

Configuring Transparent Bridging

Our Cisco IOS software bridging functionality 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 describes how to configure transparent bridging and source-route transparent (SRT) bridging. This chapter also describes the concepts of virtual networking, transparent bridging of virtual LANs (VLANs), and routing between VLANs. For a complete description of the commands mentioned in this chapter, refer to the “Transparent Bridging Commands” chapter in the *Bridging and IBM Networking Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

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:

- Configure Transparent Bridging and SRT Bridging
- Configure Transparently Bridged Virtual LANs (VLANs)
- Configure Routing between VLANs
- Configure Transparent Bridging over WANs
- Configure Concurrent Routing and Bridging
- Configure Integrated Routing and Bridging
- 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 “Transparent and SRT Bridging Configuration Examples” section 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 <i>bridge-group</i> 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 be a member of only 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 <i>bridge-group</i> |

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 Transparently Bridged Virtual LANs (VLANs)

Traditionally, a bridge group is an independently bridged subnetwork. In this definition, bridge groups cannot exchange traffic with other bridge groups, nor can they multiplex or demultiplex different streams of bridged traffic. Our transparently bridged VLAN feature permits a bridge group to extend outside the router to identify traffic switched within the bridge group.

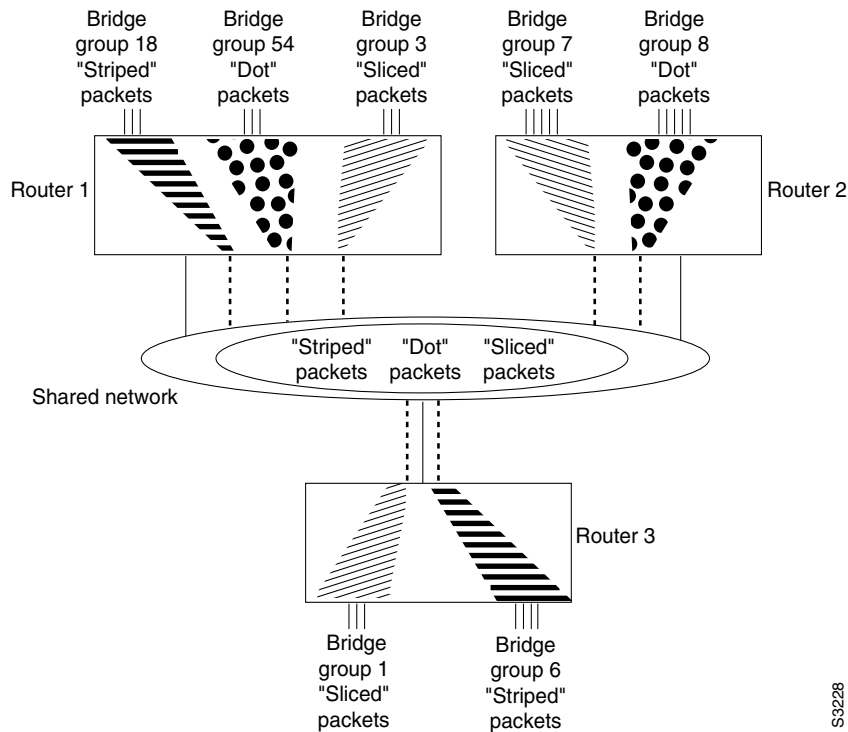
While bridge groups remain internal organizations of network interfaces functioning as distinct bridges within a router, transparent bridging on subinterfaces permits bridge groups to be used to multiplex different streams of bridged traffic on a LAN or HDLC serial interface. In this way, bridged traffic may be switched out of one bridge group on one router, multiplexed across a subinterface, and demultiplexed into a second bridge group on a second router. Together, the first bridge group and the second bridge group form a transparently bridged VLAN. This approach can be extended to impose logical topologies upon transparently bridged networks.

The primary application of transparently bridged VLANs constructed in this way is to separate traffic between bridge groups of local network interfaces, to multiplex bridged traffic from several bridge groups on a shared interface (LAN or HDLC serial), and to form VLANs composed of collections of bridge groups on several routers. These VLANs improve performance because they reduce the propagation of locally bridged traffic, and they improve security benefits because they completely separate traffic.

In Figure 32, different bridge groups on different routers are configured into three VLANs that span the bridged network. Each bridge group consists of conventionally bridged local interfaces and a subinterface on the backbone FDDI LAN. Bridged traffic on the subinterface is encapsulated and “colored” with a VLAN identifier known as a *security association identifier* common to all bridge groups participating in the VLAN. In addition, bridges only accept packets bearing security association identifiers for which they have a configured subinterface. Thus, a bridge group is configured to participate in a VLAN if it contains a subinterface configured with the VLAN’s characteristic security association identifier. See the section “Transparently Bridged VLANs Configuration Example” later in this chapter for an example configuration of the topology shown in Figure 32.

Note The 802.10 encapsulation used to “color” transparently bridged packets on subinterfaces might increase the size of a packet so that it exceeds the MTU size of the LAN from which the packet originated. To avoid MTU violations on the shared network, the originating LANs must either have a smaller native MTU than the shared network (as is the case from Ethernet to FDDI), or the MTU on all packet sources on the originating LAN must be configured to be at least 16 bytes less than the MTU of the shared network.

Figure 32 Transparently Bridged VLANs on an FDDI Backbone



To configure a VLAN on a transparently bridged network, perform the following tasks, beginning in interface configuration mode:

| Task | Command |
|--|--|
| Specify a subinterface. | interface <i>type slot/port.subinterface-number</i> |
| Specify the IEEE 802.10 Security data exchange security association identifier. (In other words, specify the “color.”) | encapsulation sde <i>said</i> |
| Associate the subinterface with an existing bridge group. | bridge-group <i>bridge-group</i> |

Note Transparently bridged VLANs are supported in conjunction with only the IEEE Spanning-Tree Protocol. When you logically segment a transparently bridged network into VLANs, each VLAN computes its own spanning-tree topology. Configuring each VLAN to compute its own spanning-tree topology provides much greater stability than running a single spanning tree throughout. Traffic bridged within one VLAN is unaffected by physical topology changes occurring within another VLAN.

Note The current implementation of SDE encapsulation is not recommended for serial or Ethernet media.

Configure Routing between VLANs

Virtual networking provides a mechanism whereby you can define logical topologies to overlay a physical switched infrastructure, and so establish autonomous VLAN domains. By definition, VLANs provide traffic separation and logical network partitioning. To communicate between VLANs a routing function is required.

Note Our VLAN Routing implementation is designed to operate across all router platforms. However, the Inter-Switch Link (ISL) VLAN trunking protocol currently is defined on 100 BaseTX/FX Fast Ethernet interfaces only and therefore is appropriate to the Cisco 7000 and higher-end platforms only. The IEEE 802.10 protocol can run over any LAN or HDLC serial interface. VLAN traffic is fast switched. The actual format of these VLAN encapsulations are detailed in the *IEEE Standard 802.10-1992 Secure Data Exchange* and in the *Inter-Switch Link (ISL) Protocol Specification*.

Our VLAN Routing implementation treats the ISL and 802.10 protocols as encapsulation types. On a physical router interface that receives and transmits VLAN packets, you can select an arbitrary subinterface and map it to the particular VLAN “color” embedded within the VLAN header. This mapping allows you to selectively control how LAN traffic is routed or switched outside of its own VLAN domain. In the VLAN routing paradigm, a switched VLAN corresponds to a single routed subnet, and the network address is assigned to the subinterface.

To route a received VLAN packet the Cisco IOS software VLAN switching code first extracts the VLAN ID from the packet header (this is a 10-bit field in the case of ISL and a 4-byte entity known as the security association identifier in the case of IEEE 802.10), then demultiplexes the VLAN ID value into a subinterface of the receiving port. If the VLAN color does not resolve to a subinterface, the Cisco IOS software can transparently bridge the foreign packet natively (without modifying the VLAN header) on the condition that the Cisco IOS software is configured to bridge on the subinterface itself. For VLAN packets that bear an ID corresponding to a configured subinterface, received packets are then classified by protocol type before running the appropriate protocol specific fast switching engine. If the subinterface is assigned to a bridge group then non-routed packets are de-encapsulated before they are bridged. This is termed “fall-back bridging” and is most appropriate for non-routable traffic types.

In Figure 33, Router A provides inter-VLAN connectivity between multiple Cisco switching platforms where there are three distinct virtual topologies present. For example, for VLAN 300 across the two Catalyst 1200A segments, traffic originating on LAN interface 1 is “tagged” with a VLAN ID of 300 as it is switched onto the FDDI ring. This ID allows the remote Catalyst 1200A to

make an intelligent forwarding decision and only switch the traffic to local interfaces configured as belonging to the same VLAN broadcast domain. Router A provides an inter-VLAN mechanism that lets Router A function as a gateway for stations on a given LAN segment by transmitting VLAN encapsulated traffic to and from other switched VLAN domains or simply transmitting traffic in native (non-VLAN) format.

Figure 33 Inter-VLAN Connectivity between Multiple Switching Platforms

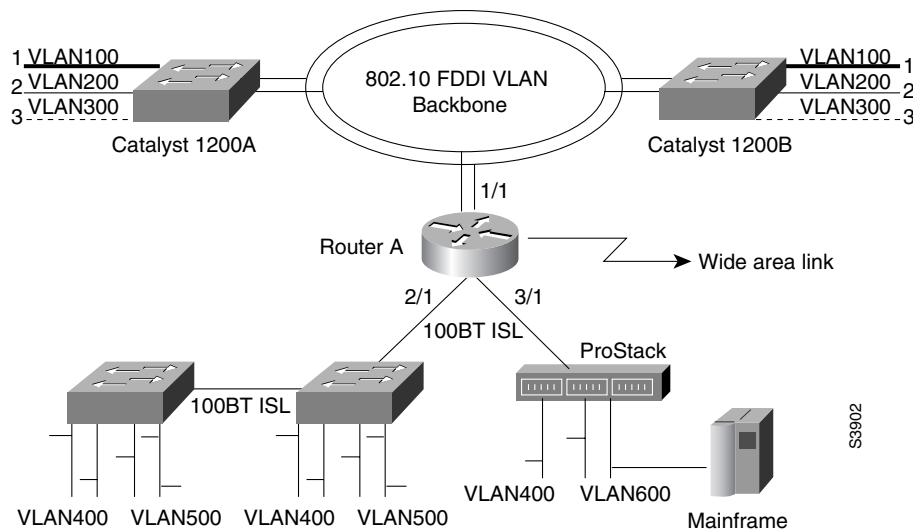


Figure 33 illustrates the following scenarios:

- Clients on VLAN 300 want to establish sessions with a server attached to a port in a different VLAN (600). In this scenario, packets originating on LAN interface 3 of the Catalyst 1200B switch are tagged with an 802.10 header with a security association identifier of 300 as they are forwarded onto the FDDI ring. Router A can accept these packets because it is configured to route VLAN 300, classify and make a layer 3 forwarding decision based on the destination network address and the route out (in this case Fast Ethernet 3/1), and adding the ISL VLAN header (color 200) appropriate to the destination subnet as the traffic is switched.
- There is a network requirement to bridge two VLANs together through the system rather than selectively route certain protocols. In this scenario the two VLAN IDs are placed in the same bridge group. Note that they form a single broadcast domain and spanning tree, effectively forming a single VLAN.

See the section “Routing Between VLANs Configuration Example” later in this chapter for an example configuration of the topology shown in Figure 33.

To configure routing between VLANs, perform the following tasks, beginning in interface configuration mode:

| Task | Command |
|---|--|
| Specify a subinterface. | interface <i>type slot/port.subinterface-number</i> |
| Specify the encapsulation type (either ISL or SDE) and the VLAN domain. | encapsulation { <i>sde isl</i> } <i>domain</i> |
| Associate the subinterface with the VLAN. | bridge-group <i>bridge-group</i> |

Configure Transparent Bridging over WANs

You can configure transparent bridging over a variety of networks, as described in the following sections:

- Configure Fast-Switched Transparent Bridging over ATM
- Configure Transparent Bridging over DDR
- Configure Transparent Bridging over Frame Relay
- Configure Transparent Bridging over Multiprotocol L2PB
- Configure Transparent Bridging over SMDS
- Configure Transparent Bridging over X.25

Configure Fast-Switched Transparent Bridging over ATM

Our bridging implementation supports IEEE 802.3 frame formats and IEEE 802.10 frame formats. Our implementation can transparently bridge ARPA style ethernet packets (also known as ethernet version 2).

Fast-switched transparent bridging over ATM supports AAL5-SNAP encapsulated packets only. All bridged AAL5-SNAP encapsulated packets are fast switched. Fast-switched transparent bridging supports Ethernet, FDDI, and Token Ring packets sent in AAL5-SNAP encapsulation over ATM. See the section “Fast-Switched Transparent Bridging over ATM Example (Cisco 7000)” for an example configuration of fast-switched transparent bridging over ATM.

For more information on configuring ATM, refer to the “Configuring ATM” chapter in the *Wide-Area Networking Configuration Guide*.

Configure Transparent Bridging over DDR

The Cisco IOS software supports transparent bridging over DDR and provides you some flexibility in controlling access and configuring the interface.

To configure DDR for bridging, complete the tasks in the following sections:

- Define the Protocols to Bridge
- Specify the Bridging Protocol
- Determine Access for Bridging
- Configure an Interface for Bridging

For an example of configuring transparent bridging over DDR, see the section “Transparent Bridging over DDR Examples” section.

Define the Protocols to Bridge

IP packets are routed by default unless they are explicitly bridged; all others are bridged by default unless they are explicitly routed.

To bridge IP packets, complete the following task in global configuration mode:

| Task | Command |
|---------------------|----------------------|
| Disable IP routing. | no ip routing |

If you choose *not* to bridge another protocol, use the relevant command to enable routing of that protocol. For more information about tasks and commands, refer to the relevant protocol chapter in either the *Network Protocols Configuration Guide, Part 1*, the *Network Protocols Configuration Guide, Part 2*, or the *Network Protocols Configuration Guide, Part 3*.

Specify the Bridging Protocol

You must specify the type of spanning-tree bridging protocol to use and also identify a bridge group. To specify the Spanning-Tree Protocol and a bridge group number, complete the following task in global configuration mode:

| Task | Command |
|--|--|
| Define the type of Spanning-Tree Protocol and identify a bridge group. | bridge <i>bridge-group</i> protocol { ieee dec } |

The bridge-group number is used when you configure the interface and assign it to a bridge group. Packets are bridged only among members of the same bridge group.

Determine Access for Bridging

You can determine access by either permitting all bridge packets or by controlling access according to Ethernet type codes.

To permit all transparent bridge packets, complete the following task in global configuration mode:

| Task | Command |
|---|--|
| Define a dialer list that permits all transparent bridge packets. | dialer-list <i>dialer-group</i> protocol bridge permit |

To control access by Ethernet type codes, complete the following tasks in global configuration mode:

| Task | Command |
|---|---|
| Permit packets according to Ethernet type codes (access list numbers must be in the range 200–299). | access-list <i>access-list-number</i> { permit deny } <i>type-code</i> [<i>mask</i>] |
| Define a dialer list for the specified access list. | dialer-list <i>dialer-group</i> protocol bridge list <i>access-list-number</i> |

For a table of some common Ethernet types codes, see the “Ethernet Types Codes” appendix in the *Bridging and IBM Networking Command Reference*.

Configure an Interface for Bridging

You can configure serial interfaces or ISDN interfaces for DDR bridging. To configure an interface for DDR bridging, complete the following tasks, starting in global configuration mode:

| Task | Command |
|--|---|
| Specify the serial or ISDN interface and enter interface configuration mode. | interface <i>type number</i> |
| Configure the dial string to call. or Configure a dialer bridge map. | dialer string <i>dial-string</i> dialer map bridge [<i>name hostname</i>] [broadcast] <i>dial-string[:isdn-subaddress]</i> |
| Assign the specified interface to a bridge group. | bridge-group <i>bridge-group</i> |

Configure Transparent Bridging over Frame Relay

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.

Fast-Switched Transparent Bridging

The transparent bridging software provides fast-switched transparent bridging for Frame Relay encapsulated serial and High-Speed Serial Interface (HSSI) networks.

SVCs are not supported for transparent bridging in this release. All the PVCs configured on a subinterfaces must belong to the same bridge group.

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 <i>dldci broadcast</i> |

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 “Configuring Frame Relay” chapter in the *Wide-Area Networking Configuration Guide*.

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 “Configuring Frame Relay” chapter in the *Wide-Area Networking Configuration Guide*.

Configure Transparent Bridging over Multiprotocol LAPB

Our software implements transparent bridging over multiprotocol LAPB encapsulation on serial interfaces. To configure transparent bridging over multiprotocol LAPB, perform the following tasks, beginning in global configuration mode:

| Task | Command |
|---|---|
| Specify the serial interface. | interface serial <i>number</i> |
| Specify no IP address to the interface. | no ip address |
| Configure multiprotocol LAPB encapsulation. | encapsulation lpb multi |
| Assign the interface to a bridge group. | bridge-group <i>bridge-group</i> |
| Specify the type of Spanning-Tree Protocol. | bridge <i>bridge-group</i> protocol {ieee dec} |

Note Transparent bridging over multiprotocol LAPB requires use of the **encapsulation lpb multi** command. You cannot use the **encapsulation lpb protocol** command with a **bridge** keyword to configure this feature.

For an example of configuring transparent bridging over multiprotocol LAPB, see the section “Transparent Bridging over Multiprotocol LAPB Example” later in this chapter.

Configure Transparent Bridging over SMDS

We support fast-switched transparent bridging for Switched Multimegabit Data Service (SMDS) encapsulated serial and HSSI networks. Standard bridging commands are used to enable bridging on an SMDS interface.

To enable transparent bridging over SMDS, perform the following tasks, beginning in interface configuration mode:

| Task | Command |
|--|--|
| Specify the serial interface. | interface serial <i>number</i> |
| Configure SMDS encapsulation on the serial interface. | encapsulation smds |
| Associate the interface with a bridge group. | bridge-group <i>bridge-group</i> |
| Enable transparent bridging of packets across an SMDS network. | smds multicast bridge <i>smds-address</i> |

Broadcast Address Resolution Protocol (ARP) packets are treated differently in transparent bridging over an SMDS network than in 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.

Our implementation of IEEE 802.6i transparent bridging for SMDS supports 802.3, 802.5, and FDDI frame formats. The router can accept frames with or without frame check sequence (FCS). Fast-switched transparent bridging is the default and is not configurable. If a packet cannot be fast switched, it is process switched.

An example configuration is provided in the section “Fast-Switched Transparent Bridging over SMDS Example” later in this chapter. For more information on SMDS, refer to the “Configuring SMDS” chapter in the *Wide-Area Networking Configuration Guide*.

Configure Transparent Bridging over X.25

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 IP-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 <i>x.121-address</i> broadcast [<i>options-keywords</i>] |

For more information about configuring X.25, refer to the “Configuring X.25 and LAPB” chapter in the *Wide-Area Networking Configuration Guide*.

Configure Concurrent Routing and Bridging

You can configure the Cisco IOS software to route a given protocol among one group of interfaces and concurrently bridge that protocol among a separate group of interfaces, all within one router. The given protocol is not switched between the two groups. Rather, routed traffic is confined to the routed interfaces and bridged traffic is confined to the bridged interfaces. A protocol may be either routed or bridged on a given interface, but not both.

The concurrent routing and bridging capability is, by default, disabled. While concurrent routing and bridging is disabled, the Cisco IOS software absorbs and discards bridgeable packets in protocols that are configured for routing on any interface in the router.

When concurrent routing and bridging is first enabled in the presence of existing bridge groups, it will generate a bridge route configuration command for any protocol for which any interface in the bridge group is configured for routing. This is a precaution that applies only when concurrent routing and bridging is not already enabled, bridge groups exist, and the **bridge crb** command is encountered.

To enable concurrent routing and bridging in the Cisco IOS software, perform the following task in global configuration mode:

| Task | Command |
|---|-------------------|
| Enable concurrent routing and bridging. | bridge crb |

Information about which protocols are routed and which are bridged is stored in a table, which can be displayed with the **show interfaces crb** privileged EXEC command.

When concurrent routing and bridging has been enabled, you must configure an explicit bridge route command for any protocol that is to be routed on the interfaces in a bridge group in addition to any required protocol-specific interface configuration.

To configure specific protocols to be routed in a bridge group, perform the following task in interface configuration mode:

| Task | Command |
|--|---|
| Specify a protocol to be routed on a bridge group. | bridge bridge-group route protocol |

Configure Integrated Routing and Bridging

Perform one or more of the following tasks to configure integrated routing and bridging on your router:

- Assign a Bridge Group Number and Define the Spanning-Tree Protocol
- Configure Interfaces
- Enable Integrated Routing and Bridging
- Configure the Bridge-Group Virtual Interface
- Configure Protocols for Routing or Bridging

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

Prior to configuring the router for integrated routing and bridging, you must enable bridging by setting up a bridge group number and specifying a Spanning-Tree Protocol. 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 <i>bridge-group</i> 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.

Configure Interfaces

To configure a router interface in the Cisco IOS software, perform the following tasks, starting in global configuration mode:

| Task | Command |
|---|---|
| Specify the interface and enter interface configuration mode. | interface <i>type number</i> |
| Assign bridge-groups to appropriate interfaces. | bridge-group <i>bridge-group</i> |

Enable Integrated Routing and Bridging

After you have set up the interfaces in the router, you can enable integrated routing and bridging.

To enable integrated routing and bridging in the Cisco IOS software, perform the following task in global configuration mode:

| Task | Command |
|---|-------------------|
| Enable integrated routing and bridging. | bridge irb |

Use the **show interfaces irb** privileged EXEC command to display the protocols that a given bridged interface can route to the other routed interface when the packet is routable, and to display the protocols that a given bridged interface bridges.

Configure the Bridge-Group Virtual Interface

The bridge-group virtual interface resides in the router. It acts like a normal routed interface that does not support bridging, but represents the entire corresponding bridge group to routed interfaces within the router. The bridge-group virtual interface is assigned the number of the bridge group that it represents. The bridge-group virtual interface number is the link between the bridge-group virtual interface and its bridge group. Because the bridge-group virtual interface is a virtual routed interface, it has all the network layer attributes, such as a network address and the ability to perform filtering. Only one bridge-group virtual interface is supported for each bridge group.

When you enable routing for a given protocol on the bridge-group virtual interface, packets coming from a routed interface but destined for a host in a bridged domain are routed to the bridge-group virtual interface, and are forwarded to the corresponding bridged interface. All traffic routed to the bridge-group virtual interface is forwarded to the corresponding bridge group as bridged traffic. All routable traffic received on a bridged interface is routed to other routed interfaces as if it is coming directly from the bridge-group virtual interface.

To create a bridge-group virtual interface, perform the following task in interface configuration mode:

| Task | Command |
|--|--|
| Enable a bridge-group virtual interface. | interface bvi <i>bridge-group</i> |

When you intend to bridge and route a given protocol in the same bridge group, you must configure the network-layer attributes of the protocol on the bridge-group virtual interface. Do not configure protocol attributes on the bridged interfaces. No bridging attributes can be configured on the bridge-group virtual interface.

Although it is generally the case that all bridged segments belonging to a bridge group are represented as a single segment or network to the routing protocol, there are situations where several individual networks coexist within the same bridged segment. To make it possible for the routed domain to learn about the other networks behind the bridge-group virtual interface, configure a secondary address on the bridge-group virtual interface to add the corresponding network to the routing process.

Configure Protocols for Routing or Bridging

When integrated routing and bridging is enabled, the default route/bridge behavior in a bridge group is to bridge all packets.

You could then explicitly configure the bridge group to route a particular protocol, so that routable packets of this protocol are routed, while non-routable packets of this protocol or packets for protocols for which the bridge group is not explicitly configured to route will be bridged.

You could also explicitly configure the bridge group so that it does not bridge a particular protocol, so that routable packets of this protocol are routed when the bridge is explicitly configured to route this protocol, and non-routable packets are dropped because bridging is disabled for this protocol.

Note Packets of non-routable protocols such as LAT are only bridged. You cannot disable bridging for the non-routable traffic.

To configure specific protocols to be routed or bridged in a bridge group, perform one or more of the following tasks in global configuration mode:

| Task | Command |
|---|--|
| Specify a protocol to be routed in a bridge group. | bridge <i>bridge-group</i> route <i>protocol</i> |
| Specify that a protocol is not to be routed in a bridge group. | no bridge <i>bridge-group</i> route <i>protocol</i> |
| Specify that a protocol is to be bridged in the bridge group. | bridge <i>bridge-group</i> bridge <i>protocol</i> |
| Specify that a protocol is not to be bridged in the bridge group. | no bridge <i>bridge-group</i> bridge <i>protocol</i> |

For example, to bridge AppleTalk, bridge and route IPX, and route IP in the same bridge group, you would do the following:

- Bridge AppleTalk: Because integrated routing and bridging bridges everything by default, no configuration is required to bridge AppleTalk.
- Bridge and route IPX: After using the **bridge irb** command to enable integrated routing and bridging, and the **interface bvi** command to create the bridge-group virtual interface for the bridge group, you would use the **bridge route** command to both bridge and route IPX (bridging is already enabled by default; the **bridge route** command enables routing).
- Route IP: Use the **bridge route** command to enable routing, and then use the **no bridge bridge** command to disable bridging.

Note When integrated routing and bridging is not enabled, routing a given protocol means that protocol is not bridged, and bridging a protocol means that protocol is not routed. When integrated routing and bridging is enabled, the disjunct relationship between routing and bridging is broken down, and a given protocol can be switched between routed and bridged interfaces on a selective, independent basis.

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
- Configure Bridge Table Aging Time

Disable IP Routing

If you want to bridge IP, you must disable IP routing because IP routing is enabled by default on the Cisco IOS software. 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 ciscoBus2 controller, significantly improving performance. Autonomous bridging is a high-speed switching feature that allows bridged

traffic to be forwarded and flooded on the ciscoBus2 controller between resident interfaces. If you are using the ciscoBus2 controller, you can maximize performance by enabling autonomous bridging on the following ciscoBus2 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 ciscoBus2 controller. These interfaces include the CTR, FCI with encapsulation bridging, and HSSI with encapsulation other than HDLC, 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 ciscoBus2 controller and some that are not, the ciscoBus2 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 ciscoBus2 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 ciscoBus2 controller). | bridge-group <i>bridge-group</i> cbus-bridging |

Note You can filter by MAC-level address on an interface only 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 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 by the Cisco IOS software 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 <i>bridge-group</i> 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 routers or bridges within the domain. The devices in the domain, and only those devices, 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 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 <i>bridge-group</i> domain <i>domain-number</i> |

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 <i>bridge-group</i> acquire |
| Remove the ability to filter out all frames except those whose addresses have been statically configured into the forwarding cache. | bridge <i>bridge-group</i> 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 <i>bridge-group</i> 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 <i>bridge-group</i> aging-time <i>seconds</i> |

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:

- Set Filters at the MAC layer
- 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 Cisco IOS software.

- 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.
- Configuring bridging access lists of type 700 may cause a momentary interruption of traffic flow.

For more information on access lists, refer to the “Configuring Traffic Filters” chapter of the *Security Configuration Guide*.

Set Filters at the MAC layer

You can filter transmission of frames at the MAC layer by performing tasks in one of the following sections:

- 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 task in the following section:

- 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 <i>bridge-group</i> address <i>mac-address</i> { forward discard } [<i>interface</i>] |

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 <i>access-list-number</i> { permit deny } <i>address mask</i> |

The vendor code is the first three bytes of the MAC address (left to right). For an example of how to filter by vendor code, see “Multicast or Broadcast Packets Bridging Example” later in this chapter.

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. Therefore, an access list that works for one medium might not work for others.

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 <i>bridge-group</i> input-address-list <i>access-list-number</i> |
| Assign an access list to an interface for filtering by the MAC destination addresses. | bridge-group <i>bridge-group</i> output-address-list <i>access-list-number</i> |

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 Protocol type 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 protocol type 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 *Bridging and IBM Networking Command Reference*.

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

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

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 <i>bridge-group</i> input-type-list <i>access-list-number</i> |
| Add a filter for Ethernet- and SNAP-encapsulated packets on output. | bridge-group <i>bridge-group</i> output-type-list <i>access-list-number</i> |

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 <i>bridge-group</i> input-lsap-list <i>access-list-number</i> |
| Add a filter for IEEE 802-encapsulated packets on output. | bridge-group <i>bridge-group</i> output-lsap-list <i>access-list-number</i> |

Access lists for Ethernet- and IEEE 802-encapsulated packets affect only bridging functions. You cannot 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 <i>access-list-number</i> { permit deny } <i>source source-mask destination destination-mask offset size operator operand</i> |

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 <i>bridge-group</i> input-pattern-list <i>access-list-number</i> |
| Apply an extended access list to the packet being sent by an interface. | bridge-group <i>bridge-group</i> output-pattern-list <i>access-list-number</i> |

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 Cisco IOS software. 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 might 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 <i>bridge-group</i> 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 with which to deny access upon input. | bridge-group <i>bridge-group</i> input-lat-service-deny <i>group-list</i> |
| Specify the group codes with which to permit access upon input. | bridge-group <i>bridge-group</i> input-lat-service-permit <i>group-list</i> |

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 conditions 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 <i>bridge-group</i> output-lat-service-deny <i>group-list</i> |
| Specify the group codes with which to permit access upon output. | bridge-group <i>bridge-group</i> output-lat-service-permit <i>group-list</i> |

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 Bridge Protocol Data Unit (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 *Configuration Fundamentals Command Reference* 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 <i>bridge-group</i> priority <i>number</i> |

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 <i>bridge-group</i> priority <i>number</i> |

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 *Bridging and IBM Networking Command Reference* 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 <i>bridge-group</i> path-cost <i>cost</i> |

Adjust Bridge Protocol Data Unit (BPDU) Intervals

You can adjust BPDU intervals as described in the following sections:

- Adjust the Interval between Hello BPDUs
- Define the Forward Delay Interval
- Define 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 <i>bridge-group</i> hello-time <i>seconds</i> |

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 <i>bridge-group</i> forward-time <i>seconds</i> |

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 <i>bridge-group</i> max-age <i>seconds</i> |

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 <i>bridge-group</i> spanning-disabled |

Tune the Transparently Bridged Network

The following sections describe how to configure features that enhance network performance by reducing the number of packets that traverse the backbone network:

- Configure Circuit Groups
- Configure Constrained Multicast Flooding

Configure Circuit Groups

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. As long as one port of the circuit group is in the forwarding state, all ports in that circuit group will participate in load distribution regardless of their spanning-tree port states. This process guarantees that the computed spanning tree is still adaptive to any topology change and the load is distributed among the multiple segments. Deterministic load distribution 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.

If a circuit-group port goes down and up as a result of configuration or a line protocol change, the spanning-tree algorithm will bypass port transition and will time out necessary timers to force the eligible circuit-group ports to enter the forwarding state. This avoids the long disruption time caused by spanning-tree topology recomputation and therefore resumes the load distribution as quickly as possible.

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 <i>bridge-group</i> circuit-group <i>circuit-group</i> |

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 <i>bridge-group</i> circuit-group <i>circuit-group</i> pause <i>milliseconds</i> |

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 <i>bridge-group</i> circuit-group <i>circuit-group</i> 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.

Configure Constrained Multicast Flooding

In a transparent bridge, multicast packets are flooded on all forwarding ports on the bridge. For some protocols, it is possible for a bridge to determine the membership of multicast groups, and constrain the flooding of multicasts to a subset of the forwarding ports. Constrained multicast flooding enables a bridge to determine group membership of IP multicast groups dynamically and flood multicast packets only on those ports that reach group members.

To enable constrained multicast flooding, perform the following task in global configuration mode:

| Task | Command |
|---|--------------------------|
| Enable constrained multicast flooding for all configured bridge groups. | bridge cmf |

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 <i>bridge-group</i> |
| Remove multicast-group state information and clear the transmit and receive counts. | clear bridge [<i>bridge-group</i>] multicast [router-ports groups counts] [<i>group-address</i>] [<i>interface-unit</i>] [counts] |
| Reinitialize the Silicon Switch Processor (SSP) on the Cisco 7000 series. | clear sse |
| Remove VLAN statistics from any statically or system configured entries. | clear vlan statistics |
| Display classes of entries in the bridge forwarding database. | show bridge [<i>bridge-group</i>] [<i>interface</i>] [<i>address</i> [<i>mask</i>]] [verbose] |
| Display the interfaces configured in each circuit group and show whether they are participating in load distribution. | show bridge [<i>bridge-group</i>] circuit-group [<i>circuit-group</i>] [<i>src-mac-address</i>] [<i>dst-mac-address</i>] |
| Display transparent bridging multicast state information. | show bridge [<i>bridge-group</i>] multicast [router-ports groups] [<i>group-address</i>] |
| Display information about configured bridge groups. | show bridge group [verbose] |
| Display IEEE 802.10 transparently bridged VLAN configuration. | show bridge vlan |
| Display the configuration for each interface that has been configured for routing or bridging. | show interfaces crb |
| Display the protocols that can be routed or bridged for the specified interface. | show interfaces [<i>interface</i>] irb |
| Display the spanning-tree topology known to the router, including whether or not filtering is in effect. | show span |
| Display a summary of SSP statistics. | show sse summary |
| Display a summary of VLAN subinterfaces. | show vlans |

Transparent and SRT Bridging Configuration Examples

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

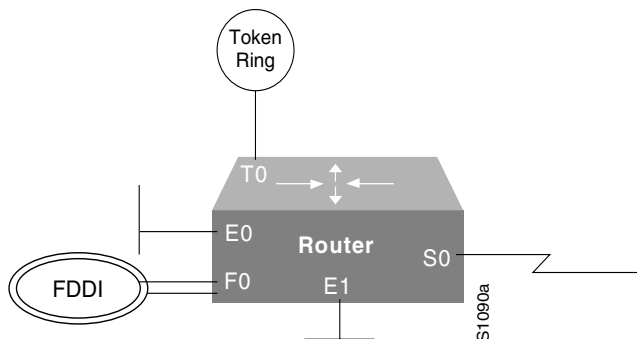
- Basic Bridging Example
- Concurrent Routing and Bridging Example
- Basic Integrated Routing and Bridging Example
- Complex Integrated Routing and Bridging Example
- Integrated Routing and Bridging with Multiple Bridge Groups Example
- Transparently Bridged VLANs Configuration Example

- Ethernet to FDDI 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
- Transparent Bridging over Multiprotocol LAPB Example
- Fast-Switched Transparent Bridging over ATM Example (Cisco 7000)
- Complex Transparent Bridging Network Topology Example

Basic Bridging Example

Figure 34 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 34 Example of Basic Bridging



The configuration file for the router depicted in Figure 34 would 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
```

Concurrent Routing and Bridging Example

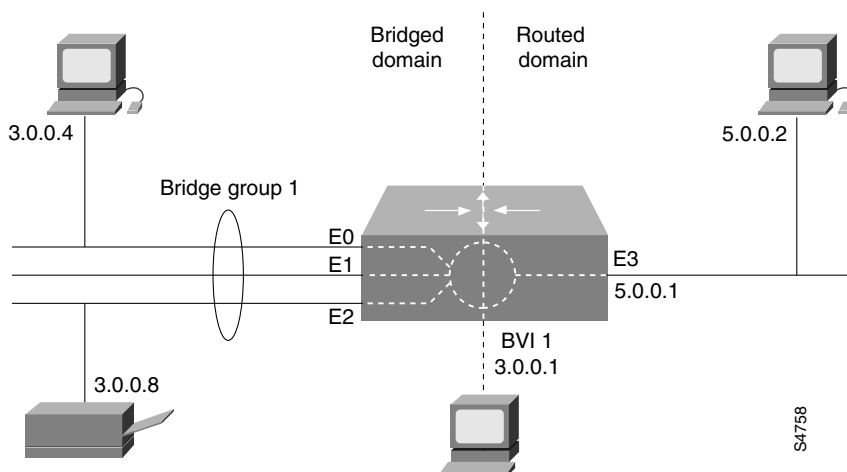
In the following example DecNet and IPX are concurrently routed and bridged. IP and AppleTalk are routed on all interfaces, DecNet and IP are routed on all interfaces not in the bridge group, and all protocols other than IP and AppleTalk are bridged on all interfaces in the bridge group:

```
!  
ipx routing 0000.0c36.7a43  
appletalk routing  
!  
decnet routing 9.65  
decnet node-type routing-iv  
!  
!  
interface Ethernet0/0  
 ip address 172.19.160.65 255.255.255.0  
 ipx network 160  
 appletalk address 160.65  
 decnet cost 7  
!  
interface Ethernet0/1  
 ip address 172.19.161.65 255.255.255.0  
 ipx network 161  
 appletalk address 161.65  
 decnet cost 7  
!  
interface Ethernet0/2  
 ip address 172.19.162.65 255.255.255.0  
 appletalk address 162.65  
 bridge-group 1  
!  
interface Ethernet0/3  
 ip address 172.19.14.65 255.255.255.0  
 appletalk address 14.65  
 appletalk zone california  
 bridge-group 1  
!  
router igrp 666  
 network 172.19.0.0  
!  
bridge crb  
 bridge 1 protocol ieee  
 bridge 1 route appletalk  
 bridge 1 route ip  
!
```

Basic Integrated Routing and Bridging Example

Figure 35 is an example of integrated routing and bridging that uses bridge group 1 to bridge and route IP. The router has three bridged Ethernet interfaces and one routed Ethernet interface.

Figure 35 Basic IP Routing using Integrated Routing and Bridging



The relevant portions of the configuration for the router are listed below.

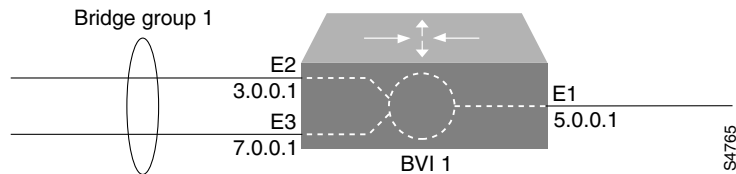
```

interface Ethernet 0
  bridge-group 1
  !
interface Ethernet 1
  bridge-group 1
  !
interface Ethernet 2
  bridge-group 1
  !
interface Ethernet 3
  ip address 5.0.0.1 255.0.0.0
  !
interface BVI 1
  ip address 3.0.0.1 255.0.0.0
  !
bridge irb
bridge 1 protocol ieee
bridge 1 route ip
    
```

Complex Integrated Routing and Bridging Example

Figure 36 is a more complex example of integrated routing and bridging, where bridge group 1 is used to route IP traffic, bridge IPX traffic, and bridge and route AppleTalk traffic.

Figure 36 Complex Integrated Routing and Bridging Example



The relevant portions of the configuration for the router are listed below.

```

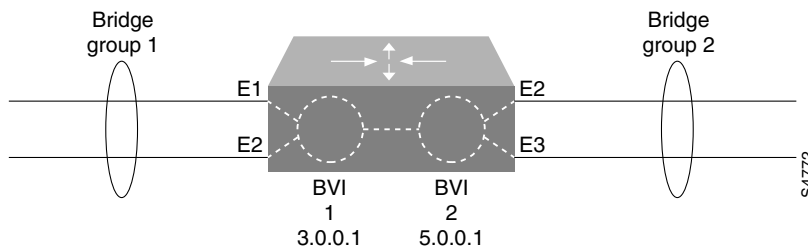
appletalk routing
!
interface Ethernet 1
 ip address 5.0.0.1 255.0.0.0
 appletalk cable-range 35-35 35.1
 appletalk zone Engineering
!
interface Ethernet 2
 ip address 3.0.0.1 255.0.0.0
 bridge-group 1
!
interface Ethernet 3
 ip address 7.0.0.1 255.0.0.0
 bridge-group 1
!
interface BVI 1
 no ip address
 appletalk cable-range 33-33 33.1
 appletalk zone Accounting
!
bridge irb
bridge 1 protocol ieee
 bridge 1 route appletalk
 bridge 1 route ip
 no bridge 1 bridge ip

```

Integrated Routing and Bridging with Multiple Bridge Groups Example

In the example illustrated in Figure 37 integrated routing and bridging is used to route and bridge IP between two bridge groups.

Figure 37 Integrated Routing and Bridging with Multiple Bridge Groups



The relevant portions of the configuration for the router are listed below.

```
interface Ethernet 1
  bridge-group 1
  !
interface Ethernet 2
  bridge-group 1
  !
interface Ethernet 3
  bridge-group 2
  !
interface Ethernet 4
  bridge-group 2
  !
interface BVI 1
  ip address 3.0.0.1 255.0.0.0
  !
interface BVI 2
  ip address 5.0.0.1 255.0.0.0
  !
bridge irb
bridge 1 protocol ieee
  bridge 1 route ip
bridge 2 protocol ieee
  bridge 2 route ip
```

Transparently Bridged VLANs Configuration Example

The following example shows the configuration for the topology in Figure 32. The “striped” VLAN is identified as security association identifier 45; the “dot” VLAN is identified as security association identifier 1008; the “sliced” VLAN is identified as security association identifier 4321. Note that the assignment of bridge group, interface, and subinterface numbers is of local significance only. You must coordinate only the configuration of a common Security Association Identifier across bridges.

Router One

```
bridge 18 protocol ieee
interface ethernet 0/1
  bridge-group 18
  !
interface ethernet 0/2
```

```
    bridge-group 18
  !
interface ethernet 0/3
  bridge-group 18
  !
interface fddi 4/0.8
  encapsulation sde 45
  bridge-group 18
  !
  bridge 54 protocol ieee

interface ethernet 1/1
  bridge-group 54

interface ethernet 1/2
  bridge-group 54

interface ethernet 1/3
  bridge-group 54

interface fddi 4/0.13
  encapsulation sde 1008
  bridge-group 54
  !
  bridge 3 protocol ieee

interface ethernet 2/1
  bridge-group 3

interface ethernet 2/2
  bridge-group 3

interface ethernet 2/3
  bridge-group 3

interface fddi 4/0.30
  encapsulation sde 4321
  bridge-group 3
```

Router Two

```
bridge 7 protocol ieee
interface ethernet 0/1
  bridge-group 7

interface ethernet 0/2
  bridge-group 7

interface ethernet 0/3
  bridge-group 7

interface ethernet 0/4
  bridge-group 7

interface fddi 2/0.11
  encapsulation sde 4321
  bridge-group 7
  !
  bridge 8 protocol ieee
interface ethernet 1/1
  bridge-group 8

interface ethernet 1/2
  bridge-group 8
```

```
interface ethernet 1/3
  bridge-group 8

interface ethernet 1/4
  bridge-group 8

interface fddi 2/0.14
  encapsulation sde 1008
  bridge-group 8
```

Router Three

```
bridge 1 protocol ieee
interface ethernet 0/1
  bridge-group 1
!
interface ethernet 0/2
  bridge-group 1
!
interface ethernet 0/3
  bridge-group 1
!
interface fddi 2/0.5
  encapsulation sde 4321
  bridge-group 1
!
bridge 6 protocol ieee
interface ethernet 1/1
  bridge-group 6
!
interface ethernet 1/2
  bridge-group 6
!
interface ethernet 1/3
  bridge-group 6
!
interface fddi 2/0.3
  encapsulation sde 45
  bridge-group 6
```

Routing Between VLANs Configuration Example

The following example shows the configuration for the topology shown in Figure 33. IP traffic is routed to and from switched VLAN domains 300, 400, and 600 to any other IP routing interface, as is IPX for VLANs 500 and 600. Because Fast Ethernet interfaces 2/1.20 and 3/1.40 are combined in bridge group 50, all other nonrouted traffic is bridged between these two subinterfaces.

```
interface FDDI 1/0.10
  ip address 131.108.1.1 255.255.255.0
  encap sde 300
!
interface FastEthernet 2/1.20.
  ip address 171.69.2.2 255.255.255.0
  encap isl 400
  bridge-group 50
!
interface FastEthernet 2/1.30
  ipx network 1000
  encap isl 500
!
interface FastEthernet 3/1.40
```

```

ip address 198.92.3.3 255.255.255.0
ipx network 1001
encap isl 600
bridge-group 50
!
bridge 50 protocol ieee
!
```

Ethernet to FDDI 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
```

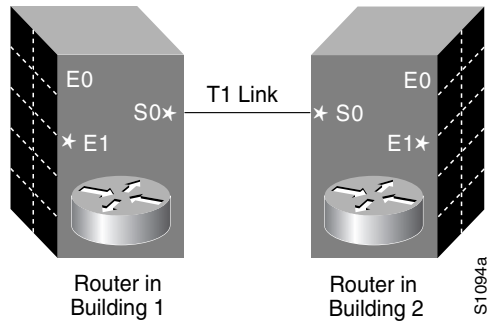
Ethernet Bridging Example

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 38 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 38 Ethernet Bridging Configuration Example



Router/Bridge in Building 1

The configuration file for the router 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
```

Router/Bridge in Building 2

The configuration file for the router 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

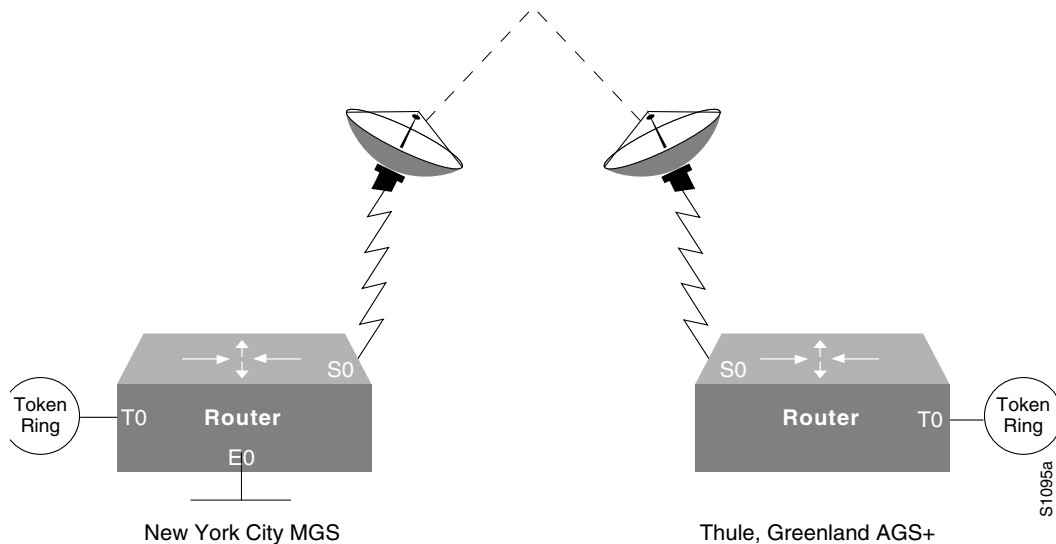
```

SRT Bridging Example

In Figure 39, 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 39 Network Configuration Example



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

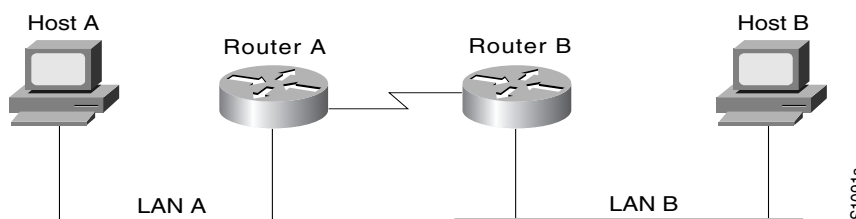
```
interface tokenring 0
 ip address 150.136.10.1 255.255.255.128
 bridge-group 1
 !
interface serial 0
 ip address 150.136.11.1 255.255.255.128
 bridge-group 1
 !
bridge 1 protocol ieee
```

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 40.

Figure 40 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
```

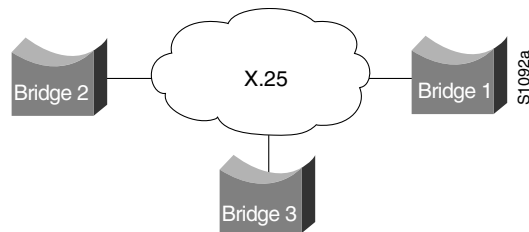
The correct access list would be as follows:

```
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 41 is an example configuration illustrating three bridges connected to each other through an X.25 network.

Figure 41 X.25 Bridging Examples



Following are the configuration commands for each of the bridges depicted in Figure 41.

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
```

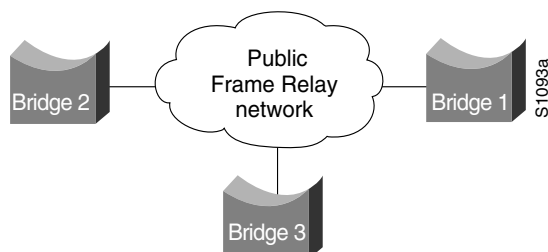
Configuration for Bridge 3

```
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 42 illustrates three bridges connected to each other through a Frame Relay network.

Figure 42 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 IP-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
```

Transparent Bridging over Multiprotocol LAPB Example

The following example illustrates a router configured for transparent bridging over multiprotocol LAPB encapsulation:

```
!
no ip routing
!
interface ethernet 1
  no ip address
  no mop enabled
  bridge-group 1
!
interface serial 0
  no ip address
  encapsulation lapb multi
  bridge-group 1
!
bridge 1 protocol ieee
```

Fast-Switched Transparent Bridging over ATM Example (Cisco 7000)

The following configuration example enables fast-switched transparent bridging over ATM:

```
interface atm 4/0
 ip address 1.1.1.1 255.0.0.0
 atm pvc 1 1 1 aal5snap
 atm pvc 2 2 2 aal5snap
 atm pvc 3 3 3 aal5snap
 bridge-group 1
 !
 bridge 1 protocol dec
```

Transparent Bridging over DDR Examples

The following two examples differ only in the packets that cause calls to be placed. The first example specifies by protocol (any bridge packet is permitted to cause a call to be made); the second example allows a finer granularity by specifying the Ethernet type codes of bridge packets.

The first example configures the serial 1 interface for DDR bridging. Any bridge packet is permitted to cause a call to be placed.

```
no ip routing
 !
 interface Serial1
 no ip address
 encapsulation ppp
 dialer in-band
 dialer enable-timeout 3
 dialer map bridge name urk broadcast 8985
 dialer hold-queue 10
 dialer-group 1
 ppp authentication chap
 bridge-group 1
 pulse-time 1
 !
 dialer-list 1 protocol bridge permit
 bridge 1 protocol ieee
 bridge 1 hello 10
```

The second example also configures the serial 1 interface for DDR bridging. However, this example includes an **access-list** command that specifies the Ethernet type codes that can cause calls to be placed and a **dialer list protocol list** command that refers to the specified access list.

```
no ip routing
 !
 interface Serial1
 no ip address
 encapsulation ppp
 dialer in-band
 dialer enable-timeout 3
 dialer map bridge name urk broadcast 8985
 dialer hold-queue 10
 dialer-group 1
 ppp authentication chap
 bridge-group 1
 pulse-time 1
 !
 access-list 200 permit 0x0800 0xFFFF8
 !
 dialer-list 1 protocol bridge list 200
 bridge 1 protocol ieee
 bridge 1 hello 10
```

Fast-Switched Transparent Bridging over SMDS Example

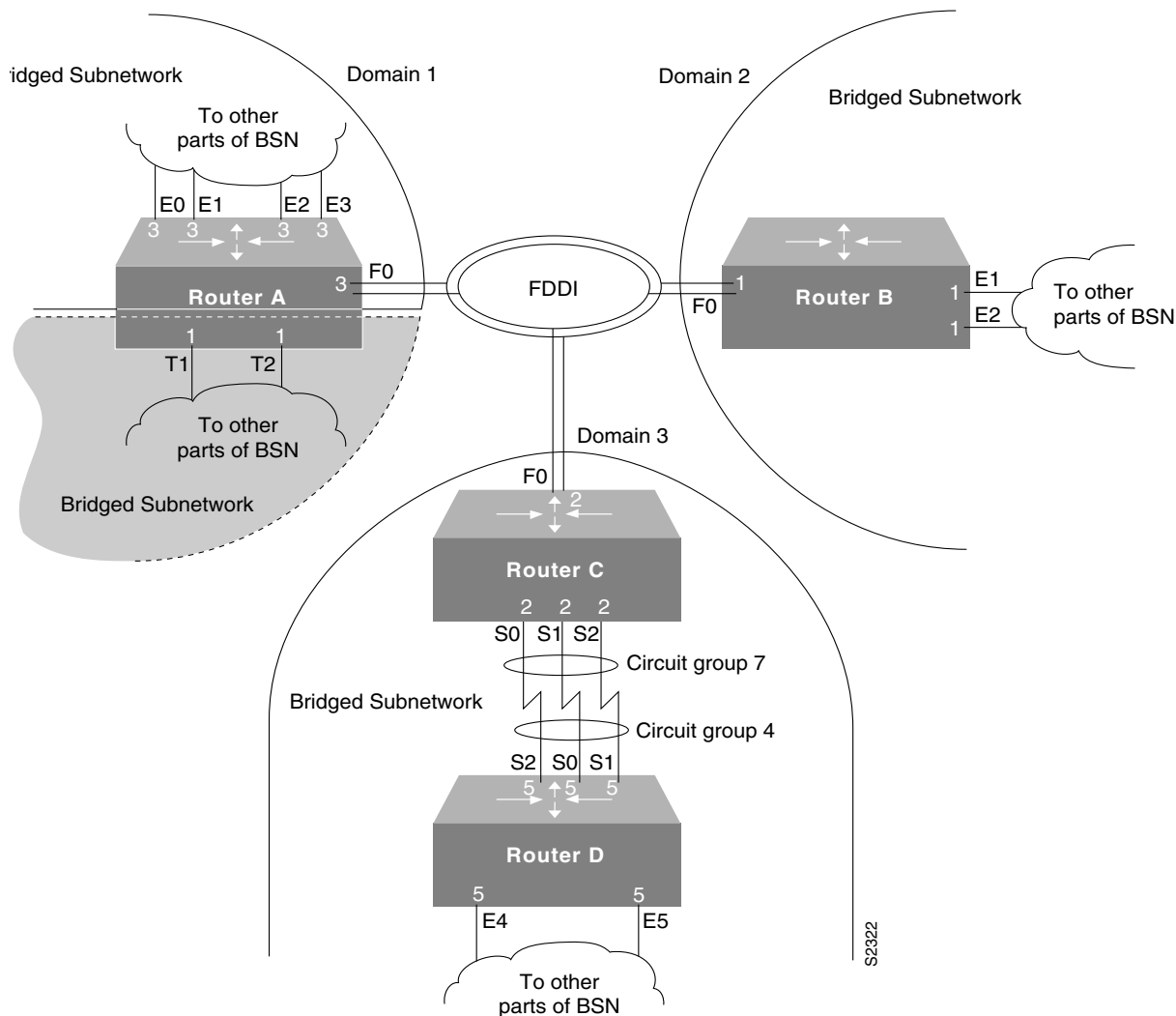
The following configuration example enables fast-switched transparent bridging over SMDS:

```
interface serial 0
 encapsulation smds
 bridge-group 1
 smds multicast bridge c141.5797.1313.ffff
```

Complex Transparent Bridging Network Topology Example

Figure 43 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 43 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 be used in this manner only 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 Cisco IOS software 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.

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 ethernet 0
  bridge-group 3
!
interface ethernet 1
  bridge-group 3
!
interface ethernet 2
  bridge-group 3
!
interface ethernet 3
  bridge-group 3
!
interface fddi 0
  bridge-group 3
!
interface tokenring 1
  bridge-group 1
!
interface tokenring 2
  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 ethernet 1
  bridge-group 1
!
interface ethernet 2
  bridge-group 1
!
interface fddi 0
  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 fddi 0
  bridge-group 2
!
interface serial 0
  bridge-group 2
  bridge-group 2 circuit-group 7
!
interface serial 1
  bridge-group 2
  bridge-group 2 circuit-group 7
!
interface serial 2
  bridge-group 2
  bridge-group 2 circuit-group 7
!
bridge 2 domain 3
bridge 2 protocol ieee

```

Configuration for Router D

```

interface ethernet 4
  bridge-group 5
!
interface ethernet 5
  bridge-group 5
!

```

Transparent and SRT Bridging Configuration Examples

```
interface serial 0
  bridge-group 5
  bridge-group 5 circuit-group 4
!
interface serial 1
  bridge-group 5
  bridge-group 5 circuit-group 4
!
interface serial 2
  bridge-group 5
  bridge-group 5 circuit-group 4
!
bridge 5 domain 3
bridge 5 protocol ieee
```