3

```
interface serial 0
encapsulation x25
x25 address 31370019565
bridge-group 5
x25 map bridge 31370019027 broadcast
x25 map bridge 31370019134 broadcast
!
bridge 5 protocol ieee
```

Frame-Relay Transparent Bridging Examples

Figure 21-6 illustrates three bridges connected to each other through a Frame Relay network.

Figure 21-6 Frame-Relay Bridging Example



Bridging in a Frame Relay Network with No Multicasts

The Frame Relay bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated for transmission across a Frame Relay network. The command specifies Internet-to-DLCI address mapping and maintains a table of both the Ethernet and DLCIs.

Following are the configuration commands for each of the bridges in a network that does not support a multicast facility.

Configuration for Bridge 1

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9 255.255.255.0
!
interface serial 0
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 134 broadcast
frame-relay map bridge 565 broadcast
!
bridge 5 protocol ieee
```

Configuration for Bridge 2

```
interface serial 1
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 27 broadcast
frame-relay map bridge 565 broadcast
!
bridge 5 protocol ieee
```

Configuration for Bridge 3

```
interface serial 0
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 27 broadcast
frame-relay map bridge 134 broadcast
!
bridge 5 protocol ieee
```

Bridging in a Frame Relay Network with Multicasts

The multicast facility is used to learn about the other bridges on the network, eliminating the need for the **frame-relay map** commands.

Following are the configuration commands for each of the bridges in a network that supports a multicast facility.

Configuration for Bridge 1

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9 255.255.255.0
!
interface serial 0
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
```

Configuration for Bridge 2

```
interface serial 1
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
```

Configuration for Bridge 3

```
interface serial 0
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
```

Complex Transparent Bridging Network Topology Example

Figure 21-7 shows a network topology made up of four bridged subnetworks. Each bridged subnetwork is defined by the scope of a spanning tree. However, the scope of each spanning tree is not shown in detail because it is unnecessary for purposes of this discussion. Instead, it is shown by a half cloud labeled “To other parts of BSN.”

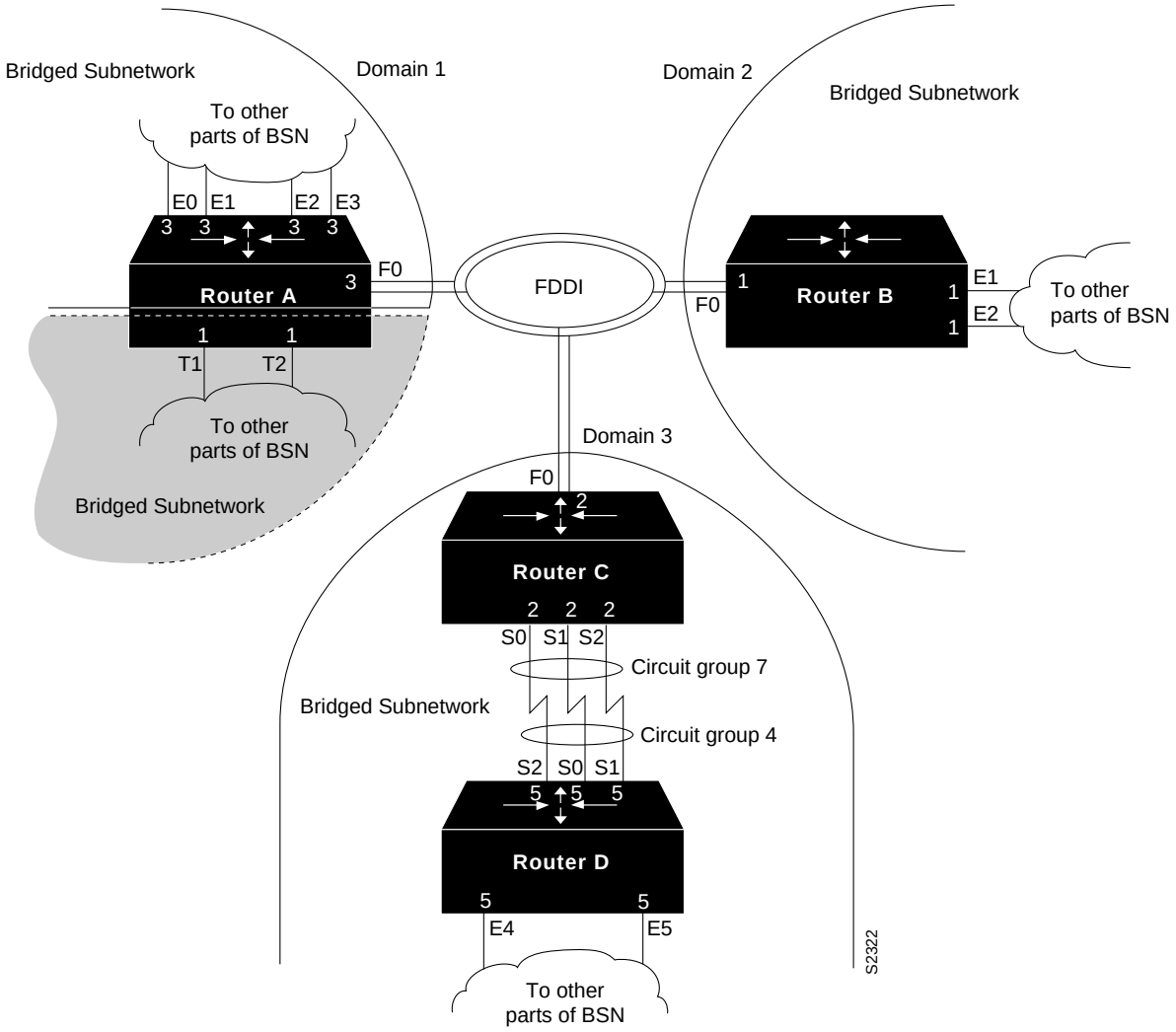


Figure 21-7 Bridged Subnetworks with Domains

For proper bridging operation, the bridged subnetworks cannot have connections between them, but they can be connected to the same backbone. In this example, three of the four bridged subnetworks are connected to the FDDI backbone and each belongs to a separate domain. Domains used in this topology allow the bridged subnetworks to be independent of one another while still bridging traffic onto the backbone destined for other connected bridged subnetworks. Domains can only be used in this manner if the bridged subnetworks have a single point of attachment to one another. In this case, the connection to the FDDI backbone is that single point of attachment. Each router on which a domain is configured and that has a single point of attachment to the other bridged subnetworks, checks whether a BPDU on the backbone is its own. If the BPDU does not belong to the bridged subnetwork, the router ignores the BPDU.

Separate bridged subnetworks, as in this example, allow spanning-tree reconfiguration of individual bridged subnetworks without disrupting bridging among the other bridged subnetworks.

Note To get spanning-tree information by bridge group, use the **show span** command. Included in this information is the root bridge of the spanning tree. The root bridge for each spanning tree can be any router in the spanning tree. Refer to the *Internetworking Technology Overview* publication for information about root bridge election.

The routers in this network are configured for bridging and demonstrate some of the bridging features available.

Configuration for Router A

Router A demonstrates multiple bridge groups in one router for bridged traffic separation.

In Router A, the Token Ring interfaces are bridged together entirely independently of the other bridged interfaces in the router and belong to bridge group 1. Bridge group 1 does not use a bridge domain because the interfaces are bridged independently of other bridged subnetworks in the network topology and it has no connection to the FDDI backbone.

Also in Router A, the Ethernet interfaces belong to bridge group 3. Bridge group 3 has a connection to the FDDI backbone and has a domain defined for it so that it can ignore BPDUs for other bridged subnetworks.

```
interface Ethernet0
bridge-group 3
!
interface Ethernet1
bridge-group 3
!
interface Ethernet2
bridge-group 3
!
interface Ethernet3
bridge-group 3
!
interface Fddi0
bridge-group 3
!
interface TokenRing1
bridge-group 1
!
interface TokenRing2
bridge-group 1
!
bridge 1 protocol ieee
bridge 3 domain 1
bridge 3 protocol ieee
```

Configuration for Router B

Router B demonstrates a simple bridge configuration. It is connected to the FDDI backbone and has domain 2 defined. As such it can bridge traffic with the other FDDI-connected BSNs. Note that bridge group 1 has no relationship to bridge group 1 in Router A; bridge groups are an organization internal to each router.

```
interface Ethernet1
bridge-group 1
!
interface Ethernet2
bridge-group 1
!
interface Fddi0
bridge-group 1
!
bridge 1 domain 2
bridge 1 protocol ieee
```

Configuration for Router C

Router C and Router D combine to demonstrate load balancing by means of circuit groups. Circuit groups are used to load balance across multiple parallel serial lines between a pair of routers. The router on each end of the serial lines must have a circuit group defined. The circuit group number can be the same or can be different. In this example, they are different.

Router C and Router D are configured with the same domain, because they must understand one another's BPDUs. If they were configured with separate domains, Router D would ignore Router C's BPDUs and vice versa.

```
interface Fddi0
bridge-group 2
!
interface Serial0
bridge-group 2
bridge-group 2 circuit 7
!
interface Serial1
bridge-group 2
bridge-group 2 circuit 7
!
interface Serial2
bridge-group 2
bridge-group 2 circuit 7
!
bridge 2 domain 3
bridge 2 protocol ieee
```

Configuration for Router D

```
interface Ethernet4
bridge-group 5
!
interface Ethernet5
bridge-group 5
!
interface Serial0
bridge-group 5
bridge-group 5 circuit 4
!
interface Serial1
bridge-group 5
bridge-group 5 circuit 4
!
interface Serial2
bridge-group 5
bridge-group 5 circuit 4
!
bridge 5 domain 3
bridge 5 protocol ieee
```