Router Products Configuration Guide

The products and specifications, configurations, and other technical information regarding the products contained in this manual are subject to change without notice. All statements, technical information, and recommendations contained in this manual are believed to be accurate and reliable but are presented without warranty of any kind, express or implied, and users must take full responsibility for their application of any products specified in this manual.

This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual for this device, may cause interference to radio communications. This equipment has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case users at their own expense will be required to take whatever measures may be required to correct the interference.

Possession, use, or copying of the software described in this publication is authorized only pursuant to a valid written license from Digital or an authorized sublicensor.

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

The following are trademarks of Digital Equipment Corporation: DDCMP, DEC, DECnet, DECNIS, DECserver, DECsystem, DECwindows, Digital, DNA, OpenVMS, ULTRIX, VAX, VAXstation, VMS, VMScluster, and the DIGITAL logo.

Portions of this document is used with permission of Cisco Systems, Incorporated. Copyright © 1990 - 1995, Cisco Systems, Inc.

The following third-party software may be included with your product and will be subject to the software license agreement:

CiscoWorks software and documentation are based in part on HP OpenView under license from the Hewlett-Packard Company. HP OpenView is a trademark of the Hewlett-Packard Company. Copyright © 1992, 1993 Hewlett-Packard Company.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB’s public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

Network Time Protocol (NTP). Copyright © 1992, David L. Mills. The University of Delaware makes no representations about the suitability of this software for any purpose.

TABLE OF CONTENTS

Chapter 5

LIST OF FIGURES

LIST OF TABLES

About This Manual

This section discusses the objectives, audience, organization, and conventions of the *Router Products Configuration Guide*.

Document Objectives

This publication describes the tasks necessary to configure and maintain your router. It includes task overviews, expanded descriptions of tasks, and comprehensive configuration examples. It does not provide complete command syntax descriptions, and therefore must be used in conjunction with the *Router Products Command Reference* publication.

Audience

This publication is intended primarily for network administrators who will be configuring and maintaining routers but are not necessarily familiar with the tasks involved, the relationship between them, or the commands necessary to perform particular tasks.

Document Organization

This publication is divided into six main parts. Each part comprises chapters describing related tasks or functions. The organization of parts and chapters in this publication matches the organization of parts and chapters in the *Router Products Command Reference*, except that this document does not contain appendixes. The parts in this publication are as follows:

- Part 1, "Product Introduction," contains an overview of the router and task descriptions for the system user interface and command parser. Begin your system configuration process with this part of the manual.
- Part 2, "System and Interface Configuration and Management," describes the tasks pertaining to booting, terminal sessions and modem lines, system management, and system interfaces. It also describes the command interpreter, or EXEC.
- Part 3, "Wide-Area Networking," describes the tasks pertaining to ATM, DDR, Frame Relay, ISDN, SMDS, and X.25. The chapters are arranged in alphabetical order for ease of use.

- Part 4, “Routing Protocols,” contains chapters that describe how to configure each supported network protocol. These protocols include Apollo Domain, AppleTalk, Banyan VINES, DECnet, IP, ISO CLNS, Novell IPX, and XNS (including Ungermann-Bass and 3Com variations). This part also contains a chapter that discusses IP routing protocols, which include IGRP, BGP, RIP, OSPF, IS-IS, and ISO-IGRP. The chapters are arranged in alphabetical order for ease of use.
- Part 5, “Bridging,” contains chapters that describe how to configure transparent bridging, source-route bridging, source-route transparent (SRT) bridging, and source-route translational bridging (SR/TLB) on our routers.
- Part 6, “IBM Networking,” contains chapters that describe how to configure the SDLC transport and serial tunneling mechanisms in an IBM local-area network. Also included are the tasks for configuring the local acknowledgment feature, managing your source-route bridges with LAN Network Manager, and configuring SDLLC and QLLC conversion, our IBM network protocol translation features. Part 6 also contains chapters that describe how to configure SNA Downstream Physical Unit (DSPU) support and SNA Frame Relay Access Support. The IBM Channel Attach chapter describes how to configure a Channel Interface Processor (CIP).

Document Conventions

Software and hardware documentation uses the following conventions:

- The symbol ^ represents the Control key.

  For example, the key combinations ^D and Ctrl-D mean hold down the Control key while you press the D key. Keys are indicated in capitals, but are not case sensitive.

- A string is defined as a nonquoted set of characters.

  For example, when setting up a community string for SNMP to “public,” do not use quotes around the string, or the string will include the quotation marks.

Command descriptions use these conventions:

- Vertical bars (|) separate alternative, mutually exclusive, elements.
- Square brackets ([ ]) indicate optional elements.
- Braces ({ }) indicate a required choice.
- Braces within square brackets ([{ }]) indicate a required choice within an optional element.
- **Boldface** indicates commands and keywords that are entered literally as shown.
- *Italics* indicate arguments for which you supply values; in contexts that do not allow italics, arguments are enclosed in angle brackets (< >).

Examples use these conventions:

- Examples that contain system prompts denote interactive sessions, indicating that the user enters commands at the prompt. The system prompt indicates the current command mode. For example, the prompt `router(config)#` indicates global configuration mode.
- Terminal sessions and information the system displays are in `screen` font.
- Information you enter is in **`boldface screen`** font.
- Nonprinting characters, such as passwords, are in angle brackets (< >).
- Default responses to system prompts are in square brackets ([ ]).

- Exclamation points (!) at the beginning of a line indicate a comment line. They are also displayed by the router for certain processes.

Note Means *reader take note*. Notes contain helpful suggestions or references to materials not contained in this manual.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

CHAPTER 16

Configuring DECnet

Digital Equipment Corporation designed the DECnet stack of protocols in the 1970's as part of its Digital Network Architecture (DNA). DNA supports DECnet routing over Ethernet, Token Ring, Fiber Distributed Data Interface (FDDI), High-Level Data Link Control (HDLC), Point-to-Point Protocol (PPP), Frame Relay, Switched Multimegabit Data Service (SMDS), X.25, and IEEE 802.2.

DECnet supports both connectionless and connection-oriented network layers implemented by Open System Interconnection (OSI) protocols. DECnet's most recent product release is called Phase V, which is equivalent to International Organization for Standardization (ISO) Connectionless Network Service (CLNS). Phase V is compatible with the previous release, Phase IV. Phase IV was similar to OSI routing, while Phase V implements full OSI routing including support for end system-to-intermediate system (ES-IS) and intermediate system-to-intermediate system (IS-IS) connections. An end system (ES) is a nonrouting network node; an intermediate system (IS) refers to a router. ES-IS support allows ESs and ISs to discover each other. IS-IS provides routing between ISs only.

DECnet Phase IV Prime supports inherent Media Access Control (MAC) addresses, which allows DECnet nodes to coexist with systems running other protocols that have MAC address restrictions.

This chapter describes how to configure our implementation of the DECnet routing protocol. For a complete description of the commands in this chapter, refer to the "DECnet Commands" chapter of the *Router Products Command Reference* publication. For historical background and a technical overview of DECnet, see the *Internetworking Technology Overview* publication.

Cisco's Implementation of DECnet

DECnet support on a Cisco router includes local-area and wide-area DECnet Phase IV routing over Ethernet, Token Ring, FDDI, and serial lines (X.25, Frame Relay, SMDS). The following are the specifics of Cisco's support:

- Cisco routers interoperate with Digital routers, and Digital hosts do not differentiate between a Cisco router and a Digital router.
- The router uses HDLC framing rather than Digital Data Communications Message Protocol (DDCMP) framing for point-to-point lines.
- If you construct a network using both Cisco Systems and Digital equipment, you must ensure that each point-to-point line has the same type of equipment on both ends.
- Cisco and DECnet Phase IV routers have incompatible X.25 support.
- As with point-to-point lines, you must use a single vendor's equipment on the X.25 portion of your network.

- You can configure your Cisco equipment running Software Release 9.1 or later to interoperate with Digital equipment, or you can configure your Cisco equipment to operate with other Cisco routers that use prior versions of router software.
- Cisco gives you additional security options through access lists.
- Cisco routers support the address translation gateway (ATG).

  ATG allows the router to participate in multiple, independent DECnet networks. In case of duplicate addressing, ATG establishes a user-specified address translation table for selected nodes between networks.
- Digital uses some nonroutable protocols that are not part of the DECnet stack.

  For example, neither Cisco nor Digital routers can route the Maintenance Operation Protocol (MOP) and local area transport (LAT); instead, these protocols must be bridged.
- The parameters in Cisco Systems' implementation of DECnet are a subset of the parameters you can modify in Digital's Network Control Program (NCP).

  Cisco uses the same names, the same range of allowable values, and the same defaults wherever possible. You must use the configuration commands to set DECnet parameters. Cisco's DECnet implementation does not set parameters by communicating with NCP.
- Cisco supports DECnet Phase IV-to-Phase V conversion:
  - Cost information is represented in native mode for the Phase IV or Phase V protocols.
  - Digital has defined algorithms for mapping a subset of the Phase V address space onto the Phase IV address space and for converting Phase IV and Phase V packets back and forth in order to support Phase IV hosts in Phase V networks and vice versa.
- Cisco's implementation differs from Digital's in the following ways:
  - You can add Phase V support without modifying your existing Phase IV support.
  - Cisco's implementation delays converting packets from Phase IV to Phase V, while Digital's implementation converts as soon as possible.

DECnet Configuration Task List

To configure DECnet routing, complete the tasks in the following sections. Only the first task is required; the remaining are optional.

- Enable DECnet Routing
- Configure DECnet on Token Rings
- Configure Address Translation
- Specify Name-to-DECnet Address Mapping
- Enable Phase IV-to-Phase V Conversion
- Propagate Phase IV Areas through an OSI Backbone
- Establish the Routing Table Size
- Configure Level 1 Routers
- Configure Level 2 Routers
- Specify Designated Routers
- Configure Static Routing

- Control Access to DECnet Networks
- Enhance DECnet Performance
- Configure DECnet over DDR
- Configure DECnet over WANs
- Monitor and Maintain the DECnet Network

See the end of this chapter for configuration examples.

Enable DECnet Routing

In order to enable DECnet routing, you must complete the tasks in the following sections:

- Either Enable DECnet Phase IV Routing or Enable DECnet Phase IV Prime Routing
- Assign a DECnet Cost to Each Interface
- Specify the DECnet Node Type

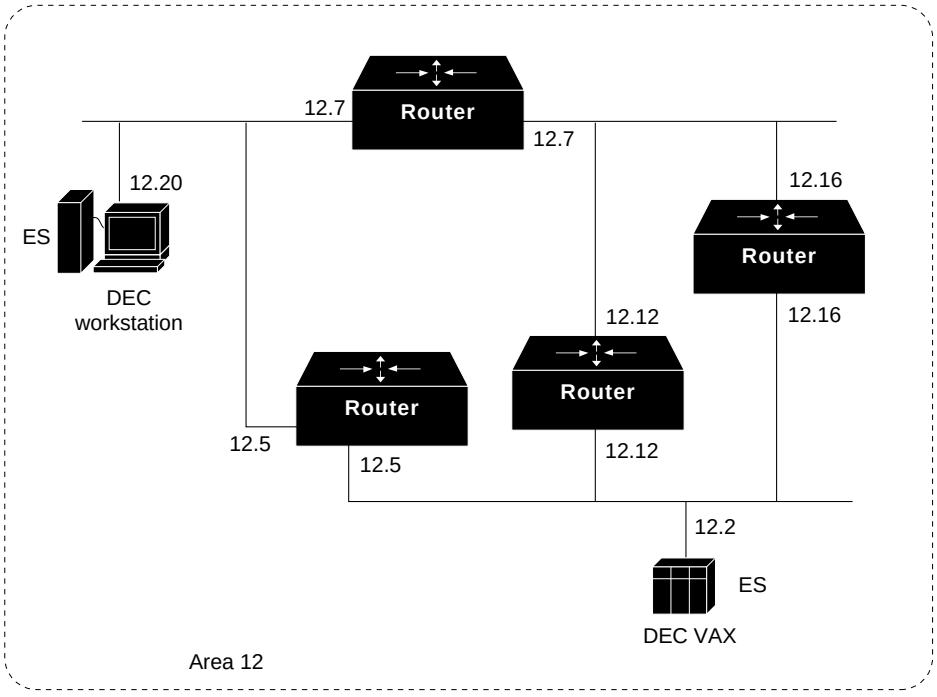
Enable DECnet Phase IV Routing

To enable DECnet Phase IV routing, perform the following task in global configuration mode:

Task	Command
Enable the DECnet Phase IV routing protocol on a global basis.	**decnet** [*network-number*] **routing** *decnet-address*

A DECnet host exists as a *node* in an *area.* An area spans many routers, and a single interface can have many areas attached to it. Therefore, if a router exists on many cables, it uses the same area and node for itself on all of them. Note how this differs from other routing protocols, where each interface is given a different internetwork address. Figure 16-1 shows the DECnet approach.

Figure 16-1 DECnet Nodes and Areas



Enabling DECnet changes the MAC addresses of the router's interfaces. This is not a problem on routers equipped with nonvolatile memory. On systems that attempt to get their Internet Protocol (IP) network addresses from network servers instead of from nonvolatile memory, there might be a problem with the hardware addresses changing and confusing other IP-speaking hosts. If you are attempting to use DECnet on such a configuration, be sure to set all global DECnet parameters before enabling DECnet routing on the interfaces.

With DECnet Phase IV Prime, the change of MAC addresses is not an issue because you can change the MAC address of the interface.

Note If you plan to use DECnet and Internet Packet Exchange (IPX) routing concurrently on the same interface, you should enable DECnet routing first, then enable IPX routing without specifying the optional MAC address. If you do this in the reverse order (that is, enable IPX, then DECnet), IPX routing will be disrupted.

Once you have enabled DECnet routing, you can obtain MAC addresses by using the **show interfaces** EXEC command. To disable DECnet routing, use the **no decnet routing** command.

Enable DECnet Phase IV Prime Routing

DECnet Phase IV requires that a MAC station address be constructed using DECnet addressing conventions, with a standard high-order byte string (AA-00-04-00) concatenated with the byte-swapped DECnet node address. This can cause problems in configurations in which DECnet nodes need to coexist with systems running protocols that have other MAC address restrictions.

DECnet Phase IV Prime allows an arbitrary MAC address on the local-area network (LAN). An address can be assigned globally (that is, assigned by the IEEE), or it can be assigned locally by a system administrator.

To enable or disable DECnet Phase IV Prime, perform one of the following tasks as appropriate in global configuration mode:

Task	Command
Specify Phase IV Prime routing.	**decnet** [*network-number*] **routing iv-prime** *decnet-address*
Stop DECnet Phase IV or Phase IV Prime routing.	**no decnet routing**

Optionally, you can map a DECnet multicast address to a Token Ring functional address other than the default functional address. To do so, perform the following task in interface configuration mode:

Task	Command
Specify the type of multicast address and the functional address to which the multicast ID will map.	**decnet multicast-map** *multicast-address-type functional-address*

Assign a DECnet Cost to Each Interface

After you have enabled DECnet routing on the router, you must assign a cost to each interface over which you want DECnet to run. Assigning a cost to an interface enables DECnet on the interface and, using a standard formula, assigns a different MAC address than that “burned in” by the manufacturer. This section describes how to assign a cost to each interface.

DECnet routing decisions are based on cost, an arbitrary measure used to compare paths on the internetwork. Costs are based on such measures as hop count or media bandwidth.The lower the cost, the better the path. You must assign a cost to each interface.

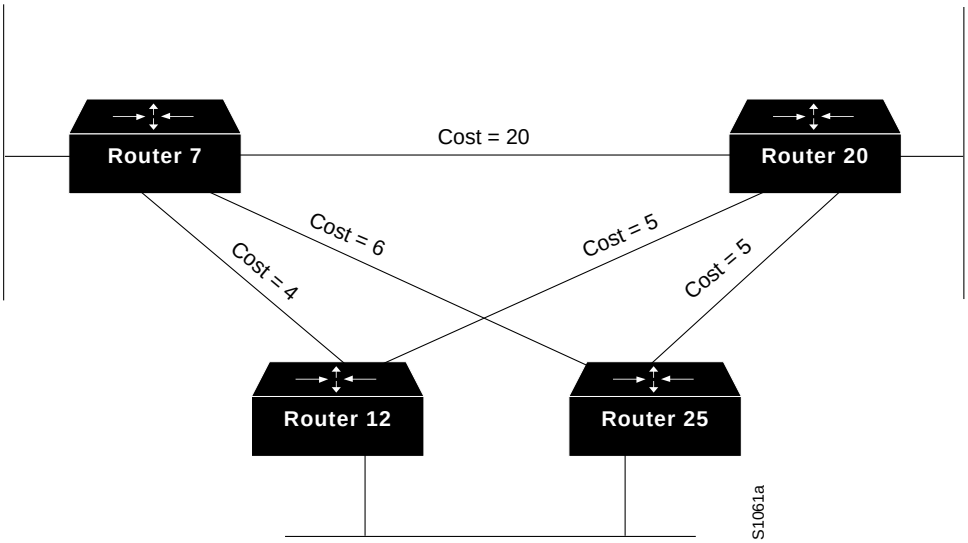
To assign a cost to each interface for DECnet Phase IV Prime, perform the following tasks in interface configuration mode:

Task	Command
Assign a cost to an interface.	**decnet cost** *cost-value*

Most DECnet installations have individualized routing strategies for using costs. Therefore, check the routing strategy used at your installation to ensure that the costs you specify are consistent with those set for other hosts on the network.

Figure 16-2 shows four routers, three Ethernets, and the various routes linking them. Each link has a different cost associated with it. The least expensive route from Router 7 to Router 20 is via Router 12.

Figure 16-2 DECnet Cost Values



Specify the DECnet Node Type

DECnet routing nodes are referred to as either Level 1 or Level 2 routers. You must specify the router's node type. A Level 1 router exchanges packets with other end nodes and routers in the same area and ignores Level 2 packets; this is called *intra-area routing*. Level 2 routers participate in the DECnet routing protocol with other routers and route packets to and from routers in other areas; this is called *interarea routing*. Level 2 routers also act as Level 1 routers in their own area.

The keyword **area** indicates a Level 2, interarea, router. The keyword **routing-iv** indicates a Level 1, intra-area, router; this is the default. In Level 1 mode, the router sends packets destined for other areas to a designated interarea router, which forwards them outside the area.

To specify the node types, perform one of the following tasks in global configuration mode:

Task	Command
Specify an interarea node type of the router.	**decnet** [*network-number*] **node-type area**
Specify an intra-area node type of the router.	**decnet** [*network-number*] **node-type routing-iv**

A simple example of how to configure DECnet is found in the "DECnet Configuration Examples" section at the end of this chapter.

Configure DECnet on Token Rings

If any Cisco routers are running Software Release 9.0 or earlier, you can use the Token Ring as a backbone or transit network for DECnet routing but you cannot communicate with non-Cisco DECnet nodes on the Token Ring.

If all Cisco routers are running Software Release 9.1 or later, you can set DECnet encapsulation to allow Cisco interoperation with non-Cisco equipment.

If you have both 9.0 and 9.1 routers in the same network, and you want them to interoperate, you must set the encapsulation type to **pre-dec** on the 9.1 routers.

To run DECnet on Token Ring interfaces, you must complete the following steps in the order specified:

Step 1 Enable DECnet routing on the Token Ring interface (see further discussion on Token Ring interfaces in the Interface Commands chapter).

Step 2 Set the DECnet encapsulation mode for the interface.

To complete these steps, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Enable DECnet on the Token Ring interface, and then enter interface configuration mode.	**interface tokenring** *number*[1]
Step 2	Configure the DECnet encapsulation mode for the specified interface.	**decnet encapsulation** {**pre-dec** | **dec**}

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

Use the keyword **dec** with routers running Software Release 9.1 or later. Use the keyword **pre-dec** with routers running Software Release 9.0 or earlier, or in a network where routers running 9.0 and routers running 9.1 must interoperate.

Configure Address Translation

If you set up multiple networks, we recommend that you configure address translation in order to avoid problems with duplicate addressing between networks. If you have multiple DECnet networks, you must establish an address translation table for selected nodes between networks. This eliminates any potential problems of duplicate addressing occurring between networks. The address translation gateway (ATG) allows you to define multiple DECnet networks and map between them.

Configuring ATG allows the router to route traffic for multiple independent DECnet networks and to establish a user-specified address translation for selected nodes between networks. Address translation allows connectivity between DECnet networks that might not otherwise be possible due to address conflicts (duplicate addresses) between them. Configuring ATG can be done over all media types.

When you use ATG, all the DECnet configuration commands implicitly apply to network number 0 unless you specify otherwise.

To translate a virtual DECnet address to a real network address, perform the following task in global configuration mode:

Task	Command
Establish a translation entry to translate a virtual DECnet address to a real DECnet address for the router.	**decnet** *first-network* **map** *virtual-address second-network real-address*

To display the address mapping information used by the DECnet ATG, use the **show decnet map** EXEC command.

A simple example of how to configure address translation can be found in the "DECnet Configuration Examples" section at the end of this chapter.

Make a "Poor Man's Routing" Connection

As an additional feature and security precaution, DECnet "Poor Man's Routing" can be used between nodes outside of the translation map as long as those nodes have access to nodes that are in the map. For example, as illustrated in Figure 16-5 in the "DECnet Configuration Examples" section later in this chapter, a user on node B could issue the following VMS operating system command:

$ dir A::D::E::

When a Poor Man's Routing connection is made between two networks, only the two adjacent nodes between the networks will have any direct knowledge about the other network. Application-level network access can then be specified to route through the connection.

Note We do not support Poor Man's Routing directly; the intermediate nodes must be VMS systems with Poor Man's Routing enabled in file access language (FAL).

Specify Name-to-DECnet Address Mapping

You can define a name-to-DECnet address mapping, which can be used instead of typing the set of numbers associated with a DECnet address.

To define a name-to-DECnet address mapping, perform the following task in global configuration mode:

Task	Command
Define a name-to-DECnet address mapping.	**decnet host** *name decnet-address*

The assigned DECnet name is displayed, where applicable, in **decnet route** and **show hosts** EXEC command output.

Enable Phase IV-to-Phase V Conversion

Routers that have conversion enabled advertise reachability to both Phase IV and Phase V hosts in both Phase IV and Phase V routing updates. If you have Phase IV hosts in Phase V networks and vice versa, you must enable Phase IV-to-Phase V conversion (and vice versa) in order for all nodes to communicate with each other. To enable DECnet conversion, you must have both DECnet and ISO CLNS configured on your router. Then perform the following task in global configuration mode:

Task	Command
Enable DECnet Phase IV-to-Phase V (and vice versa) conversion on the router.	**decnet conversion***nsap-prefix*

Make sure that the area you specify in the **decnet conversion** global configuration command is the same as the area you specified in the ISO CLNS address. You must also enable CLNS on all interfaces, even if the router has only Phase IV hosts on some of the interfaces. This enables information about those routers to be included in link state packets and consequently, enables other routers to be informed about the routers connected by that interface.

An example of how to enable a Phase IV area through an OSI backbone can be found in the "Phase IV-to-Phase V Conversion Example" section later in this chapter.

Propagate Phase IV Areas through an OSI Backbone

One limitation of the Phase IV-to-Phase V conversion has been the inability to propagate Phase IV area routes through OSI clouds. Using the "advertise" feature, you can explicitly configure any DECnet Phase IV areas that you want to propagate outward. You configure the border routers at the Phase IV/Phase V junction.

When distant routers send a packet destined across the cloud to a border router, the router converts the route and sends it as an OSI packet. In order for the converting router to have the corresponding OSI entry to which to convert the Phase IV packet, the other border router at the Phase IV/V junction must inject "static discard" routes. In this way, the first router converts the packet from Phase IV to Phase V, sending it through the cloud, and at the other end, the router advertising the static discard route converts the packet back to Phase IV and discards the Phase V packet. In effect, a fake entry is created in the Phase IV area table to propagate this information to other routers. This entry will not overwrite a native Phase IV entry if one already exists in the table.

To enable Phase IV areas to propagate through an OSI backbone on the router, perform the following task in global configuration mode:

Task	Command
Enable DECnet Phase IV areas to propagate through an OSI backbone on the router.	**decnet advertise** *decnet-area hops cost*

To enable the border router *at the far end* to convert the Phase V packet back to Phase IV, it must advertise a static discard route. To configure the far border router, perform the following task in global configuration mode:

Task	Command
Advertise a static discard route on the far-end border router	**clns route** *nsap-prefix* **discard**[1]

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

An example of how to enable a Phase IV area through an OSI backbone can be found in the "Configuring Phase IV Areas through an OSI Backbone Example" section at the end of this chapter.

Establish the Routing Table Size

You can configure the maximum number of addresses and areas allowed in the router's routing table. It is best to keep routing updates small. All areas or nodes that cannot be reached must be advertised as unreachable to the router. When configuring the routing table size, indicate the maximum node and area numbers that can exist in the network. In general, all routers on the network should use the same values for maximum addresses and nodes.

To establish the routing table size, perform either or both of the following tasks in global configuration mode:

Task	Command
Set the maximum node address that can exist in the network on the router.	**decnet** [*network-number*] **max-address** *value*
Set the largest number of areas that the router can handle in its routing table.	**decnet** [*network-number*] **max-area** *area-number*

Configure Level 1 Routers

Perform any of the tasks in the following section for the routers you have configured as Level 1 (intra-area) routers. In Level 1 mode, the router sends packets destined for other areas to a designated interarea router, which forwards them outside the area.

Set Areas as Unreachable

You can set the maximum cost that the router considers usable for intra-area routing. The router ignores routes within its local area that have a cost greater than the value you specify.

You also can set the maximum number of hops, or traversal of different paths that the router considers usable for intra-area routing. The router ignores routes within its local area that have a value greater than you specify.

To set certain intra-areas as unreachable based on cost value or hop count, perform either or both of the following tasks in global configuration mode:

Task	Command
Set the maximum cost value for intra-area routing on the router.	**decnet** [*network-number*] **max-cost** *cost*
Set the maximum hop count value for intra-area routing on the router.	**decnet** [*network-number*] **max-hops** *hop-count*

Configure Level 2 Routers

Perform any of the tasks in the following section for the routers you have configured as Level 2 (interarea) routers. In Level 2 mode, the router sends packets destined for other areas via the least-cost path to another interarea router.

Set Areas as Unreachable

You can set the maximum cost for a usable route to a distant area. The router treats as unreachable any route with a cost greater than the value you specify.

You also can set the maximum number of hops for a usable route to a distant area. The router treats as unreachable any route with a hop count greater than the value you specify.

To set certain interareas as unreachable based on cost value or hop count, perform either or both of the following tasks in global configuration mode:

Task	Command
Set the maximum cost specification value for interarea routing on the router.	**decnet** [*network-number*] **area-max-cost** *value*

Task	Command
Set the maximum hop count value for interarea routing on the router.	**decnet** [*network-number*] **area-max-hops** *value*

Specify Designated Routers

You can determine the router to which all end nodes on an Ethernet communicate if they do not know where else to send a packet. This router is called the *designated* router and is the router with the highest priority. When two or more routers on a single Ethernet in a single area share the same highest priority, the router with the highest node number is selected. You can reset a router's priority to help ensure that it is elected designated router in its area. This is specified on a per-interface basis.

To specify designated routers, perform the following task in interface configuration mode:

Task	Command
Assign or change a priority number to a router on a per-interface basis to receive packets for which no destination is specified.	**decnet router-priority** *value*

Configure Static Routing

Static routing is used when it is not possible or desirable to use dynamic routing. The following are some instances of when you would use static routing:

- The routers do not support the same dynamic routing protocol.
- Your network includes WAN links that involve paying for connect time or per packet.
- You want routers to advertise connectivity to external networks but you are not running an interdomain routing protocol.
- You must interoperate with another vendor's equipment that does not support any of the dynamic routing protocols that we support.
- The router operates over X.25, Frame Relay, or SMDS networks.

Note An interface that is configured for static routing cannot reroute around failed links.

To configure static routing, complete any of the tasks in the following sections:

- Configure a Static Route
- Configure a Static Route for an Interface
- Configure a Default Static Route
- Configure a Default Static Route for an Interface

Configure a Static Route

You can configure a specific static route and apply it globally even when you use dynamic routing.

To apply a specific static route globally, perform the following task in global configuration mode:

Task	Command
Configure a specific static route.	**decnet route** *decnet-address next-hop-address* [*hops*] [*cost*]

Configure a Static Route for an Interface

You can select a specific interface for a specific static route when you do not know the address of your neighbor.

To apply a specific static route to a specific interface, perform the following task in global configuration mode:

Task	Command
Configure a specific static route for a specific interface.	**decnet route** *decnet-address next-hop-type number* [*snpa-address*] [*hops* [*cost*]]

Configure a Default Static Route

You can configure a default static route and apply it globally even when you use dynamic routing.

To apply a default static route globally, perform the following task in global configuration mode:

Task	Command
Configure a default route.	**decnet route default** *next-hop-address* [*hops* [*cost*]]

Configure a Default Static Route for an Interface

You can configure a specific interface for a default static route when you do not know the address of your neighbor.

To apply a default static route to a specific interface, perform the following task in global configuration mode:

Task	Command
Configure a specific default route for a specific interface.	**decnet route default** *next-hop-type number* [*snpa-address*] [*hops* [*cost*]]

Configure DECnet Static Route Propagation

When you use static routes or default static routes, you can specify whether the static routes are propagated. By default, DECnet static routes will not be propagated to other routers.

To enable or disable static route propagation, perform the following tasks in global configuration mode:

Task	Command
Enable static route propagation.	**decnet propagate static**
Disable static route propagation.	**no decnet propagate static**

Control Access to DECnet Networks

We provide several layers of access control for network security. You can complete any or all of the tasks in the following sections:

- Create an Access List Based on Source Addresses
- Create an Access List Based on Source and Destination Addresses
- Add Filters to Access Lists
- Configure Access Groups
- Configure Routing Filters

Create an Access List Based on Source Addresses

You can configure lists globally to control access by source addresses. The standard form of the DECnet access list has a source DECnet address followed by a source-mask address, with bits set wherever the corresponding bits in the address should be ignored. DECnet addresses are written in the form *area.node*. For example, 50.4 is area 50, node 4. All addresses and masks are in decimal notation.

To create a standard DECnet access list, perform the following task in global configuration mode:

Task	Command
Create an access list to restrict access to a single address.	**access-list** *access-list-number* {**permit** | **deny**} *source source-mask*

To disable the list, use the **no access-list** command.

Create an Access List Based on Source and Destination Addresses

The extended form of the DECnet access list has a source DECnet address and mask pair, followed by a destination DECnet address and mask pair.

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

Task	Command
Create an extended access list for several addresses.	**access-list** *access-list-number* {**permit** | **deny**} *source source-mask* [*destination destination-mask*]

To disable the extended access list, use the **no access-list** command.

Add Filters to Access Lists

DECnet access lists can be used to filter *connect initiate* packets. With these packets, you can filter by DECnet object type, such as MAIL.

To add filters to access lists, perform the following task in global configuration mode:

Task	Command
Add filtering (by DECnet object type) to an access list.	**access-list** *access-list-number* {**permit** | **deny**} *source source-mask [destination destination-mask* {**eq** | **neq**} *[[source-object*] [*destination-object*] [*identification*]] **any**]

Configure Access Groups

You can restrict access to specific interfaces by applying an access list to them. Interfaces that are associated with the same access list are considered to be an access group.

To configure access groups, perform the following task in interface configuration mode:

Task	Command
Assign an access list to a specified interface.	**decnet access-group** *access-list-number*

Configure Routing Filters

You can control access to hello messages or routing information being received or sent out on an interface. Addresses that are not in the access list are shown in the update message as unreachable.

To configure routing filters, perform either or both of the following tasks, as needed, in interface configuration mode:

Task	Command
Control access to hello messages or routing information received on a specified interface.	**decnet in-routing-filter** *access-list-number*
Control access to routing information being sent out on a specified interface.	**decnet out-routing-filter** *access-list-number*

Enhance DECnet Performance

To optimize internetwork performance, complete any or all of the tasks in the following sections:

- Set Maximum Equal-Cost Paths
- Establish Selection for Paths of Equal Cost
- Set Maximum Visits
- Adjust the Hello Timer
- Disable Fast Switching
- Set the Congestion Threshold
- Adjust the Broadcast Routing Timer

Set Maximum Equal-Cost Paths

You can set the maximum number of equal-cost paths to a destination on a global basis. Limiting the number of equal-cost paths can save memory on routers with limited memory or very large configurations. Additionally, in networks with a large number of multiple paths and end systems with limited ability to cache out-of-sequence packets, performance might suffer when traffic is split between many paths.

To set maximum equal-cost paths, perform the following task in global configuration mode:

Task	Command
Set the maximum number of equal-cost paths to a destination. Paths are set in the routing table.	**decnet** [*network-number*] **max-paths** *value*

Use the **show decnet route** EXEC command to display the first hop route to a specified address and to show all equal-cost paths to a single destination.

Establish Selection for Paths of Equal Cost

You can establish one of two methods for selecting among paths of equal cost on the router: on a round-robin basis, which is the default, or by configuring the router so that traffic for any higher-layer session is always routed over the same path.

In the round-robin or *normal* mode, the first packet is sent to the first node, the second packet to the second node, and so on. If the final node is reached before all packets are sent, the next packet in line is sent to the first node, then to the second node, and so forth.

The *interim* mode supports older implementations of DECnet (VMS Versions 4.5 and earlier) that do not support out-of-order packet caching. Other sessions might take another path, thus using equal-cost paths that a router might have for a particular destination.

To select normal or interim mode on the router, perform one of the following tasks in global configuration mode:

Task	Command
Specify that traffic is routed over equal-cost paths on a round-robin basis.	**decnet path-split-mode normal**
Specify that traffic is always routed over the same path.	**decnet path-split-mode interim**

Set Maximum Visits

You can determine the number of times that a packet can pass through a router. The router ignores packets that have a value greater than the amount of visits you specify. Digital recommends that the value be at least twice the number of maximum hops, to allow packets to reach their destinations when routes are changing.

To set the number of times a packet can pass through a router, perform the following task in global configuration mode:

Task	Command
Set the number of times a packet can pass through a router.	**decnet** [*network-number*] **max-visits** *value*

Adjust the Hello Timer

Hosts use the hello messages to identify the hosts with which they can communicate directly. The router sends hello messages every 15 seconds by default. On extremely slow serial lines, you might want to increase this value on a per-interface basis to reduce overhead.

To adjust the interval for sending hello messages, perform the following task in interface configuration mode:

Task	Command
Adjust the interval (in seconds) for sending hello messages on interfaces with DECnet enabled.	**decnet hello-timer** *seconds*

Disable Fast Switching

By default, our DECnet routing software implements fast switching of DECnet packets. You might want to disable fast switching in order to save memory space on interface cards and help avoid congestion when high-bandwidth interfaces are writing large amounts of information to low-bandwidth interfaces. This is especially important when using rates slower than T1.

To disable fast switching of DECnet packets, perform the following task in interface configuration mode:

Task	Command
Disable fast switching of DECnet packets on a per-interface basis.	**no decnet route-cache**

Set the Congestion Threshold

If a router configured for DECnet experiences congestion, it sets the *congestion-experienced* bit. You can define the congestion threshold on a per-interface basis. By setting this threshold, you will cause the system to set the congestion-experienced bit if the output queue has more than the specified number of packets in it.

To set the congestion threshold, perform the following task in interface configuration mode:

Task	Command
Set the congestion threshold.	**decnet congestion-threshold** *number*

Adjust the Broadcast Routing Timer

Other routers use broadcast updates to construct local routing tables. Increasing the time between routing updates on a per-interface basis reduces the amount of unnecessary network traffic. Digital calls this parameter the *broadcast routing timer* because Digital uses a different timer for serial lines. Our DECnet implementation does not make this distinction.

To adjust the broadcast routing timer, perform the following task in interface configuration mode:

Task	Command
Adjust how often the router sends routing updates that list all the hosts that the router can reach on a per-interface basis.	**decnet routing-timer** *seconds*

Configure DECnet over DDR

Dial-on-demand routing (DDR) is now supported for DECnet. Refer to the “Configuring DDR” chapter in this publication.

Configure DECnet over WANs

You can configure DECnet over X.25, SMDS, and Frame Relay networks. To do this, configure the appropriate address mappings as described in the "Configuring X.25 and LAPB," Configuring SMDS," and "Configuring Frame Relay" chapters, respectively.

Monitor and Maintain the DECnet Network

On page 15-15, replace the section "Monitor and Maintain the DECnet Network" with this revised section.

To clear counters, test network node reachability, and display information about DECnet networks, perform the following tasks in EXEC mode:

Task	Command
Clear the DECnet counters.	**clear decnet counters**
Test network node reachability.	**ping decnet** {*host* | *address*}
Display the global DECnet parameters.	**show decnet**
Display the global DECnet status and configuration for all interfaces, or the status and configuration for a specified interface, including address, paths, cost, access lists, and more.	**show decnet interface** [*interface unit*]
List a router's address mapping information used by the DECnet ATG.	**show decnet map**
Display all Phase IV and Phase IV Prime neighbors and the MAC address associated with each neighbor.	**show decnet neighbors**
Display a router's DECnet routing table.	**show decnet route** [*decnet-address*]
Display a router's static DECnet routing table.	**show decnet static**
List a router's DECnet traffic statistics, including datagrams sent, received, and forwarded.	**show decnet traffic**

DECnet Configuration Examples

The following sections provide examples that show some common DECnet configuration activities:

- Enabling DECnet Example
- Phase IV-to-Phase V Conversion Example
- Configuring Phase IV Areas through an OSI Backbone Example
- Configuring Address Translation Example
- Configuring DECnet Phase IV Prime Examples

Enabling DECnet Example

The following example illustrates the commands required for enabling DECnet. DECnet routing is established on a router at address 4.27. The node is configured as a Level 2, or interarea router. A cost of four is set for the Ethernet 0 interface. A cost of ten is set for the serial 1 interface.

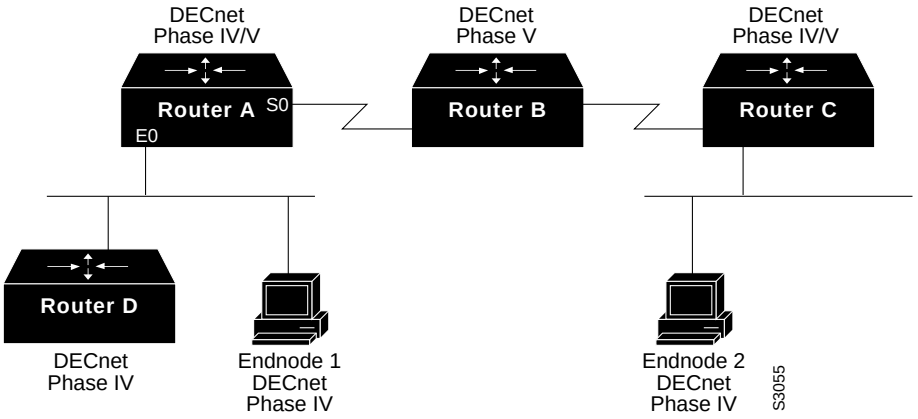
```
decnet routing 4.27
decnet node area
interface ethernet 0
decnet cost 4
interface serial 1
decnet cost 10
```

Phase IV-to-Phase V Conversion Example

The following example, illustrated in Figure 16-3, shows that for the DECnet Phase IV-to-Phase V conversion to work properly, CLNS ISIS must be configured on certain interfaces.

Note that although Router A has only Phase IV hosts connected by its Ethernet 0 interface, that interface must be configured for CLNS ISIS for Router A to convert the Phase IV adjacency information into Phase V. If Router A's Ethernet interface 0 is not configured for CLNS ISIS, Router B will never get information about Router D and endnode 1.

Figure 16-3 Sample Phase IV/V Network



Configuration for Router A

```
decnet routing 1.1
decnet conversion 49
clns routing
router isis
net 49.0001.aa00.0400.0104.00
interface e 0
clns router isis
decnet cost 4
interface s 0
clns router isis
```

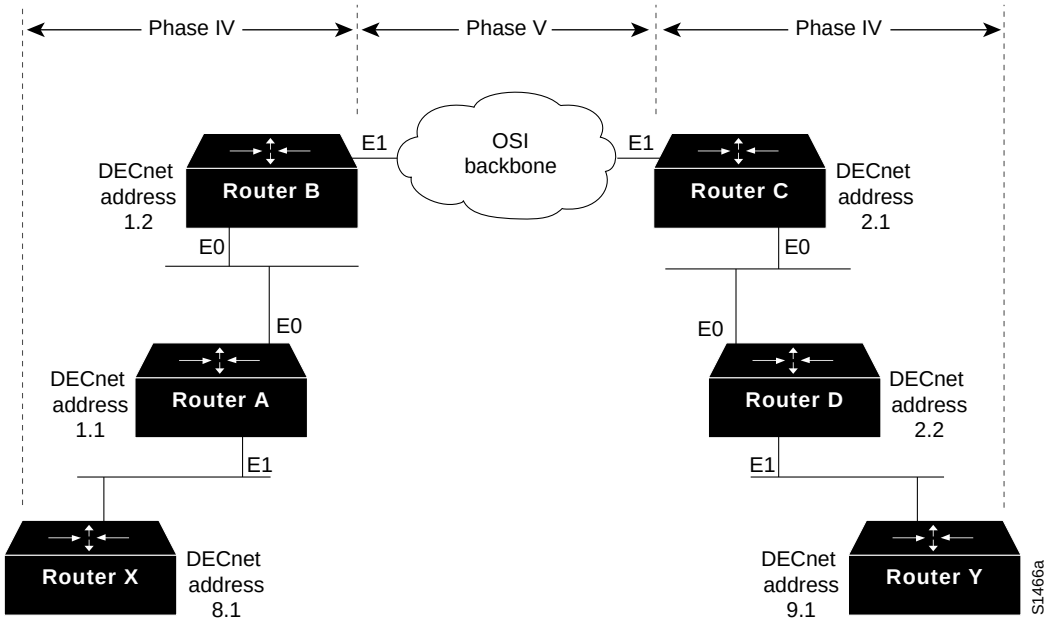
Sample Configuration for Router D

```
decnet routing 1.10
interface e 0
decnet cost 4
```

Configuring Phase IV Areas through an OSI Backbone Example

The following example illustrates how to configure border routers to propagate Phase IV areas through an OSI backbone using the advertise feature. In this example, Router X in area 8 wants to communicate with Router Y in area 9. Figure 16-4 illustrates the network, and the configurations that follow illustrate the commands required for enabling the advertise feature.

Figure 16-4 Sample Phase IV/Phase V Network



Configuration for Router B

```
decnet conversion 49
!Propagate Area 9 reachability information
decnet advertise 9 4 2
!Create dummy OSI route to force conversion to Phase IV
clns route 49.0008 discard
```

Configuration for Router C

```
decnet conversion 49
!Propagate Area 8 reachability information
decnet advertise 8 6 3
!Create dummy OSI route to force conversion to Phase IV
clns route 49.0009 discard
```

Router A's routing table will then contain the following, as displayed with the **show decnet route** EXEC command:

```
   Area       Cost  Hops   Next Hop to Node       Expires  Prio

*1              0     0     (Local) -> 1.1
*8              4     1   Ethernet1 -> 8.1           35     64    A
*9              5     2   Ethernet0 -> 1.2
   Node       Cost  Hops   Next Hop to Node       Expires  Prio

*(Area)         0     0     (Local) -> 1.1
```

```
*1.1            0     0     (Local) -> 1.1
*1.2            4     1   Ethernet4 -> 1.2           38     64    VA
```

Router B's routing table will then contain the following:

```
  Area        Cost  Hops   Next Hop to Node       Expires  Prio

*1              0     0     (Local) -> 1.2
*8              8     2   Ethernet0 -> 1.1
*9              4     2       (OSI) -> 1.2
  Node        Cost  Hops   Next Hop to Node       Expires  Prio

*(Area)         0     0     (Local) -> 1.2
*1.1            4     1   Ethernet0 -> 1.1           37     64    VA
*1.2            0     0     (Local) -> 1.2
```

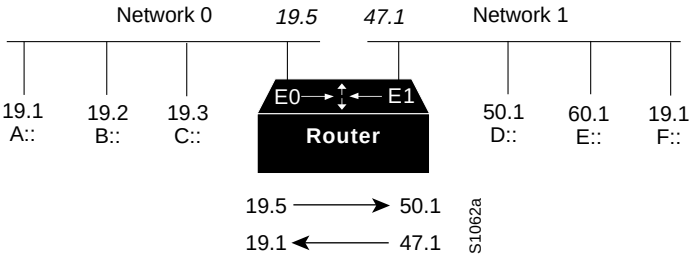
Router C's router table will then contain the following:

```
  Area        Cost  Hops   Next Hop to Node       Expires  Prio

*2              0     0     (Local) -> 2.1
*8              6     3       (OSI) -> 2.1
*9              8     2   Ethernet0 -> 2.2
  Node        Cost  Hops   Next Hop to Node       Expires  Prio

*(Area)         0     0     (Local) -> 2.1
*2.1            0     0     (Local) -> 2.1
*2.2            4     1   Ethernet0 -> 2.2           33     64    VA
```

Configuring Address Translation Example

In Figure 16-5, the router is connected to two DECnet networks using Ethernet. The following example illustrates how to configure an ATG between Network 0 and Network 1.

Figure 16-5 ATG Configuration Example



In Network 0, the router is configured at address 19.4 and is a Level 1 router. In Network 1, the router is configured at address 50.5 and is an area router. At this point, no routing information is exchanged between the two networks. Each network in the router has a separate routing table.

```
decnet 0 routing 19.4
decnet 0 node routing-iv
interface ethernet 0
decnet 0 cost 1
!
decnet 1 routing 50.5
decnet 1 node area
interface ethernet 1
decnet 1 cost 1
```

To establish a translation map, enter these commands:

```
decnet 0 map 19.5 1 50.1
decnet 1 map 47.1 0 19.1
```

Packets in Network 0 sent to virtual address 19.5 will be routed to Network 1, and the destination address will be translated to 50.1. Packets sent to virtual address 47.1 in Network 1 will be routed to Network 0 as 19.1.

Table 16-1 defines the parameters for the translation map.

Table 16-1 A Packet Exchange between Nodes A and D

Source		Destination	
A packet addressed as:	19.1	19.5	is received on Ethernet 0 as 19.5
Translates to:	47.1	50.1	and is transmitted out Ethernet 1 as 50.1
A reply packet:	50.1	47.1	is received on Ethernet 1
Translates to:	19.5	19.1	and is transmitted on Ethernet 0

Network 0 uses a block of addresses from its area to map the remote nodes. In Network 0, the router will advertise nodes 19.5 and 19.6. These nodes must not already exist in Network 0.

Network 1 uses another area for the address translation. Since the router will be advertising the availability of area 47, that area should not already exist in Network 1, because DECnet area fragmentation could occur.

Only nodes that exist in the maps on both networks will be able to communicate directly. Network 0 node 19.1 will be able to communicate with Network 1 node 50.1 (as 19.5), but will not be able to communicate directly with Network 1 node 60.1.

When naming nodes, use the appropriate address in each network. See the lists that follow for examples.

Network 0 VMS NCP Command File Sample

```
$ MCR NCP
define node 19.1 name A
define node 19.2 name B
define node 19.3 name C
define node 19.4 name GS
define node 19.5 name D
define node 19.6 name F
```

Network 1 VMS NCP Command File Sample

```
$ MCR NCP
define node 50.1 name D
define node 50.5 name GS
define node 60.1 name E
define node 19.1 name F
define node 47.1 name A
define node 47.2 name C
```

Configuring DECnet Phase IV Prime Examples

This section includes examples of configuring DECnet Phase IV Prime support for inherent MAC addresses. The comments in these examples point out some possible configuration errors, in addition to explaining correct command lines.

In the following example, Ethernet 0 interface is configured for DECnet Phase IV Prime.

```
decnet routing iv-prime 1.1
interface ethernet 0
decnet cost 10
! Interface Ethernet 0 will have aa-00-04-00 form of MAC address. Router is
! bilingual on interface Ethernet 0.
```

In the following example, Token Ring 1 interface is configured with a MAC address that is not supported by DECnet Phase IV.

```
decnet routing 2.1
interface Token Ring 1
decnet cost 5
mac-address 0000.0c00.62e6
! Interface Token Ring 1 has MAC address as set
! This is an error because the token ring interface has a MAC address that is
! not Phase IV-compatible, and the router is not running Phase IV Prime.
```

In the following example, the router is not configured to support DECnet Phase IV Prime until later in the configuration.

```
interface tokenring 1
decnet cost 5
mac-address 0000.0c00.62e6
! invalid configuration, since router is only Phase IV.
decnet routing iv-prime 5.5
! Become a Phase IV Prime router

interface tokenring 1
mac-address 0000.0c00.62e6
! Valid configuration since the router is now running Phase IV Prime.
```

The following example shows valid and invalid ways of using the **decnet multicast-map** command.

```
decnet routing iv-prime 3.4

interface tokenring 1
decnet multicast-map phiv-prime-all-bridges c000.2000.0000
! Invalid value (phiv-prime-all-bridges) for multicast ID string

interface tokenring 1
decnet multicast-map iv-prime-all-routers d000.2000.0000
! Invalid value (d000.2000.0000) for functional address

interface tokenring 1
decnet multicast-map iv-prime-all-routers c000.2000.0000
! This will work. The command redefines the multicast to functional address
! mapping for the "all Phase IV Prime routers" multicast.
```

CHAPTER 17

Configuring IP

The Internet Protocol (IP) is a packet-based protocol used to exchange data over computer networks. IP handles addressing, fragmentation, reassembly, and protocol demultiplexing. It is the foundation on which all other IP protocols, collectively referred to as the IP Protocol suite, are built. IP is a network-layer protocol that contains addressing and control information that allows data packets to be routed.

The Transmission Control Protocol (TCP) is built upon the IP layer. TCP is a connection-oriented protocol that specifies the format of data and acknowledgments used in the transfer of data. TCP also specifies the procedures that the computers use to ensure that the data arrives correctly. TCP allows multiple applications on a system to communicate concurrently because it handles all demultiplexing of the incoming traffic among the application programs.

This chapter describes how to configure the IP protocol. For a complete description of the commands in this chapter, refer to the "IP Commands" chapter of the *Router Products Command Reference* publication. For information on configuring the various IP routing protocols, refer to the "Configuring IP Routing Protocols" chapter of this manual. For historical background and a technical overview of IP, see the *Internetworking Technology Overview* publication.

Cisco's Implementation of IP

Cisco's implementation of IP provides most of the major services contained in the various protocol specifications. Cisco routers also provide the TCP and User Datagram Protocol (UDP) services called Echo and Discard, which are described in RFCs 862 and 863, respectively.

Cisco supports both TCP and UDP at the transport layer, for maximum flexibility in services. Cisco also supports all standards for IP broadcasts.

IP Configuration Task List

A number of tasks are associated with configuring IP. A basic and required task for configuring IP is to assign IP addresses to network interfaces. Doing so enables the interfaces and allows communication with hosts on those interfaces using IP. Associated with this task are decisions about subnetting and masking the IP addresses.

To configure IP, complete the tasks in the following sections:

- Assign IP Addresses to Network Interfaces
- Configure Address Resolution Methods
- Disable IP Routing
- Configure a Routing Process

- Configure Broadcast Packet Handling
- Configure IP Services
- Filter IP Packets
- Configure the Hot Standby Protocol
- Configure Basic IP Security Options
- Configure Extended IP Security Options
- Configure the DNSIX Audit Trail Facility
- Configure IP Accounting
- Configure Performance Parameters
- Configure IP over WANs
- Monitor and Maintain the IP Network

Remember that not all of the tasks in these sections are required. The tasks you need to perform will depend on your network and your needs.

At the end of this chapter, the examples in the "IP Configuration Examples" illustrate how you might configure your network using IP.

Assign IP Addresses to Network Interfaces

IP addresses identify locations to which IP datagrams can be sent. See the *Internetworking Technology Overview* publication for detailed information on IP addresses.

Some IP addresses are reserved for special uses and cannot be used for host, subnet, or network addresses. Table 17-1 lists ranges of IP addresses and shows which addresses are reserved and which are available for use.

Table 17-1 Reserved and Available IP Addresses

Class	Address or Range	Status
A	0.0.0.0 1.0.0.0 through 126.0.0.0 127.0.0.0	Reserved Available Reserved
B	128.0.0.0 128.1.0.0 through 191.254.0.0 191.255.0.0	Reserved Available Reserved
C	192.0.0.0 192.0.1.0 through 223.255.254 223.255.255.0	Reserved Available Reserved
D	224.0.0.0 through 239.255.255.255	Multicast group addresses
E	240.0.0.0 through 255.255.255.254 255.255.255.255	Reserved Broadcast

The official description of IP addresses is found in RFC 1166, "Internet Numbers."

To receive an assigned network number, contact your Internet service provider.

To assign an IP address and a network mask to a network interface on the router, perform the following task in interface configuration mode:

Task	Command
Set an IP address for an interface.	**ip address** *ip-address mask*

A mask identifies the bits that denote the network number in an IP address. When you use the mask to subnet a network, the mask is then referred to as a *subnet mask*. Subnets are described in the *Internetworking Technology Overview* publication.

Note We only support network masks that use contiguous bits that are flush left against the network field.

The tasks required to enable additional, optional, IP addressing features are contained in the following sections:

- Assign Multiple IP Addresses to Network Interfaces
- Enable Use of Subnet Zero
- Enable Classless Routing Behavior
- Enable IP Processing on a Serial Interface

Assign Multiple IP Addresses to Network Interfaces

The software supports multiple IP addresses per interface. You can specify an unlimited number of secondary addresses. Secondary IP addresses can be used in a variety of situations. The following are the most common applications:

- There might not be enough host addresses for a particular network segment. For example, your subnetting allows up to 254 hosts per logical subnet, but on one physical subnet you need to have 300 host addresses. Using secondary IP addresses on the routers allows you to have two logical subnets using one physical subnet.
- Many older networks were built using Level 2 bridges, and were not subnetted. The judicious use of secondary addresses can aid in the transition to a subnetted, router-based network. Routers on an older, bridged segment can easily be made aware that there are many subnets on that segment.
- Two subnets of a single network might otherwise be separated by another network. You can create a single network from subnets that are physically separated by another network by using a secondary address. In these instances, the first network is *extended*, or layered on top of the second network. Note that a subnet cannot appear on more than one active interface of the router at a time.

Note If any router on a network segment uses a secondary address, all other routers on that same segment must also use a secondary address from the same network or subnet.

To assign multiple IP addresses to network interfaces, perform the following task in interface configuration mode:

Task	Command
Assign multiple IP addresses to network interfaces.	**ip address** *ip-address mask* **secondary**

Note IP routing protocols sometimes treat secondary addresses differently when sending routing updates. See the description of IP split horizon in the "Configuring IP Routing Protocols" chapter for details.

See the "IP Configuration Examples" section at the end of the chapter for an example of creating a network from separated subnets.

Enable Use of Subnet Zero

Subnetting with a subnet address of zero is illegal and strongly discouraged (as stated in RFC 791) because of the confusion that can arise between a network and a subnet that have the same addresses. For example, if network 131.108.0.0 is subnetted as 255.255.255.0, subnet zero would be written as 131.108.0.0—which is identical to the network address.

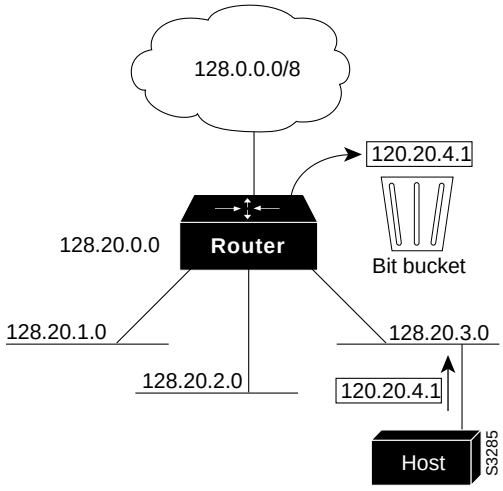
You can use the all zeros and all ones subnet (131.108.255.0), even though it is discouraged. Configuring interfaces for the all ones subnet is explicitly allowed. However, if you need the entire subnet space for your IP address, perform the following task in global configuration mode to enable subnet zero:

Task	Command
Enable the use of subnet zero for interface addresses and routing updates.	**ip subnet-zero**

Enable Classless Routing Behavior

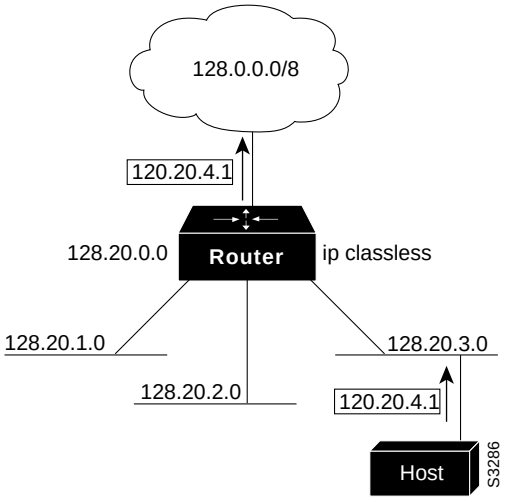
At times the router might receive packets destined for a subnet of a network that has no network default route. Figure 17-1 shows the router in network 128.20.0.0 connected to subnets 128.20.1.0, 128.20.2.0, and 128.20.3.0. Suppose the host sends a packet to 120.20.4.1. By default, if the router receives a packet destined for a subnet it does not recognize, and there is no network default route, the router discards the packet.

Figure 17-1 No IP Classless Routing



In Figure 17-2, classless routing is enabled in the router. Therefore, when the host sends a packet to 120.20.4.1, instead of discarding the packet, the router forwards the packet to the best supernet route.

Figure 17-2 IP Classless Routing



To have the router forward packets destined for unrecognized subnets to the best supernet route possible, perform the following task in global configuration mode:

Task	Command
Enable classless routing behavior.	**ip classless**

Enable IP Processing on a Serial Interface

You might want to enable IP processing on a serial or tunnel interface without assigning an explicit IP address to the interface. Whenever the unnumbered interface generates a packet (for example, for a routing update), it uses the address of the interface you specified as the source address of the IP packet. It also uses the specified interface address in determining which routing processes are sending updates over the unnumbered interface. Restrictions are as follows:

- Serial interfaces using HDLC, PPP, LAPB, and Frame Relay encapsulations, as well as SLIP and tunnel interfaces, can be unnumbered. It is not possible to use the unnumbered interface feature with X.25 or SMDS encapsulations.
- You cannot use the **ping** EXEC command to determine whether the interface is up, because the interface has no IP address. The Simple Network Management Protocol (SNMP) can be used to remotely monitor interface status.
- You cannot netboot a runnable image over an unnumbered serial interface.
- You cannot support IP security options on an unnumbered interface.

If you are configuring Intermediate System-to-Intermediate System (IS-IS) across a serial line, you should configure the serial interfaces as unnumbered. This allows you to conform with RFC 1195, which states that IP addresses are not required on each interface.

Note Using an unnumbered serial line between different major networks requires special care. If at each end of the link there are different major networks assigned to the interfaces you specified as unnumbered, any routing protocols running across the serial line should be configured to not advertise subnet information.

To enable IP processing on an unnumbered serial interface, perform the following task in interface configuration mode:

Task	Command
Enable IP processing on a serial or tunnel interface without assigning an explicit IP address to the interface.	**ip unnumbered** *interface-name*

The interface you specify must be the name of another interface in the router that has an IP address, not another unnumbered interface.

The interface you specify also must be enabled (listed as “up” in the **show interfaces** command display).

An example of how to configure serial interfaces can be found in the “IP Configuration Examples” section at the end of the chapter.

Configure Address Resolution Methods

Our IP implementation allows you to control interface-specific handling of IP addresses by facilitating address resolution, name services, and other functions. The following sections describe how to configure address resolution methods:

- Establish Address Resolution
- Map Host Names to IP Addresses

- Configure HP Probe Proxy Name Requests
- Configure the Next Hop Resolution Protocol

Establish Address Resolution

A device in the IP can have both a local address, which uniquely identifies the device on its local segment or LAN, and a network address, which identifies the network the device belongs to. The local address is more properly known as a data link address because it is contained in the data link layer (Layer 2 of the OSI model) part of the packet header and is read by data link devices (bridges and all device interfaces, for example). The more technically inclined will refer to local addresses as MAC addresses, because the Media Access Control (MAC) sublayer within the data link layer processes addresses for the layer.

To communicate with a device on Ethernet, for example, the router first must determine the 48-bit MAC or local data link address of that device. The process of determining the local data link address from an IP address is called *address resolution.* The process of determining the IP address from a local data link address is called *reverse address resolution.* The router uses three forms of address resolution: Address Resolution Protocol (ARP), proxy ARP, and Probe (which is similar to ARP). The router also uses the Reverse Address Resolution Protocol (RARP). The ARP, proxy ARP, and RARP protocols are defined in RFCs 826, 1027, and 903, respectively. Probe is a protocol developed by the Hewlett-Packard Company for use on IEEE-802.3 networks.

The Address Resolution Protocol (ARP) is used to associate IP addresses with media or MAC addresses. Taking an IP address as input, ARP determines the associated media address. Once a media or MAC address is determined, the IP address/media address association is stored in an ARP cache for rapid retrieval. Then the IP datagram is encapsulated in a link-layer frame and sent over the network. Encapsulation of IP datagrams and ARP requests and replies on IEEE 802 networks other than Ethernet is specified by the Subnetwork Access Protocol (SNAP).

The Reverse Address Resolution Protocol (RARP) works the same way as ARP, except that the RARP Request packet requests an IP address instead of a local data link address. Use of RARP requires a RARP server on the same network segment as the router interface. RARP often is used by diskless nodes that do not know their IP addresses when they boot. Our routers attempt to use RARP if they do not know the IP address of an interface at startup. Also, the routers are able to act as RARP servers by responding to RARP requests that they are able to answer. See the “Loading System Images, Microcode Images, and Configuration Files” chapter to learn how to configure a router as a RARP server.

Perform the following tasks to set address resolution on the router:

- Define a Static ARP Cache
- Set ARP Encapsulations
- Disable Proxy ARP
- Configure Local-Area Mobility

The procedures for performing these tasks are described in the following sections.

Define a Static ARP Cache

ARP and other address resolution protocols provide a dynamic mapping between IP addresses and media addresses. Because most hosts support dynamic address resolution, you generally do not need to specify static ARP cache entries. If you do need to define them, you can do so globally. Doing this task installs a permanent entry in the ARP cache. The router uses this entry to translate 32-bit IP addresses into 48-bit hardware addresses.

Optionally, you can specify that the router respond to ARP requests as if it were the owner of the specified IP address, and you also have the option of specifying an ARP entry timeout period when you define ARP entries.

The following two tables list the tasks to provide dynamic mapping between IP addresses and media address.

Perform either of the following tasks in global configuration mode:

Task	Command
Globally associate an IP address with a media (hardware) address in the ARP cache.	**arp** *ip-address hardware-address type*
Specify that the router respond to ARP requests as if it were the owner of the specified IP address.	**arp** *ip-address hardware-address type* **alias**

Perform the following task in interface configuration mode:

Task	Command
Set the length of time an ARP cache entry will stay in the cache.	**arp timeout** *seconds*

To display the type of ARP being used on a particular interface and also display the ARP timeout value, use the **show interfaces** EXEC command. Use the **show arp** EXEC command to examine the contents of the ARP cache. Use the **show ip arp** EXEC command to show IP entries. To remove all nonstatic entries from the ARP cache, use the privileged EXEC command **clear arp-cache**.

Set ARP Encapsulations

By default, standard Ethernet-style ARP encapsulation (represented by the **arpa** keyword) is enabled on the IP interface. You can change this encapsulation method to SNAP or HP Probe, as required by your network, to control the interface-specific handling of IP address resolution into 48-bit Ethernet hardware addresses.

When you set HP Probe encapsulation, the router uses the Probe protocol whenever it attempts to resolve an IEEE-802.3 or Ethernet local data link address. The subset of Probe that performs address resolution is called Virtual Address Request and Reply. Using Probe, the router can communicate transparently with Hewlett-Packard IEEE-802.3 hosts that use this type of data encapsulation. You must explicitly configure all interfaces for Probe that will use Probe.

To specify the ARP encapsulation type, perform the following task in interface configuration mode:

Task	Command
Specify one of three ARP encapsulation methods for a specified interface.	**arp** {**arpa** | **probe** | **snap**}

Disable Proxy ARP

The router uses proxy ARP, as defined in RFC 1027, to help hosts with no knowledge of routing determine the media addresses of hosts on other networks or subnets. For example, if the router receives an ARP request for a host that is not on the same network as the ARP request sender, and if the router has the best route to that host, then the router sends an ARP reply packet giving its own local data link address. The host that sent the ARP request then sends its packets to the router, which forwards them to the intended host. Proxy ARP is enabled by default.

To disable proxy ARP, perform the following task in interface configuration mode, as necessary, for your network:

Task	Command
Disable proxy ARP on the interface.	**no ip proxy-arp**

Configure Local-Area Mobility

Local-area mobility provides the ability to relocate IP hosts within a limited area without reassigning host IP addresses and without changes to the host software. Local-area mobility is supported on Ethernet, Token Ring, and FDDI interfaces only.

To create a mobility area with only one router, perform the following tasks:

Task		Command
Step 1	Enabled bridging on the router.	**bridge** *group* **protocol** {**dec** | **ieee**}[1]
Step 2	Enter interface configuration mode.	See Table 2-1.
Step 3	Enable local-area mobility.	**ip mobile arp** [**timers** *keepalive hold-time*] [**access-group** *access-list-number*]
Step 4	Configure bridging on the interface.	**bridge-group** *group*[1]

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

To create larger mobility areas, you must first redistribute the mobile routes into your IGP. The IGP must support host routes. You can use Enhanced IGRP, OSPF, or ISIS; you can also use RIP in some cases, but this is not recommended. To redistribute the mobile routes into your existing IGP configuration, perform the following tasks:

Task		Command
Step 1	Enter router configuration mode.	**router** {**eigrp** *autonomous-system* | **isis** [*tag*] | **ospf** *process-id*}
Step 2	Set default metric values.	**default-metric** *number* or **default-metric** *bandwidth delay reliability loading mtu*
Step 3	Redistribute the mobile routes.	**redistribute mobile**

If your IGP supports summarization, you should also restrict the mobile area so that it falls completely inside an IGP summarization area. This lets hosts roam within the mobile area without affecting routing outside the area.

The mobile area must consist of a contiguous set of subnets.

Hosts that roam within a mobile area should rely on a configured default router for their routing.

Map Host Names to IP Addresses

Each unique IP address can have a host name associated with it. The router maintains a cache of host name-to-address mappings for use by the EXEC **connect**, **telnet**, **ping** and related Telnet support operations. This cache speeds the process of converting names to addresses.

IP defines a naming scheme that allows a device to be identified by its location in the IP. This is a hierarchical naming scheme that provides for *domains*. Domain names are pieced together with periods (.) as the delimiting characters. For example, Cisco Systems is a commercial organization that the IP identifies by a *com* domain name, so its domain name is *cisco.com*. A specific device in this domain, the File Transfer Protocol (FTP) system for example, is identified as *ftp.cisco.com*.

To keep track of domain names, IP has defined the concept of a *name server* whose job it is to hold a cache, or database, of names mapped to IP addresses. To map domain names to IP addresses, you must first identify the host names, then specify a name server, and enable the Domain Name System (DNS), the Internet's global naming scheme that uniquely identifies network devices. You do these by performing the tasks in the following sections:

- Map IP Addresses to Host Names
- Specify the Domain Name
- Specify a Name Server
- Disable the DNS
- Use the DNS to Discover ISO CLNS Addresses

The following sections describe these tasks.

Map IP Addresses to Host Names

The router maintains a table of host names and their corresponding addresses, also called a host name-to-address mapping. Higher-layer protocols such as Telnet use host names to identify network devices (hosts). The router and other network devices must be able to associate host names with IP addresses to communicate with other IP devices. Host names and IP addresses can be associated with one another through static or dynamic means.

Manually assigning host names to addresses is useful when dynamic mapping is not available.

To assign host names to addresses, perform the following task in global configuration mode:

Task	Command
Statically associate host names with IP addresses.	**ip host** *name* [*tcp-port-number*] *address1* [*address2...address8*]

Specify the Domain Name

You can specify a default domain name that the router software will use to complete domain name requests. You can specify either a single domain name or a list of domain names. Any IP host name that does not contain a domain name will have the domain name you specify appended to it before being added to the host table.

To specify a domain name or names, perform either of the following tasks in global configuration mode:

Task	Command
Define a default domain name that the router will use to complete unqualified host names.	**ip domain-name** *name*
or	
Define a list of default domain names to complete unqualified host names.	**ip domain-list** *name*

See the "IP Configuration Examples" section at the end of this chapter for an example of establishing IP domains.

Specify a Name Server

To specify one or more hosts (up to six) that can function as a name server to supply name information for the DNS, perform the following task in global configuration mode:

Task	Command
Specify one or more hosts that supply name information.	**ip name-server** *server-address1* [[*server-address2*]...*server-address6*]

Disable the DNS

If your network devices require connectivity with devices in networks for which you do not control name assignment, you can assign device names that uniquely identify your devices within the entire internetwork. The Internet's global naming scheme, the DNS, accomplishes this task. This service is enabled by default.

To disable the DNS, perform the following task in global configuration mode:

Task	Command
Disable DNS-based host name-to-address translation.	**no ip domain-lookup**

See the "IP Configuration Examples" section at the end of this chapter for an example of enabling the DNS.

Use the DNS to Discover ISO CLNS Addresses

If your router has both IP and International Organization for Standardization Connectionless Network Service (ISO CLNS) enabled and you want to use ISO CLNS NSAP (Network Service Access Point) addresses, you can use the DNS to query these addresses, as documented in RFC 1348. This feature is enabled by default.

To disable DNS queries for ISO CLNS addresses, perform the following task in global configuration mode:

Task	Command
Disable DNS queries for ISO CLNS addresses.	**no ip domain-lookup nsap**

Configure HP Probe Proxy Name Requests

HP Probe Proxy support allows a router to respond to HP Probe Proxy name requests. These requests are typically used at sites that have Hewlett-Packard (HP) equipment and are already using HP Probe. Tasks associated with HP Probe Proxy are shown in the following two tables.

To configure HP Probe Proxy, perform the following task in interface configuration mode:

Task	Command
Allow the router to respond to HP Probe Proxy name requests.	**ip probe proxy**

Perform the following task in global configuration mode:

Task	Command
Enter the host name of an HP host (for which the router is acting as a proxy) into the host table.	**ip hp-host** *hostname ip-address*

See the "IP Configuration Examples" section at the end of this chapter for an example of configuring HP hosts on a network segment.

Configure the Next Hop Resolution Protocol

Routers and hosts can use Next Hop Resolution Protocol (NHRP) to discover the addresses of other routers and hosts connected to a nonbroadcast, multiaccess (NBMA) network. In the past, partially meshed NBMA networks had to be configured with overlapping LIS (logically independent IP subnets). In such configurations, packets might have had to make several hops over the NBMA network before arriving at the exit router (the router nearest the destination network). In addition, such NBMA networks (whether partially or fully meshed) have typically required tedious static configurations. These static configurations provided the mapping between network layer addresses (such as IP) and NBMA addresses (such as E.164 addresses for SMDS).

NHRP provides an ARP-like solution that alleviates these NBMA network problems. With NHRP, systems attached to an NBMA network can dynamically learn the NBMA address of the other systems that are part of that network. These systems can then directly communicate without requiring traffic to use an intermediate hop.

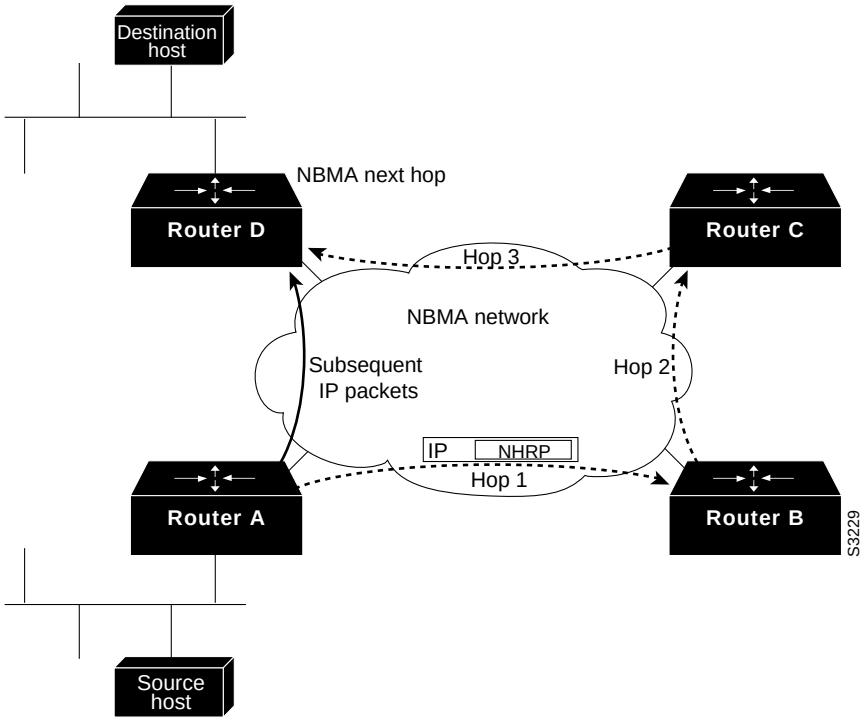
The NBMA network can be considered to be nonbroadcast either because it technically does not support broadcasting (for example, an X.25 network) or because broadcasting is not feasible (for example, an SMDS broadcast group or an extended Ethernet that would be too large).

Cisco's Implementation of NHRP

Cisco's initial implementation of NHRP supports only IP Version 4. Currently, NHRP can run on ATM, Ethernet, SMDS, and multipoint tunnel networks. Although NHRP is available on Ethernet, it is not necessary to implement NHRP over Ethernet media because Ethernet is capable of broadcasting.

Figure 17-3 illustrates four routers connected to an NBMA network. Within the network are ATM or SMDS switches necessary for the routers to communicate with each other. Assume that the switches have virtual circuit connections represented by hops 1, 2, and 3 of the figure. When Router A attempts to forward an IP packet from the source host to the destination host, NHRP is triggered. On behalf of the source host, Router A sends an NHRP request packet encapsulated in an IP packet, which takes three hops across the network to reach Router D, connected to the destination host. After receiving a positive NHRP reply, Router D is determined to be the "NBMA next hop," and Router A sends subsequent IP packets for the destination to Router D in one hop.

Figure 17-3 Next Hop Resolution Protocol (NHRP)



With NHRP, once the NBMA next hop is determined, the source can either start sending IP packets to the destination (in a connectionless NBMA network such as SMDS) or first establish a connection to the destination with the desired bandwidth and QOS characteristics (in a connection-oriented NBMA network such as ATM).

Other address resolution methods can be in use while NHRP is deployed. Hosts that can use only the LIS model might require ARP servers and services over NBMA networks, and deployed hosts might not implement NHRP but might continue to support ARP variations. NHRP is designed to eliminate the suboptimal routing that results from the LIS model, and can be deployed with existing ARP services without interfering with them.

NHRP can be used to facilitate building a virtual private network. In this context, a virtual private network consists of a virtual Layer 3 network that is built on top of an actual Layer 3 network. The topology you can use over the virtual private network can be largely independent of the underlying network, and the protocols you run over it can be completely independent of it.

Connected to the NBMA network are one or more Next Hop Servers, which implement NHRP. All routers running Release 10.3 or later are capable of implementing NHRP and thus can act as Next Hop Servers. A host or router that is not an NHRP speaker must be configured with the identity of the Next Hop Server that serves it.

Each Next Hop Server serves a set of destination hosts, which might or might not be directly connected to the NBMA network. Next Hop Servers cooperatively resolve the NBMA next hop addresses within their NBMA network. In addition to NHRP, Next Hop Servers typically participate in protocols used to disseminate routing information across (and beyond the boundaries of) the NBMA network, and might support ARP service also.

A Next Hop Server maintains a “next-hop resolution” cache, which is a table of IP-to-NBMA address mappings. The table is created from information gleaned from NHRP register packets, extracted from NHRP request or reply packets that traverse the Next Hop Server as they are forwarded, or through other means such as ARP and preconfigured tables.

Modes of Operation

NHRP supports two modes of operation: fabric and server modes. The modes differ in the way the Next Hop Server updates the destination address in the IP packet containing the NHRP Request.

Hosts attached directly to the NBMA network have no knowledge of whether NHRP is deployed in server or fabric mode and host configuration is the same in each case. Regardless of which mode is used, NHRP clients must be configured with the IP address and NBMA address of at least one Next Hop Server. In practice, a host's default router should also be its Next Hop Server.

Fabric Mode

In fabric mode, it is expected that all routers within the NBMA network are NHRP-capable. A Next Hop Server serving a destination must lie along the routed path to that destination. In practice, this means that all egress routers must double as Next Hop Servers serving the destinations beyond them, and that hosts on the NBMA network are served by routers that double as Next Hop Servers.

Server Mode

In server mode, few Next Hop Servers exist in an NBMA network. This might occur in networks having routers that do not support NHRP or networks that have many directly attached hosts and relatively few routers.

Server mode requires static configuration of Next Hop Server identity in the client stations (hosts or routers). The client station must be configured with the IP address of one or more Next Hop Servers, and there must be a path to that Next Hop Server (either directly, in which case the Next Hop Server's NBMA address must be known, or indirectly, through a router whose NBMA address is known). If there are multiple Next Hop Servers, they must be configured with each others' addresses, the identities of the destinations they each serve, and a logical NBMA network identifier. (This static configuration requirement, which might also involve authentication, tends to limit the number of Next Hop Servers.)

If the NBMA network offers a group addressing or multicast feature, the client station can be configured with a group address assigned to the group of Next Hop Servers. The client might then submit NHRP requests to the group address, eliciting a response from one or more Next Hop Servers, depending on the response strategy selected.

The servers can also be configured with the group or multicast address of their peers, and a Next Hop Server might use this address to forward NHRP requests it cannot satisfy to its peers. This might elicit a response. The Next Hop Server would then forward the NHRP reply to the NHRP request originator. The purpose of using group addressing or a similar multicast mechanism in this scenario is to eliminate the need to preconfigure each Next Hop Server in a logical NBMA network with both the individual identities of other Next Hop Servers and the destinations they serve. It reduces the number of Next Hop Servers that might be traversed to process an NHRP request (in those configurations where Next Hop Servers either respond or forward via the multicast, only two Next Hop Servers would be traversed) and allows the Next Hop Server that serves the NHRP request originator to cache next hop information associated with the reply.

NHRP Configuration Task List

To configure NHRP, perform the tasks described in the following sections. The first task is required, the remainder are optional.

- Enable NHRP on an Interface
- Configure a Station's Static IP-to-NBMA Address Mapping

- Statically Configure a Next Hop Server (Server Mode)
- Configure NHRP Authentication
- Control NHRP Initiation
- Suppress Forward and Reverse Record Options
- Specify the NHRP Responder Address
- Change the Time Period NBMA Addresses Are Advertised as Valid
- Configure a GRE Tunnel for Multipoint Operation

For NHRP configuration examples, see the section "IP Configuration Examples" later in this chapter.

Enable NHRP on an Interface

To enable NHRP for an interface on a router, perform the following task in interface configuration mode. In general, all NHRP stations within a logical NBMA network must be configured with the same network identifier.

Task	Command
Enable NHRP on an interface.	**ip nhrp network-id** *number*

For an example of enabling NHRP, see the section "Enabling NHRP Example" at the end of this chapter.

Configure a Station's Static IP-to-NBMA Address Mapping

To participate in NHRP, a station connected to an NBMA network should be configured with the IP and NBMA addresses of its Next Hop Server(s). The format of the NBMA address depends on the medium you are using. For example, ATM uses an NSAP address, Ethernet uses a MAC address, and SMDS uses an E.164 address.

Alternatively, the station should be configured with a means of acquiring those addresses, that is, the group address that can be used to reach the Next Hop Servers.

A third possibility is that the Next Hop Server(s) can be physically located on the stations's default or peer routers, so their IP addresses can be obtained from the station's IP forwarding table.

If the station is attached to several link layer networks (including logical NBMA networks), the station should also be configured to receive routing information from its Next Hop Server(s) and peer routers so that it can determine which IP networks are reachable through which link layer networks.

To configure static IP-to-NBMA address mapping on a station (host or router), perform the following task in interface configuration mode:

Task	Command
Configure static IP-to-NBMA address mapping.	**ip nhrp map** *ip-address nbma-address*

Statically Configure a Next Hop Server (Server Mode)

A Next Hop Server is configured with its own identity, a set of IP address prefixes that correspond to the IP addresses of the stations it serves, a logical NBMA network identifier, and in the case of server mode, the identities of other Next Hop Servers in the same logical NBMA network. If a served station is attached to several link layer networks, the Next Hop Server might also need to be configured to advertise routing information to such stations.

If a Next Hop Server acts as an egress router for stations connected to link layer networks other than the NBMA network, the Next Hop Server must also be configured to exchange routing information between the NBMA network and these other link layer networks.

In all cases, routing information is exchanged using conventional intradomain or interdomain routing protocols.

To statically configure a Next Hop Server, perform the following task in interface configuration mode:

Task	Command
Statically configure a Next Hop Server.	**ip nhrp nhs** *nhs-address* [*net-address* [*netmask*]]

To configure multiple networks that the Next Hop Server serves, repeat the **ip nhrp nhs** command with the same Next Hop Server address, but different IP network addresses. To configure additional Next Hop Servers, repeat the **ip nhrp nhs** command.

In the absence of static address configurations, a Next Hop Server operates in fabric mode and must itself learn the NBMA addresses of the stations it serves dynamically through NHRP.

Configure NHRP Authentication

Configuring an authentication string ensures that only routers configured with the same string can intercommunicate using NHRP. Therefore, if the authentication scheme is to be used, the same string must be configured in all routers configured for NHRP on a fabric. To specify the authentication string for NHRP on an interface, perform the following task in interface configuration mode:

Task	Command
Specify an authentication string.	**ip nhrp authentication** *string*

Control NHRP Initiation

You can specify an IP access list that is used to decide which IP packets can trigger the sending of NHRP requests. By default, all non-NHRP packets can trigger NHRP requests. To limit which IP packets trigger NHRP requests, you must define an access list and then apply it to the interface.

To define an access list, perform one of the following tasks in global configuration mode:

Task	Command
Define a standard IP access list.	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*]
Define an extended IP access list.	**access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**tos** *tos*] [**established**]

Then apply the IP access list to the interface by performing the following task in interface configuration mode:

Task	Command
Specify an IP access list that controls NHRP requests.	**ip nhrp interest** *access-list-number*

Suppress Forward and Reverse Record Options

To dynamically detect link-layer filtering in NBMA networks (for example, SMDS address screens), and to provide loop detection and diagnostic capabilities, NHRP incorporates a Route Record in requests and replies. The Route Record options contain the network (and link layer) addresses of all intermediate Next Hop Servers between source and destination (in the forward direction) and between destination and source (in the reverse direction).

By default, forward record options and reverse record options are included in NHRP request and reply packets. To suppress the use of these options, perform the following task in interface configuration mode:

Task	Command
Suppress forward and reverse record options.	**no ip nhrp record**

Specify the NHRP Responder Address

If an NHRP requestor wants to know which Next Hop Server generates an NHRP reply packet, it can request that information by including the responder address option in its NHRP request packet. The Next Hop Server that generates the NHRP reply packet then complies by inserting its own IP address in the NHRP reply. The Next Hop Server uses the primary IP address of the specified interface.

To specify which interface the Next Hop Server uses for the NHRP responder IP address, perform the following task in interface configuration mode.

Task	Command
Specify which interface the Next Hop Server uses to determine the NHRP responder address.	**ip nhrp responder** *type number*

If an NHRP reply packet being forwarded by a Next Hop Server contains that Next Hop Server's own IP address, the Next Hop Server generates an Error Indication of type "NHRP Loop Detected" and discards the reply.

Change the Time Period NBMA Addresses Are Advertised as Valid

You can change the length of time that NBMA addresses are advertised as valid in positive and negative NHRP responses. In this context, advertised means how long the router tells other routers to keep the addresses it is providing in NHRP responses. The default length of time for each response is 7200 seconds (2 hours). To change the length of time, perform the following task in interface configuration mode:

Task	Command
Specify the number of seconds that NBMA addresses are advertised as valid in positive or negative NHRP responses.	**ip nhrp holdtime** *seconds-positive* [*seconds-negative*]

Configure a GRE Tunnel for Multipoint Operation

You can enable a generic route encapsulation (GRE) tunnel to operate in multipoint fashion. A tunnel network of multipoint tunnel interfaces can be thought of as an NBMA network. To configure the tunnel, perform the following tasks in interface configuration mode:

Task	Command
Enable a GRE tunnel to be used in multipoint fashion.	**tunnel mode gre multipoint**
Configure a tunnel identification key.	**tunnel key** *key-number*

The tunnel key should correspond to the NHRP network identifier specified in the **ip nhrp network-id** command. For an example of NHRP configured on a multipoint tunnel, see the section "NHRP on a Multipoint Tunnel Example" at the end of this chapter.

Disable IP Routing

Every router ships with IP routing automatically enabled. If you choose to set up the router to bridge rather than route IP datagrams, you must disable IP routing. To disable IP routing, perform the following task in global configuration mode:

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

When IP routing is disabled, the router will act as an IP end host for IP packets destined for or sourced by it, whether or not bridging is enabled for those IP packets not destined for the router. To reenable IP routing, use the **ip routing** command.

Routing Assistance When IP Routing Is Disabled

The router software provides three methods by which the router can learn about routes to other networks when IP routing is disabled and the router is acting as an IP host:

- Proxy ARP
- A default gateway (also known as default router)
- The router discovery mechanism

When IP routing is disabled, the default gateway feature and the router discovery client are enabled, and proxy ARP is disabled. When IP routing is enabled, the default gateway feature is disabled and you can configure proxy ARP and the router discovery servers.

Proxy ARP

The most common method of learning about other routes is by using proxy ARP. Proxy ARP, defined in RFC 1027, enables an Ethernet host with no knowledge of routing to communicate with hosts on other networks or subnets. Such a host assumes that all hosts are on the same local Ethernet and that it can use ARP to determine their hardware addresses.

Under proxy ARP, if a router receives an ARP Request for a host that is not on the same network as the ARP Request sender, the router evaluates whether it has the best route to that host. If the router does have the best route, it sends an ARP Reply packet giving its own Ethernet hardware address.

The host that sent the ARP Request then sends its packets to the router, which forwards them to the intended host. The software treats all networks as if they are local and performs ARP requests for every IP address. This feature is enabled by default.

Proxy ARP works as long as other routers support it. Many other routers, especially host-based routing software, do not support it.

Default Gateway

Another method for locating routes is to define a default router (or gateway). The software sends all nonlocal packets to this router, which either routes them appropriately or sends an IP Control Message Protocol (ICMP) redirect message back to the router, telling it of a better route. The ICMP redirect message indicates which local router the host should use. The software caches the redirect messages and routes each packet thereafter as efficiently as possible. The limitations of this method are that there is no means of detecting when the default router has crashed or is unavailable and no method of picking another router if one of these events should occur.

To set up a default gateway for a host, perform the following task in global configuration mode:

Task	Command
Set up a default gateway (router).	**ip default-gateway** *ip-address*

To display the address of the default gateway, use the **show ip redirects** EXEC command.

Router Discovery Mechanism

The router software provides a third method, called *router discovery*, by which the router can dynamically learn about routes to other networks using the Gateway Discovery Protocol (GDP) or the ICMP Router Discovery Protocol (IRDP) for detecting routers. The software is also capable of wire-tapping Routing Information Protocol (RIP) and Interior Gateway Routing Protocol (IGRP) routing updates and inferring the location of routers from those updates. The server/client implementation of router discovery does not actually examine or store the full routing tables sent by routers, it merely keeps track of which systems are sending such data.

This mechanism supports the following protocols:

- Gateway Discovery Protocol (GDP)
- ICMP Router Discovery Protocol (IRDP)
- Routing Information Protocol (RIP)
- Interior Gateway Routing Protocol (IGRP)

You can configure these protocols in any combination. When possible, we recommend that you use GDP or IRDP because they allow each router to specify *both* a priority and the time after which a router should be assumed down if no further packets are received. Routers discovered using IGRP are assigned an arbitrary priority of 60. Routers discovered through RIP are assigned a priority of 50. For IGRP and RIP, the software attempts to measure the time between updates and will assume that the router is down if no updates are received for 2.5 times that interval.

Each router discovered becomes a candidate for the default router. The list of candidates is scanned and a new highest-priority router is selected when any of the following events occur:

- When a higher-priority router is discovered (the list of routers is polled at 5-minute intervals).
- When the current default router is declared down.

- When a TCP connection is about to time out because of excessive retransmissions. In this case, the server flushes the ARP cache and the ICMP redirect cache and picks a new default router in an attempt to find a successful route to the destination.

To configure the router discovery feature using the GDP routing protocol, perform the following task in interface configuration mode:

Task	Command
Use the GDP protocol to configure router discovery.	**ip gdp gdp**

To configure the router discovery feature using the IRDP routing protocol, perform the following task in interface configuration mode:

Task	Command
Use the IRDP protocol to configure router discovery.	**ip gdp irdp**

To configure the router discovery feature using the RIP routing protocol, perform the following task in interface configuration mode:

Task	Command
Use the RIP protocol to configure router discovery.	**ip gdp rip**

To configure the router discovery feature using the IGRP routing protocol, perform the following task in interface configuration mode:

Task	Command
Use the IGRP protocol to configure router discovery.	**ip gdp igrp**

Configure a Routing Process

At this point in the configuration process, you can choose to configure one or more of the many routing protocols that are available based on your individual network needs. Routing protocols provide topology information of an internetwork. Refer to the “Configuring IP Routing Protocols” chapter for the tasks involved in configuring IP routing protocols. If you want to continue to perform basic IP configuration tasks, continue reading the following sections.

Configure Broadcast Packet Handling

A *broadcast* is a data packet destined for all hosts on a particular physical network. Network hosts recognize broadcasts by special addresses. Broadcasts are heavily used by some protocols, including several important Internet protocols. Control of broadcast messages is an essential part of the IP network administrator’s job.

Our routers support two kinds of broadcasting: *directed broadcasting* and *flooding.* A directed broadcast is a packet sent to a specific network or series of networks, while a flooded broadcast packet is sent to every network. A directed broadcast address includes the network or subnet fields.

Several early IP implementations do not use the current broadcast address standard. Instead, they use the old standard, which calls for all zeros instead of all ones to indicate broadcast addresses. Many of these implementations do not recognize an all-ones broadcast address and fail to respond to the broadcast correctly. Others forward all-ones broadcasts, which causes a serious network overload known as a *broadcast storm*. Implementations that exhibit these problems include systems based on versions of BSD UNIX prior to Version 4.3.

Routers provide some protection from broadcast storms by limiting their extent to the local cable. Bridges (including intelligent bridges), because they are Layer 2 devices, forward broadcasts to all network segments, thus propagating all broadcast storms.

The best solution to the broadcast storm problem is to use a single broadcast address scheme on a network. Most modern IP implementations allow the network manager to set the address to be used as the broadcast address. Many implementations, including the one on our router, can accept and interpret all possible forms of broadcast addresses.

For detailed discussions of broadcast issues in general, see RFC 919, "Broadcasting Internet Datagrams," and RFC 922, "Broadcasting IP Datagrams in the Presence of Subnets." The router support for Internet broadcasts generally complies with RFC 919 and RFC 922; however, it does not support multisubnet broadcasts as defined in RFC 922.

The current broadcast address standard provides specific addressing schemes for forwarding broadcasts. Perform the tasks in the following sections to enable these schemes:

- Enable Directed Broadcast-to-Physical Broadcast Translation
- Forward UDP Broadcast Packets and Protocols
- Establish an IP Broadcast Address
- Flood IP Broadcasts

See the "IP Configuration Examples" section at the end of this chapter for broadcasting configuration examples.

Enable Directed Broadcast-to-Physical Broadcast Translation

To enable forwarding of directed broadcasts on an interface where the broadcast becomes a physical broadcast, perform one of the tasks that follow. By default, this feature is enabled only for those protocols configured using the **ip forward-protocol** global configuration command. You can specify an access list to control which broadcasts are forwarded. When an access list is specified, only those IP packets permitted by the access list are eligible to be translated from directed broadcasts to physical broadcasts.

Perform either of the following tasks in interface configuration mode as required for your network:

Task	Command
Enable directed broadcast-to-physical broadcast translation on an interface.	**ip directed-broadcast** [*access-list-number*]
Disable directed broadcast-to-physical broadcast translation on an interface.	**no ip directed-broadcast** [*access-list-number*]

Forward UDP Broadcast Packets and Protocols

Network hosts occasionally use UDP broadcasts to determine address, configuration, and name information. If such a host is on a network segment that does not include a server, UDP broadcasts are normally not forwarded. You can remedy this situation by configuring the interface of your router to forward certain classes of broadcasts to a helper address. You can have more than one helper address per interface.

You can specify a UDP destination port to control which UDP services are forwarded. You can specify multiple UDP protocols. You can also specify the Network Disk (ND) protocol, which is used by older diskless Sun workstations, and you can specify the network security protocol SDNS. By default, both UDP and ND forwarding are enabled if a helper address has been defined for an interface. The description for the **ip forward-protocol** command in the *Router Products Command Reference* publication lists the ports that are forwarded by default if you do not specify any UDP ports.

If you do not specify any UDP ports when you configure the forwarding of UDP broadcasts, you are configuring the router to act as a BOOTP forwarding agent. BOOTP packets carry Dynamic Host Configuration Protocol (DHCP) information. (DHCP is defined in RFC 1531.) This means that the router is now compatible with DHCP clients.

To enable forwarding and to specify the destination address, perform the following task in interface configuration mode:

Task	Command
Enable forwarding and specify the destination address for forwarding UDP broadcast packets, including BOOTP.	**ip helper-address** *address*

To specify which protocols will be forwarded, perform the following task in global configuration mode:

Task	Command
Specify which protocols will be forwarded over which ports.	**ip forward-protocol** {**udp** [*port*] | **nd** | **sdns**}

See the "IP Configuration Examples" section at the end of this chapter for an example of how to configure helper addresses.

Establish an IP Broadcast Address

The router supports IP broadcasts on both local- and wide-area networks. There are several ways to indicate an IP broadcast address. Currently, the most popular way, and the default, is an address consisting of all ones (255.255.255.255), although the routers can be configured to generate any form of IP broadcast address. Our routers also can receive and understand any form of IP broadcast.

To set the router's IP broadcast address, perform the following task in interface configuration mode:

Task	Command
Establish a different broadcast address (other than 255.255.255.255).	**ip broadcast-address** [*ip-address*]

If the router does not have nonvolatile memory, and you need to specify the broadcast address to use before the router has been configured, you have to change the IP broadcast address by setting jumpers in the processor configuration register. Setting bit 10 causes the router to use all zeros. Bit 10 interacts with bit 14, which controls the network and subnet portions of the broadcast address. Setting bit 14 causes the router to include the network and subnet portions of its address in the broadcast address. Table 17-2 shows the combined effect of setting bits 10 and 14.

Table 17-2 Configuration Register Settings for Broadcast Address Destination

Bit 14	Bit 10	Address (<net><host>)
Out	Out	<ones><ones>
Out	In	<zeros><zeros>
In	In	<net><zeros>
In	Out	<net><ones>

Some router platforms allow the configuration register to be set through the software; see the "Loading System Images, Microcode Images, and Configuration Files" chapter for details. For other router platforms, the configuration register can only be changed through hardware; see the appropriate hardware installation and maintenance manual for your system.

Flood IP Broadcasts

You can allow IP broadcasts to be flooded throughout your internetwork in a controlled fashion using the database created by the bridging spanning-tree protocol. Turning on this feature also prevents loops. In order to support this capability, the routing software must include the transparent bridging and bridging must be configured on each interface that is to participate in the flooding. If bridging is not configured on an interface, it still will be able to receive broadcasts, but the interface will never forward broadcasts it receives and the router will never use that interface to send broadcasts received on a different interface.

Packets that are forwarded to a single network address using the IP helper address mechanism can be flooded. Only one copy of the packet is sent on each network segment.

In order to be considered for flooding, packets must meet the following criteria. (Note that these are the same conditions used to consider packets forwarding via IP helper addresses.)

- The packet must be a MAC-level broadcast.
- The packet must be an IP-level broadcast.
- The packet must be a TFTP, DNS, Time, NetBIOS, ND, or BootP packet, or a UDP protocol specified by the **ip forward-protocol udp** global configuration command.
- The packet's time-to-live (TTL) value must be at least two.

A flooded UDP datagram is given the destination address you specified with the **ip broadcast-address** command on the output interface. The destination address can be set to any desired address. Thus, the destination address may change as the datagram propagates through the network. The source address is never changed. The TTL value is decremented.

After a decision has been made to send the datagram out on an interface (and the destination address possibly changed), the datagram is handed to the normal IP output routines and is therefore subject to access lists if they are present on the output interface.

To use the bridging spanning-tree database to flood UDP datagrams, perform the following task in global configuration mode:

Task	Command
Use the bridging spanning-tree database to flood UDP datagrams.	**ip forward-protocol spanning-tree**

If no actual bridging is desired, you can configure a type-code bridging filter that will deny all packet types from being bridged. Refer to the "Configuring Transparent Bridging" chapter for more information about using access lists to filter bridged traffic. The spanning-tree database is still available to the IP forwarding code to use for the flooding.

Speed Up Flooding of UDP Datagrams

You can speed up flooding of UDP datagrams using the spanning-tree algorithm. Used in conjunction with the **ip forward-protocol spanning-tree** command, this feature boosts the performance of spanning tree-based UDP flooding by a factor of about four to five times. The feature, called *turboflooding*, is supported over Ethernet interfaces configured for ARPA encapsulated, Fiber Distributed Data Interface (FDDI), and HDLC-encapsulated serial interfaces, but is not supported on Token Ring interfaces. As long as the Token Rings and the non-HDLC serial interfaces are not part of the bridge group being used for UDP flooding, turboflooding will behave normally.

To enable turboflooding, perform the following task in global configuration mode:

Task	Command
Use the bridging spanning-tree database to speed up flooding of UDP datagrams.	**ip forward-protocol turbo-flood**

Configure IP Services

The IP suite offers a number of services that control and manage IP connections. Many of these services are provided by the Internet Control Message Protocol (ICMP). ICMP messages are sent by routers to hosts or other routers when a problem is discovered with the Internet header. For detailed information on ICMP, see RFC 792.

To configure IP services, complete the tasks in the following sections:

- Disable ICMP Protocol Unreachable Messages
- Disable ICMP Redirect Messages
- Understand Path MTU Discovery
- Set the MTU Packet Size
- Enable ICMP Mask Reply Messages
- Disable IP Source Routing
- Configure Simplex Ethernet Interfaces

See the "IP Configuration Examples" section at the end of this chapter for examples of ICMP services.

Disable ICMP Protocol Unreachable Messages

If the router receives a nonbroadcast packet destined for itself that uses an unknown protocol, it sends an ICMP Protocol Unreachable message back to the source. Similarly, if the router receives a packet that it is unable to deliver to the ultimate destination because it knows of no route to the destination address, it sends an ICMP Host Unreachable message to the source. This feature is enabled by default.

You can disable this service by performing the following task in interface configuration mode:

Task	Command
Disable the sending of ICMP Protocol Unreachable and Host Unreachable messages.	**no ip unreachables**

Disable ICMP Redirect Messages

Routes sometimes can become less than optimal. For example, it is possible for the router to be forced to resend a packet through the same interface on which it was received. If this happens, the router sends an ICMP Redirect message to the packet's originator telling it that it is on a subnet directly connected to the router, and that it must forward the packet to another system on the same subnet. It does so because the originating host presumably could have sent that packet to the next hop without involving the router at all. The Redirect message instructs the sender to remove the router from the route and substitute a specified device representing a more direct path. This feature is enabled by default.

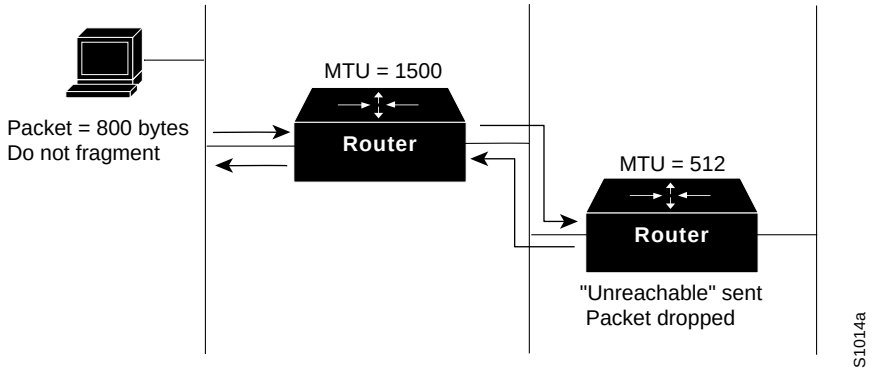
You can disable the sending of ICMP Redirect messages by performing the following task in interface configuration mode:

Task	Command
Disable the sending of ICMP Redirect messages to learn routes.	**no ip redirects**

Understand Path MTU Discovery

Our routers support the IP Path MTU Discovery mechanism, as defined in RFC 1191. IP Path MTU Discovery allows a host to dynamically discover and cope with differences in the maximum allowable maximum transmission unit (MTU) size of the various links along the path. Sometimes a router is unable to forward a datagram because it requires fragmentation (the packet is larger than the MTU you set for the interface with the **ip mtu** command), but the "Don't fragment" (DF) bit is set. The router sends a message to the sending host, alerting it to the problem. The host will have to fragment packets for the destination so that they fit the smallest packet size of all the links along the path. This technique is shown in Figure 17-4.

Figure 17-4 IP Path MTU Discovery



IP Path MTU Discovery is useful when a link in a network goes down, forcing use of another, different MTU-sized link (and different routers). As shown in Figure 17-4, suppose one is trying to send IP packets over a network where the MTU in the first router is set to 1500 bytes, but then reaches a router where the MTU is set to 512 bytes. If the datagram's "Don't fragment" bit is set, the datagram would be dropped because the 512-byte router is unable to forward it. All packets larger than 512 bytes will be dropped in this case. The second router returns an ICMP Destination Unreachable message to the source of the datagram with its Code field indicating "Fragmentation needed and DF set." To support IP Path MTU Discovery, it would also include the MTU of the next-hop network link in the low-order bits of an unused header field.

IP Path MTU Discovery is also useful when a connection is first being established and the sender has no information at all about the intervening links. It is always advisable to use the largest MTU that the links will bear; the larger the MTU, the fewer packets the host needs to send.

Note IP Path MTU Discovery is a process initiated by end hosts. If an end host does not support IP Path MTU Discovery, a router will have no mechanism available to avoid fragmenting datagrams generated by the end host.

The Cisco 7000, AGS+, and Cisco 4000 routers support fast switching of IP packets between Ethernet and FDDI interfaces. When packets are being sent from FDDI to Ethernet interfaces and you are not using IP Path MTU Discovery, FDDI packets with data lengths larger than 1500 bytes will be fragmented into multiple Ethernet packets. This will slow performance. If the majority of your router traffic travels off the FDDI ring, you might want to either lower the MTU size on your host FDDI interfaces to 1500 bytes or run IP Path MTU Discovery on your hosts.

Because the CTR card does not support the switching of frames larger than 4472 bytes, some interoperability problems may occur if CTR cards are intermixed with other Token Ring cards on the same network. You can minimize this by setting lower (and the same) IP maximum packet sizes for all devices on the network with the **ip mtu** interface command.

To enable Path MTU Discovery for connections initiated by the router (when the router is acting as a host), see the section "Enable Path MTU Discovery" later in this chapter.

Set the MTU Packet Size

All interfaces have a default MTU packet size. You can adjust the IP MTU size so that if an IP packet exceeds the MTU set for a router's interface, the router will fragment it.

Changing the MTU value (with the **mtu** interface configuration command) can affect the IP MTU value. If the current IP MTU value is the same as the MTU value, and you change the MTU value, the IP MTU value will be modified automatically to match the new MTU. However, the reverse is not true; changing the IP MTU value has no effect on the value for the **mtu** interface configuration command.

Also, all devices on a physical medium must have the same protocol MTU in order to operate.

To set the MTU packet size for a specified interface, perform the following task in interface configuration mode:

Task	Command
Set the IP MTU packet size for an interface.	**ip mtu** *bytes*

Enable ICMP Mask Reply Messages

Occasionally, network devices need to know the subnet mask for a particular subnetwork in the internetwork. To achieve this information, such devices can send ICMP Mask Request messages. These messages are responded to by ICMP Mask Reply messages from devices that have the requested information. The router can respond to ICMP Mask Request messages if this function is enabled.

To enable the sending of ICMP Mask Reply messages, perform the following task in interface configuration mode:

Task	Command
Enable the sending of ICMP Mask Reply messages.	**ip mask-reply**

Disable IP Source Routing

The router examines IP header options on every packet. It supports the IP header options *Strict Source Route*, *Loose Source Route*, *Record Route*, and *Time Stamp*, which are defined in RFC 791. If the router finds a packet with one of these options enabled, it performs the appropriate action. If it finds a packet with an invalid option, it sends an ICMP Parameter Problem message to the source of the packet and discards the packet.

IP provides a provision allowing the source IP host to specify a route through the IP network. This provision is known as *source routing*. Source routing is specified as an option in the IP header. If source routing is specified, the router forwards the packet according to the specified source route. This feature is employed when you want to force a packet to take a certain route through the network. The default is to perform source routing.

You can disable IP source-route header options by performing the following task in global configuration mode:

Task	Command
Cause the router to discard any IP datagram containing a source-route option.	**no ip source-route**

Configure Simplex Ethernet Interfaces

You can configure simplex Ethernet interfaces. This feature is useful for setting up dynamic IP routing over a simplex circuit; that is, a circuit that receives only or transmits only. When a route is learned on a receive-only interface, the interface designated as the source of the route is converted to the interface you specify. When packets are routed out this specified interface, they are sent to the IP address of the source of the routing update. To reach this IP address on a transmit-only Ethernet link, a static ARP entry mapping this IP address to the hardware address of the other end of the link is required.

To assign a transmit interface to a receive-only interface, perform the following task in interface configuration mode:

Task	Command
Assign a transmit interface to a receive-only interface.	**transmit-interface** *interface-name*

See the "IP Configuration Examples" section at the end of this chapter for an example of configuring a simplex Ethernet interface.

Filter IP Packets

Packet filtering helps control packet movement through the network. Such control can help limit network traffic and restrict network use by certain users or devices. To permit or deny packets from crossing specified router interfaces, we provide *access lists*.

You can use access lists in several ways:

- To control the transmission of packets on an interface
- To control virtual terminal line access
- To restrict contents of routing updates

This section summarizes how to create access lists and how to apply them.

See the "IP Configuration Examples" section at the end of this chapter for examples of configuring access lists.

An access list is a sequential collection of permit and deny conditions that apply to IP addresses. The router tests addresses against the conditions in an access list one by one. The first match determines whether the router accepts or rejects the address. Because the router stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the router rejects the address.

The two steps involved in using access lists are as follows:

Step 1 Create an access list by specifying an access list number and access conditions.

Step 2 Apply the access list to interfaces or terminal lines.

These steps are described in the next sections.

Create Standard and Extended Access Lists

Caution This release introduces substantial changes to IP access lists. These extensions are backward compatible; migrating from existing releases to this image will convert your access lists automatically. However, previous releases are not upwardly compatible with these changes. Thus, if you save an access list with this image and then use older software, the resulting access list will not be interpreted correctly. **This could cause you severe security problems.** Save your old configuration file before booting Release 10.3 images.

The software supports two styles of access lists for IP:

- Standard IP access lists use source addresses for matching operations.
- Extended IP access lists use source and destination addresses for matching operations, as well as optional protocol type information for finer granularity of control.

To create a standard access list, perform one of the following tasks in global configuration mode:

Task	Command
Define a standard IP access list using a source address and wildcard.	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*]
Define a standard IP access list using an abbreviation for the source and source mask of 0.0.0.0 255.255.255.255.	**access-list** *access-list-number* {**deny** | **permit**} **any**

To create an extended access list, perform one of the following tasks in global configuration mode:

Task	Command
Define an extended IP access list number and the access conditions.	**access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**tos** *tos*] [**established**]
Define an extended IP access list using an abbreviation for a source and source wildcard of 0.0.0.0 255.255.255.255 and an abbreviation for a destination and destination wildcard of 0.0.0.0 255.255.255.255.	**access-list** *access-list-number* {**deny** | **permit**} *protocol* **any any**
Define an extended IP access list using an abbreviation for a source and source wildcard of *source* 0.0.0.0 and an abbreviation for a destination and destination wildcard of *destination* 0.0.0.0.	**access-list** *access-list-number* {**deny** | **permit**} *protocol* **host** *source* **host** *destination*

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.

Note Keep in mind when making the standard and extended access list that by default, the end of the access list contains an implicit deny statement for everything if it did not find a match before reaching the end. Further, with standard access lists, if you omit the mask from an associated IP host address access list specification, 0.0.0.0 is assumed to be the mask.

Refer to the "IP Configuration Examples" section at the end of this chapter for examples of implicit masks.

Apply an Access List to an Interface or Terminal Line

After an access list is created, you can apply it to one or more interfaces. Access lists can be applied on *either* outbound or inbound interfaces. The following two tables show how this task is accomplished for both terminal lines and network interfaces.

Perform the following task in line configuration mode:

Task	Command
Restrict incoming and outgoing connections between a particular virtual terminal line (into a device) and the addresses in an access list.	**access-class** *access-list-number* {**in** | **out**}

Perform the following task in interface configuration mode:

Task	Command
Control access to an interface.	**ip access-group** *access-list-number* {**in** | **out**}

Note Autonomous switching is not used when you have extended access lists.

For inbound access lists, after receiving a packet, the router checks the source address of the packet against the access list. If the access list permits the address, the router continues to process the packet. If the access list rejects the address, the router discards the packet and returns an ICMP Host Unreachable message.

For outbound access lists, after receiving and routing a packet to a controlled interface, the router checks the source address of the packet against the access list. If the access list permits the address, the router transmits the packet. If the access list rejects the address, the router discards the packet and returns an ICMP Host Unreachable message.

When you apply an access list (standard or extended) that has not yet been defined to an interface, the router will act as if the access list has not been applied to the interface and will accept all packets. Remember this behavior if you use undefined access lists as a means of security in your network.

Note Set identical restrictions on all the virtual terminal lines, because a user can attempt to connect to any of them.

Configure the Hot Standby Protocol

The Hot Standby Router Protocol provides high network availability because it routes IP traffic from hosts on Ethernet, FDDI, or Token Ring networks without relying on the availability of any single router.

This feature is useful for hosts that do not support a router discovery protocol such as IRDP and do not have the functionality to switch to a new router when their selected router reloads or loses power. Because existing TCP sessions can survive the *failover*, this protocol also provides a more transparent means of recovery for hosts that dynamically select a next hop for routing IP traffic.

When the Hot Standby Router Protocol is configured on a network segment, it provides a virtual MAC address and an IP address that is shared among routers in a group of routers that are running the Hot Standby Router Protocol. One of these routers is selected by the protocol to be the active router. The active router receives and routes packets destined for the group's MAC address. For *n* routers running the Hot Standby Router Protocol, there are *n*+1 IP and MAC addresses assigned.

The Hot Standby Router Protocol detects when the designated active router fails, at which point a selected standby router assumes control of the Hot Standby group's MAC and IP addresses. A new standby router is also selected at that time.

Routers that are running the Hot Standby Router Protocol send and receive multicast UDP-based hello packets to detect router failure and to designate active and standby routers.

You can configure multiple Hot Standby groups on an interface, thereby making fuller use of the redundant routers. To do so, specify a group number for each Hot Standby command you configure for the interface.

Note Token Ring interfaces allow up to three Hot Standby groups each.

Note The Cisco 1000 series, Cisco 2500 series, Cisco 3000 series, and Cisco 4000 series that use Lance Ethernet hardware do not support multiple Hot Standby groups on a single Ethernet interface.

To enable the Hot Standby Router Protocol on an interface, perform the following task in interface configuration mode:

Task	Command
Enable the Hot Standby Router Protocol.	**standby** [*group-number*] **ip** [*ip-address*]

To configure other Hot Standby group attributes that affect how the local router participates in the Hot Standby Router Protocol, perform one or more of the following tasks in interface configuration mode:

Task	Command
Configure the time between hello packets and the holdtime before other routers declare the active router to be down.	**standby** [*group-number*] **timers** *hellotime holdtime*
Set the router's Hot Standby priority, used in choosing the active router.	**standby** [*group-number*] **priority** *priority-number*
Specify that, if the local router has priority over the current active router, the local router should attempt to take its place as the active router.	**standby** [*group-number*] **preempt**
Configure the interface to track other interfaces, so that if one of the other interfaces goes down, the router's Hot Standby priority is lowered.	**standby** [*group-number*] **track** *type number* [*interface-priority*]
Select an authentication string to be carried in all Hot Standby Router Protocol messages.	**standby** [*group-number*] **authentication** *string*

Configure Basic IP Security Options

Our IP Security Option (IPSO) support addresses both the basic and extended security options as described in RFC 1108. Our implementation is only minimally compliant with RFC 1108, because our router only accepts and generates a four-byte IPSO. IPSO is generally used to comply with the U.S. Government's DoD security policy.

Our basic IPSO support provides the following features:

- Defines security level on a per-interface basis
- Defines single-level or multilevel interfaces
- Provides a label for incoming packets
- Strips labels on a per-interface basis
- Reorders options to put any basic security options first

To configure basic IPSO, complete the tasks in the following sections:

- Enable IPSO and Set the Security Classifications
- Specify How IP Security Options Are Processed

Enable IPSO and Set the Security Classifications

To enable IPSO and set security classifications on an interface, perform either of the following tasks in interface configuration mode:

Task	Command
Set an interface to the requested IPSO classification and authorities.	**ip security dedicated** *level authority* [*authority...*]
or	
Set an interface to the requested IPSO range of classifications and authorities.	**ip security multilevel** *level1* [*authority1...*] **to** *level2 authority2* [*authority2...*]

Use the **no ip security** command to reset an interface to its default state.

Specify How IP Security Options Are Processed

To specify how IP security options are processed, perform any of the following optional tasks in interface configuration mode:

Task	Command
Enable an interface to ignore the authorities field of all incoming packets.	**ip security ignore-authorities**
Classify packets that have no IPSO with an implicit security label.	**ip security implicit-labelling** [*level authority* [*authority...*]]
Accept packets on an interface that has an extended security option present.	**ip security extended-allowed**
Ensure that all packets leaving the router on an interface contain a basic security option.	**ip security add**
Remove any basic security option that might be present on a packet leaving the router through an interface.	**ip security strip**

Task	Command
Prioritize security options on a packet.	**ip security first**
Treat as valid any packets that have Reserved1 through Reserved4 security levels.	**ip security reserved-allowed**

Default Values for Command Keywords

In order to fully comply with IPSO, the default values for the minor keywords have become complex. Default value usages include the following:

- The default for all of the minor keywords is *off*, with the exception of **implicit-labelling** and **add.**
- The default value of **implicit-labelling** is *on* if the interface is unclassified Genser; otherwise it is *off*.
- The default value for **add** is *on* if the interface is not "unclassified Genser"; and otherwise it is *off*.

Table 17-3 provides a list of all default values.

Table 17-3 Default Security Keyword Values

Interface Type	Level	Authority	Implicit Labeling	Add IPSO
None	None	None	On	Off
Dedicated	Unclassified	Genser	On	Off
Dedicated	Any	Any	Off	On
Multilevel	Any	Any	Off	On

The default value for any interface is "dedicated, unclassified Genser." Note that this implies implicit labeling. This might seem unusual, but it makes the system entirely transparent to packets without options. This is the setting generated when you specify the **no ip security** interface configuration command.

Configure Extended IP Security Options

Our extended IPSO support is compliant with the Department of Defense Intelligence Information System Network Security for Information Exchange (DNSIX) specification documents. Extended IPSO functionality can unconditionally accept or reject Internet traffic that contains extended security options by comparing those options to configured allowable values. This support allows DNSIX networks to use additional security information to achieve a higher level of security than that achievable with basic IPSO.

We also support a subset of the security features defined in the DNSIX Version 2.1 specification. Specifically, we support DNSIX definitions of the following:

- How extended IPSO is processed
- Audit trail facility

There are two kinds of extended IPSO fields defined by the DNSIX 2.1 specification and supported by our implementation of extended IPSO—Network Level Extended Security Option (NLESO) and Auxiliary Extended Security Option (AESO) fields.

NLESO processing requires that security options be checked against configured allowable information, source, and compartment bit values and requires that the router be capable of inserting extended security options in the IP header.

AESO is similar to NLESO, except that its contents are not checked and are assumed to be valid if its source is listed in the AESO table.

To configure extended IPSO, complete the tasks in the following sections:

- Configure Global Default Settings
- Attach ESOs to an Interface
- Attach AESOs to an Interface

DNSIX Version 2.1 causes slow-switching code.

Configure Global Default Settings

To configure global default setting for extended IPSO, including AESOs, perform the following task in global configuration mode:

Task	Command
Configure system-wide default settings.	**ip security eso-info** *source compartment-size default-bit*

Attach ESOs to an Interface

To specify the minimum and maximum sensitivity levels for an interface, perform the following tasks in interface configuration mode:

Task	Command
Set the minimum sensitivity level for an interface.	**ip security eso-min** *source compartment-bits*
Set the maximum sensitivity level for an interface.	**ip security eso-max** *source compartment-bits*

Attach AESOs to an Interface

To specify the extended IPSO sources that are to be treated as AESO sources, perform the following task in interface configuration mode:

Task	Command
Specify AESO sources.	**ip security aeso** *source compartment-bits*

Configure the DNSIX Audit Trail Facility

The audit trail facility is a UDP-based protocol that generate an audit trail of IPSO security violations. This facility allows the system to report security failures on incoming and outgoing packets. The Audit Trail Facility sends DNSIX audit trail messages when a datagram is rejected because of IPSO security violations. This feature allows you to configure organization-specific security information.

The DNSIX audit trail facility consists of two protocols:

- DNSIX Message Deliver Protocol (DMDP) provides a basic message-delivery mechanism for all DNSIX elements.
- Network Audit Trail Protocol (NAT) provides a buffered logging facility for applications to use to generate auditing information. This information is then passed on to DMDP.

To configure the DNSIX auditing facility, complete the tasks in the following sections:

- Enable the DNSIX Audit Trail Facility
- Specify Hosts to Receive Audit Trail Messages
- Specify Transmission Parameters

Enable the DNSIX Audit Trail Facility

To enable the DNSIX audit trail facility, perform the following task in global configuration mode:

Task	Command
Start the audit writing module.	**dnsix-nat source** *ip-address*

Specify Hosts to Receive Audit Trail Messages

To define and change primary and secondary addresses of the host to receive audit messages, perform the following task in global configuration mode:

Task	Command
Specify the primary address for the audit trail.	**dnsix-nat primary** *ip-address*
Specify the secondary address for the audit trail.	**dnsix-nat secondary** *ip-address*
Specify the address of a collection center that is authorized to change primary and secondary addresses. Specified hosts are authorized to change the destination of audit messages.	**dnsix-nat authorized-redirection** *ip-address*

Specify Transmission Parameters

To specify transmission paramters, perform the following tasks in global configuration mode:

Task	Command
Specify the number of records in a packet before it is sent to a collection center.	**dnsix-nat trasmit-count** *count*
Specify the number of transmit retries for DMDP.	**dnsix-dmdp retries** *count*

Configure IP Accounting

Our IP accounting support provides basic IP accounting functions. By enabling IP accounting, users can see the number of bytes and packets switched through the router on a source and destination IP address basis. Only transit IP traffic is measured and only on an outbound basis; traffic generated by the router or terminating in the router is not included in the accounting statistics. To maintain accurate accounting totals, the router software maintains two accounting databases: an active and a checkpointed database.

Our IP accounting support also provides information identifying IP traffic that fails IP access lists. Identifying IP source addresses that violate IP access lists alerts you to possible attempts to breach security. The data also indicates that you should verify IP access list configurations. To make this feature available to users, you must enable IP accounting of access list violations using the **ip accounting access-violations** command. Users can then display the number of bytes and packets from a single source that attempted to breach security against the access list for the source destination pair. By default, IP accounting displays the number of packets that have passed access lists and were routed.

To enable IP accounting, perform one of the following tasks for each interface in interface configuration mode:

Task	Command
Enable basic IP accounting.	**ip accounting**
Enable IP accounting with the ability to identify IP traffic that fails IP access lists.	**ip accounting access-violations**

To configure other IP accounting functions, perform one or more of the following tasks in global configuration mode:

Task	Command
Set the maximum number of accounting entries to be created.	**ip accounting-threshold** *threshold*
Filter accounting information for hosts.	**ip accounting-list** *ip-address mask*
Control the number of transit records that will be stored in the IP accounting database.	**ip accounting-transits** *count*

To display IP access violations for a specific IP accounting database, perform the following task in EXEC mode:

Task	Command
Display IP access-violation information.	**show ip accounting** [**checkpoint**] **access-violations**

To display IP access violations, you must give the **access-violations** keyword on the command. If you do not specify the keyword, the command defaults to displaying the number of packets that have passed access lists and were routed. The access violations output displays the number of the access list failed by the last packet for the source and destination pair. The number of packets reveals how aggressive the attack is upon a specific destination.

Use the EXEC command **show ip accounting** to display the active accounting database. To display the checkpointed database, use the **show ip accounting checkpoint** EXEC command. The **clear ip accounting** EXEC command clears the active database and creates the checkpointed database.

Configure Performance Parameters

To tune IP performance, complete the tasks in the following sections:

- Compress TCP Packet Headers
- Set the TCP Connection Attempt Time
- Enable Path MTU Discovery

- Enable Fast Switching
- Enable Fast Switching on the Same Interface
- Enable SSE Fast Switching
- Enable IP Autonomous Switching
- Control Route Cache Invalidation

Compress TCP Packet Headers

You can compress the headers of your TCP/IP packets in order to reduce their size, thereby increasing performance. Header compression is particularly useful on networks with a large percentage of small packets, such as those supporting many Telnet connections. This feature only compresses the TCP header, so it has no effect on UDP packets or other protocol headers. The TCP header compression technique, described fully in RFC 1144, is supported on serial lines using HDLC or PPP encapsulation. You must enable compression on both ends of a serial connection.

You can optionally specify outgoing packets to be compressed only if TCP incoming packets on the same interface are compressed. If you do not specify this option, the router will compress all traffic. The default is no compression.

You also can specify the total number of header compression connections that can exist on an interface. You should configure one connection for each TCP connection through the specified interface.

To enable compression, perform either of the following optional tasks in interface configuration mode:

Task	Command
Enable TCP header compression.	**ip tcp header-compression** [**passive**]
Specify the total number of header compression connections that can exist on an interface.	**ip tcp compression-connections** *number*

Note When compression is enabled, fast switching is disabled. Fast processors can handle several fast interfaces, such as T1s, that are running header compression. However, you should think carefully about your network's traffic characteristics before compressing TCP headers. You might want to use the monitoring commands to help compare network utilization before and after enabling header compression.

Set the TCP Connection Attempt Time

You can set the amount of time the router will wait to attempt to establish a TCP connection. In previous versions of router software, the system would wait a fixed 30 seconds when attempting to do so. This amount of time is not sufficient in networks that have dial-up asynchronous connections, such as a network consisting of dial-on-demand links that are implemented over modems because it will affect your ability to Telnet over the link (from the router) if the link must be brought up.

Because the connection attempt time is a host parameter, it does not pertain to traffic going through the router, just to traffic originated at the router.

To set the TCP connection attempt time, perform the following task in global configuration mode:

Task	Command
Set the amount of time the router will wait to attempt to establish a TCP connection.	**ip tcp synwait-time** *seconds*

Enable Path MTU Discovery

Path MTU Discovery is a method for maximizing the use of available bandwidth in the network between the end points of a TCP connection, and is described in RFC 1191. By default, this feature is disabled. Existing connections are not affected when this feature is turned on or off. To enable Path MTU Discovery, perform the following task in interface configuration mode:

Task	Command
Enable Path MTU Discovery.	**ip tcp path-mtu-discovery**

Customers using TCP connections to move bulk data between systems on distinct subnets would benefit most by enabling this feature. This might include customers using RSRB with TCP encapsulation, STUN, X.25 Remote Switching (also known as XOT or X.25 over TCP), and some protocol translation configurations.

The **ip tcp path-mtu-discovery** command is to enable Path MTU Discovery for connections initiated by the router when it is acting as a host. For a discussion of how the router supports Path MTU Discovery when the router is acting as a router, see the section “Understand Path MTU Discovery” earlier in this chapter.

Enable Fast Switching

Fast switching involves the use of a high-speed switching cache for IP routing. With fast switching, destination IP addresses are stored in the high-speed cache so that some time-consuming table lookups need not be done. Our routers generally offer better packet transfer performance when fast switching is enabled. However, if you have bursts of traffic, or if slow-speed serial links (64K and below) are being fed from higher-speed media such as T1 or Ethernet, then disabling fast switching can reduce the packet drop rate to some extent. Fast switching allows outgoing packets to be load balanced on a *per-destination* basis.

To enable or disable fast switching, perform the following tasks in interface configuration mode:

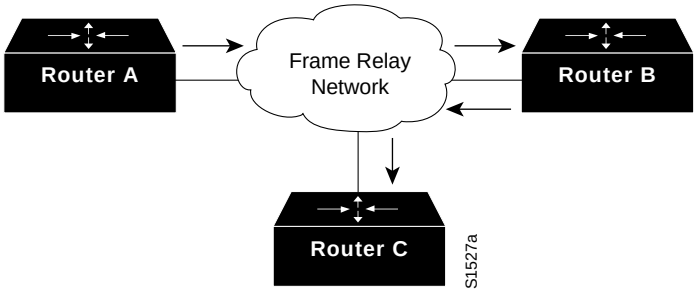
Task	Command
Enable fast-switching (use of a high-speed route cache for IP routing).	**ip route-cache**
Disable fast switching and enable load balancing on a per-packet basis.	**no ip route-cache**

Enable Fast Switching on the Same Interface

You can enable IP fast switching when the input and output interfaces are the same interface. This normally is not recommended, though it is useful when you have partially meshed media such as Frame Relay. You could use this feature on other interfaces, although it is not recommended because it would interfere with redirection.

Figure 17-5 illustrates a scenario where this is desirable. Router A has a DLCI to Router B, and Router B has a data link connection identifier (DLCI) to Router C. There is no DLCI between Routers A and C; traffic between them must go in and out of Router B through the same interface.

Figure 17-5 IP Fast Switching on the Same Interface



Perform the following task in interface configuration mode to allow IP fast switching on the same interface:

Task	Command
Enable the fast switching of packets out of the same interface on which they arrived.	**ip route-cache same-interface**

Enable SSE Fast Switching

The silicon switching engine (SSE) is on the Silicon Switch Processor (SSP) board in the Cisco 7000. SSE switching contributes to very fast packet processing by allowing the SSE to perform switching independently of the system processor.

To enable SSE fast switching, perform the following task in interface configuration mode:

Task	Command
Enable the SSE fast switching.	**ip route-cache sse**

The SSE can perform both standard and extended IP access list checking. That is, packets passing either standard or extended IP output access list checks can be SSE-switched. SSE switching of IP packets is not supported if input access lists are used. Prior to Release 10.3, when extended IP access lists were configured on a router having an SSE, IP traffic was fast-switched instead of SSE-switched.

Enable IP Autonomous Switching

Autonomous switching contributes to faster packet processing by allowing the high-speed router bus to switch packets independently without interrupting the system processor. This feature works only in Cisco 7000 or AGS+ systems with high-speed network controller cards, and with a switch processor or ciscoBus controller card running microcode Version 1.4 or later.

By default, IP autonomous switching is not enabled.

Perform any of the following tasks in interface configuration mode as needed for your network:

Task	Command
Enable both fast switching and autonomous switching.	**ip route-cache cbus**
Disable both fast switching and autonomous switching on an interface.	**no ip route-cache**
Disable only autonomous switching on an interface.	**no ip route-cache cbus**

Note Autonomous switching works only in high-end systems with high-speed network controller cards, such as the CSC-HSCI, CSC-MEC, CSC-FCI, CSC-C2/FCIT, and CSC-C2CTR Token Ring card, and with a ciscoBus controller card running microcode Version 1.4 or later. (See the information on microcode revisions in the microcode release notes accompanying this publication for other microcode revision requirements.)

Control Route Cache Invalidation

The high-speed route cache used by IP fast switching and autonomous switching is invalidated when the IP routing table changes. By default, the invalidation of the cache is delayed slightly to avoid excessive CPU load while the routing table is changing.

To control route cache invalidation, perform the following tasks in global configuration mode as needed for your network:

Task	Command
Allow immediate invalidation of the cache.	**no ip cache-invalidate-delay**
Delay invalidation of the cache.	**ip cache-invalidate-delay** [*minimum maximum quiet threshold*]

Note This task normally should not be necessary. It should be performed only under the guidance of technical staff. Incorrect configuration can seriously degrade the performance of your router.

Configure IP over WANs

You can configure IP over X.25, SMDS, Frame Relay, and DDR networks.To do this, configure the address mappings, as described in the appropriate wide-area networking chapters.

Monitor and Maintain the IP Network

To monitor and maintain your network, perform the tasks in the following sections:

- Clear Caches, Tables, and Databases
- Specify the Format of Network Masks
- Display System and Network Statistics
- Monitor and Maintain NHRP

Clear Caches, Tables, and Databases

You can remove all contents of a particular cache, table, or database. Clearing a cache, table, or database can become necessary when the contents of the particular structure have become or are suspected to be invalid.

The following table lists the tasks associated with clearing caches, tables, and databases. All are performed in EXEC mode.

Task	Command
Clear the IP ARP cache and the fast-switching cache.	**clear arp-cache**
Remove one or all entries from the host name and address cache.	**clear host** {*name* | *****}
Clear the active IP accounting or checkpointed database when IP accounting is enabled.	**clear ip accounting** [**checkpoint**]
Remove one or more routes from the IP routing table.	**clear ip route** {*network* [*mask*] | *****}
Cause the SSE route processor on the Cisco 7000 to recompute the program for IP and download it again.	**clear ip sse**
Cause the route processor on the Cisco 7000 to be reinitialized.	**clear sse**

Specify the Format of Network Masks

IP uses a 32-bit mask that indicates which address bits belong to the network and subnetwork fields and which bits belong to the host field. This is called a netmask. By default, **show** commands display an IP address and then its netmask in dotted decimal notation. For example, a subnet would be displayed as 131.108.11.55 255.255.255.0.

You might find it more convenient to display the network mask in hexadecimal format or bitcount format instead. The hexadecimal format is commonly used on UNIX systems. The above example would be displayed as 131.108.11.55 0XFFFFFF00.

The bitcount format for displaying network masks is to append a slash (/) and the total number of bits in the netmask to the address itself. The above example would be displayed as 131.108.11.55/24.

To specify the format in which netmasks appear for the current session, perform the following task in EXEC mode:

Task	Command
Specify the format of network masks for the current session.	**term ip netmask-format** {**bitcount** | **decimal** | **hexadecimal**}

To configure the format in which netmasks appear for an individual line, perform the following task in line configuration mode:

Task	Command
Configure the format of network masks for a line.	**ip netmask-format** {**bitcount** | **decimal** | **hexadecimal**}

Display System and Network Statistics

You can display specific router statistics such as the contents of IP routing tables, caches, and databases. Information provided can be used to determine resource utilization and solve network problems. You also can display information about node reachability and discover the routing path that your router's packets are taking through the network.

These tasks are summarized in the table that follows. See the "IP Commands" chapter in the *Router Products Command Reference* for details about the commands listed in these tasks. Perform the following tasks in privileged EXEC mode:

Task	Command
Display the contents of all current access lists.	**show access-lists**
Display the entries in the ARP table for the router.	**show arp**
Display state information and the current configuration of the DNSIX audit writing module.	**show dnsix**
Display the default domain name, style of lookup service, the name server hosts, and the cached list of host names and addresses.	**show hosts**
Display the contents of current IP access lists.	**show ip access-list** [*access-list-number*]
Display the active IP accounting or checkpointed database.	**show ip accounting** [**checkpoint**]
Display IP addresses mapped to TCP ports (aliases).	**show ip aliases**
Display the IP ARP cache.	**show ip arp**
Display the routing table cache used to fast switch IP traffic.	**show ip cache** [*prefix mask*] [*type number*]
Display the usability status of interfaces.	**show ip interface** [*type number*]
Display the masks used for network addresses and the number of subnets using each mask.	**show ip masks** *address*
Display the address of a default gateway.	**show ip redirects**
Display the current state of the routing table.	**show ip route** [*address* [*mask*]] | [*protocol*]
Display the current state of the routing table in summary form.	**show ip route summary**
Show statistics on TCP header compression.	**show ip tcp header-compression**
Display IP protocol statistics.	**show ip traffic**
Display a summary of SSP statistics.	**show sse summary**
Display the status of the standby router.	**show standby**
Test network node reachability (privileged).	**ping** [*protocol*] {*host* | *address*}
Test network node reachability using a simple ping facility (user).	**ping** [*protocol*] {*host* | *address*}
Trace packet routes through the network (privileged).	**trace** [*destination*]
Trace packet routes through the network (user).	**trace ip** *destination*

Monitor and Maintain NHRP

To monitor the NHRP cache or traffic, perform either of the following tasks in EXEC mode:

Task	Command
Display the IP NHRP cache, optionally limited to dynamic or static cache entries for a specific interface.	**show ip nhrp** [**dynamic** | **static**] [*type number*]
Display NHRP traffic statistics.	**show ip nhrp traffic**

The NHRP cache can contain static entries caused by statically configured addresses and dynamic entries caused by the router learning addresses from NHRP packets. To clear static entries, use the **no ip nhrp map** command. To clear the NHRP cache of dynamic entries, perform the following task in EXEC mode:

Task	Command
Clear the IP NHRP cache of dynamic entries.	**clear ip nhrp**

IP Configuration Examples

The following sections provide IP configuration examples:

- Serial Interfaces Configuration Example
- Creating a Network from Separated Subnets Example
- Dynamic Lookup Example
- Establishing IP Domains Example
- Configuring HP Hosts on a Network Segment Example
- Enabling NHRP Example
- NHRP on a Multipoint Tunnel Example
- NHRP Over ATM Example
- Helper Addresses Example
- Broadcasting Examples
- Customizing ICMP Services Example
- Simplex Ethernet Interfaces Example
- Access List Examples
- IPSO Configuration Examples
- Ping Command Example

Serial Interfaces Configuration Example

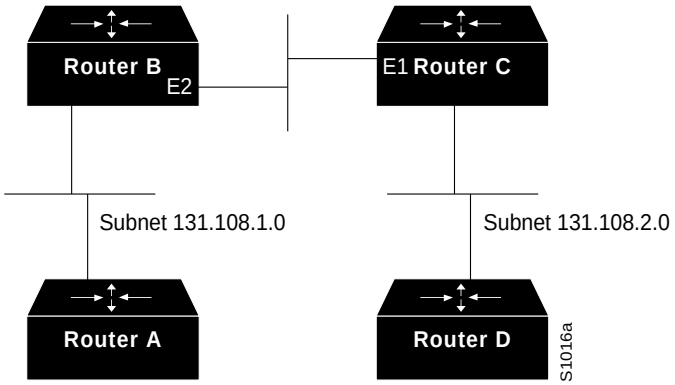
In the following example, the second serial interface (serial 1) is given Ethernet 0's address. The serial interface is unnumbered.

```
interface ethernet 0
ip address 145.22.4.67 255.255.255.0
interface serial 1
ip unnumbered ethernet 0
```

Creating a Network from Separated Subnets Example

In the following example, subnets 1 and 2 of network 131.108.0.0 are separated by a backbone, as shown in Figure 17-6. The two networks are brought into the same logical network through the use of secondary addresses.

Figure 17-6 Creating a Network from Separated Subnets



The following examples show the configurations for Routers B and C.

Configuration for Router B

```
interface ethernet 2
ip address 192.5.10.1 255.255.255.0
ip address 131.108.3.1 255.255.255.0 secondary
```

Configuration for Router C

```
interface ethernet 1
ip address 192.5.10.2 255.255.255.0
ip address 131.108.3.2 255.255.255.0 secondary
```

Dynamic Lookup Example

A cache of host name-to-address mappings is used by **connect**, **telnet**, **ping**, **trace**, **write net**, and **configure net** EXEC commands to speed the process of converting names to addresses. The commands used in this example specify the form of dynamic name lookup to be used. Static name lookup also can be configured.

The following example configures the host name-to-address mapping process for the router. IP DNS-based translation is specified, the addresses of the name servers are specified, and the default domain name is given.

```
! IP Domain Name System (DNS)-based host name-to-address translation is enabled
ip domain-lookup
! Specifies host 131.108.1.111 as the primary name server and host 131.108.1.2
! as the secondary server
ip name-server 131.108.1.111 131.108.1.2
```

```
! Defines cisco.com as the default domain name the router uses to complete
! unqualified host names
ip domain-name cisco.com
```

Establishing IP Domains Example

The example that follows establishes a domain list with several alternate domain names.

```
ip domain-list csi.com
ip domain-list telecomprog.edu
ip domain-list merit.edu
```

Configuring HP Hosts on a Network Segment Example

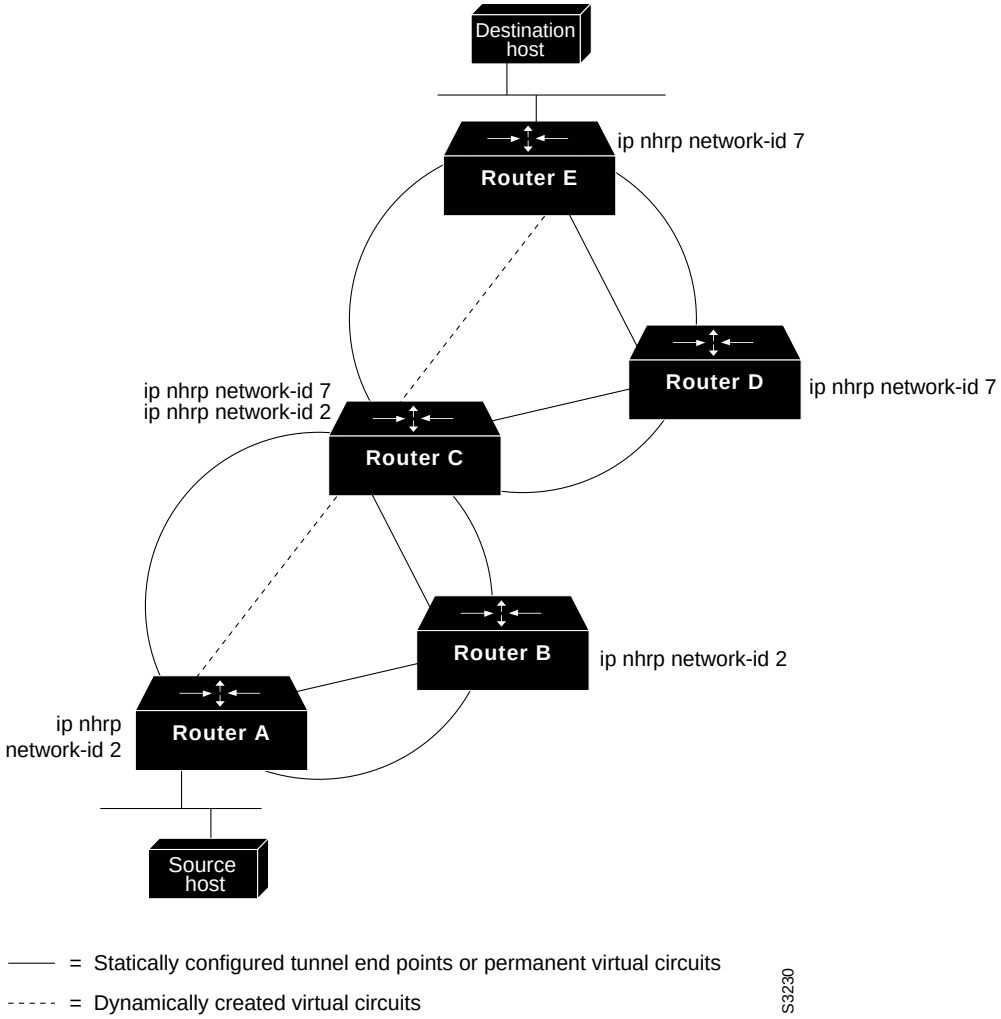
The following example has a network segment with Hewlett-Packard devices on it. The commands listed customize the router's first Ethernet port to respond to Probe name requests for bl4zip and to use Probe as well as ARP.

```
ip hp-host bl4zip 131.24.6.27
interface ethernet 0
arp probe
ip probe proxy
```

Enabling NHRP Example

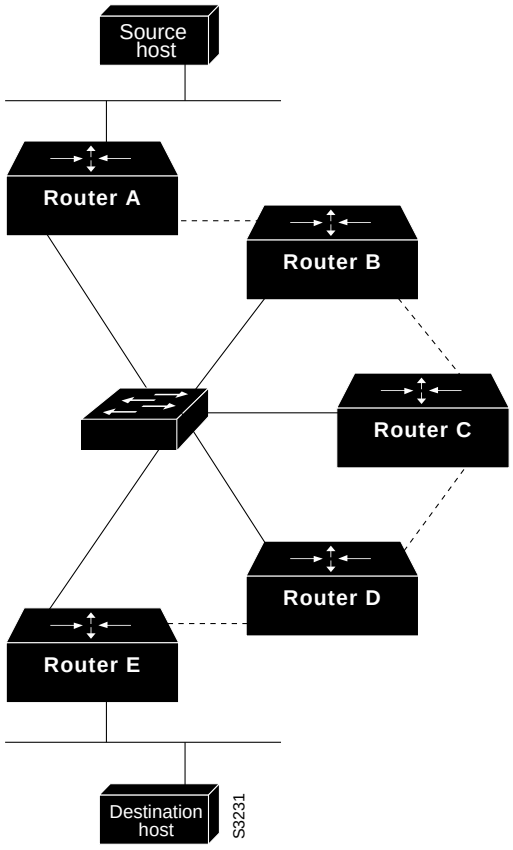
A logical NBMA network is considered the group of interfaces and hosts participating in NHRP and having the same network identifier. Figure 17-7 illustrates two logical NBMA networks (shown as circles) configured over a single physical NBMA network. Router A can communicate with Routers B and C because they share the same network identifier (2). Router C can also communicate with Routers D and E, as they share network identifier 7. After address resolution is complete, Router A can send IP packets to Router C in one hop, and Router C can send them to Router E in one hop, as shown by the dotted lines.

Figure 17-7 Two Logical NBMA Networks over One Physical NBMA Network



The physical configuration of the five routers in Figure 17-7 might actually be that shown in Figure 17-8. The source host is connected to Router A and the destination host is connected to Router E. The same switch serves all five routers, making one physical NBMA network.

Figure 17-8 Physical Configuration of a Sample NBMA



Refer again to Figure 17-7. Initially, before NHRP has resolved any NBMA addresses, IP packets from the source host to the destination host travel through all five routers connected to the switch before reaching the destination. When Router A first forwards the IP packet toward the destination host, Router A also generates an NHRP request for the destination host's IP address. The request is forwarded to Router C, whereupon a reply is generated. Router C replies because it is the egress router between the two logical NBMA networks.

Similarly, Router C generates an NHRP request of its own, which Router E replies to. In this example, subsequent IP traffic between the source and the destination still requires two hops to traverse the NBMA network, since the IP traffic must be forwarded between the two logical NBMA networks. Only one hop would be required if the NBMA network were not logically divided.

NHRP on a Multipoint Tunnel Example

With multipoint tunnels, a single tunnel interface may be connected to multiple neighboring routers. Unlike point-to-point tunnels, a tunnel destination need not be configured. In fact, if configured, the tunnel destination must correspond to an IP multicast address. Broadcast or multicast packets to be sent over the tunnel interface can then be transmitted by sending the GRE packet to the multicast address configured as the tunnel destination.

Multipoint tunnels require that you configure a tunnel key. Otherwise, unexpected GRE traffic could easily be received by the tunnel interface. For simplicity, it is recommended that the tunnel key correspond to the NHRP network identifier.

In the following example, Routers A, B, C, and D all share a common Ethernet segment. Minimal connectivity over the multipoint tunnel network is configured, thus creating a network that can be treated as a partially meshed NBMA network. Due to the static NHRP map entries, Router A knows how to reach Router B, Router B knows how to reach Router C, Router C know how to reach Router D, and Router D knows how to reach Router A.

When Router A initially attempts to send an IP packet to Router D, the packet is forwarded through Routers B and C. Through NHRP, the routers quickly learn each other's NBMA addresses (in this case, IP addresses assigned to the underlying Ethernet network). The partially meshed tunnel network readily becomes fully meshed, at which point any of the routers can directly communicate over the tunnel network without their IP traffic requiring an intermediate hop.

The significant portions of the configurations for Routers A, B, C, and D follow.

Router A

```
interface tunnel 0
no ip redirects
ip address 11.0.0.1 255.0.0.0
ip nhrp map 11.0.0.2 10.0.0.2
ip nhrp network-id 1
ip nhrp nhs 11.0.0.2
tunnel source ethernet 0
tunnel mode gre multipoint
tunnel key 1

interface ethernet 0
ip address 10.0.0.1 255.0.0.0
```

Router B

```
interface tunnel 0
no ip redirects
ip address 11.0.0.2 255.0.0.0
ip nhrp map 11.0.0.3 10.0.0.3
ip nhrp network-id 1
ip nhrp nhs 11.0.0.3
tunnel source ethernet 0
tunnel mode gre multipoint
tunnel key 1

interface ethernet 0
ip address 10.0.0.2 255.0.0.0
```

Router C

```
interface tunnel 0
no ip redirects
ip address 11.0.0.3 255.0.0.0
ip nhrp map 11.0.0.4 10.0.0.4
ip nhrp network-id 1
ip nhrp nhs 11.0.0.4
tunnel source ethernet 0
tunnel mode gre multipoint
tunnel key 1

interface ethernet 0
ip address 10.0.0.3 255.0.0.0
```

Router D

```
interface tunnel 0
no ip redirects
ip address 11.0.0.4 255.0.0.0
ip nhrp map 11.0.0.1 10.0.0.1
ip nhrp network-id 1
ip nhrp nhs 11.0.0.1
tunnel source ethernet 0
tunnel mode gre multipoint
tunnel key 1

interface ethernet 0
ip address 10.0.0.4 255.0.0.0
```

NHRP Over ATM Example

The following example shows a configuration of three routers using NHRP over ATM in fabric mode. Additionally, subinterfaces and dynamic routing are used. Router A obtains an OSPF route which it can use to reach the LIS where Router B resides. Router A can then initially reach Router B through Router C. Router A and Router B are able to directly communicate without Router C once NHRP has resolved Router A's and Router C's respective NSAP addresses.

The significant portions of the configurations for Routers A, B, and C follow.

Router A

```
interface ATM0/0
 ip address 10.1.0.1 255.255.0.0
 ip nhrp network-id 1
 map-group a
 atm nsap-address 11.1111.11.111111.1111.1111.1111.1111.1111.1111.11
 atm rate-queue 1 10
 atm pvc 1 0 5 qsaal

router ospf 1
 network 10.0.0.0 0.255.255.255 area 0

map-list a
ip 10.1.0.3 atm-nsap 33.3333.33.333333.3333.3333.3333.3333.3333.3333.33
```

Router B

```
interface ATM0/0
 ip address 10.2.0.2 255.255.0.0
 ip nhrp network-id 1
 map-group a
 atm nsap-address 22.2222.22.222222.2222.2222.2222.2222.2222.2222.22
 atm rate-queue 1 10
 atm pvc 2 0 5 qsaal

router ospf 1
 network 10.0.0.0 0.255.255.255 area 0

map-list a
ip 10.2.0.3 atm-nsap 33.3333.33.333333.3333.3333.3333.3333.3333.3333.33
```

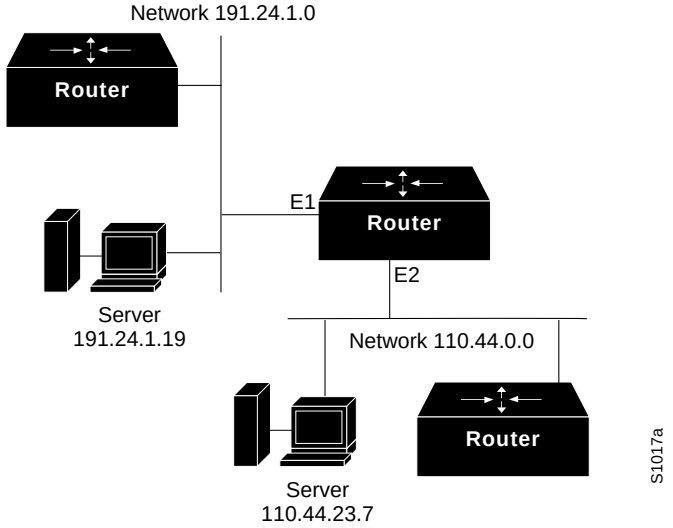
Router C

```
interface ATM0/0
 no ip address
 atm rate-queue 1 10
 atm pvc 2 0 5 qsaal

interface ATM0/0.1 multipoint
 ip address 10.1.0.3 255.255.0.0
 ip nhrp network-id 1
 map-group a
 atm nsap-address 33.3333.33.333333.3333.3333.3333.3333.3333.3333.33
 atm rate-queue 1 10

interface ATM0/0.2 multipoint
 ip address 10.2.0.3 255.255.0.0
 ip nhrp network-id 1
 map-group b
 atm nsap-address 33.3333.33.333333.3333.3333.3333.3333.3333.3333.33
 atm rate-queue 1 10

router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
 neighbor 10.1.0.1 priority 1
 neighbor 10.2.0.2 priority 1

map-list a
ip 10.1.0.1 atm-nsap 11.1111.11.111111.1111.1111.1111.1111.1111.1111.11

map-list b
ip 10.2.0.2 atm-nsap 22.2222.22.222222.2222.2222.2222.2222.2222.2222.22
```

Helper Addresses Example

In the following example, one router is on network 191.24.1.0 and the other is on network 110.44.0.0, and you want to permit IP broadcasts from hosts on either network segment to reach both servers. Figure 17-9 illustrates how to configure the router that connects network 110 to network 191.24.1.

Figure 17-9 IP Helper Addresses



The following example shows the configuration:

```
ip forward-protocol udp
!
interface ethernet 1
ip helper-address 110.44.23.7
interface ethernet 2
ip helper-address 191.24.1.19
```

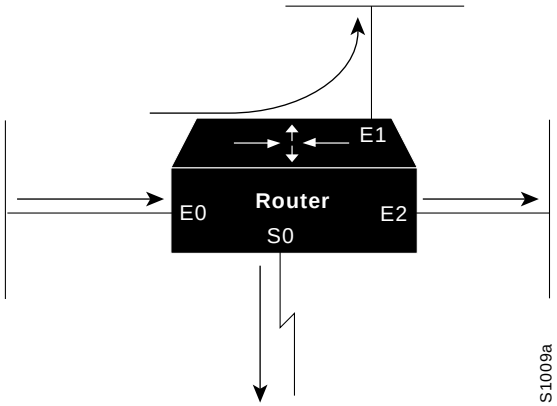
Broadcasting Examples

Our routers support two types of broadcasting: directed broadcasting and flooding. A directed broadcast is a packet sent to a specific network or series of networks, while a flooded broadcast packet is sent to every network. The following examples describe configurations for both types of broadcasting.

Flooded Broadcast Example

Figure 17-10 shows a flooded broadcast packet being sent to every network. The packet that is incoming from interface E0 is flooded to interfaces E1, E2 and S0.

Figure 17-10 IP Flooded Broadcast



A directed broadcast address includes the network or subnet fields. For example, if the network address is 128.1.0.0, the address 128.1.255.255 indicates all hosts on network 128.1.0.0. This would be a directed broadcast. If network 128.1.0.0 has a subnet mask of 255.255.255.0 (the third octet is the subnet field), the address 128.1.5.255 specifies all hosts on subnet 5 of network 128.1.0.0—another directed broadcast.

Flooding of IP Broadcasts Example

In the following example, flooding of IP broadcasts is enabled on all interfaces (two Ethernet and two serial). No bridging is permitted. The access list denies all protocols. No specific UDP protocols are listed by a separate **ip forward-protocol udp** interface configuration command, so the default protocols (TFTP, DNS, Time, NetBIOS, and BootP) will be flooded.

```
ip forward-protocol spanning-tree
bridge 1 protocol dec
access-list 201 deny 0x0000 0xFFFF
```

```
interface ethernet 0
bridge-group 1
bridge-group 1 input-type-list 201
interface ethernet 1
bridge-group 1
bridge-group 1 input-type-list 201
interface serial 0
bridge-group 1
bridge-group 1 input-type-list 201
interface serial 1
bridge-group 1
bridge-group 1 input-type-list 201
```

Customizing ICMP Services Example

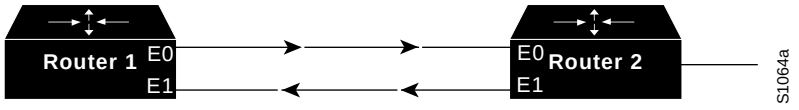
The example that follows changes some of the ICMP defaults for the first Ethernet interface 0. Disabling the sending of redirects could mean that you do not think your routers on this segment will ever have to send a redirect. Disabling the Unreachables messages will have a secondary effect—it also will disable IP Path MTU Discovery, because path discovery works by having routers send Unreachables messages. If you have a network segment with a small number of devices and an absolutely reliable traffic pattern—which could easily happen on a segment with a small number of little-used user devices—you would be disabling options that your router would be unlikely to use anyway.

```
interface ethernet 0
no ip unreachables
no ip redirects
```

Simplex Ethernet Interfaces Example

The following is an example of configuring a simplex Ethernet interface. Figure 17-11 illustrates how to configure IP on two routers sharing transmit-only and receive-only Ethernet connections.

Figure 17-11 Simplex Ethernet Connections



Configuration for Router 1

```
interface ethernet 0
ip address 128.9.1.1
!
interface ethernet 1
ip address 128.9.1.2
transmit-interface ethernet 0
!
!use show interfaces command to find router2-MAC-address-E0
arp 128.9.1.4 router2-MAC-address-E0 arpa
```

Configuration for Router 2

```
interface ethernet 0
ip address 128.9.1.3
```

```
transmit-interface ethernet 1
!
interface ethernet 1
ip address 128.9.1.4
!
!use show interfaces command to find router1-MAC-address-E1
arp 128.9.1.1 router1-MAC-address-E1 arpa
```

Access List Examples

In the following example, network 36.0.0.0 is a Class A network whose second octet specifies a subnet; that is, its subnet mask is 255.255.0.0. The third and fourth octets of a network 36.0.0.0 address specify a particular host. Using access list 2, the router would accept one address on subnet 48 and reject all others on that subnet. The last line of the list shows that the router would accept addresses on all other network 36.0.0.0 subnets.

```
access-list 2 permit 36.48.0.3
access-list 2 deny 36.48.0.0  0.0.255.255
access-list 2 permit 36.0.0.0  0.255.255.255
interface ethernet 0
ip access-group 2 in
```

Examples of Implicit Masks in Access Lists

IP access lists contain *implicit* masks. For instance, if you omit the mask from an associated IP host address access list specification, 0.0.0.0 is assumed to be the mask. Consider the following example configuration:

```
access-list 1 permit 0.0.0.0
access-list 1 permit 131.108.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
```

For this example, the following masks are implied in the first two lines:

```
access-list 1 permit 0.0.0.0 0.0.0.0
access-list 1 permit 131.108.0.0 0.0.0.0
```

The last line in the configuration (using the deny keyword) can be left off, because IP access lists implicitly *deny* all other access. This is equivalent to finishing the access list with the following command statement:

```
access-list 1 deny 0.0.0.0 255.255.255.255
```

The following access list only allows access for those hosts on the three specified networks. It assumes that subnetting is not used; the masks apply to the host portions of the network addresses. Any hosts with a source address that does not match the access list statements will be rejected.

```
access-list 1 permit 192.5.34.0  0.0.0.255
access-list 1 permit 128.88.0.0  0.0.255.255
access-list 1 permit 36.0.0.0  0.255.255.255
! (Note: all other access implicitly denied)
```

To specify a large number of individual addresses more easily, you can omit the address mask that is all zeros from the **access-list** global configuration command. Thus, the following two configuration commands are identical in effect:

```
access-list 2 permit 36.48.0.3
access-list 2 permit 36.48.0.3  0.0.0.0
```

Examples of Configuring Extended Access Lists

In the following example, the first line permits any incoming TCP connections with destination ports greater than 1023. The second line permits incoming TCP connections to the SMTP port of host 128.88.1.2. The last line permits incoming ICMP messages for error feedback.

```
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.0.0 0.0.255.255 gt 1023
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.1.2 0.0.0.0 eq 25
access-list 102 permit icmp 0.0.0.0 255.255.255.255 128.88.0.0 255.255.255.255
interface ethernet 0
ip access-group 102 in
```

For another example of using an extended access list, suppose you have a network connected to the Internet, and you want any host on an Ethernet to be able to form TCP connections to any host on the Internet. However, you do not want IP hosts to be able to form TCP connections to hosts on the Ethernet except to the mail (SMTP) port of a dedicated mail host.

SMTP uses TCP port 25 on one end of the connection and a random port number on the other end. The same two port numbers are used throughout the life of the connection. Mail packets coming in from the Internet will have a destination port of 25. Outbound packets will have the port numbers reversed. The fact that the secure system behind the router always will be accepting mail connections on port 25 is what makes it possible to separately control incoming and outgoing services.The access list can be configured on either the outbound or inbound interface.

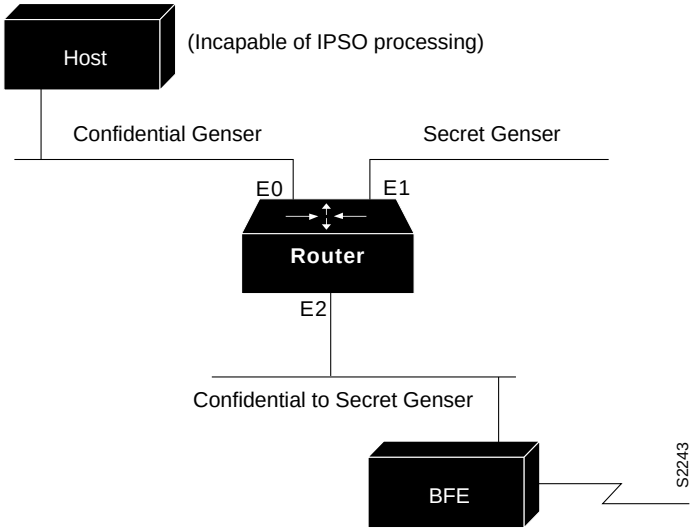
In the following example, the Ethernet network is a Class B network with the address 128.88.0.0, and the mail host's address is 128.88.1.2. The keyword **established** is used only for the TCP protocol to indicate an established connection. A match occurs if the TCP datagram has the ACK or RST bits set, which indicate that the packet belongs to an existing connection.

```
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.0.0 0.0.255.255 established
access-list 102 permit tcp 0.0.0.0 255.255.255.255 128.88.1.2 0.0.0.0 eq 25
interface ethernet 0
ip access-group 102 in
```

IPSO Configuration Examples

In the following example, three Ethernet interfaces are presented. These interfaces are running at security levels of Confidential Genser, Secret Genser, and Confidential to Secret Genser, as shown in
Figure 17-12.

Figure 17-12 IPSO Security Levels



The following commands set up interfaces for the configuration in Figure 17-12:

```
interface ethernet 0
ip security dedicated confidential genser
interface ethernet 1
ip security dedicated secret genser
interface ethernet 2
ip security multilevel confidential genser to secret genser
```

It is possible for the setup to be much more complex.

In the following example, there are devices on Ethernet 0 that cannot generate a security option, and so must accept packets without a security option. These hosts do not understand security options; therefore, never place one on such interfaces. Furthermore, there are hosts on the other two networks that are using the extended security option to communicate information, so you must allow these to pass through the system. Finally, there also is a host (a Blacker Front End; see the "Configuring X.25 and LABP" chapter for more information about Blacker emergency mode) on Ethernet 2 that requires the security option to be the first option present, and this condition also must be specified. The new configuration follows.

```
interface ethernet 0
ip security dedicated confidential genser
ip security implicit-labelling
ip security strip
interface ethernet 1
ip security dedicated secret genser
ip security extended-allowed
!
interface ethernet 2
ip security multilevel confidential genser to secret genser
ip security extended-allowed
ip security first
```

Ping Command Example

You can specify the router address to use as the source address for ping packets. In the following example, it is 131.108.105.62:

```
Sandbox# ping
Protocol [ip]:
Target IP address: 131.108.1.111
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: yes
Source address: 131.108.105.62
Type of service [0]:
Set DF bit in IP header? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 131.108.1.111, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/4/4 ms
```

CHAPTER 18

Configuring IP Routing Protocols

This chapter describes how to configure the various Internet Protocol (IP) routing protocols. For a complete description of the commands listed in this chapter, refer to the "IP Routing Protocols Commands" chapter of the *Router Products Command Reference* publication. For information on configuring the IP protocol, refer to the "Configuring IP" chapter of this manual. For historical background and a technical overview of IP routing protocols, see the *Internetworking Technology Overview* publication.

Cisco's Implementation of IP Routing Protocols

Cisco's implementation of each of the IP routing protocols is discussed in detail at the beginning of the individual protocol sections throughout this chapter.

IP routing protocols are divided into two classes: interior gateway protocols (IGPs) and exterior gateway protocols (EGPs). The IGPs and EGPs that Cisco supports are listed in the following sections.

Note Many routing protocol specifications refer to routers as *gateways*, so the word *gateway* often appears as part of routing protocol names. However, a router usually is defined as a Layer 3 internetworking device, whereas a protocol translation gateway usually is defined as a Layer 7 internetworking device. The reader should understand that whether a routing protocol name contains the word "gateway" or not, routing protocol activities occur at Layer 3 of the OSI reference model.

Interior Gateway Protocols

Interior protocols are used for routing networks that are under a common network administration. All IP interior gateway protocols must be specified with a list of associated networks before routing activities can begin. A routing process listens to updates from other routers on these networks and broadcasts its own routing information on those same networks. The interior routing protocols supported are as follows:

- Internet Gateway Routing Protocol (IGRP)
- Enhanced Internet Gateway Routing Protocol (Enhanced IGRP)
- Open Shortest Path First (OSPF)
- Routing Information Protocol (RIP)
- Intermediate System-to-Intermediate System (IS-IS)

Exterior Gateway Protocols

Exterior protocols are used to exchange routing information between networks that do not share a common administration. IP exterior gateway protocols require three sets of information before routing can begin:

- A list of neighbor (or peer) routers with which to exchange routing information
- A list of networks to advertise as directly reachable
- The autonomous system number of the local router

The supported exterior gateway protocols are as follows:

- Border Gateway Protocol (BGP)
- Exterior Gateway Protocol (EGP)

Router Discovery Protocols

Our routers also support two router discovery protocols, Gateway Discovery Protocol (GDP) and ICMP Router Discovery Protocol (IRDP), which allow hosts to locate routers.

GDP was developed by Cisco and is not an industry standard. Unsupported example GDP clients can be obtained upon request from Cisco. Our IRDP implementation fully conforms to the router discovery protocol outlined in RFC 1256.

Multiple Routing Protocols

You can configure multiple routing protocols in a single router to connect networks that use different routing protocols. You can, for example, run RIP on one subnetted network, IGRP on another subnetted network, and exchange routing information between them in a controlled fashion. The available routing protocols were not designed to interoperate with one another, so each protocol collects different types of information and reacts to topology changes in its own way. For example, RIP uses a hop-count metric and IGRP uses a five-element vector of metric information. In the case where routing information is being exchanged between different networks that use different routing protocols, there are many configuration options that allow you to filter the exchange of routing information.

Our routers can handle simultaneous operation of up to 30 dynamic IP routing processes. The combination of routing processes on a router can consist of the following protocols (with the limits noted):

- Up to 30 IGRP routing processes
- Up to 30 OSPF routing processes
- One RIP routing process
- One IS-IS process
- One BGP routing process
- Up to 30 EGP routing processes

IP Routing Protocols Task List

With any of the IP routing protocols, you need to create the routing process, associate networks with the routing process, and customize the routing protocol for your particular network.

You will need to perform some combination of the tasks in the following sections to configure IP routing protocols:

- Determine a Routing Process
- Configure IGRP
- Configure Enhanced IGRP
- Configure OSPF
- Configure RIP
- Configure IS-IS
- Configure BGP
- Configure EGP
- Configure GDP
- Configure IRDP
- Configure IP Multicast Routing
- Configure Routing Protocol-Independent Features
- Monitor and Maintain the IP Network

See the end of this chapter for IP routing protocol configuration examples.

Determine a Routing Process

Choosing a routing protocol is a complex task. When choosing a routing protocol, consider (at least) the following:

- Internetwork size and complexity
- Support for variable-length subnet masks (VLSM); IS-IS, static routes, and OSPF support VLSM.
- Internetwork traffic levels
- Security needs
- Reliability needs
- Internetwork delay characteristics
- Organizational policies
- Organizational acceptance of change

The following sections describe the configuration tasks associated with each supported routing protocol. This publication does not provide in-depth information on how to choose routing protocols; you must choose routing protocols that best suit your needs. For detailed information on the technology behind the major routing protocols, see the *Internetworking Technology Overview* manual or other internetworking publications.

Configure IGRP

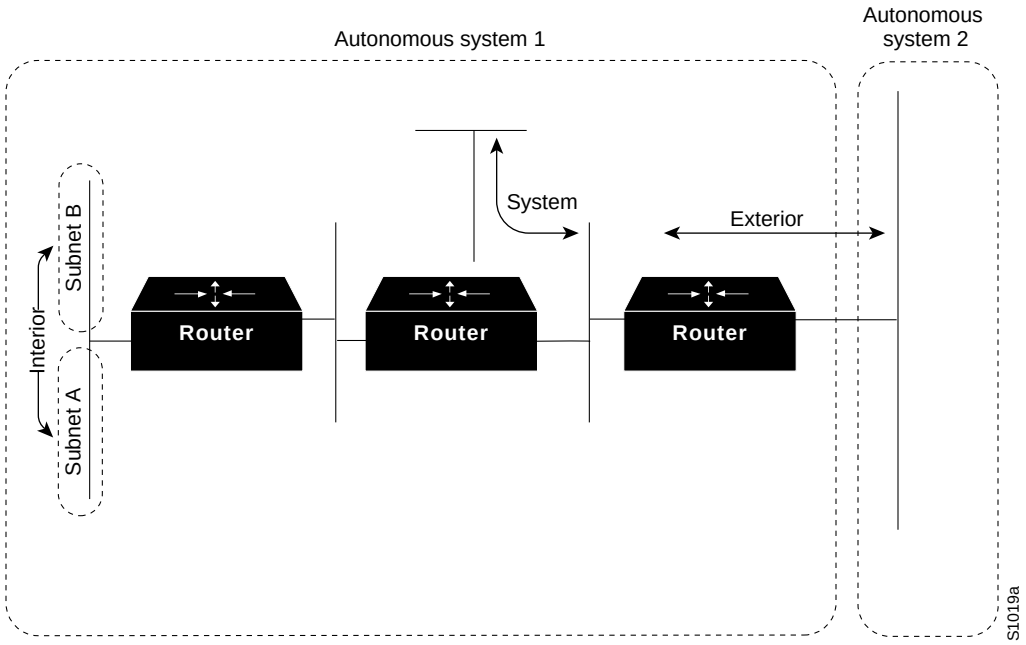
The Interior Gateway Routing Protocol (IGRP) is a dynamic distance-vector routing protocol designed by Cisco Systems in the mid-1980s for routing in an autonomous system that contains large, arbitrarily complex networks with diverse bandwidth and delay characteristics.

Cisco's IGRP Implementation

IGRP uses a combination of user-configurable metrics including internetwork delay, bandwidth, reliability, and load.

IGRP also advertises three types of routes: interior, system, and exterior, as shown in Figure 18-1. Interior routes are routes between subnets in the network attached to a router interface. If the network attached to a router is not subnetted, IGRP does not advertise interior routes.

Figure 18-1 Interior, System, and Exterior Routes



System routes are routes to networks within an autonomous system. The router derives system routes from directly connected network interfaces and system route information provided by other IGRP-speaking routers. System routes do not include subnet information.

Exterior routes are routes to networks outside the autonomous system that are considered when identifying a *gateway of last resort*. The router chooses a gateway of last resort from the list of exterior routes that IGRP provides. The router uses the gateway (router) of last resort if it does not have a better route for a packet and the destination is not a connected network. If the autonomous system has more than one connection to an external network, different routers can choose different exterior routers as the gateway of last resort.

IGRP Updates

By default, a router running IGRP sends an update broadcast every 90 seconds. It declares a route inaccessible if it does not receive an update from the first router in the route within three update periods (270 seconds). After seven update periods (630 seconds), the router removes the route from the routing table.

IGRP uses *flash update* and *poison reverse updates* to speed up the convergence of the routing algorithm. Flash update is the sending of an update sooner than the standard periodic update interval of notifying other routers of a metric change. Poison reverse updates are intended to defeat larger routing loops caused by increases in routing metrics. The poison reverse updates are sent to remove a route and place it in *holddown*, which keeps new routing information from being used for a certain period of time.

IGRP Configuration Task List

To configure IGRP, perform the tasks in the following sections. It is only mandatory to create the IGRP routing process; the other tasks described are optional.

- Create the IGRP Routing Process
- Allow Point-to-Point Updates for IGRP
- Define Unequal-Cost Load Balancing
- Control Traffic Distribution
- Adjust the IGRP Metric Weights
- Disable Holddown
- Enforce a Maximum Network Diameter
- Validate Source IP Addresses

Create the IGRP Routing Process

To create the IGRP routing process, perform the following required tasks:

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Enable an IGRP routing process, which places you in router configuration mode.	**router igrp** *process number*
Step 3	Associate networks with an IGRP routing process.	**network** *network-number*

IGRP sends updates to the interfaces in the specified networks. If an interface's network is not specified, it will not be advertised in any IGRP update.

It is not necessary to have a registered autonomous system number to use IGRP. If you do not have a registered number, you are free to create your own. We recommend that if you do have a registered number, you use it to identify the IGRP process.

Allow Point-to-Point Updates for IGRP

Because IGRP is normally a broadcast protocol, in order for IGRP routing updates to reach point-to-point or nonbroadcast networks, you must configure the router to permit this exchange of routing information.

To permit information exchange, perform the following task in router configuration mode:

Task	Command
Define a neighboring router with which to exchange point-to-point routing information.	**neighbor** *ip-address*

To control the set of interfaces that you want to exchange routing updates with, you can disable the sending of routing updates on specified interfaces by configuring the **passive-interface** command. See the discussion on filtering in the section in this chapter titled "Filter Routing Information."

Define Unequal-Cost Load Balancing

IGRP can simultaneously use an asymmetric set of paths for a given destination. This feature is known as *unequal-cost load balancing*. Unequal-cost load balancing allows traffic to be distributed among multiple (up to four) unequal-cost paths to provide greater overall throughput and reliability. Alternate path variance (that is, the difference in desirability between the primary and alternate paths) is used to determine the *feasibility* of a potential route. An alternate route is *feasible* if the next router in the path is *closer* to the destination (has a lower metric value) than the current router and if the metric for the entire alternate path is *within* the variance. Only paths that are feasible can be used for load balancing and included in the routing table. These conditions limit the number of cases in which load balancing can occur, but ensure that the dynamics of the network will remain stable.

The following general rules apply to IGRP unequal-cost load balancing:

- IGRP will accept up to four paths for a given destination network.
- The local best metric must be greater than the metric learned from the next router; that is, the next-hop router must be closer (have a smaller metric value) to the destination than the local best metric.
- The alternative path metric must be within the specified *variance* of the local best metric. The multiplier times the local best metric for the destination must be greater than or equal to the metric through the next router.

If these conditions are met, the route is deemed feasible and can be added to the routing table.

By default, the amount of variance is set to one (equal-cost load balancing). You can define how much worse an alternate path can be before that path is disallowed by performing the following task in router configuration mode:

Task	Command
Define the variance associated with a particular path.	**variance** *multiplier*

Note By using the variance feature, the router can balance traffic across all feasible paths and can immediately converge to a new path if one of the paths should fail.

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of configuring IGRP feasible successor.

Control Traffic Distribution

By default, if IGRP or Enhanced IGRP have multiple routes of unequal cost to the same destination, the router will distribute traffic among the different routes by giving each route a share of the traffic in inverse proportion to its metric. If you want to have faster convergence to alternate routes but you do not want to send traffic across inferior routes in the normal case, you might prefer to have no traffic flow along routes with higher metrics.

To control how traffic is distributed among multiple routes of unequal cost, perform the following task in router configuration mode:

Task	Command
Distribute traffic proportionately to the ratios of metrics, or by the minimum-cost route.	**traffic-share** {**balanced** | **min**}

Adjust the IGRP Metric Weights

You have the option of altering the default behavior of IGRP routing and metric computations. This allows, for example, tuning system behavior to allow for transmissions via satellite. Although IGRP metric defaults were carefully selected to provide excellent operation in most networks, you can adjust the IGRP metric. Adjusting IGRP metric weights can dramatically affect network performance, however, so ensure you make all metric adjustments carefully.

To adjust the IGRP metric weights, perform the following task in router configuration mode. Due to the complexity of this task, we recommend that you only perform it with guidance from an experienced system designer.

Task	Command
Adjust the IGRP metric.	**metric weights** *tos k1 k2 k3 k4 k5*

By default, the IGRP composite metric is a 24-bit quantity that is a sum of the segment delays and the lowest segment bandwidth (scaled and inverted) for a given route. For a network of homogeneous media, this metric reduces to a hop count. For a network of mixed media (FDDI, Ethernet, and serial lines running from 9600 bps to T1 rates), the route with the lowest metric reflects the most desirable path to a destination.

Disable Holddown

When a router learns that a network is at a greater distance than was previously known, or it learns the network is down, the route to that network is placed into holddown. During the holddown period, the route is advertised, but incoming advertisements about that network from any router other than the one that originally advertised the network's new metric will be ignored. This mechanism is often used to help avoid routing loops in the network, but has the effect of increasing the topology convergence time. To disable holddowns with IGRP, perform the following task in router configuration mode. All routers in an IGRP autonomous system must be consistent in their use of holddowns.

Task	Command
Disable the IGRP holddown period.	**no metric holddown**

Enforce a Maximum Network Diameter

The router enforces a maximum diameter to the IGRP network. Routes whose hop counts exceed this diameter will not be advertised. The default maximum diameter is 100 hops. The maximum diameter is 255 hops.

To configure the maximum diameter, perform the following task in router configuration mode:

Task	Command
Configure the maximum network diameter.	**metric maximum-hops** *hops*

Validate Source IP Addresses

To disable the default function that validates the source IP addresses of incoming routing updates, perform the following task in router configuration mode:

Task	Command
Disable the checking and validation of the source IP address of incoming routing updates.	**no validate-update-source**

Configure Enhanced IGRP

Enhanced IGRP is an enhanced version of the Interior Gateway Routing Protocol (IGRP) developed by Cisco Systems, Inc. Enhanced IGRP uses the same distance vector algorithm and distance information as IGRP. However, the convergence properties and the operating efficiency of Enhanced IGRP have improved significantly over IGRP.

The convergence technology is based on research conducted at SRI International and employs an algorithm referred to as the Diffusing Update Algorithm (DUAL). This algorithm guarantees loop-free operation at every instant throughout a route computation and allows all routers involved in a topology change to synchronize at the same time. Routers that are not affected by topology changes are not involved in recomputations. The convergence time with DUAL rivals that of any other existing routing protocol.

Cisco's Enhanced IGRP Implementation

IP Enhanced IGRP provides the following features:

- Automatic redistribution. IP IGRP routes can be automatically redistributed into Enhanced IGRP, and IP Enhanced IGRP routes can be automatically redistributed into IGRP. If desired, you can turn off redistribution. You can also completely turn off IP Enhanced IGRP and IP IGRP on the router or on individual interfaces.
- Increased network width. With IP RIP, the largest possible width of your network is 15 hops. When IP Enhanced IGRP is enabled, the largest possible width is 224 hops. Because the Enhanced IGRP metric is large enough to support thousands of hops, the only barrier to expanding the network is the transport layer hop counter. Cisco works around this problem by incrementing the transport control field only when an IP packet has traversed 15 routers and the next hop to the destination was learned via Enhanced IGRP. When a RIP route is being used as the next hop to the destination, the transport control field is incremented as usual.

Enhanced IGRP offers the following features:

- Fast convergence. The DUAL algorithm allows routing information to converge as quickly as any currently available routing protocol.
- Partial updates. Enhanced IGRP sends incremental updates when the state of a destination changes, instead of sending the entire contents of the routing table. This feature minimizes the bandwidth required for Enhanced IGRP packets.
- Less CPU usage than IGRP. This occurs because full update packets do not have to be processed each time they are received.
- Neighbor discovery mechanism. This is a simple hello mechanism used to learn about neighboring routers. It is protocol-independent.
- Variable-length subnet masks.
- Arbitrary route summarization.
- Scaling. Enhanced IGRP scales to large networks.

Enhanced IGRP has four basic components:

- Neighbor discovery/recovery
- Reliable transport protocol
- DUAL finite state machine
- Protocol-dependent modules

Neighbor discovery/recovery is the process that routers use to dynamically learn of other routers on their directly attached networks. Routers must also discover when their neighbors become unreachable or inoperative. Neighbor discovery/recovery is achieved with low overhead by periodically sending small hello packets. As long as hello packets are received, a router can determine that a neighbor is alive and functioning. Once this status is determined, the neighboring routers can exchange routing information.

The reliable transport protocol is responsible for guaranteed, ordered delivery of Enhanced IGRP packets to all neighbors. It supports intermixed transmission of multicast and unicast packets. Some Enhanced IGRP packets must be transmitted reliably and others need not be. For efficiency, reliability is provided only when necessary. For example, on a multiaccess network that has multicast capabilities, such as Ethernet, it is not necessary to send hellos reliably to all neighbors individually. Therefore, Enhanced IGRP sends a single multicast hello with an indication in the packet informing the receivers that the packet need not be acknowledged. Other types of packets, such as updates, require acknowledgment, and this is indicated in the packet. The reliable transport has a provision to send multicast packets quickly when there are unacknowledged packets pending. Doing so helps ensure that convergence time remains low in the presence of varying speed links.

The DUAL finite state machine embodies the decision process for all route computations. It tracks all routes advertised by all neighbors. DUAL uses the distance information, known as a metric, to select efficient, loop-free paths. DUAL selects routes to be inserted into a routing table based on feasible successors. A successor is a neighboring router used for packet forwarding that has a least-cost path to a destination that is guaranteed not to be part of a routing loop. When there are no feasible successors but there are neighbors advertising the destination, a recomputation must occur. This is the process whereby a new successor is determined. The amount of time it takes to recompute the route affects the convergence time. Even though the recomputation is not processor intensive, it is advantageous to avoid recomputation if it is not necessary. When a topology change occurs, DUAL will test for feasible successors. If there are feasible successors, it will use any it finds in order to avoid unnecessary recomputation.

The protocol-dependent modules are responsible for network layer protocol-specific tasks. An example is the IP Enhanced IGRP module, which is responsible for sending and receiving Enhanced IGRP packets that are encapsulated in IP. It is also responsible for parsing Enhanced IGRP packets and informing DUAL of the new information received. IP Enhanced IGRP asks DUAL to make routing decisions, but the results are stored in the IP routing table. Also, IP Enhanced IGRP is responsible for redistributing routes learned by other IP routing protocols.

Enhanced IGRP Configuration Task List

To configure IP Enhanced IGRP, complete the tasks in the following sections. At a minimum, you must enable IP Enhanced IGRP. The remaining tasks are optional.

- Enable IP Enhanced IGRP
- Transition from IGRP to Enhanced IGRP
- Configure IP Enhanced IGRP-Specific Parameters
- Configure Protocol-Independent Parameters

See the end of this chapter for configuraiton examples.

Enable IP Enhanced IGRP

To create an IP Enhanced IGRP routing process, perform the following tasks:

Task		Command
Step 1	Enable an IP Enhanced IGRP routing process in global configuration mode.	**router eigrp** *process-number*
Step 2	Associate networks with an IP Enhanced IGRP routing process in router configuration mode.	**network** *network-number*

IP Enhanced IGRP sends updates to the interfaces in the specified network(s). If you do not specify an interface's network, it will not be advertised in any IP Enhanced IGRP update.

Transition from IGRP to Enhanced IGRP

If you have routers on your network that are configured for IGRP and you want to make a transition to routing Enhanced IGRP, you need to designate transition routers that have both IGRP and Enhanced IGRP configured. In these cases, perform the tasks as noted in the previous section, "Enable IP Enhanced IGRP," and also read the section on configuring IGRP in this chapter. You must use the same autonomous system number in order for routes to be redistributed automatically.

Configure IP Enhanced IGRP-Specific Parameters

To configure IP Enhanced IGRP-specific parameters, perform one or more of the following tasks:

- Define Unequal-Cost Load Balancing
- Adjust the IP Enhanced IGRP Metric Weights
- Disable Route Summarization
- Configure Summary Aggregate Addresses

Define Unequal-Cost Load Balancing

IP Enhanced IGRP can simultaneously use an asymmetric set of paths for a given destination. This feature is known as *unequal-cost load balancing.* Unequal-cost load balancing allows traffic to be distributed among up to four unequal-cost paths to provide greater overall throughput and reliability. Alternate path variance (the difference in desirability between the primary and alternate paths) is used to determine the feasibility of a potential route. An alternate route is feasible if the next router in the path is closer to the destination (has a lower metric value) than the current router and if the metric for the entire alternate path is within the variance. Only paths that are feasible can be used for load balancing and included in the routing table. These conditions limit the number of cases in which load balancing can occur, but ensure that the dynamics of the network will remain stable.

The following general rules apply to IP Enhanced IGRP unequal-cost load balancing:

- IP Enhanced IGRP will accept up to four paths for a given destination network.
- The local best metric must be greater than the metric learned from the next router; that is, the next-hop router must be closer (have a smaller metric value) to the destination than the local best metric.
- The alternative path metric must be within the specified *variance* of the local best metric. The multiplier times the local best metric for the destination must be greater than or equal to the metric through the next router.

If these conditions are met, the route is deemed feasible and can be added to the routing table.

By default, the amount of variance is set to one (equal-cost load balancing). To change the variance to define how much worse an alternate path can be before that path is disallowed, perform the following task in router configuration mode:

Task	Command
Define the variance associated with a particular path.	**variance** *multiplier*

Note By using the variance feature, the router can balance traffic across all feasible paths and can immediately converge to a new path if one of the paths should fail.

Adjust the IP Enhanced IGRP Metric Weights

You can adjust the default behavior of IP Enhanced IGRP routing and metric computations. For example, this allows you to tune system behavior to allow for satellite transmission. Although IP Enhanced IGRP metric defaults have been carefully selected to provide excellent operation in most networks, you can adjust the IP Enhanced IGRP metric. Adjusting IP Enhanced IGRP metric weights can dramatically affect network performance, so be careful if you adjust them.

To adjust the IP Enhanced IGRP metric weights, perform the following task in router configuration mode:

Task	Command
Adjust the IP Enhanced IGRP metric.	**metric weights** *tos k1 k2 k3 k4 k5*

Note Because of the complexity of this task, it is not recommended unless it is done with guidance from an experienced network designer.

By default, the IP Enhanced IGRP composite metric is a 32-bit quantity that is a sum of the segment delays and the lowest segment bandwidth (scaled and inverted) for a given route. For a network of homogeneous media, this metric reduces to a hop count. For a network of mixed media (FDDI, Ethernet, and serial lines running from 9600 bps to T1 rates), the route with the lowest metric reflects the most desirable path to a destination.

Disable Route Summarization

You can configure IP Enhanced IGRP to perform automatic summarization of subnet routes into network-level routes. For example, you can configure subnet 131.108.1.0 to be advertised as 131.108.0.0 over interfaces that have subnets of 192.31.7.0 configured. Automatic summarization is performed when there are two or more **network** router configuration commands configured for the IP Enhanced IGRP process. By default, this feature is enabled.

To disable automatic summarization, perform the following task in router configuration mode:

Task	Command
Disable automatic summarization.	**no auto-summary**

Route summarization works in conjunction with the **ip summary-address eigrp** interface configuration command, in which additional summarization can be performed. If auto-summary is in effect, there usually is no need to configure network level summaries using the **ip summary-address eigrp** command.

Configure Summary Aggregate Addresses

You can configure a summary aggregate address for a specified interface. If there are any more-specific routes in the routing table, IP Enhanced IGRP will advertise the summary address out the interface with a metric equal to the minimum of all more-specific routes.

To configure a summary aggregate address, perform the following task in interface configuration mode:

Task	Command
Configure a summary aggregate address.	**ip summary-address eigrp** *autonomous-system-number address mask*

Configure Protocol-Independent Parameters

To configure protocol-independent parameters, perform one or more of the following tasks:

- Redistribute Routing Information
- Set Metrics for Redistributed Routes
- Filter Routing Information
- Adjust the Interval between Hello Packets and the Hold Time
- Disable Split Horizon

Redistribute Routing Information

In addition to running multiple routing protocols simultaneously, the router can redistribute information from one routing protocol to another. For example, you can instruct the router to readvertise IP Enhanced IGRP-derived routes using the RIP protocol, or to readvertise static routes using the IP Enhanced IGRP protocol. This capability applies to all the IP-based routing protocols.

You may also conditionally control the redistribution of routes between routing domains by defining a method known as *route maps* between the two domains.

To redistribute routes from one protocol into another, perform the following task in router configuration mode:

Task	Command
Redistribute routes from one routing protocol into another.	**redistribute** *protocol autonomous-system-number* [**route-map** *map-tag*]

To define route maps, perform the following task in global configuration mode:

Task	Command
Define any route maps needed to control redistribution.	**route-map** *map-tag* {**permit** | **deny**} *sequence-number*

By default, the redistribution of default information between IP Enhanced IGRP processes is enabled. To disable the redistribution, perform the following task in router configuration mode:

Task	Command
Disable the redistribution of default information between IP Enhanced IGRP processes.	**no default-information allowed** {**in** | **out**}

Set Metrics for Redistributed Routes

The metrics of one routing protocol do not necessarily translate into the metrics of another. For example, the RIP metric is a hop count and the IP Enhanced IGRP metric is a combination of five quantities. In such situations, an artificial metric is assigned to the redistributed route. Because of this unavoidable tampering with dynamic information, carelessly exchanging routing information between different routing protocols can create routing loops, which can seriously degrade network operation.

To set metrics for redistributed routes, perform the first task when redistributing from IP Enhanced IGRP, and perform the second task when redistributing into IP Enhanced IGRP. Each task is done in router configuration mode.

Task	Command
Cause the current routing protocol to use the same metric value for all redistributed routes.	**default-metric** *number*
Cause the IP Enhanced IGRP routing protocol to use the same metric value for all non-IGRP redistributed routes.	**default-metric** *bandwidth delay reliability loading mtu*

Filter Routing Information

You can filter routing protocol information by performing the following tasks:

- Suppress the sending of routing updates on a particular router interface. Doing so prevents other systems on an interface from learning about routes dynamically.
- Suppress networks from being advertised in routing updates. Doing so prevents other routers from learning a particular router's interpretation of one or more routes.
- Suppress a routing protocol from both sending and receiving updates on a particular interface. You usually perform this task when a wildcard command has been used to configure the routing protocol for more router interfaces than is desirable.
- Suppress networks listed in updates from being accepted and acted upon by a routing process. Doing so keeps a router from using certain routes.
- Filter on the source of routing information. You perform this task to prioritize routing information from different sources, because the accuracy of the routing information can vary.
- Apply an offset to routing metrics. Doing so provides a local mechanism for increasing the value of routing metrics.

Use the information in the following sections to perform these tasks.

Prevent Routing Updates through an Interface

To prevent other routers on a local network from learning about routes dynamically, you can keep routing update messages from being sent through a router interface. This feature applies to all IP-based routing protocols except BGP and EGP.

To prevent routing updates through a specified interface, perform the following task in router configuration mode:

Task	Command
Suppress the sending of routing updates through a router interface.	**passive-interface** *type unit*

Control the Advertising of Routes in Routing Updates

To control which routers learn about routes, you can control the advertising of routesin routing updates. To do this, perform the following task in router configuration mode:

Task	Command
Control the advertising of routes in routing updates.	**distribute-list** *access-list-number* **out** [*interface-name* | *routing-process* | *autonomous-system-number*]

Control the Processing of Routing Updates

To control the processing of routes listed in incoming updates, perform the following task in router configuration mode:

Task	Command
Control which incoming route updates are processes.	**distribute-list** *access-list-number* **in** [*interface-name*]

Apply Offsets to Routing Metrics

To provide a local mechanism for increasing the value of routing metrics, you can apply an offset to routing metrics. To do so, perform the following task in router configuration mode:

Task	Command
Apply an offset to routing metrics.	**offset-list** {**in** | **out**} *offset* [*access-list-number*]

Filter Sources of Routing Information

An administrative distance is a rating of the trustworthiness of a routing information source, such as an individual router or a group of routers. In a large network, some routing protocols and some routers can be more reliable than others as sources of routing information. Also, when multiple routing processes are running in the same router for IP, the same route may be advertised by more than one routing process. Specifying administrative distance values enables the router to discriminate between sources of routing information. The router always picks the route whose routing protocol has the lowest administrative distance.

There are no general guidelines for assigning administrative distances, because each network has its own requirements. You must determine a reasonable matrix of administrative distances for the network as a whole. Table 18-1 shows the default administrative distance for various routing information sources.

Table 18-1 Default Administrative Distances

Route Source	Default Distance
Connected interface	0
Static route	1
Enhanced IGRP summary route	5
External BGP	20
Internal Enhanced IGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
External Enhanced IGRP	170
Internal BGP	200
Unknown	255

For example, consider a router using IP Enhanced IGRP and RIP. Suppose you trust the IP Enhanced IGRP-derived routing information more than the RIP-derived routing information. Because the default IP Enhanced IGRP administrative distance is lower than that for RIP, the router uses the IP Enhanced IGRP-derived information and ignores the RIP-derived information. However, if you lose the source of the IP Enhanced IGRP-derived information (for example, because of a power shutdown), the router uses the RIP-derived information until the IP Enhanced IGRP-derived information reappears.

Note You can also use administrative distance to rate the routing information from routers running the same routing protocol. This application is generally discouraged if you are unfamiliar with this particular use of administrative distance, since it can result in inconsistent routing information, including forwarding loops.

To filter sources of routing information, perform the following tasks in router configuration mode:

Task	Command
Filter on routing information sources.	**distance eigrp** *internal-distance external-distance*

Adjust the Interval between Hello Packets and the Hold Time

You can adjust the interval between hello packets and the hold time.

Routers periodically send hello packets to each other to dynamically learn of other routers on their directly attached networks. The routers use this information to discover who their neighbors are and to learn when their neighbors become unreachable or inoperative. By default, hello packets are sent every 5 seconds.

You can configure the hold time on a specified interface for the IP Enhanced IGRP routing process designated by the autonomous system number. The hold time is advertised in hello packets and indicates to neighbors the length of time they should consider the sender valid. The default hold time is three times the hello interval, or 15 seconds.

To change the interval between hello packets, perform the following task in interface configuration mode:

Task	Command
Configure the hello interval for an IP Enhanced IGRP routing process.	**ip hello-interval eigrp** *autonomous-system-number seconds*

On very congested and large networks, 15 seconds may not be sufficient time for all routers to receive hello packets from their neighbors. In this case, you may want to increase the hold time.

To change the hold time, perform the following task in interface configuration mode:

Task	Command
Configure the hold time for an IP Enhanced IGRP routing process.	**ip hold-time eigrp** *autonomous-system-number seconds*

Note Do not adjust the hold time without advising technical support.

Disable Split Horizon

Split horizon controls the sending of IP Enhanced IGRP update and query packets. When split horizon is enabled on an interface, these packets are not sent for destinations for which this interface is the next hop. This reduces the possibility of routing loops.

By default, split horizon is enabled on all interfaces.

Split horizon blocks information about routes from being advertised by a router out any interface from which that information originated. This behavior usually optimizes communications among multiple routers, particularly when links are broken. However, with nonbroadcast networks, such as Frame Relay and SMDS, situations can arise for which this behavior is less than ideal. For these situations, you may wish to disable split horizon.

To disable split horizon, perform the following task in interface configuration mode:

Task	Command
Disable split horizon.	**no ip split-horizon eigrp** *autonomous-system-number*

Configure OSPF

Open Shortest Path First (OSPF) is an IGP developed by the OSPF working group of the Internet Engineering Task Force (IETF). Designed expressly for IP networks, OSPF supports IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication and uses IP multicast when sending/receiving packets.

We support RFC 1253, Open Shortest Path First (OSPF) MIB, August 1991. The OSPF MIB defines an IP routing protocol that provides management information related to OSPF and is supported by Cisco routers.

Cisco's OSPF Implementation

Cisco's implementation conforms to the OSPF Version 2 specifications detailed in the Internet RFC 1583. The list that follows outlines key features supported in Cisco's OSPF implementation:

- Stub areas—Definition of stub areas is supported.
- Route redistribution—Routes learned via any IP routing protocol can be redistributed into any other IP routing protocol. At the intradomain level, this means that OSPF can import routes learned via IGRP, RIP, and IS-IS. OSPF routes can also be exported into IGRP, RIP, and IS-IS. At the interdomain level, OSPF can import routes learned via EGP and BGP. OSPF routes can be exported into EGP and BGP.
- Authentication—Authentication among neighboring routers within an area is supported.
- Routing interface parameters—Configurable parameters supported include interface output cost, retransmission interval, interface transmit delay, router priority, router "dead" and hello intervals, and authentication key.
- Virtual links—Virtual links are supported.

Note In order to take advantage of the OSPF stub area support, *default routing* must be used in the stub area.

OSPF Configuration Task List

OSPF typically requires coordination among many internal routers, *area border routers* (routers connected to multiple areas), and autonomous system boundary routers. At a minimum, OSPF-based routers can be configured with all default parameter values, no authentication, and interfaces assigned to areas. If you intend to customize your environment, you must ensure coordinated configurations of all routers.

To configure OSPF, complete the tasks in the following sections. Enabling OSPF is mandatory; the other tasks are optional but might be required for your application.

- Enable OSPF
- Configure OSPF Interface Parameters
- Configure OSPF over Different Physical Networks
- Configure OSPF Area Parameters
- Configure Route Summarization between OSPF Areas
- Configure Route Summarization when Redistributing Routes into OSPF
- Create Virtual Links
- Generate a Default Route
- Configure Lookup of DNS Names
- Force the Router ID Choice with a Loopback Interface
- Disable Default OSPF Metric Calculation Based on Bandwidth
- Configure OSPF on Simplex Ethernet Interfaces
- Configure Route Calculation Timers

In addition, you can specify route redistribution; see the task "Redistribute Routing Information" later in this chapter for information on how to configure route redistribution.

Enable OSPF

As with other routing protocols, enabling OSPF requires that you create an OSPF routing process, specify the range of IP addresses to be associated with the routing process, and assign area IDs to be associated with that range of IP addresses. Perform the following tasks, starting in global configuration mode:

Task		Command
Step 1	Enable OSPF routing, which places you in router configuration mode.	**router ospf** *process-id*
Step 2	Define an interface on which OSPF runs and define the area ID for that interface.	**network** *address wildcard-mask* **area** *area-id*

Configure OSPF Interface Parameters

Our OSPF implementation allows you to alter certain interface-specific OSPF parameters, as needed.You are not required to alter any of these parameters, but some interface parameters must be consistent across all routers in an attached network. Therefore, be sure that if you do configure any of these parameters, the configurations for all routers on your network have compatible values.

In interface configuration mode, specify any of the following interface parameters as needed for your network:

Task	Command
Explicitly specify the cost of sending a packet on an OSPF interface.	**ip ospf cost** *cost*

Task	Command
Specify the number of seconds between link state advertisement retransmissions for adjacencies belonging to an OSPF interface.	**ip ospf retransmit-interval** *seconds*
Set the estimated number of seconds it takes to transmit a link state update packet on an OSPF interface.	**ip ospf transmit-delay** *seconds*
Set router priority to help determine the OSPF designated router for a network.	**ip ospf priority** *number*
Specify the length of time, in seconds, between the hello packets that a router sends on an OSPF interface.	**ip ospf hello-interval** *seconds*
Set the number of seconds that a router's hello packets must not have been seen before its neighbors declare the OSPF router down.	**ip ospf dead-interval** *seconds*
Assign a specific password to be used by neighboring OSPF routers on a network segment that is using OSPF's simple password authentication.	**ip ospf authentication-key** *password*

Configure OSPF over Different Physical Networks

OSPF classifies different media into three types of networks by default:

- Broadcast networks (Ethernet, Token Ring, FDDI)
- Nonbroadcast, multiaccess networks (SMDS, Frame Relay, X.25)
- Point-to-point networks (HDLC, PPP)

You can configure your network as either a broadcast or a nonbroadcast multiaccess network.

X.25 and Frame Relay provide an optional broadcast capability that can be configured in the map to allow OSPF to run as a broadcast network. See the **x25 map** and **frame-relay map** command descriptions in the *Router Products Command Reference* publication for more detail.

Configure Your OSPF Network Type

You have the choice of configuring your OSPF network type to either broadcast or nonbroadcast multiaccess, regardless of the default media type. Using this feature, you can configure broadcast networks as nonbroadcast multiaccess networks when, for example, you have routers in your network that do not support multicast addressing. You also can configure nonbroadcast multiaccess networks, such as X.25, Frame Relay, and SMDS, as broadcast networks. This feature saves you from having to configure neighbors, as described in the section "Configure OSPF for Nonbroadcast Networks."

Configuring nonbroadcast multiaccess networks as either broadcast or nonbroadcast assumes that there are virtual circuits from every router to every router or fully-meshed network. This is not true for some cases, for example, due to cost constraints or when you have only a partially-meshed network. In these cases, you can configure the OSPF network type as a point-to-multipoint network. Routing between two routers that are not directly connected will go through the router that has virtual circuits to both routers. Note that you do not need to configure neighbors when using this feature.

To configure your OSPF network type, perform the following task in interface configuration mode:

Task	Command
Configure the OSPF network type for a specified interface.	**ip ospf network** {**broadcast** | **non-broadcast** | **point-to-multipoint**}

Configure OSPF for Nonbroadcast Networks

Because there might be many routers attached to an OSPF network, a *designated router* is selected for the network. It is necessary to use special configuration parameters in the designated router selection if broadcast capability is not configured.

These parameters need only be configured in those routers that are themselves eligible to become the designated router or backup designated router (in other words, routers with a nonzero router priority value).

To configure routers that interconnect to nonbroadcast networks, perform the following task in router configuration mode

Task	Command
Configure routers interconnecting to nonbroadcast networks.	**neighbor** *ip-address* [**priority** *number*] [**poll-interval** *seconds*]

You can specify the following neighbor parameters, as required:

- Priority for a neighboring router
- Nonbroadcast poll interval
- Interface through which the neighbor is reachabl

Configure OSPF Area Parameters

Our OSPF software allows you to configure several area parameters. These area parameters, shown in the following table, include authentication, defining stub areas, and assigning specific costs to the default summary route. *Authentication* allows password-based protection against unauthorized access to an area. *Stub areas* are areas into which information on external routes is not sent. Instead, there is a default external route generated by the area border router into the stub area for destinations outside the autonomous system.

In router configuration mode, specify any of the following area parameters as needed for your network:

Task	Command
Enable authentication for an OSPF area.	**area** *area-id* **authentication**
Define an area to be a stub area.	**area** *area-id* **stub**
Assign a specific cost to the default summary route used for the stub area.	**area** *area-id* **default-cost** *cost*

Configure Route Summarization between OSPF Areas

Route summarization is the consolidation of advertised addresses. This feature causes a single summary route to be advertised to other areas by an area border router. In OSPF, an area border router will advertise networks in one area into another area. If the network numbers in an area are

assigned in a way such that they are contiguous, you can configure the area border router to advertise a summary route that covers all the individual networks within the area that fall into the specified range.

To specify an address range, perform the following task in router configuration mode:

Task	Command
Specify an address range for which a single route will be advertised.	**area** *area-id* **range** *address mask*

Configure Route Summarization when Redistributing Routes into OSPF

When redistributing routes from other protocols into OSPF (as described in the section "Configure Routing Protocol-Independent Features" later in this chapter, each route is advertised individually in an external link state advertisement (LSA). However, you can configure the router to advertise a single route for all the redistributed routes that are covered by a specified network address and mask. Doing so helps decrease the size of the OSPF link state database.

To have the router advertise one summary route for all redistributed routes covered by a network address and mask, perform the following task in router configuration mode:

Task	Command
Specifiy an address and mask that covers redistributed routes, so only one summary route is advertised.	**summary-address** *address mask*

Create Virtual Links

In OSPF, all areas must be connected to a backbone area. If there is a break in backbone continuity, or the backbone is purposefully partitioned, you can establish a *virtual link*. The two end points of a virtual link are Area Border Routers. The virtual link must be configured in both routers. The configuration information in each router consists of the other virtual endpoint (the other Area Border Router), and the nonbackbone area that the two routers have in common (called the *transit area*). Note that virtual links cannot be configured through stub areas.

To establish a virtual link, perform the following task in router configuration mode:

Task	Command
Establish a virtual link.	**area** *area-id* **virtual-link** *router-id* [**hello-interval** *seconds*] [**retransmit-interval** *seconds*] [**transmit-delay** *seconds*] [**dead-interval** *seconds*] [**authentication-key** *password*]

To display information about virtual links, use the **show ip ospf virtual-links** EXEC command. To display the router ID of an OSPF router, use the **show ip ospf** EXEC command.

Generate a Default Route

You can force an autonomous system boundary router to generate a default route into an OSPF routing domain. Whenever you specifically configure redistribution of routes into an OSPF routing domain, the router automatically becomes an autonomous system boundary router. However, an autonomous system boundary router does not, by default, generate a *default route* into the OSPF routing domain.

To force the autonomous system boundary router to generate a default route, perform the following task in router configuration mode:

Task	Command
Force the autonomous system boundary router to generate a default route into the OSPF routing domain.	**default-information originate** [**always**] [**metric** *metric-value*] [**metric-type** *type-value*] [**route-map** *map-name*]

See also the discussion of redistribution of routes in the "Configure Routing Protocol-Independent Features" section later in this chapter.

Configure Lookup of DNS Names

You can configure OSPF to look up Domain Name System (DNS) names for use in all OSPF **show** command displays. This feature makes it easier to identify a router, because it is displayed by name rather than by its router ID or neighbor ID.

To configure DNS name lookup, perform the following task in global configuration mode:

Task	Command
Configure DNS name lookup.	**ip ospf-name-lookup**

Force the Router ID Choice with a Loopback Interface

OSPF uses the largest IP address configured on the router's interfaces as its router ID. If the interface associated with this IP address is ever brought down, or if the address is removed, the OSPF process must recalculate a new router ID and resend all of its routing information out its interfaces.

If a loopback interface is configured with an IP address, the router will use this IP address as its router ID, even if other interfaces have larger IP addresses. Since loopback interfaces never go down, greater stability in the routing table is achieved.

OSPF automatically prefers a loopback interface over any other kind, and it chooses the first loopback interface found. If no loopback interfaces are present, the highest IP address in the router is chosen. You cannot tell OSPF to use any particular interface.

To configure an IP address on a loopback interface, perform the following tasks, starting in global configuration mode:

Task		Command
Step 1	Create a loopback interface, which places you in interface configuration mode.	**interface loopback 0**[1]
Step 2	Assign an IP address to this interface.	**ip address** *address mask*

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

Disable Default OSPF Metric Calculation Based on Bandwidth

In Cisco IOS Release 10.2 and earlier, OSPF assigned default OSPF metrics to interfaces regardless of the interface bandwidth. It gave both 64K and T1 links the same metric (1562), and thus required an explicit **ip ospf cost** command in order to take advantage of the faster link.

In Cisco IOS Release 10.3, by default, OSPF calculates the OSPF metric for an interface according to the bandwidth of the interface. For example, a 64K link gets a metric of 1562, while a T1 link gets a metric of 64. To disable this feature, perform the following task in router configuration mode:

Task	Command
Disable default OSPF metric calculations based on interface bandwidth, resulting in a fixed default metric assignment.	**no ospf auto-cost-determination**

Configure OSPF on Simplex Ethernet Interfaces

Because simplex interfaces between two routers on an Ethernet represent only one network segment, for OSPF you have to configure the transmitting interface to be a passive interface. This prevents OSPF from sending hello packets for the transmitting interface. Both routers are able to see each other via the hello packet generated for the receiving interface.

To configure OSPF on simplex Ethernet interfaces, perform the following task in router configuration mode:

Task	Command
Suppress the sending of hello packets through the specified interface.	**passive-interface** *interface*

Configure Route Calculation Timers

You can configure the delay time between when OSPF receives a topology change and when it starts a Shortest Path First (SPF) calculation. You can also configure the hold time between two consecutive SPF calculations. To do this, perform the following task in router configuration mode:

Task	Command
Configure route calculation timers.	**timers spf** *spf-delay spf-holdtime*

Configure RIP

The Routing Information Protocol (RIP) is a relatively old but still commonly used IGP created for use in small, homogeneous networks. It is a classical distance-vector routing protocol.

RIP uses broadcast User Datagram Protocol (UDP) data packets to exchange routing information. Each router sends routing information updates every 30 seconds; this process is termed *advertising.* If a router does not receive an update from another router for 180 seconds or more, it marks the routes served by the nonupdating router as being unusable. If there is still no update after 240 seconds, the router removes all routing table entries for the nonupdating router.

The measure, or metric, that RIP uses to rate the value of different routes is the *hop count.* The hop count is the number of routers that can be traversed in a route. A directly connected network has a metric of zero; an unreachable network has a metric of 16. This small range of metrics makes RIP unsuitable as a routing protocol for large networks. If the router has a default network path, RIP advertises a route that links the router to the pseudonetwork 0.0.0.0. The network 0.0.0.0 does not exist; RIP treats 0.0.0.0 as a network to implement the default routing feature. Our routers will advertise the default network if a default was learned by RIP, or if the router has a gateway of last resort and RIP is configured with a default metric.

RIP sends updates to the interfaces in the specified networks. If an interface's network is not specified, it will not be advertised in any RIP update.

For information about filtering RIP information, see the "Filter Routing Information" section later in this chapter. RIP is documented in RFC 1058.

To configure RIP, perform the following tasks, starting in global configuration mode:

Task		Command
Step 1	Enable a RIP routing process, which places you in router configuration mode.	**router rip**
Step 2	Associate a network with a RIP routing process.	**network** *network-number*

Running IGRP and RIP Concurrently

It is possible to run IGRP and RIP concurrently. The IGRP information will override the RIP information by default because of IGRP's administrative distance.

However, running IGRP and RIP concurrently does not work well when the network topology changes. Because IGRP and RIP have different update timers and because they require different amounts of time to propagate routing updates, one part of the network will end up believing IGRP routes and another part will end up believing RIP routes. This will result in routing loops. Even though these loops do not exist for very long, the time to live (TTL) will quickly reach zero, and ICMP will send a "TTL exceeded" message. This message will cause most applications to stop attempting network connections.

Validate Source IP Addresses

To disable the default function that validates the source IP addresses of incoming routing updates, perform the following task in router configuration mode:

Task	Command
Disable the checking and validation of the source IP address of incoming routing updates.	**no validate-update-source**

Allow Point-to-Point Updates for RIP

Because RIP is normally a broadcast protocol, in order for RIP routing updates to reach point-to-point or nonbroadcast networks, you must configure the router to permit this exchange of routing information.

You configure the router to permit this exchange of routing information by performing the following task in router configuration mode:

Task	Command
Define a neighboring router with which to exchange point-to-point routing information.	**neighbor** *ip-address*

To control the set of interfaces that you want to exchange routing updates with, you can disable the sending of routing updates on specified interfaces by configuring the **passive-interface** command. See the discussion on filtering in the section in this chapter titled "Filter Routing Information."

Configure IS-IS

IS-IS, which stands for Intermediate System-to-Intermediate System, is an International Organization for Standardization (ISO) dynamic routing specification. IS-IS is described in ISO 10589. Cisco's implementation of IS-IS allows you to configure IS-IS as an IP routing protocol on your router.

IS-IS Configuration Task List

To configure IS-IS, complete the tasks in the following sections. Only enabling IS-IS is required; the remainder of the tasks are optional although you might be required to perform them depending upon your specific application.

- Enable IS-IS
- Configure IS-IS Interface Parameters
- Configure Miscellaneous IS-IS Parameters

In addition, you can filter routing information (see the task "Filter Routing Information" later in this chapter for information on how to do this), and specify route redistribution (see the task "Redistribute Routing Information" later in this chapter for information on how to do this).

Enable IS-IS

As with other routing protocols, enabling IS-IS requires that you create an IS-IS routing process and assign it to specific networks. You can specify *only one* IS-IS process per router. Only one IS-IS process is allowed whether you run it in integrated mode, ISO CLNS only, or IP only.

Network Entity Titles (NETs) define the area addresses for the IS-IS area. Multiple NETs per router are allowed, up to a maximum of three. Refer to the "Configuring ISO CLNS" chapter for a more detailed discussion of NETs.

Perform the following tasks to enable IS-IS on the router:

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Enable IS-IS routing and specify an IS-IS process for IP, which places you in router configuration mode.	**router isis** [*tag*]
Step 3	Configure NETs for the routing process; you can specify a name for a NET as well as an address.	**net** *network-entity-title*
Step 4	Enter interface configuration mode.	See Table 2-1.
Step 5	Specify the interfaces that should be actively routing IS-IS.	**ip router isis** [*tag*]

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of configuring IS-IS as an IP routing protocol.

Configure IS-IS Interface Parameters

Our IS-IS implementation allows you to alter certain interface-specific IS-IS parameters. You can do the following:

- Configure IS-IS link state metrics
- Set the advertised hello interval
- Set the advertised CSNP interval
- Set the retransmission interval
- Specify designated router election
- Specify the interface circuit type
- Assign a password for an interface

You are not required to alter any of these parameters, but some interface parameters must be consistent across all routers in an attached network. Therefore, be sure that if you do configure any of these parameters, the configurations for all routers on the network have compatible values.

Configure IS-IS Link-State Metrics

You can configure a cost for a specified interface. The only metric that is supported by the router and that you can configure is the *default-metric*, which you can configure for Level 1 and/or Level 2 routing. The other metrics currently are not supported.

To configure the metric for the specified interface, perform the following task in interface configuration mode:

Task	Command
Configure the metric (or cost) for the specified interface.	**isis metric** *default-metric* [*delay-metric* [*expense-metric* [*error-metric*]]] {**level-1** | **level-2**}

Set the Advertised Hello Interval

You can specify the length of time, in seconds, between hello packets that the router sends on the interface.

To specify the length of time between hello packets for the specified interface, perform the following task in interface configuration mode:

Task	Command
Specify the length of time, in seconds, between hello packets the router sends on the specified interface.	**isis hello-interval** *seconds* {**level-1** | **level-2**}

The hello interval can be configured independently for Level 1 and Level 2, except on serial point-to-point interfaces. (Because there is only a single type of hello packet sent on serial links, it is independent of Level 1 or Level 2.) Specify an optional level for X.25, SMDS, and Frame Relay multiaccess networks.

Set the Advertised CSNP Interval

Complete Sequence Number PDUs (CSNPs) are sent by the designated router to maintain database synchronization. You can configure the IS-IS CSNP interval for the interface.

To configure the CSNP interval for the specified interface, perform the following task in interface configuration mode:

Task	Command
Configure the IS-IS CSNP interval for the specified interface.	**isis csnp-interval** *seconds* {**level-1** | **level-2**}

This feature does not apply to serial point-to-point interfaces. It applies to WAN connections if the WAN is viewed as a multiaccess meshed network.

Set the Retransmission Interval

You can configure the number of seconds between retransmission of IS-IS link state PDUs (LSPs) for point-to-point links.

To set the retransmission level, perform the following task in interface configuration mode:

Task	Command
Configure the number of seconds between retransmission of IS-IS LSPs for point-to-point links.	**isis retransmit-interval** *seconds*

The value you specify should be an integer greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links.

Specify Designated Router Election

You can configure the priority to use for designated router election. Priorities can be configured for Level 1 and Level 2 individually.

To specify the designated router election, perform the following task in interface configuration mode:

Task	Command
Configure the priority to use for designated router election.	**isis priority** *value* {**level-1** | **level-2**}

Specify the Interface Circuit Type

You can specify adjacency levels on a specified interface. This parameter is also referred to as the interface circuit type.

To specify the interface circuit type, perform the following task in interface configuration mode:

Task	Command
Configure the type of adjacency desired for neighbors on the specified interface (the interface circuit type).	**isis circuit-type** {**level-1** | **level-1-2** | **level-2-only}**

Assign a Password for an Interface

You can assign different passwords for different routing levels. Specifying Level 1 or Level 2 configures the password for only Level 1 or Level 2 routing, respectively. If you do not specify a level, the default is Level 1. By default, authentication is disabled.

To configure a password for the specified level, perform the following task in interface configuration mode:

Task	Command
Configure the authentication password for a specified interface.	**isis password** *password* {**level-1** | **level-2**}

Configure Miscellaneous IS-IS Parameters

You can configure the following miscellaneous, optional IS-IS parameters:

- Generate a default route
- Specify router level support
- Configure IS-IS authentication passwords
- Summarize address ranges

Generate a Default Route

You can force a default route into an IS-IS routing domain. Whenever you specifically configure redistribution of routes into an IS-IS routing domain, the router does not, by default, generate a *default route* into the IS-IS routing domain. The following feature allows you to force the boundary router do this.

To generate a default route, perform the following task in router configuration mode:

Task	Command
Force a default route into the IS-IS routing domain.	**default-information originate** [**metric** *metric-value*] [**metric-type** *type-value*] {**level-1** | **level-1-2** | **level-2**} [**route-map** *map-name*]

See also the discussion of redistribution of routes in the "Configure Routing Protocol-Independent Features" section later in this chapter.

Specify Router-Level Support

You can configure the router to act as a Level 1 (intra-area) router, as both a Level 1 router and a Level 2 (interarea) router, or as an interarea router only.

To specify router level support, perform the following task in router configuration mode:

Task	Command
Configure the level at which the router should operate.	**is-type** {**level-1** | **level-1-2** | **level-2-only**}

Configure IS-IS Authentication Passwords

You can assign passwords to areas and domains.

The area authentication password is inserted in Level 1 (station router level) LSPs, CSNPs, and Partial Sequence Number PDUs (PSNPs). The routing domain authentication password is inserted in Level 2 (the area router level) LSP, CSNP, and PSNPs.

To configure either area or domain authentication passwords, perform the following tasks in router configuration mode:

Task	Command
Configure the area authentication password.	**area-password** *password*
Configure the routing domain authentication password.	**domain-password** *password*

Summarize Address Ranges

You can create aggregate addresses that are represented in the routing table by a summary address. This process is called route summarization. One summary address can include multiple groups of addresses for a given level. Routes learned from other routing protocols also can be summarized. The metric used to advertise the summary is the smallest metric of all the more specific routes.

To create a summary of addresses for a given level, perform the following task in router configuration mode:

Task	Command
Create a summary of addresses for a given level.	**summary-address** *address mask* {**level-1** | **level-1-2** | **level-2**}

Configure BGP

The Border Gateway Protocol (BGP), as defined in RFCs 1163 and 1267, allows you to set up an interdomain routing system that automatically guarantees the loop-free exchange of routing information between autonomous systems.

Cisco's BGP Implementation

In BGP, each route consists of a network number, a list of autonomous systems that information has passed through (called the *autonomous system path*), and a list of other *path attributes*. We support BGP Versions 2, 3, and 4, as defined in RFCs 1163, 1267, and 1654, respectively.

The primary function of a BGP system is to exchange network reachability information with other BGP systems, including information about the list of autonomous system paths. This information can be used to construct a graph of autonomous system connectivity from which routing loops can be pruned and with which autonomous system-level policy decisions can be enforced.

You can configure the value for the multiple exit discriminator (MULTI_EXIT_DISC, or MED) metric attribute using route maps. (The name of this metric for BGP Versions 2 and 3 is INTER_AS.) When an update is sent to an IBGP peer, the MED will be passed along without any change. This will enable all the peers in the same autonomous system to make a consistent path selection.

A third-party next-hop router address is used in the NEXT_HOP attribute, regardless of the autonomous system of that third-party router. The router automatically calculates the value for this attribute.

Transitive, optional path attributes are passed along to other BGP-speaking routers. The current BGP implementation does not generate such attributes.

BGP Version 4 (BGP4) supports classless interdomain routing (CIDR), which lets you reduce the size of your routing tables by creating aggregate routes, resulting in *supernets*. CIDR eliminates the concept of network classes within BGP and supports the advertising of IP prefixes. CIDR routes can be carried by OSPF and ISIS-IP.

See the "Using Route Maps with BGP" section for examples of how to use route maps to redistribute BGP4 routes.

How BGP Selects Paths

The BGP process selects a single autonomous system path to use and to pass along to other BGP-speaking routers. Cisco's BGP implementation has a reasonable set of factory defaults that can be overridden by administrative weights. The algorithm for path selection is as follows:

- If the next hop is inaccessible, do not consider it.
- Consider larger BGP administrative weights first.
- If the routers have the same weight, consider the route with higher local preference.
- If the routes have the same local preference, prefer the route that the specified router originated.
- If no route was originated, prefer the shorter autonomous system path.
- If all paths are external, prefer the lowest origin code (IGP < EGP < INCOMPLETE).
- If origin codes are the same and all the paths are from the same autonomous system, prefer the path with the lowest MULTI_EXIT_DISC METRIC. A missing metric is treated as zero.
- If the autonomous system paths are of the same length, prefer external paths over internal paths.
- If IGP synchronization is disabled and only internal paths remain, prefer the path through the closest neighbor.
- Prefer the route with the lowest IP address value for the BGP router ID.

BGP Configuration Task List

To configure BGP, complete the tasks in the following sections:

- Enable BGP Routing
- Configure BGP Neighbors
- Reset BGP Connections

The tasks in the following sections are optional:

- Configure BGP Route Filtering by Neighbor
- Configure BGP Path Filtering by Neighbor
- Configure BGP Community Filtering
- Disable Next-Hop Processing on BGP Updates
- Configure BGP Administrative Weights
- Configure BGP Interactions with IGPs
- Configure a Common Autonomous System

- Configure a Routing Domain Confederation
- Configure Miscellaneous BGP Parameters

Enable BGP Routing

To enable BGP routing, establish a BGP routing process on the router and specify those networks within the router's autonomous system to be advertised. Perform the following steps. There is a limit of 200 networks that can be advertised from one autonomous system.

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Enable a BGP routing process, which places you in router configuration mode.	**router bgp** *autonomous-system*
Step 3	Flag a network as local to this autonomous system.	**network** *network-number* **mask** *network-mask*

Note For exterior protocols, a reference to an IP network from the **network** router configuration command only controls which networks are advertised. This is in contrast to interior gateway protocols, such as IGRP, which also use the **network** command to determine where to send updates.

Configure BGP Neighbors

Like other exterior gateway protocols (EGPs), BGP must completely understand the relationships it has with its neighbors. BGP supports two kinds of neighbors: internal and external. Internal neighbors are in the same autonomous system; external neighbors are in different ASs. Normally, external neighbors are adjacent to each other and share a subnet, while internal neighbors may be anywhere in the same autonomous system.

To configure BGP neighbors, perform the following task in router configuration mode:

Task	Command
Specify a BGP neighbor.	**neighbor** *ip-address* **remote-as** *number*

You also can configure neighbor templates that use a word argument rather than an IP address to configure BGP neighbors. This is an advanced feature requiring a well-thought-out network architecture. Do not use this feature without thoroughly understanding its application.

Perform the following tasks in router configuration mode to configure BGP neighbor templates:

Task	Command
Support anonymous neighbor peers by configuring a neighbor template.	**neighbor** *template-name* **neighbor-list** *access-list-number*
Treat neighbors that have been accepted by a template as if they were configured by hand.	**neighbor** *template-name* **configure-neighbors**

Reset BGP Connections

Once you have defined two routers to be BGP neighbors, they will form a BGP connection and exchange routing information. If you subsequently change a BGP filter, weight, distance, version, or timer, or make a similar configuration change, you need to reset BGP connections for the configuration change to take effect. Perform either of the following tasks in EXEC mode to reset BGP connections:

Task	Command
Reset a particular BGP connection.	**clear ip bgp** *address*
Reset all BGP connections.	**clear ip bgp ***

To automatically reset BGP sessions, perform the following task in router configuration mode:

Task	Command
Automatically reset BGP sessions of any directly adjacent external peer if the link used to reach it goes down.	**bgp fast-external-fallover**

Configure BGP Route Filtering by Neighbor

If you want to restrict the routing information that the router learns or advertises, you can filter BGP routing updates to and from particular neighbors. To do this, define an access list and apply it to the updates. Distribute-list filters are applied to network numbers and not autonomous system paths.

To filter BGP routing updates, perform the following task in router configuration mode:

Task	Command
Filter BGP routing updates to/from neighbors as specified in an access list.	**neighbor** *ip-address* **distribute-list** *access-list-number* {**in** | **out**}

Configure BGP Path Filtering by Neighbor

In addition to filtering routing updates based on network numbers, you can specify an access list filter on both incoming and outbound updates based on the BGP autonomous system paths. Each filter is an access list based on regular expressions. To do this, define an autonomous system path access list and apply it to updates to and from particular neighbors. See the "Regular Expressions" appendix in the *Router Products Command Reference* publication for more information on forming regular expressions.

Perform the following tasks to configure BGP path filtering:

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Define a BGP-related access list.	**ip as-path access-list** *access-list-number* {**permit** | **deny**} *as-regular-expression*
Step 3	Enter router configuration mode.	See Table 2-1.
Step 4	Establish a BGP filter.	**neighbor** *ip-address* **filter-list** *access-list-number* {**in** | **out** | **weight** *weight*}

Configure BGP Community Filtering

BGP supports transit policies via controlled distribution of routing information. The distribution of routing information is based on one of three values:

- IP address (see the section "Configure BGP Route Filtering by Neighbor" earlier in this chapter).
- The value of the AS_PATH attribute (see the section "Configure BGP Path Filtering by Neighbor" earlier in this chapter).
- The value of the COMMUNITIES attribute (as described in this section).

The COMMUNITIES attribute is a way to group destinations into communities and apply routing decisions based on the communities. This method simplifies a BGP speaker's configuration that controls distribution of routing information.

A community is a group of destinations that share some common attribute. Each destination can belong to multiple communities. Autonomous system administrators can define which communities a destination belongs to. By default, all destinations belong to the general Internet community. The community is carried as the COMMUNITIES attribute.

The COMMUNITIES attribute is an optional, transitive, global attribute in the numerical range from 1 to 4,294,967,200 or set to one of these values:

- **internet**—Advertise this route to the Internet community.
- **no-export**—Do not advertise this route to EBGP peers.
- **no-advertise**—Do not advertise this route to any peer (internal or external).

Based on the community, you can control which routing information to accept, prefer, or distribute to other neighbors. A BGP speaker can set, append, or modify the community of a route when you learn, advertise, or redistribute routes. When routes are aggregated, the resulting aggregate has a COMMUNITIES attribute that contains all communities from all the initial routes.

To create a community list, perform the following task in global configuration mode:

Task	Command
Create a community list.	**ip community-list** *community-list-number* {**permit** | **deny**} *community-number*

To set the COMMUNITIES attribute, see the **match community-list** and **set community** commands in the section "Redistribute Routing Information" later in this chapter.

Disable Next-Hop Processing on BGP Updates

You can configure the router to disable next-hop processing for BGP updates to a neighbor. This is useful in non-meshed networks such as Frame Relay or X.25 where BGP neighbors might not have direct access to all other neighbors on the same IP subnet.

To disable next-hop processing, perform the following task in router configuration mode:

Task	Command
Disable next-hop processing on BGP updates to a neighbor.	**neighbor** *ip-address* **next-hop-self**

Configure BGP Administrative Weights

An administrative weight is a number that you can assign to a path so that you can control the path selection process. The administrative weight is local to the router. A weight can be a number from 0 to 65535. Paths that the router originates have weight 32768 by default, other paths have weight zero. If you have particular neighbors that you want to prefer for most of your traffic, you can assign a weight to all paths learned from a neighbor.

Perform the following task in router configuration mode to configure BGP administrative weights:

Task	Command
Specify a weight for all paths from a neighbor.	**neighbor** *ip-address* **weight** *weight*

In addition, you can assign weights based on autonomous system path access lists. A given weight becomes the weight of the path if the autonomous system path is accepted by the access list. Any number of weight filters are allowed.

Perform the following tasks to assign weights based on autonomous system path access lists:

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Define a BGP-related access list.	**ip as-path access-list** *access-list-number* {**permit** | **deny**} *as-regular-expression*
Step 3	Enter router configuration mode.	See Table 2-1.
Step 4	Configure set administrative weight on all incoming routes matching an autonomous system path filter.	**neighbor** *ip-address* **filter-list** *access-list-number* **weight** *weight*

Configure BGP Interactions with IGPs

If your autonomous system will be passing traffic through it from another autonomous system to a third autonomous system, it is very important that your autonomous system be consistent about the routes that it advertises. For example, if your BGP were to advertise a route before all routers in your network had learned about the route through your IGP, your autonomous system could receive traffic that some routers cannot yet route. To prevent this from happening, BGP must wait until the IGP has propagated routing information across your autonomous system. This causes BGP to be *synchronized* with the IGP. Synchronization is enabled by default.

In some cases, you do not need synchronization. If you will not be passing traffic from a different autonomous system through your autonomous system, or if all routers in your autonomous system will be running BGP, you can disable synchronization. Disabling this feature can allow you to carry fewer routes in your IGP, increase the number of paths that BGP can select, and allow BGP to converge more quickly, however you must run BGP on all routers in your autonomous system and there must be a full IBGP connectivity mesh between these routers. To disable synchronization, perform the following task in router configuration mode:

Task	Command
Disable synchronization between BGP and an IGP.	**no synchronization**

When you disable synchronization, you should also clear BGP routes using the **clear ip bgp** command.

In general, you will not want to redistribute most BGP routes into your IGP. A common design is to redistribute one or two routes and to make them exterior routes in IGRP or have your BGP speakers generate a default route for your autonomous system. When redistributing from BGP into IGP, only the routes learned using EBGP get redistributed.

In most circumstances, you also will not want to redistribute your IGP into BGP. Just list the networks in your autonomous system with **network** router configuration commands and your networks will be advertised. Networks that are listed this way are referred to as *local networks* and have a BGP origin attribute of "IGP." They must appear in the main IP routing table and can have any source; for example, they can be directly connected or learned via an IGP. The BGP routing process periodically scans the main IP routing table to detect the presence or absence of local networks, updating the BGP routing table as appropriate.

If you do perform redistribution into BGP, you must be very careful about the routes that can be in your IGP, especially if the routes were redistributed from BGP into the IGP elsewhere. This creates a situation where BGP is potentially injecting information into the IGP and then sending such information back into BGP and vice versa.

Networks that are redistributed into BGP from the EGP protocol will be given the BGP origin attribute "EGP." Other networks that are redistributed into BGP will have the BGP origin attribute of "incomplete." The origin attribute in our implementation is only used in the path selection process.

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of synchronization.

Configure Aggregate Addresses

CIDR lets you create aggregate routes, or *supernets*, to minimize the size of routing tables. You can configure aggregate routes in BGP either by redistributing an aggregate route into BGP or by using the conditional aggregation feature described in the next task table.

To create an aggregate address in the routing table, perform one or more of the following tasks in router configuration mode:

Task	Command
Create an aggregate entry in the BGP routing table. Advertise general information.	**aggregate-address** *address mask*
Advertised information will include all elements of all paths.	**aggregate-address** *address mask* **as-set**
Advertise summary addresses only.	**aggregate-address** *address-mask* **summary-only**
Suppress selected more-specific routes.	**aggregate-address** *address mask* **suppress-map** *map-tag*

Specify Automatic Summarization of Network Numbers

To disable automatic network number summarization when redistributing to BGP from IGPs, perform the following task in router configuration mode:

Task	Command
Disable automatic network summarization.	**no auto-summary**

Configure a Common Autonomous System

BGP requires internal BGP (IBGP) neighbors to be fully meshed, which is a processor-intensive situation. One way to reduce the IBGP mesh is to divide the autonomous system (AS) into multiple autonomous systems and specify that they are under common administration. Each autonomous system is fully meshed and has external BGP (EBGP) sessions with the peers in other autonomous systems. Even though the peers in different autonomous systems have EBGP sessions, they exchange routing information as if they are IBGP peers. Specifically, the next-hop and local preference information is preserved. This enables to you to retain a single Interior Gateway Protocol (IGP) for all of the autonomous systems.

To specify an autonomous system under a common administration, perform the following task in router configuration mode:

Task	Command
Specify a common BGP autonomous system.	**bgp common-as** *autonomous-system* [*autonomous-system ...*]

An alternative method to reduce the IBGP mesh is to use the Routing Domain Confederation feature described in the section “Configure a Routing Domain Confederation.”

Configure a Routing Domain Confederation

Another way to reduce the IBGP mesh is to divide an autonomous system into multiple autonomous systems and group them into a single confederation. Each autonomous system is fully meshed within itself, and has a few connections to another autonomous system in the same confederation. Even though the peers in different autonomous systems have EBGP sessions, they exchange routing information as if they are IBGP peers. Specifically, the next-hop and local preference information is preserved. This enables to you to retain a single Interior Gateway Protocol (IGP) for all of the autonomous systems. To the outside world, the confederation looks like a single autonomous system.

To configure a BGP confederation and the autonomous systems that belong to it, perform the following tasks in router configuration mode:

Task	Command
Configure a BGP confederation.	**bgp confederation identifier** *autonomous-system*
Specify the autonomous systems that belong to the confederation.	**bgp confederation peers** *autonomous-system* [*autonomous-system ...*]

An alternative method to reduce the IBGP mesh is to use the common administration method described in the section “Configure a Common Autonomous System” earlier in this chapter.

Configure Miscellaneous BGP Parameters

You can adjust several miscellaneous BGP parameters, as indicated in the following subsections.

Configure Neighbor Options

To provide BGP routing information to a large number of neighbors, you can configure BGP to accept neighbors based on an access list. If a neighbor attempts to initiate a BGP connection, its address must be accepted by the access list for the connection to be accepted. If you do this, the

router will not attempt to initiate a BGP connection to these neighbors, so the neighbors must be explicitly configured to initiate the BGP connection. If no access list is specified, all connections are accepted.

By default, no COMMUNITIES attribute is sent to a neighbor. You can specify that the COMMUNITIES attribute be sent to the neighbor at an IP address.

If a neighbor is running a different version of BGP, you should configure the version of BGP that the neighbor is speaking.

External BGP peers normally must reside on a directly connected network. Sometimes it is useful to relax this restriction in order to test BGP; do so by specifying the **neighbor ebgp-multihop** command

For internal BGP, you might want to allow your BGP connections to stay up regardless of which interfaces are available on the router. To do this, you first configure a *loopback* interface and assign it an IP address. Next, configure the BGP update source to be the loopback interface. Finally, configure your neighbor to use the address on the loopback interface.

You can also set the minimum interval of time between BGP routing updates and apply a route map to incoming and outgoing routes.

Configure any of the following neighbor options in router configuration mode:

Task	Command
Specify an access list of BGP neighbors.	**neighbor any** [*access-list-number*]
Specify that the COMMUNITIES attribute be sent to the neighbor at this IP address.	**neighbor** *ip-address* **send-community**
Specify the BGP version to use when communicating with a neighbor.	**neighbor** *ip-address* **version** *value*
Allow internal BGP sessions to use any operational interface for TCP connections.	**neighbor** *ip-address* **update-source** *interface*
Allow BGP sessions even when the neighbor is not on a directly connected segment.	**neighbor** *ip-address* **ebgp-multihop**
Set the minimum interval between sending BGP routing updates.	**neighbor** {*address* | *tag*} **advertisement-interval** *seconds*
Apply a route map to incoming or outgoing routes.	**neighbor** {*address* | *tag*} **route-map** *route-map-name* {**in** | **out**}

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for examples of configuring BGP neighbor options.

Set the Network Weight

To set the absolute weight for a network, perform the following task in router configuration mode:

Task	Command
Set the weight for a network.	**network** *address* **weight** *weight*

Indicate Backdoor Routes

You can indicate which networks are reachable using a *backdoor* route that the border router should use. A backdoor network is treated as a local network, except that it is not advertised. To configure backdoor routes, perform the following task in router configuration mode:

Task	Command
Indicate reachable networks through backdoor routes.	**network** *address* **backdoor**

Update IP Routing Table

To modify metric and tag information when the IP routing table is updated with BGP learned routes, perform the following task in router configuration mode:

Task	Command
Apply route-map to routes when updating the IP routing table.	**table-map** *route-map name*

Set Administrative Distance

Administrative distance is a measure of the ability of a routing protocol to provide optimal routes. BGP uses three different administrative distances—external, internal, and local. Routes learned through external BGP are given the external distance, routes learned with internal BGP are given the internal distance, and routes that are part of this autonomous system are given the local distance. To assign a BGP administrative distance, perform the following task in router configuration mode:

Task	Command
Assign a BGP administrative distance.	**distance bgp** *external-distance internal-distance local-distance*

Changing the administrative distance of BGP routes is considered dangerous and generally is not recommended. The external distance should be lower than any other dynamic routing protocol, and the internal and local distances should be higher than any other dynamic routing protocol.

Adjust BGP Timers

BGP uses certain timers to control periodic activities such as the sending of keepalive messages and the interval after not receiving a keepalive message after which the router declares a peer dead. You can adjust these timers. When a connection is started, BGP will negotiate the hold time with the neighbor. The smaller of the two hold times will be chosen. The keepalive timer is then set based on the negotiated holdtime and the configured keepalive time. To adjust BGP timers, perform the following task in router configuration mode:

Task	Command
Adjust BGP timers.	**timers bgp** *keepalive holdtime*

Configure the MULTI_EXIT_DISC METRIC

BGP uses the MULTI_EXIT_DISC METRIC as a hint to external neighbors about preferred paths. (The name of this metric for BGP Versions 2 and 3 is INTER_AS.) If you have a router that traffic should avoid, you can configure that router with a higher MULTI_EXIT_DISC METRIC. Doing this sets the MULTI_EXIT_DISC METRIC on all paths that the router advertises. Perform the following task in router configuration mode:

Task	Command
Set an MULTI_EXIT_DISC METRIC.	**default-metric** *number*

Change the Local Preference Value

You can define a particular path as more or less preferable than other paths by changing the default local preference value of 100. To assign a different default local preference value, perform the following task in router configuration mode:

Task	Command
Change the default local preference value.	**bgp default local-preference** *value*

You can use route maps to change the default local preference of specific paths. See the "Using Route Maps with BGP" section for examples.

Redistribute Network 0.0.0.0

To redistribute network 0.0.0.0, perform the following task in router configuration mode:

Task	Command
Allow the redistribution of network 0.0.0.0 into BGP.	**default-information originate**

Configure EGP

The Exterior Gateway Protocol (EGP), specified in RFC 904, is an older EGP used for communicating with certain routers in the Defense Data Network (DDN) that the U.S. Department of Defense designates as *core routers*. EGP also was used extensively when attaching to the National Science Foundation Network (NSFnet) and other large backbone networks.

An exterior router uses EGP to advertise its knowledge of routes to networks within its autonomous system. It sends these advertisements to the core routers, which then readvertise their collected routing information to the exterior router. A neighbor or peer router is any router with which the router communicates using EGP.

Cisco's EGP Implementation

Cisco's implementation of EGP supports three primary functions, as specified in RFC 904:

- Routers running EGP establish a set of neighbors, and these neighbors share reachability information.
- EGP routers poll their neighbors periodically to see if they are "alive."
- EGP routers send update messages containing information about the reachability of networks within their autonomous systems.

EGP Configuration Task List

To enable EGP routing on your router, complete the tasks in the following sections. The tasks in the first two sections are mandatory; the tasks in the other sections are optional.

- Enable EGP Routing
- Configure EGP Neighbor Relationships
- Adjust EGP Timers
- Configure Third-Party EGP Support
- Configure Backup Routers
- Configure Default Routes
- Define a Central Routing Information Manager (Core Gateway)

Enable EGP Routing

To enable EGP routing, you must specify an autonomous system number, generate an EGP routing process, and indicate the networks for which the EGP process will operate.

Perform these required tasks in the order given as shown in the following table:

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Specify the autonomous system that the router resides in for EGP.	**autonomous-system** *local-as*
Step 3	Enable an EGP routing process, which places you in router configuration mode.	**router egp** *remote-as*
Step 4	Specify a network to be advertised to the EGP peers of an EGP routing process.	**network** *network-number*

Note For exterior gateway protocols, a reference to an IP network from the **network** router configuration command that is learned by another routing protocol does not require a **redistribute** router configuration command. This is in contrast to interior gateway protocols, such as IGRP, which require the use of the **redistribute** command.

Configure EGP Neighbor Relationships

A router using EGP cannot dynamically determine its neighbor or peer routers. You must therefore provide a list of neighbor routers.

To specify an EGP neighbor, perform the following task in router configuration mode:

Task	Command
Specify an EGP neighbor.	**neighbor** *ip-address*

Adjust EGP Timers

The EGP timers consist of a hello timer and a poll time interval timer. The hello timer determines the frequency in seconds with which the router sends hello messages to its peer. The poll time is how frequently to exchange updates. Our implementation of EGP allows these timers to be adjusted by the user.

To adjust EGP timers, perform the following task in router configuration mode:

Task	Command
Adjust EGP timers.	**timers egp** *hello polltime*

Configure Third-Party EGP Support

EGP supports a *third-party mechanism* in which EGP tells an EGP peer that another router (the third party) on the shared network is the appropriate router for some set of destinations.

To specify third-party routers in updates, perform the following task in router configuration mode:

Task	Command
Specify a third-party through which certain destinations can be achieved.	**neighbor** *ip-address* **third-party** *third-party-ip-address* [**internal** | **external**]

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of configuring third-party EGP support.

Configure Backup Routers

You might want to provide backup in the event of site failure by having a second router belonging to a different autonomous system act as a backup to the EGP router for your autonomous system. To differentiate between the primary and secondary EGP routers, the two routers will advertise network routes with differing EGP distances or metrics. A network with a low metric is generally favored over a network with a high metric.

Networks declared as local are always announced with a metric of zero. Networks that are redistributed will be announced with a metric specified by the user. If no metric is specified, redistributed routes will be advertised with a metric of three. All redistributed networks will be advertised with the same metric. The redistributed networks can be learned from static or dynamic routes. See also the "Redistribute Routing Information" section later in this chapter.

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of configuring backup routers.

Configure Default Routes

You also can designate network 0.0.0.0 as a default route. If the next hop for the default route can be advertised as a third party, it will be included as a third party.

To enable the use of default EGP routes, perform the following task in router configuration mode:

Task	Command
Configure EGP to generate a default route.	**default-information originate**

Define a Central Routing Information Manager (Core Gateway)

Normally, an EGP process expects to communicate with neighbors from a single autonomous system. Because all neighbors are in the same autonomous system, the EGP process assumes that these neighbors all have consistent internal information. Therefore, if the EGP process is informed about a route from one of its neighbors, it will not send it out to other neighbors.

With *core EGP*, the assumption is that all neighbors are from different autonomous systems, and all have inconsistent information. In this case, the EGP process distributes routes from one neighbor to all others (but not back to the originator). This allows the EGP process to be a central clearinghouse for information with a single, central manager of routing information (sometimes called a *core gateway*). To this end, one core gateway process can be configured for each router.

To define a core gateway process, perform the following steps in the order in which they appear:

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Allow a specific router to act as a peer with any reachable autonomous system.	**router egp 0**
Step 3	Define how an EGP process determines which neighbors will be treated as peers. or	**neighbor any** [*access-list-number*]
	Allow the specified address to be used as the next hop in EGP advertisements.	**neighbor any third-party** *ip-address* [**internal** | **external]**

The EGP process defined in this way can act as a peer with any autonomous system, and information is interchanged freely between autonomous systems.

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of configuring an EGP core gateway.

Note Split horizon is performed only on a *per-gateway* basis (in other words, if an external router informs the router about a specific network, and that router is the *best* path, the router will *not* inform the originating external router about that path). Our routers can also perform per-gateway split horizon on third-party updates.

Configure GDP

The Gateway Discovery Protocol (GDP), designed by Cisco to address customer needs, allows hosts to dynamically detect the arrival of new routers, as well as determine when a router goes down. You must have host software to take advantage of this protocol.

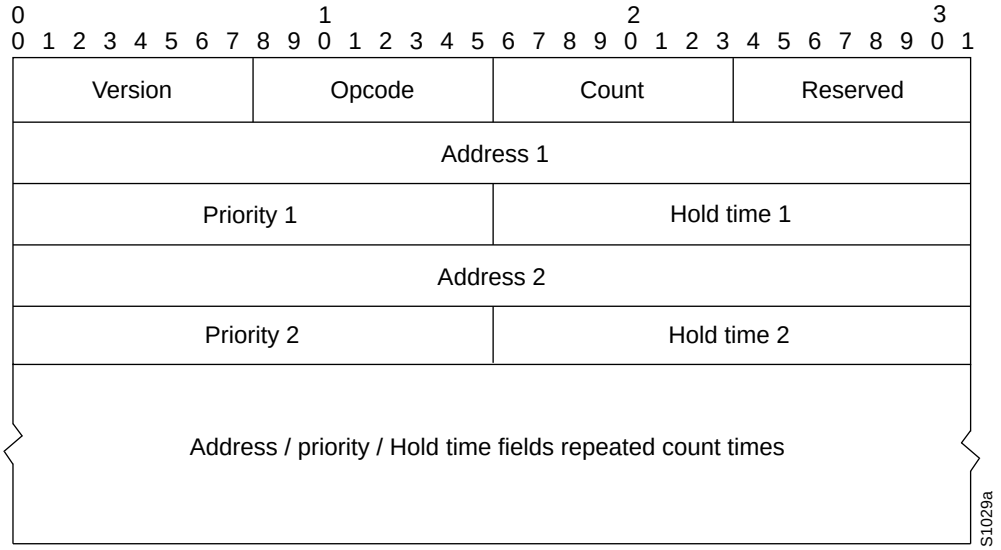
For ease of implementation on a variety of host software, GDP is based on the User Datagram Protocol (UDP). The UDP source and destination ports of GDP datagrams are both set to 1997 (decimal).

There are two types of GDP messages: *report* and *query*. On broadcast media, report message packets are periodically sent to the IP broadcast address announcing that the router is present and functioning. By listening for these report packets, a host can detect a vanishing or appearing router. If a host issues a query packet to the broadcast address, the routers each respond with a report sent

to the host's IP address. On nonbroadcast media, routers send report message packets only in response to query message packets. The protocol provides a mechanism for limiting the rate at which query messages are sent on nonbroadcast media.

Figure 18-2 shows the format of the GDP report message packet format. A GDP query message packet has a similar format, except that the count field is always zero and no address information is present.

Figure 18-2 GDP Report Message Packet Format



The fields in the Report and Query messages are as follows:

- Version—8-bit field containing the protocol version number. The current GDP version number is 1. If an unrecognized version number is found, the GDP message must be ignored.
- Opcode—8-bit field that describes the GDP message type. Unrecognized opcodes must be ignored. Opcode 1 is a report message and opcode 2 is a query message.
- Count—8-bit field that contains the number of address, priority, and hold time tuples in this message. A query message has a Count field value of zero. A report message has a count field value of 1 or greater.
- Reserved—8-bit reserved field; it must be set to zero.
- Address—32-bit fields containing the IP address of a router on the local network segment. There are no other restrictions on this address. If a host encounters an address that it believes is not on its local network segment, it should ignore that address.
- Priority—16-bit fields that indicate the relative quality of the associated address. The numerically larger the value in the priority field, the better the address should be considered.
- Hold Time—16-bit fields. On broadcast media, the number of seconds the associated address should be used as a router without hearing further report messages regarding that address. On nonbroadcast media such as X.25, this is the number of seconds the requester should wait before sending another query message.

Numerous actions can be taken by the host software listening to GDP packets. One possibility is to flush the host's ARP cache whenever a router appears or disappears. A more complex possibility is to update a host routing table based on the coming and going of routers. The particular course of action taken depends on the host software and your network requirements.

To enable GDP routing and other optional GDP tasks as required for your network, perform the following tasks in interface configuration mode:

Task	Command
Enable GDP processing on an interface.	**ip gdp**
Set the relative quality of the associated address.	**ip gdp priority** *number*
Set the GDP report period.	**ip gdp reporttime** *seconds*
Set the length of time the associated address should be used as a router without hearing further report messages regarding that address.	**ip gdp holdtime** *seconds*

Configure IRDP

Like GDP, the ICMP Router Discovery Protocol (IRDP) allows hosts to locate routers. When operating as a client, router discovery packets are generated, and when operating as a host, router discovery packets are received.

The only required task for configuring IRDP routing on a specified interface is to enable IRDP processing ona ninterface. Perform the following task in interface configuration mode:

Task	Command
Enable IRDP processing on an interface.	**ip irdp**

When you enable IRDP processing, the default parameters will apply. You can optionally change any of these IRDP parameters. Perform the following tasks in interface configuration mode:

Task	Command
Send IRDP advertisements to the all-systems multicast address (224.0.0.1) on a specified interface.	**ip irdp multicast**
Set the IRDP period for which advertisements are valid.	**ip irdp holdtime** *seconds*
Set the IRDP maximum interval between advertisements.	**ip irdp maxadvertinterval** *seconds*
Set the IRDP minimum interval between advertisements.	**ip irdp minadvertinterval** *seconds*
Set a router's IRDP preference level.	**ip irdp preference** *number*
Specify an IRDP address and preference to proxy-advertise.	**ip irdp address** *address* [*number*]

A router can proxy-advertise other machines that use IRDP; however, this is not recommended because it is possible to advertise nonexistent machines or machines that are down.

Configure IP Multicast Routing

Traditional IP communication allows a host to send packets to a single host (unicast transmission) or to all hosts (broadcast transmission). IP multicast provides a third scheme, allowing a host to send packets to a subset of all hosts (group transmission). These hosts are known as group members.

Packets delivered to group members are identified by a single multicast group address. Multicast packets are delivered to a group using best-effort reliability just like IP unicast packets.

The multicast environment consists of senders and receivers. Any host, whether it is a member of a group or not, can send to a group. However, only the members of a group receive the message.

A multicast address is chosen for the receivers in a multicast group. Senders use that address as the destination address of a datagram to reach all members of the group.

Membership in a multicast group is dynamic: hosts can join and leave at any time. There is no restriction on the location or number of members in a multicast group. A host can be a member of more than one multicast group at a time.

How active a multicast group is and what members it has can vary from group to group and from time to time. A multicast group can be active for a very long while, or it may be very short-lived. Membership in a group can change constantly. A group that has members may have no activity.

Routers executing a multicast routing protocol, such as PIM, maintain forwarding tables to forward multicast datagrams. Routers use the Internet Group Management Protocol (IGMP) to learn whether members of a group are present on their directly attached subnets. Hosts join multicast groups by sending IGMP report messages.

Cisco's Implementation of IP Multicast Routing

The Cisco IOS software supports two protocols to implement IP multicast routing:

- Internet Group Management Protocol (IGMP). IGMP is used between hosts on a LAN and the router(s) on that LAN to track which multicast groups the hosts are members of.
- Protocol-Independent Multicast (PIM). PIM is used between routers so they can track which multicast packets to forward to each other and to their directly connected LANs.

Internet Group Management Protocol (IGMP)

IGMP is used by IP hosts to report their group membership to directly connected multicast routers. IGMP is an integral part of IP. IGMP is defined in RFC 1112, *Host Extensions for IP Multicasting*.

IGMP uses group addresses, which are Class D IP addresses. The high-order four bits of a Class D address are 1110. This means that host group addresses can be in the range 224.0.0.0 to 239.255.255.255. The address 224.0.0.0 is guaranteed not to be assigned to any group. The address 224.0.0.1 is assigned to all systems on a subnet. The address 224.0.0.2 is assigned to all routers on a subnet.

Protocol-Independent Multicast (PIM) Protocol

The PIM protocol maintains the current IP multicast service mode of receiver-initiated membership. It is not dependent on a specific unicast routing protocol.

PIM is defined in the following IETF Internet drafts: *Protocol Independent Multicast (PIM): Motivation and Architecture*; *Protocol Independent Multicast (PIM), Dense Mode Protocol Specification*; *Protocol Independent Multicast (PIM), Sparse Mode Protocol Specification*; *IGMP Router Extensions for Routing to Dense Multicast Groups*; and *IGMP Router Extensions for Routing to Sparse Multicast Groups*.

PIM can operate in two modes: dense mode and sparse mode.

In dense mode, a router assumes that all other routers do want to forward multicast packets for a group. If a router receives a multicast packet and has no directly connected members or PIM neighbors present, a prune message is sent back to the source. Subsequent multicast packets are not flooded to this router on this pruned branch. PIM builds source-based multicast distribution trees.

In sparse mode, a router assumes that other routers do not want to forward multicast packets for a group unless there is an explicit request for the traffic. When hosts join a multicast group, the directly connected routers send PIM Join messages to the rendezvous point (RP). The RP keeps track of multicast groups. Hosts that send multicast packets are registered with the RP by that host's first-hop router. The RP then sends joins toward the source. At this point, packets are forwarded on a shared distribution tree. If the multicast traffic from a specific source is sufficient, the receiver's first-hop router may send joins toward the source to build a source-based distribution tree.

IP Multicast Routing Configuration Task List

To configure IP multicast routing, perform the required tasks described in the following sections:

- Enable IP Multicast Routing on the Router
- Enable PIM on an Interface

You can also perform the optional tasks described in the following sections:

- Configure a Router to Be a Member of a Group
- Configure the Host-Query Message Interval
- Control Access to IP Multicast Groups
- Modify PIM Message Timers
- Configure the TTL Threshold
- Configure DVMRP Interoperability
- Advertise Network 0.0.0.0 to DVMRP Neighbors
- Configure a DVMRP Tunnel

Configuration examples are provided at the end of this chapter.

Enable IP Multicast Routing on the Router

Enabling IP multicast routing on the router allows the router to forward multicast packets.

To enable IP multicast routing on the router, perform the following task in global configuration mode:

Task	Command
Enable IP multicast routing on the router.	**ip multicast-routing**

Enable PIM on an Interface

Enabling PIM on an interface also enables IGMP operation on that interface. An interface can be configured to be in dense mode or sparse mode. The mode describes how the router populates its multicast routing table and how the router forwards multicast packets it receives from its directly connected LANs. In populating the multicast routing table, dense-mode interfaces are always added to the table. Sparse-mode interfaces are added to the table only when periodic join messages are received from downstream routers or there is a directly connected member on the interface. When forwarding from a LAN, sparse-mode operation occurs if there is an RP known for the group. If so, the packets are encapsulated and sent toward the RP. When no RP is known, the packet is flooded in a dense-mode fashion. If the multicast traffic from a specific source is sufficient, the receiver's first-hop router may send joins toward the source to build a source-based distribution tree.

There is no default mode setting. By default, multicast routing is disabled on an interface.

To configure PIM on an interface to be in dense mode, perform the following task in interface configuration mode:

Task	Command
Enable dense-mode PIM on the interface.	**ip pim dense-mode**

For an example of how to configure a PIM interface in dense mode, see the section "Configure a Router to Operate in Dense Mode Example" later in this chapter.

To configure PIM on an interface to be in sparse mode, perform the following task in interface configuration mode:

Task	Command
Enable sparse-mode PIM on the interface.	**ip pim sparse-mode**

If you configure the router to operate in sparse mode, you must also choose one or more routers to be RPs. You do not have to configure the routers to be RPs; they learn this themselves. RPs are used by senders to a multicast group to announce their existence and by receivers of multicast packets to learn about new senders. A router can be configured so that packets for a single multicast group can use one or more RPs.

You must configure the IP address of RPs in leaf routers only. Leaf routers are those routers that are directly connected either to a multicast group member or to a sender of multicast messages.

The RP address is used by first-hop routers to send PIM register messages on behalf of a host sending a packet to the group. The RP address is also used by last-hop routers to send PIM join/prune messages to the RP to inform it about group membership. The RP does not need to know it is an RP. You need to configure the RP address only on first-hop and last-hop routers (leaf routers).

A PIM router can be an RP for more than one group. A group can have more than one RP. The conditions specified by the access-list determine for which groups the router is an RP.

To configure the address of the RP, perform the following task in global configuration mode:

Task	Command
Configure the address of a PIM RP.	**ip pim rp-address** *ip-address* [*access-list-number*]

For an example of how to configure a PIM interface in sparse mode, see the section "Configure a Router to Operate in Sparse Mode Example" later in this chapter.

Configure a Router to Be a Member of a Group

Cisco routers can be configured to be members of a multicast group. This is useful for determining multicast reachability in a network. If a router is configured to be a group member and supports the protocol that is being transmitted to the group, it can respond. An example is **ping**. A router will respond to ICMP echo request packets addressed to a group for which it is a member. Another example is the multicast traceroute tools provided in the Cisco IOS software.

To have a router join a multicast group and turn on IGMP on the router, perform the following task in interface configuration mode:

Task	Command
Join a multicast group.	**ip igmp join-group** *group-address*

Configure the Host-Query Message Interval

Multicast routers send IGMP host-query messages to discover which multicast groups are present on attached networks. These messages are sent to the all-systems group address of 224.0.0.1 with a TTL of 1.

Multicast routers send host-query messages periodically to refresh their knowledge of memberships present on their networks. If, after some number of queries, the router discovers that no local hosts are members of a multicast group, the router stops forwarding onto the local network multicast packets from remote origins for that group and sends a prune message upstream toward the source.

Multicast routers elect a PIM designated router for the LAN (subnet). This is the router with the highest IP address. The designated router is responsible for sending IGMP host-query messages to all hosts on the LAN. In sparse mode, the designated router also sends PIM register and PIM join messages towards the RP router.

By default, the designated router sends IGMP host-query messages once a minute in order to keep the IGMP overhead on hosts and networks very low. To modify this interval, perform the following task in interface configuration mode:

Task	Command
Configure the frequency at which the designated router sends IGMP host-query messages.	**ip igmp query-interval** *seconds*

Control Access to IP Multicast Groups

Multicast routers send IGMP host-query messages to determine which multicast groups have members of the router's attached local networks. The routers then forward to these group members all packets addressed to the multicast group. You can place a filter on each interface that restricts the multicast groups that hosts on the subnet serviced by the interface can join.

To filter multicast groups allowed on an interface, perform the following task in interface configuration mode:

Task	Command
Control the multicast groups that hosts on the subnet serviced by an interface can join.	**ip igmp access-group** *access-list-number*

Modify PIM Message Timers

By default, multicast routers send PIM router-query messages every 30 seconds. To modify this interval, perform the following task in interface configuration mode:

Task	Command
Configure the frequency at which multicast routers send PIM router-query messages.	**ip pim query-interval** *seconds*

Configure the TTL Threshold

The time-to-live (TTL) value controls whether packets are forwarded out of an interface. You specify the TTL value in hops. Any multicast packet with a TTL less than the interface TTL threshold is not forwarded on the interface. The default value is 0, which means that all multicast packets are forwarded on the interface. To change the default TTL threshold value, perform the following task in interface configuration mode:

Task	Command
Configure the TTL threshold of packets being forwarded out an interface.	**ip multicast-threshold** *ttl*

Configure DVMRP Interoperability

Cisco multicast routers using PIM can interoperate with non-Cisco multicast routers that use the Distance Vector Multicast Routing Protocol (DVMRP).

PIM routers dynamically discover DVMRP multicast routers on attached networks. Once a DVMRP neighbor has been discovered, the router periodically transmits DVMRP report messages advertising the unicast sources reachable in the PIM domain. By default, directly connected subnets and networks are advertised. The router forwards multicast packets that have been forwarded by DVMRP routers and in turn forwards multicast packets to DVMRP routers.

You can configure what sources are advertised and what metrics are used by using the **ip dvmrp metric** command. You can also direct all sources learned via a particular unicast routing process to be advertised into DVMRP.

It is necessary to use mrouted version 2.2 (which implements a nonpruning version of DVMRP) or version 3.2 (which implements a pruning version of DVMRP) when Cisco routers are directly connected to DVMRP routers or interoperate with DVMRP routers over an MBONE tunnel. DVMRP advertisements produced by Cisco routers can cause older versions of mrouted to corrupt their routing tables and those of their neighbors.

To configure the sources that are advertised and the metrics that are used when transmitting DVMRP report messages, perform the following task in interface configuration mode:

Task	Command
Configure the metric associated with a set of destinations for DVMRP reports.	**ip dvmrp metric** *metric* [*access-list-number*] [*protocol process-id*]

Our routers answer mrinfo requests sent by mrouted systems. The router returns information about neighbors on DVMRP tunnels. This information includes the metric, which is always set to 1, the configured TTL threshold, and the status of the tunnel.

For an example of how to configure a PIM router to interoperate with a DVMRP router, see the section “Configure DVMRP Interoperability Examples” later in this chapter.

Advertise Network 0.0.0.0 to DVMRP Neighbors

The mrouted protocol is a public domain implementation of DVMRP. If your router is a neighbor to an mrouted version 3.4 machine, you can configure the router to advertise network 0.0.0.0 to the DVMRP neighbor. You must specify whether only route 0.0.0.0 is advertised or other routes can also be specified.

To advertise network 0.0.0.0 to DVMRP neighbors on an interface, perform the following task in interface configuration mode:

Task	Command
Advertise network 0.0.0.0 to DVMRP neighbors.	**ip dvmrp default-information** {**originate** | **only**}

Configure a DVMRP Tunnel

Cisco routers can support DVMRP tunnels to the MBONE. (The MBONE is the multicast backbone of the Internet.) You can configure a DVMRP tunnel on a router if the other end is running DVMRP. The router then sends and receives multicast packets over the tunnel. This allows a PIM domain to connect to the DVMRP router in the case where all routers on the path do not support multicast routing.

When a Cisco router runs DVMRP over a tunnel, it advertises sources in DVMRP report messages much as it does on real networks. In addition, DVMRP report messages received are cached on the router and are used as part of its Reverse Path Forwarding (RPF) calculation. This allows multicast packets received over the tunnel to be forwarded by the router.

When you configure a DVMRP tunnel, you should assign a tunnel an address for two reasons:

- To enable the sending of IP packets over the tunnel
- To indicate whether the Cisco IOS software should perform DVMRP summarization

You can assign an IP address either by using the **ip address** interface configuration command or by using the **ip unnumbered** interface configuration command to configure the tunnel to be unnumbered. Either of these two methods allows IP multicast packets to flow over the tunnel. The Cisco IOS software will not advertise subnets over the tunnel if the tunnel has a different network number from the subnet. In this case, the Cisco IOS software advertises only the network number over the tunnel.

To configure a DVMRP tunnel, perform the following tasks:

Task		Command
Step 1	Specify a tunnel interface in global configuration mode. This puts the router into interface configuration mode.	**interface tunnel** *number*
Step 2	Set the tunnel interface's source address. This is the IP address of the interface on the router.	**tunnel source** *ip-address*
Step 3	Set the tunnel interface's destination address. This is the IP address of the mrouted multitask router.	**tunnel destination** *ip-address*
Step 4	Configure a DVMRP tunnel.	**tunnel mode dvmrp**
Step 5	Assign an IP address to the interface. or Configure the interface as unnumbered.	**ip address** *address mask* **ip unnumbered**

Task		Command
Step 6	Configure PIM on the interface.	**ip pim** {**dense-mode** | **sparse-mode**}
Step 7	Configure an acceptance filter for incoming DVMRP reports.	**ip dvmrp accept-filter** *access-list-number administrative-distance*

For an example of how to configure a DVMRP interoperability over a tunnel interface, see the section "Configure DVMRP Interoperability Examples" later in this chapter.

Configure Routing Protocol-Independent Features

Previous sections addressed configurations of specific routing protocols. Complete the protocol-independent tasks described in the following sections as needed:

- Use Variable-Length Subnet Masks
- Configure Static Routes
- Specify Default Routes
- Redistribute Routing Information
- Filter Routing Information
- Adjust Timers
- Enable or Disable Split Horizon

Use Variable-Length Subnet Masks

OSPF, static routes, and IS-IS support variable-length subnet masks (VLSMs). With VLSMs, you can use different masks for the same network number on different interfaces, which allows you to conserve IP addresses and more efficiently use available address space. However, using VLSMs also presents address assignment challenges for the network administrator and ongoing administrative challenges.

Refer to RFC 1219 for detailed information about VLSMs and how to correctly assign addresses.

Note Consider your decision to use VLSMs carefully. It is easy to make mistakes in address assignments and it is generally more difficult to monitor your network using VLSMs.

The best way to implement VLSMs is to keep your existing numbering plan in place and gradually migrate some networks to VLSMs to recover address space. See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of using VLSMs.

Configure Static Routes

Static routes are user-defined routes that cause packets moving between a source and a destination to take a specified path. Static routes can be important if the router cannot build a route to a particular destination. They are also useful for specifying a gateway of last resort to which all unroutable packets will be sent.

To configure static routes, perform the following task in global configuration mode:

Task	Command
Establish a static route.	**ip route** *network* [*mask*] {*address* | *interface*} [*distance*]

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of configuring static routes.

The router remembers static routes until you remove them (using the **no** form of the **ip route** global configuration command). However, you can override static routes with dynamic routing information through prudent assignment of administrative distance values. Each dynamic routing protocol has a default administrative distance, as listed in Table 18-2. If you would like a static route to be overridden by information from a dynamic routing protocol, simply ensure that the administrative distance of the static route is higher than that of the dynamic protocol.

Static routes that point to an interface will be advertised via RIP, IGRP, and other dynamic routing protocols, regardless of whether **redistribute static** commands were specified for those routing protocols. This is because static routes that point to an interface are considered in the routing table to be connected and hence lose their static nature. However, if you define a static route to an interface that is not one of the networks defined in a **network** command, no dynamic routing protocols will advertise the route unless a **redistribute static** command is specified for these protocols.

When an interface goes down, all static routes through that interface are removed from the IP routing table. Also, when the router can no longer find a valid next hop for the address specified as the forwarding router's address in a static route, the static route is removed from the IP routing table.

Table 18-2 Default Administrative Distances

Route Source	Default Distance
Connected interface	0
Static route	1
External BGP	20
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
Internal BGP	200
Unknown	255

Specify Default Routes

A router might not be able to determine the routes to all other networks. To provide complete routing capability, the common practice is to use some routers as "smart routers" and give the remaining routers default routes to the smart router. (Smart routers have routing table information for the entire internetwork.) These default routes can be passed along dynamically or can be configured into the individual routers.

Most dynamic interior routing protocols include a mechanism for causing a smart router to generate dynamic default information that is then passed along to other routers.

Specify a Default Network

If a router has a directly connected interface onto the specified default network, the dynamic routing protocols running on that router will generate or source a default route. In the case of RIP, it will advertise the pseudonetwork 0.0.0.0. In the case of IGRP, the network itself is advertised and flagged as an exterior route.

A router that is generating the default for a network also may need a default of its own. One way of doing this is to specify a static route to the network 0.0.0.0 through the appropriate router.

To define a static route to a network as the static default route, perform the following task in global configuration mode:

Task	Command
Specify a default network.	**ip default-network** *network-number*

Gateway of Last Resort

When default information is being passed along through a dynamic routing protocol, no further configuration is required. The system will periodically scan its routing table to choose the optimal default network as its default route. In the case of RIP, it will be only one choice, network 0.0.0.0. In the case of IGRP, there might be several networks that can be candidates for the system default. The router uses both administrative distance and metric information to determine the default route (gateway of last resort). The selected default route appears in the gateway of last resort display of the **show ip route** EXEC command.

If dynamic default information is not being passed to the router, candidates for the default route can be specified with the **ip default-network** command. In this usage, **ip default-network** takes a nonconnected network as an argument. If this network appears in the routing table from any source (dynamic or static), it is flagged as a candidate default route and is a possible choice as the default route for the router.

If the router has no interface on the default network but does have a route to it, it will consider this network as a candidate default path. The route candidates will be examined and the best one will be chosen based on administrative distance and metric. The gateway to the best default path will become the gateway of last resort for the router.

Redistribute Routing Information

In addition to running multiple routing protocols simultaneously, the router can redistribute information from one routing protocol to another. For example, you can instruct the router to readvertise IGRP-derived routes using the RIP protocol, or to readvertise static routes using the IGRP protocol. This applies to all of the IP-based routing protocols.

You also can conditionally control the redistribution of routes between routing domains by defining a method known as *route maps* between the two domains.

The following four tables list tasks associated with route redistribution.

To define a route map for redistribution, perform the following task in global configuration mode:

Task	Command
Define any route maps needed to control redistribution.	**route-map** *map-tag* [[**permit** | **deny**] [*sequence-number*]]

A pair of **match** and **set** commands are required to follow a **route-map** command. To define conditions for redistributing routes from one routing protocol into another, perform at least one of the following tasks in route-map configuration mode:

Task	Command
Match a BGP autonomous system path access list.	**match as-path** *path-list-number*
Match a BGP community list.	**match community-list** *community-list-number* [**exact**]
Match a standard access list.	**match ip address** *access-list-number...access-list-number*
Match the specified metric.	**match metric** *metric-value*
Match a next-hop router address passed by one of the access lists specified.	**match ip next-hop** *access-list-number...access-list-number*
Match the specified tag value.	**match tag** *tag-value...tag-value*
Match the specified next hop route out one of the interfaces specified.	**match interface** *name unit...name unit*
Match the address specified by the specified advertised access lists.	**match ip route-source** *access-list-number...access-list-number*
Match the specifed route type.	**match route-type** {**local** | **internal** | **external** [**type-1** | **type-2**] | **level-1** | **level-2**}

A pair of **match** and **set** commands are required to follow a **route-map** command. To define conditions for redistributing routes from one routing protocol into another, perform at least one of the following tasks in route-map configuration mode:

Task	Command
Set the COMMUNITIES attribute.	**set community** *community-number* [**additive**]
Assign a value to a local BGP path.	**set local-preference** *value*
Specify the BGP weight for the routing table.	**set weight** *weight*
Set the BGP origin code.	**set origin** {**igp** | **egp** *as* | **incomplete**}
Specify the address of the next hop.	**set next-hop** *next-hop*
Enable automatic computing of tag table.	**set automatic-tag**
For routes that are advertised into the specified area of the routing domain.	**set level** {**level-1** | **level-2** | **level-1-2** | **stub-area** | **backbone**}
Set the metric value to give the redistributed routes.	**set metric** *metric-value*
Set the metric type to give redistributed routes.	**set metric-type** {**internal** | **external** | **type-1** | **type-2**}
Set a tag value to associate with the redistributed routes.	**set tag** *tag-value*

To distribute routes from one routing domain into another and to control route redistribution, perform the following tasks in router configuration mode:

Task	Command
Redistribute routes from one routing protocol to another routing protocol.	**redistribute** *protocol* [*process-id*] {**level-1** | **level-1-2** | **level-2**} [**metric** *metric-value*] [**metric-type** *type-value*] [**match internal** | **external** *type-value*] [**tag** *tag-value*] [**route-map** *map-tag*] [**weight** *weight*] [**subnets**]

Task	Command
Cause the current routing protocol to use the same metric value for all redistributed routes (BGP, EGP, OSPF, RIP).	**default-metric** *number*
Cause the IGRP or enhanced IGRP routing protocol to use the same metric value for all redistributed routes.	**default-metric** *bandwidth delay reliability loading mtu*
Disable the redistribution of default information between IGRP processes. This is enabled by default.	**no default-information allowed** {**in** | **out**}

The metrics of one routing protocol do not necessarily translate into the metrics of another. For example, the RIP metric is a hop count and the IGRP metric is a combination of five quantities. In such situations, an artificial metric is assigned to the redistributed route. Because of this unavoidable tampering with dynamic information, carelessly exchanging routing information between different routing protocols can create routing loops, which can seriously degrade network operation.

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for examples of configuring redistribution and route maps.

Supported Metric Translations

This section describes supported automatic metric translations between the routing protocols. The following descriptions assume that you have not defined a default redistribution metric that replaces metric conversions.

- RIP can automatically redistribute static routes. It assigns static routes a metric of 1 (directly connected).
- EGP can automatically redistribute static routes and all dynamically derived routes. EGP assigns the metric 3 to all static and derived routes.
- BGP does not normally send metrics in its routing updates.
- IGRP can automatically redistribute static routes and information from other IGRP-routed autonomous systems. IGRP assigns static routes a metric that identifies them as directly connected. IGRP does not change the metrics of routes derived from IGRP updates from other autonomous systems.
- Note that any protocol can redistribute other routing protocols if a default metric is in effect.

Filter Routing Information

You can filter routing protocol information by performing the following tasks:

- Suppress the sending of routing updates on a particular router interface. This is done to prevent other systems on an interface from learning about routes dynamically.
- Suppress networks from being advertised in routing updates. This is done to prevent other routers from learning a particular router's interpretation of one or more routes.
- Suppress networks listed in updates from being accepted and acted upon by a routing process. This is done to keep a router from using certain routes.

- Filter on the source of routing information. This is done to prioritize routing information from different sources, because some pieces of routing information may be more accurate than others.
- Apply an offset to routing metrics. This is done to provide a local mechanism for increasing the value of routing metrics.

Note When routes are redistributed between OSPF processes, no OSPF metrics are preserved.

The following sections describe these tasks.

Suppress Routing Updates through an Interface

To prevent other routers on a local network from learning about routes dynamically, you can keep routing update messages from being sent through a router interface. This feature applies to all IP-based routing protocols except BGP and EGP.

OSPF and IS-IS behaviors are somewhat different. In OSPF, the interface address you specify as passive appears as a stub network in the OSPF domain. OSPF routing information is neither sent nor received through the specified router interface. In IS-IS, the specified IP addresses are advertised without actually running IS-IS on those interfaces.

To prevent routing updates through a specified interface, perform the following task in router configuration mode:

Task	Command
Suppress the sending of routing updates through the specified router interface.	**passive-interface** *interface*

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for examples of configuring passive interfaces.

Suppress Routes from Being Advertised in Routing Updates

To prevent other routers from learning one or more routes, you can suppress routes from being advertised in routing updates. You cannot specify an interface name in OSPF. When used for OSPF, this feature applies only to external routes.

To suppress routes from being advertised in routing updates, perform the following task in router configuration mode:

Task	Command
Permit or deny routes from being advertised in routing updates depending upon the action listed in the access list.	**distribute-list** *access-list-number* **out** [*interface-name* | *routing-process*]

Suppress Routes Listed in Updates from Being Processed

You might want to avoid processing certain routes listed in incoming updates. This feature does not apply to OSPF or IS-IS.

Perform this task in router configuration mode:

Task	Command
Suppress routes listed in updates from being processed.	**distribute-list** *access-list-number* **in** [*interface-name*]

Apply Offsets to Routing Metrics

An offset list is the mechanism for increasing incoming and outgoing metrics to routes learned via RIP and IGRP. Optionally, you can limit the offset list with either an access list or an interface. To increase the value of routing metrics, perform the following task in router configuration mode:

Task	Command
Apply an offset to routing metrics.	**offset-list** {**in** | **out**} *offset* [*access-list-number* | [*type number*]]

Filter Sources of Routing Information

An *administrative distance* is a rating of the trustworthiness of a routing information source, such as an individual router or a group of routers. In a large network, some routing protocols and some routers can be more reliable than others as sources of routing information. Also, when multiple routing processes are running in the same router for IP, it is possible for the same route to be advertised by more than one routing process. By specifying administrative distance values, you enable the router to intelligently discriminate between sources of routing information. The router will always pick the route whose routing protocol has the lowest administrative distance.

To filter sources of routing information, perform the following task in router configuration mode:

Task	Command
Filter on routing information sources.	**distance** *weight* [*address-mask* [*access-list-number*]] [**ip**]

For example, consider a router using IGRP and RIP. Suppose you trust the IGRP-derived routing information more than the RIP-derived routing information. In this example, because the default IGRP administrative distance is lower than the default RIP administrative distance, the router uses the IGRP-derived information and ignores the RIP-derived information. However, if you lose the source of the IGRP-derived information (to a power shutdown in another building, for example), the router uses the RIP-derived information until the IGRP-derived information reappears.

Note You also can use administrative distance to rate the routing information from routers running the same routing protocol. This application is generally discouraged if you are unfamiliar with this particular use of administrative distance, because it can result in inconsistent routing information, including forwarding loops.

Assigning administrative distances is a problem unique to each network and is done in response to the greatest perceived threats to the network. Even when general guidelines exist, the network manager must ultimately determine a reasonable matrix of administrative distances for the network as a whole. Table 18-3 shows the default administrative distance for various sources of routing information.

Table 18-3 Default Administrative Distances

Route Source	Default Distance
Connected interface	0
Static route	1
External BGP	20
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
Internal BGP	200
Unknown	255

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for examples of setting administrative distances.

Adjust Timers

Routing protocols use a variety of timers that determine such variables as the frequency of routing updates, the length of time before a route becomes invalid, and other parameters. You can adjust these timers to tune routing protocol performance to better suit your internetwork needs.

For IGRP and RIP, you can make the following timer adjustments:

- The rate (time in seconds between updates) at which routing updates are sent
- The interval of time (in seconds) after which a route is declared invalid
- The interval (in seconds) during which routing information regarding better paths is suppressed
- The amount of time (in seconds) that must pass before a route is removed from the routing table
- The amount of time for which routing updates will be postponed

EGP and BGP have their own **timers** commands, although some EGP timers might be set with the timers basic command. See the EGP and BGP sections, respectively.

It also is possible to tune the IP routing support in the software to enable faster convergence of the various IP routing algorithms and hence, quicker fallback to redundant routers. The total effect is to minimize disruptions to end users of the network in situations where quick recovery is essential.

The following two tables list tasks associated with adjusting routing protocol timers and the keepalive interval.

Perform the following task in router configuration mode:

Task	Command
Adjust routing protocol timers.	**timers basic** *update invalid holddown flush* [*sleeptime*]

Perform the following the following task in interface configuration mode:

Task	Command
Adjust the frequency with which the router sends messages to itself (Ethernet and Token Ring) or to the other end (HDLC-serial and PPP-serial links) to ensure that a network interface is alive for a specified interface.	**keepalive** [*seconds*][1]

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

You can also configure the *keepalive* interval, the frequency at which the router sends messages to itself (Ethernet and Token Ring) or to the other end (hdlc-serial, ppp-serial) to ensure that a network interface is alive. The interval in some previous software versions was 10 seconds; it is now adjustable in one-second increments down to one second. An interface is declared down after three update intervals have passed without receiving a keepalive packet.

When adjusting the keepalive timer for a very low bandwidth serial interface, large packets can delay the smaller keepalive packets long enough to cause the line protocol to go down. You might need to experiment to determine the best value.

Enable or Disable Split Horizon

Normally, routers that are connected to broadcast-type IP networks and that use distance-vector routing protocols employ the *split horizon* mechanism to reduce the possibility of routing loops. Split horizon blocks information about routes from being advertised by a router out any interface from which that information originated. This behavior usually optimizes communications among multiple routers, particularly when links are broken. However, with nonbroadcast networks, such as Frame Relay and SMDS, situations can arise for which this behavior is less than ideal. For these situations, you might want to disable split horizon. This applies to IGRP and RIP.

If an interface is configured with secondary IP addresses and split horizon is enabled, updates might not be sourced by every secondary address. One routing update is sourced per network number unless split horizon is disabled.

To enable or disable split horizon, perform the following tasks in interface configuration mode:

Task	Command
Enable split horizon.	**ip split-horizon**
Disable split horizon.	**no ip split-horizon**

Split horizon for Frame Relay and SMDS encapsulation is disabled by default. Split horizon is not disabled by default for interfaces using any of the X.25 encapsulations. For all other encapsulations, split horizon is enabled by default.

See the "IP Routing Protocol Configuration Examples" section at the end of this chapter for an example of using split horizon.

Note In general, changing the state of the default is not recommended unless you are certain that your application requires making a change in order to advertise routes properly. Remember: If split horizon is disabled on a serial interface (and that interface is attached to a packet-switched network), you *must* disable split horizon for all routers in any relevant multicast groups on that network.

Monitor and Maintain the IP Network

You can remove all contents of a particular cache, table, or database. You also can display specific router statistics. The following sections describe each of these tasks.

Clear Caches, Tables, and Databases

You can remove all contents of a particular cache, table, or database. Clearing a cache, table, or database can become necessary when the contents of the particular structure have become or are suspected to be invalid.

The following table lists the tasks associated with clearing caches, tables, and databases for IP routing protocols. Perform these tasks in EXEC mode:

Task	Command
Clear the IP ARP cache and the fast-switching cache.	**clear arp-cache**
Reset a particular BGP connection.	**clear ip bgp** *address*
Reset all BGP connections.	**clear ip bgp ***
Delete neighbors from the neighbor table.	**clear ip eigrp neighbors** [*ip-address* | *interface*]
Delete entries from the IGMP cache.	**clear ip igmp group** [*group-name* | *group-address* | *interface*]
Delete entries from the IP multicast routing table.	**clear ip mroute *** | {*group-name* [*source-address*] | *group-address* [*source-address*]}
Clear one or more routes from the IP routing table.	**clear ip route** {*network* [*mask*] | ***}

Display System and Network Statistics

You can display specific router statistics such as the contents of IP routing tables, caches, and databases. Information provided can be used to determine resource utilization and solve network problems. You can also display information about node reachability and discover the routing path your router's packets are taking through the network.

To display various router statistics, perform the following tasks in EXEC mode:

Task	Command
Trace a branch of a multicast tree for a specific group.	**mbranch** *group-address branch-address* [*ttl*]
Trace a branch of a multicast tree for a group in the reverse direction.	**mrbranch** *group-address branch-address* [*ttl*]
Display all BGP routes that contain subnet and supernet network masks.	**show ip bgp cidr-only**

Task	Command
Display routes that are matched by the specified autonomous system path access list.	**show ip bgp filter-list** *access-list-number*
Display routes that belong to the specified communities.	**show ip bgp community** *community-number* [**exact**]
Display routes that are permitted by the community list.	**show ip bgp community-list** *community-list-number* [**exact**]
Display the routes that match the specified regular expression entered on the command line.	**show ip bgp regexp** *regular-expression*
Display the contents of the BGP routing table.	**show ip bgp** [*network*] [*network-mask*] [**subnets**]
Display detailed information on the TCP and BGP connections to individual neighbors.	**show ip bgp neighbors** [*address*]
Display routes learned from a particular BGP neighbor.	**show ip bgp neighbors** *address* [**routes** | **paths**]
Display all BGP paths in the database.	**show ip bgp paths**
Display the status of all BGP connections.	**show ip bgp summary**
Display the entries in the DVMRP routing table.	**show ip dvmrp route** [*ip-address*]
Display statistics on EGP connections and neighbors.	**show ip egp**
Display the IP Enhanced IGRP discovered neighbors.	**show ip eigrp neighbors** [*interface unit*]
Display the IP Enhanced IGRP topology table for a given process.	**show ip eigrp topology** [*autonomous-system-number* | [[*ip-address*] *mask*]]
Display the number of packets sent and received for all or a specified IP Enhanced IGRP process.	**show ip eigrp traffic** [*autonomous-system-number*]
Display the multicast groups that are directly connected to the router and that were learned via IGMP.	**show ip igmp groups** [*group-name* | *group-address* | *interface*]
Display multicast-related information about an interface.	**show ip igmp interface** [*type number*]
Display IRDP values.	**show ip irdp**
Display the contents of the IP multicast routing table.	**show ip mroute** [*group-name* | *group-address*] [**summary**] [**count**] **show ip mroute** [*group-name* [*source-address*] | *group-address* [*source-address*]]
Display general information about OSPF routing processes in a particular router.	**show ip ospf** [*process-id*]

Task	Command
Display lists of information related to the OSPF database for a specific router.	**show ip ospf** [*process-id area-id*] **database** **show ip ospf** [*process-id area-id*] **database** [**router**] [*link-state-id*] **show ip ospf** [*process-id area-id*] **database** [**network**] [*link-state-id*] **show ip ospf** [*process-id area-id*] **database** [**summary**] [*link-state-id*] **show ip ospf** [*process-id area-id*] **database** [**asb-summary**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**external**] [*link-state-id*] **show ip ospf** [*process-id area-id*] **database** [**database-summary**]
Display OSPF-related interface information.	**show ip ospf interface** [*interface-name*]
Display OSPF-neighbor information on a per-interface basis.	**show ip ospf neighbor** [*interface-name*] [*neighbor-id*] **detail**
Display OSPF-related virtual links information.	**show ip ospf virtual-links**
Display information about interfaces configured for PIM.	**show ip pim interface** [*type number*]
List the PIM neighbors discovered by the router.	**show ip pim neighbor** [*type number*]
Display the RP routers associated with a sparse-mode multicast group.	**show ip pim rp** [*group-name* | *group-address*]
Display the parameters and current state of the active routing protocol process.	**show ip protocols**
Display the current state of the routing table.	**show ip route** [*address* [*mask*]] | [*protocol* [*process-id*]]
Display the current state of the routing table in summary form.	**show ip route summary**
Display supernets.	**show ip route supernets-only**
Display the IS-IS link state database.	**show isis database** [**level-1**] [**level-2**] [**l1**] [**l2**] [**detail**] [**lspid**]
Display all route maps configured or only the one specified.	**show route-map** [*map-name*]
Display the internal OSPF routing table entries to Area Border Router (ABR) and Autonomous System Boundary Router (ASBR).	**show ip ospf border-routers**

IP Routing Protocol Configuration Examples

The following sections provide IP routing protocol configuration examples:

- Variable-Length Subnet Masks Example
- Overriding Static Routes with Dynamic Protocols Example
- Configuring IS-IS as an IP Routing Protocol Example
- Static Routing Redistribution Example

- IGRP Redistribution Example
- RIP and IGRP Redistribution Example
- IP Enhanced IGRP Redistribution Examples
- RIP and IP Enhanced IGRP Redistribution Example
- IP Multicast Routing Configuration Examples
- OSPF Routing and Route Redistribution Examples
- BGP Route Advertisement and Redistribution Examples
- Default Metric Values Redistribution Example
- Route-Map Examples
- IGRP Feasible Successor Relationship Example
- BGP Synchronization Example
- BGP Basic Neighbor Specification Examples
- BGP Aggregate Route Examples
- Third-Party EGP Support Example
- Backup EGP Router Example
- EGP Core Gateway Example
- Autonomous System within EGP Example
- Passive Interface Examples
- Administrative Distance Examples
- Split Horizon Examples

Variable-Length Subnet Masks Example

OSPF, static routes, and IS-IS support variable-length subnet masks (VLSMs). With VLSMs, you can use different masks for the same network number on different interfaces, which allows you to conserve IP addresses and more efficiently use available address space.

In the following example, a 14-bit subnet mask is used, leaving two bits of address space reserved for serial line host addresses. There is sufficient host address space for two host endpoints on a point-to-point serial link.

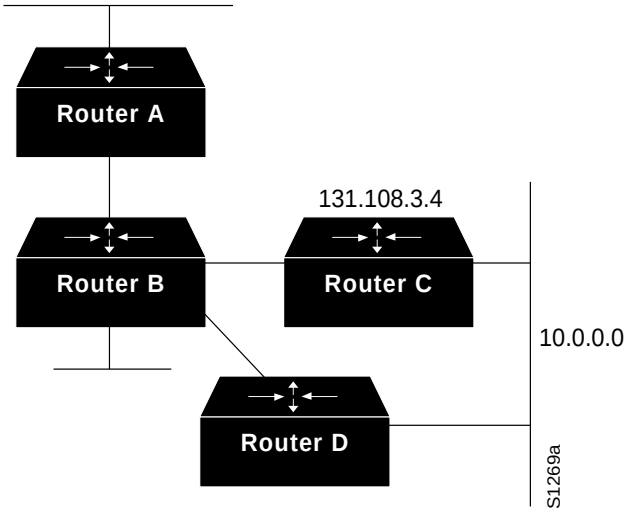
```
interface ethernet 0
ip address 131.107.1.1 255.255.255.0
! 8 bits of host address space reserved for ethernets

interface serial 0
ip address 131.107.254.1 255.255.255.252
! 2 bits of address space reserved for serial lines

! Router is configured for OSPF and assigned AS 107
router ospf 107
! Specifies network directly connected to the router
network 131.107.0.0 0.0.255.255 area 0.0.0.0
```

Overriding Static Routes with Dynamic Protocols Example

In the following example, packets for network 10.0.0.0 from Router B, where the static route is installed, will be routed through 131.108.3.4 if a route with an administrative distance less than 110 is not available. Figure 18-3 illustrates this point. The route learned by a protocol with an administrative distance of less than 110 might cause Router B to send traffic destined for network 10.0.0.0 via the alternate path—through Router D.

```
ip route 10.0.0.0 255.0.0.0 131.108.3.4 110
```

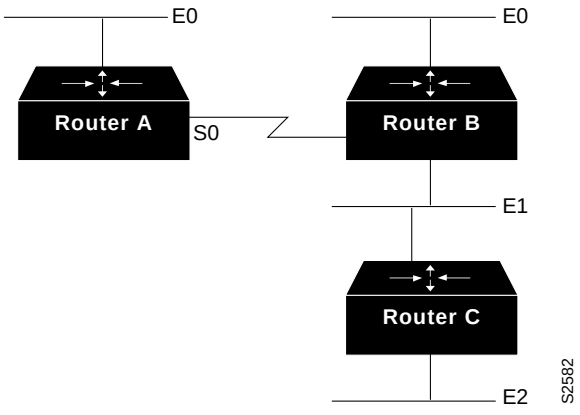
Figure 18-3 Overriding Static Routes



Configuring IS-IS as an IP Routing Protocol Example

The following example shows how you would configure three routers to run IS-IS as an IP routing protocol. Figure 18-4 illustrates the example configuration.

Figure 18-4 Illustration of IS-IS Routing



Configuration for Router A

```
router isis
net 49.0001.0000.0000.000a.00
interface e 0
ip router isis
interface s 0
ip router isis
```

Configuration for Router B

```
router isis
net 49.0001.0000.0000.000b.00
interface e 0
ip router isis
interface e 1
ip router isis
interface s 0
ip router isis
```

Configuration for Router C

```
router isis
net 49.0001.0000.0000.000c.00
interface e 1
ip router isis
interface e 2
ip router isis
```

Static Routing Redistribution Example

In the example that follows, three static routes are specified, two of which are to be advertised. Do this by specifying the **redistribute static** router configuration command, then specifying an access list that allows only those two networks to be passed to the IGRP process. Any redistributed static routes should be sourced by a single router to minimize the likelihood of creating a routing loop.

```
ip route 192.1.2.0 255.255.255.0 192.31.7.65
ip route 193.62.5.0 255.255.255.0 192.31.7.65
ip route 131.108.0.0 255.255.255.0 192.31.7.65
access-list 3 permit 192.1.2.0
access-list 3 permit 193.62.5.0
!
router igrp 109
network 192.31.7.0
default-metric 10000 100 255 1 1500
redistribute static
distribute-list 3 out static
```

IGRP Redistribution Example

Each IGRP routing process can provide routing information to only one autonomous system; the router must run a separate IGRP process and maintain a separate routing database for each autonomous system it services. However, you can transfer routing information between these routing databases.

Suppose the router has one IGRP routing process for network 15.0.0.0 in autonomous system 71 and another for network 192.31.7.0 in autonomous system 109, as the following commands specify:

```
router igrp 71
network 15.0.0.0
```

```
router igrp 109
network 192.31.7.0
```

To transfer a route to 192.31.7.0 into autonomous system 71 (without passing any other information about autonomous system 109), use the command in the following example:

```
router igrp 71
redistribute igrp 109
distribute-list 3 out igrp 109
access-list 3 permit 192.31.7.0
```

RIP and IGRP Redistribution Example

Consider a WAN at a university that uses RIP as an interior routing protocol. Assume that the university wants to connect its wide area network to a regional network, 128.1.0.0, which uses IGRP as the routing protocol. The goal in this case is to advertise the networks in the university network to the routers on the regional network. The commands for the interconnecting router are listed in the example that follows:

```
router igrp 109
network 128.1.0.0
redistribute rip
default-metric 10000 100 255 1 1500
distribute-list 10 out rip
```

In this example, the **router** global configuration command starts an IGRP routing process. The **network** router configuration command specifies that network 128.1.0.0 (the regional network) is to receive IGRP routing information. The **redistribute** router configuration command specifies that RIP-derived routing information be advertised in the routing updates. The **default-metric** router configuration command assigns an IGRP metric to all RIP-derived routes.

The **distribute-list** router configuration command instructs the router to use access list 10 (not defined in this example) to limit the entries in each outgoing update. The access list prevents unauthorized advertising of university routes to the regional network.

IP Enhanced IGRP Redistribution Examples

Each IP Enhanced IGRP routing process can provide routing information to only one autonomous system; the router must run a separate IP Enhanced IGRP process and maintain a separate routing database for each autonomous system it services. However, you can transfer routing information between these routing databases.

Suppose the router has one IP Enhanced IGRP routing process for network 15.0.0.0 in autonomous system 71 and another for network 192.31.7.0 in autonomous system 109, as the following commands specify:

```
router eigrp 71
network 15.0.0.0
router eigrp 109
network 192.31.7.0
```

To transfer a route from 192.31.7.0 into autonomous system 71 (without passing any other information about autonomous system 109), use the command in the following example:

```
router eigrp 71
redistribute eigrp 109 route-map 109-to-71
route-map 109-to-71 permit
match ip address 3
set metric 10000 100 1 255 1500
access-list 3 permit 192.31.7.0
```

The following example is an alternative way to transfer a route to 192.31.7.0 into autonomous system 71. Unlike the previous configuration, this one does not allow you to arbitrarily set the metric.

```
router eigrp 71
redistribute eigrp 109
distribute-list 3 out eigrp 109
access-list 3 permit 192.31.7.0
```

RIP and IP Enhanced IGRP Redistribution Example

This sections provides two examples of RIP and IP Enhanced IGRP redistribution, a simple one and a complex one.

Example 1: Simple Redistribution

Consider a wide-area network at a university that uses RIP as an interior routing protocol. Assume that the university wants to connect its wide-area network to a regional network, 128.1.0.0, which uses IP Enhanced IGRP as the routing protocol. The goal in this case is to advertise the networks in the university network to the routers on the regional network. The commands for the interconnecting router are listed in the example that follows:

```
router eigrp 109
network 128.1.0.0
redistribute rip
default-metric 10000 100 255 1 1500
distribute-list 10 out rip
```

In this example, the **router** global configuration command starts an IP Enhanced IGRP routing process. The **network** router configuration command specifies that network 128.1.0.0 (the regional network) is to send and receive IP Enhanced IGRP routing information. The **redistribute** router configuration command specifies that RIP-derived routing information be advertised in the routing updates. The **default-metric** router configuration command assigns an IP Enhanced IGRP metric to all RIP-derived routes.

The **distribute-list** router configuration command instructs the router to use access list 10 (not defined in this example) to limit the entries in each outgoing update. The access list prevents unauthorized advertising of university routes to the regional network.

Example 2: Complex Redistribution

The most complex redistribution case is one in which *mutual* redistribution is required between an IGP (in this case IP Enhanced IGRP) and BGP.

Suppose that BGP is running on a router somewhere else in autonomous system 1, and that the BGP routes are injected into IP Enhanced IGRP routing process 1. You must use filters to ensure that the proper routes are advertised. The example configuration for router R1 illustrates use of access filters and a distribution list to filter routes advertised to BGP neighbors. This example also illustrates configuration commands for redistribution between BGP and IP Enhanced IGRP.

```
! Configuration for router R1:
router bgp 1
network 131.108.0.0
neighbor 192.5.10.1 remote-as 2
neighbor 192.5.10.15 remote-as 1
neighbor 192.5.10.24 remote-as 3
redistribute eigrp 1
```

```
distribute-list 1 out eigrp 1
!
! All networks that should be advertised from R1 are controlled with access lists:
!
access-list 1 permit 131.108.0.0
access-list 1 permit 150.136.0.0
access-list 1 permit 128.125.0.0
!
router eigrp 1
network 131.108.0.0
network 192.5.10.0
redistribute bgp 1
```

IP Multicast Routing Configuration Examples

This section provides the following IP multicast routing configuration examples:

- Configure a Router to Operate in Dense Mode Example
- Configure a Router to Operate in Sparse Mode Example
- Configure DVMRP Interoperability Examples

Configure a Router to Operate in Dense Mode Example

The following example configures dense-mode PIM on an Ethernet interface of the router:

```
ip multicast-routing
interface ethernet 0
ip pim dense-mode
```

Configure a Router to Operate in Sparse Mode Example

The following example configures the router to operate in sparse-mode PIM. The RP router is the router whose address is 10.8.0.20.

```
access-list 1 permit 224.2.0.1
!
ip multicast-routing
ip pim rp-address 10.8.0.20 1
interface ethernet 1
ip pim sparse-mode
```

Configure DVMRP Interoperability Examples

The following example configures DVMRP interoperability for configurations when the PIM router and the DVMRP router are on the same network segment. In this example, access list 1 advertises the networks (98.92.35.0,198.92.36.0, 198.92.37.0,131.108.0.0, and 150.136.0.0 to the DVMRP router, and access list 2 is used to prevent all other networks from being advertised (**ip dvmrp metric 0**).

```
interface ethernet 0
ip address 131.119.244.244 255.255.255.0
ip pim dense-mode
ip dvmrp metric 1 1
ip dvmrp metric 0 2

access-list 1 permit 198.92.35.0 0.0.0.255
access-list 1 permit 198.92.36.0 0.0.0.255
access-list 1 permit 198.92.37.0 0.0.0.255
access-list 1 permit 131.108.0.0 0.0.255.255
```

```
access-list 1 permit 150.136.0.0 0.0.255.255
access-list 1 deny   0.0.0.0 255.255.255.255
access-list 2 permit 0.0.0.0 255.255.255.255
```

The following example configures DVMRP interoperability over a tunnel interface.

```
hostname Mbone(ACOnet)
!
boot system xx-k.Aug11 192.76.243.9
boot system flash
boot system rom
!
ip multicast-routing
!
interface tunnel 0
no ip address
ip pim dense-mode
tunnel source Ethernet0
tunnel destination 192.70.92.133
tunnel mode dvmrp
!
interface Ethernet0
description Universitat DMZ-ethernet
ip address 193.171.23.23 255.255.255.240 secondary
ip address 192.76.243.2 255.255.255.0
ip pim dense-mode
!
router igrp 11853
network 192.76.243.0
network 193.171.23.0
```

OSPF Routing and Route Redistribution Examples

OSPF typically requires coordination among many internal routers, area border routers, and autonomous system boundary routers. At a minimum, OSPF-based routers can be configured with all default parameter values, with no authentication, and with interfaces assigned to areas.

Three examples follow:

- The first is a simple configuration illustrating basic OSPF commands.
- The second example illustrates a configuration for internal, area border, and autonomous system boundary routers within a single, arbitrarily assigned, OSPF autonomous system.
- The third example illustrates a more complex configuration and the application of various tools available for controlling OSPF-based routing environments.

Example 1: Basic OSPF Configuration

The following example illustrates a simple OSPF configuration that enables OSPF routing process 9000, attaches Ethernet 0 to area 0.0.0.0, and redistributes RIP into OSPF, and OSPF into RIP:

```
interface Ethernet0
ip address 130.93.1.1 255.255.255.0
ip ospf cost 1
!
interface Ethernet 1
ip address 130.94.1.1 255.255.255.0
!
router ospf 9000
network 130.93.0.0 0.0.255.255 area 0.0.0.0
redistribute rip metric 1 subnets
!
```

```
router rip
network 130.94.0.0
redistribute ospf 9000
default-metric 1
```

Example 2: Another Basic OSPF Configuration

The following example illustrates the assignment of four area IDs to four IP address ranges. In the example, OSPF routing process 109 is initialized, and four OSPF areas are defined: 10.9.50.0, 2, 3, and 0. Areas 10.9.50.0, 2, and 3 mask specific address ranges, while Area 0 enables OSPF for *all other* networks.

```
router ospf 109
network 131.108.20.0  0.0.0.255 area 10.9.50.0
network 131.108.0.0  0.0.255.255 area 2
network 131.109.10.0  0.0.0.255 area 3
network 0.0.0.0  255.255.255.255 area 0
!
! Interface Ethernet0 is in area 10.9.50.0:
interface Ethernet 0
ip address 131.108.20.5 255.255.255.0
!
! Interface Ethernet1 is in area 2:
interface Ethernet 1
ip address 131.108.1.5 255.255.255.0
!
! Interface Ethernet2 is in area 2:
interface Ethernet 2
ip address 131.108.2.5 255.255.255.0
!
! Interface Ethernet3 is in area 3:
interface Ethernet 3
ip address 131.109.10.5 255.255.255.0
!
! Interface Ethernet4 is in area 0:
interface Ethernet 4
ip address 131.109.1.1 255.255.255.0
!
! Interface Ethernet5 is in area 0:
interface Ethernet 5
ip address 10.1.0.1 255.255.0.0
```

Each **network** router configuration command is evaluated sequentially, so the specific order of these commands in the configuration is important. The router sequentially evaluates the *address*/*wildcard-mask* pair for each interface. See the "IP Routing Protocols Commands" chapter of the Router Products Command Reference for more information.

Consider the first **network** command. Area ID 10.9.50.0 is configured for the interface on which subnet 131.108.20.0 is located. Assume that a match is determined for interface Ethernet 0. Interface Ethernet 0 is attached to Area 10.9.50.0 only.

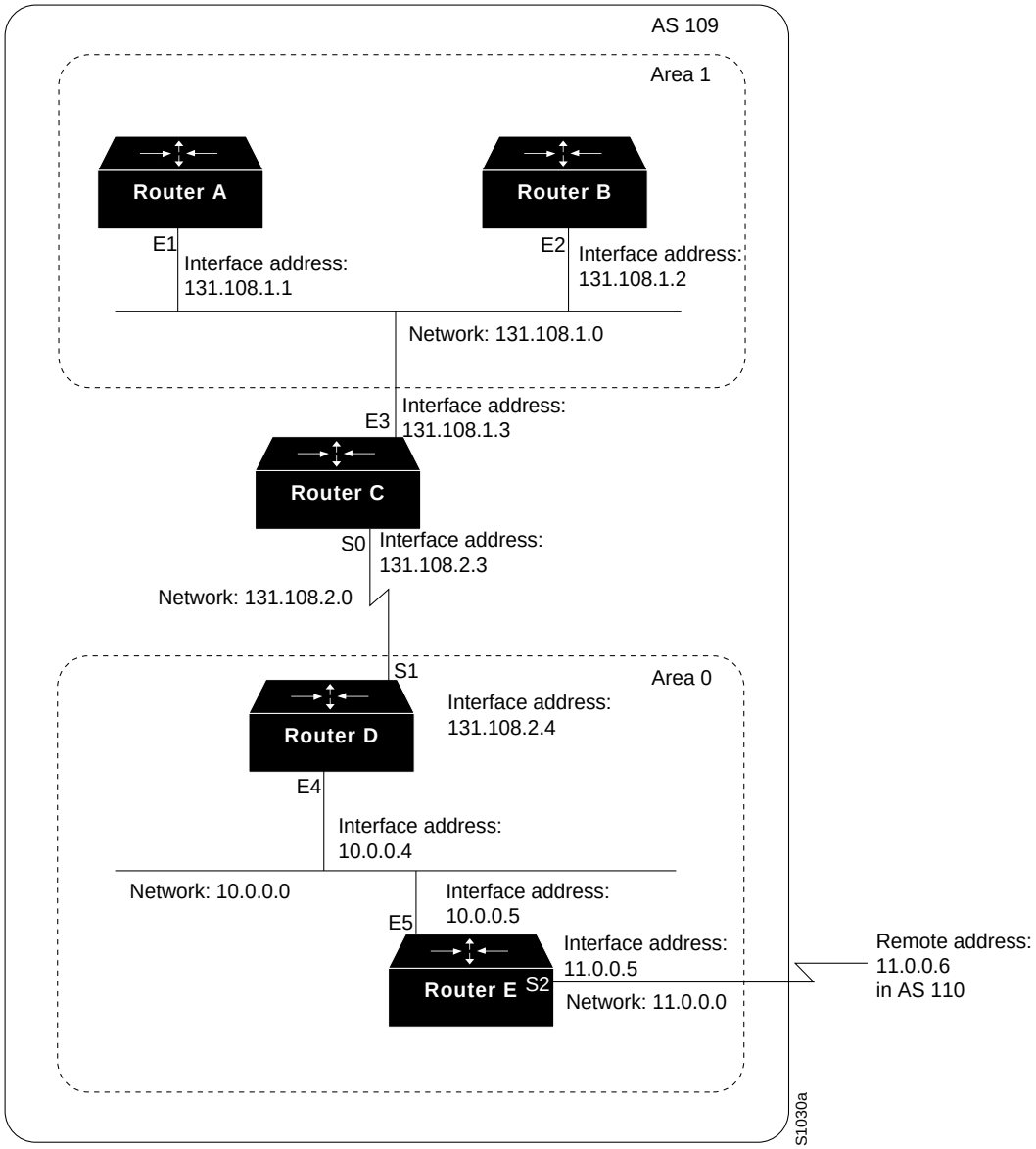
The second **network** command is evaluated next. For Area 2, the same process is then applied to all interfaces (except interface Ethernet 0). Assume that a match is determined for interface Ethernet 1. OSPF is then enabled for that interface and Ethernet 1 is attached to Area 2.

This process of attaching interfaces to OSPF areas continues for all **network** commands. Note that the last **network** command in this example is a special case. With this command all available interfaces (not explicitly attached to another area) are attached to Area 0.

Example 3: Internal, Area Border, and Autonomous System Boundary Routers

The following example outlines a configuration for several routers within a single OSPF autonomous system. Figure 18-5 provides a general network map that illustrates this example configuration.

Figure 18-5 Sample OSPF Autonomous System Network Map



In this configuration, five routers are configured in OSPF autonomous system 109:

- Router A and Router B are both internal routers within Area 1.
- Router C is an OSPF area border router; note that for Router C, Area 1 is assigned to E3 and Area 0 is assigned to S0.

- Router D is an internal router in Area 0 (backbone area); in this case, both **network** router configuration commands specify the same area (Area 0, or the backbone area).
- Router E is an OSPF autonomous system boundary router; note that BGP routes are redistributed into OSPF and that these routes are advertised by OSPF.

Note It is not necessary to include definitions of all areas in an OSPF autonomous system in the configuration of all routers in the autonomous system. You need only define the *directly* connected areas. In the example that follows, routes in Area 0 are learned by the routers in Area 1 (Router A and Router B) when the area border router (Router C) injects summary link state advertisements (LSAs) into Area 1.

Autonomous System 109 is connected to the outside world via the BGP link to the external peer at IP address 11.0.0.6.

Configuration for Router A - Internal Router

```
interface Ethernet 1
ip address 131.108.1.1 255.255.255.0

router ospf 109
network 131.108.0.0 0.0.255.255 area 1
```

Configuration for Router B - Internal Router

```
interface Ethernet 2
ip address 131.108.1.2 255.255.255.0

router ospf 109
network 131.108.0.0 0.0.255.255 area 1
```

Configuration for Router C - Area Border Router

```
interface Ethernet 3
ip address 131.108.1.3 255.255.255.0

interface Serial 0
ip address 131.108.2.3 255.255.255.0

router ospf 109
network 131.108.1.0 0.0.0.255 area 1
network 131.108.2.0 0.0.0.255 area 0
```

Configuration for Router D - Internal Router

```
interface Ethernet 4
ip address 10.0.0.4 255.0.0.0

interface Serial 1
ip address 131.108.2.4 255.255.255.0

router ospf 109
network 131.108.2.0 0.0.0.255 area 0
network 10.0.0.0 0.255.255.255 area 0
```

Configuration for Router E - Autonomous System Boundary Router

```
interface Ethernet 5
ip address 10.0.0.5 255.0.0.0

interface Serial 2
ip address 11.0.0.5 255.0.0.0

router ospf 109
network 10.0.0.0 0.255.255.255 area 0
redistribute bgp 109 metric 1 metric-type 1

router bgp 109
network 131.108.0.0
network 10.0.0.0
neighbor 11.0.0.6 remote-as 110
```

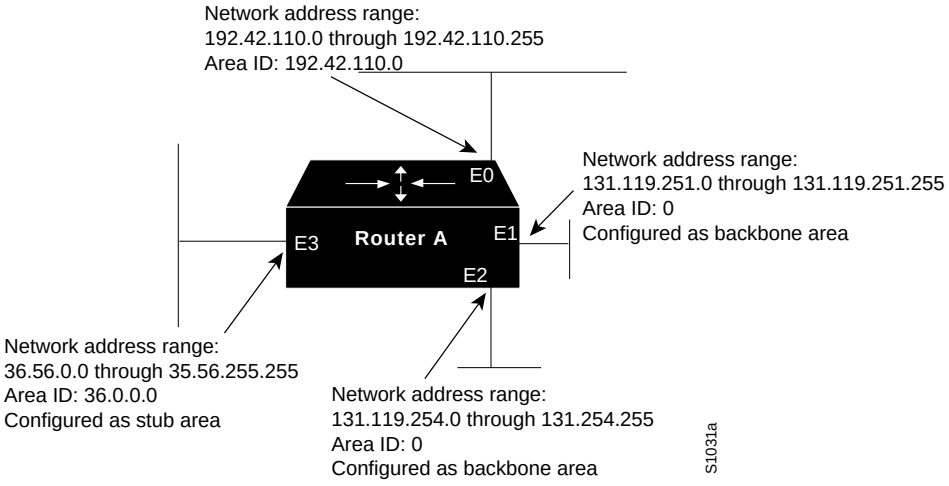
Example 4: Complex OSPF Configuration

The following example configuration accomplishes several tasks in setting up an area border router. These tasks can be split into two general categories:

- Basic OSPF configuration
- Route redistribution

The specific tasks outlined in this configuration are detailed briefly in the following descriptions. Figure 18-6 illustrates the network address ranges and area assignments for the interfaces.

Figure 18-6 Interface and Area Specifications for OSPF Example Configuration



The basic configuration tasks in this example are as follows:

- Configure address ranges for Ethernet 0 through Ethernet 3 interfaces.
- Enable OSPF on each interface.
- Set up an OSPF authentication password for each area and network.
- Assign link state metrics and other OSPF interface configuration options.

- Create a *stub area* with area id 36.0.0.0. (Note that the **authentication** and **stub** options of the **area** router configuration command are specified with separate **area** command entries, but can be merged into a single **area** command.)
- Specify the backbone area (Area 0).

Configuration tasks associated with redistribution are as follows:

- Redistribute IGRP and RIP into OSPF with various options set (including **metric-type**, **metric**, **tag**, and **subnet**).
- Redistribute IGRP and OSPF into RIP.

The following is an example OSPF configuration:

```
interface Ethernet0
ip address 192.42.110.201 255.255.255.0
ip ospf authentication-key abcdefgh
ip ospf cost 10
!
interface Ethernet1
ip address 131.119.251.201 255.255.255.0
ip ospf authentication-key ijklmnop
ip ospf cost 20
ip ospf retransmit-interval 10
ip ospf transmit-delay 2
ip ospf priority 4
!
interface Ethernet2
ip address 131.119.254.201 255.255.255.0
ip ospf authentication-key abcdefgh
ip ospf cost 10
!
interface Ethernet3
ip address 36.56.0.201 255.255.0.0
ip ospf authentication-key ijklmnop
ip ospf cost 20
ip ospf dead-interval 80
```

OSPF is on network 131.119:

```
router ospf 201
network 36.0.0.0 0.255.255.255 area 36.0.0.0
network 192.42.110.0 0.0.0.255 area 192.42.110.0
network 131.119.0.0 0.0.255.255 area 0
area 0 authentication
area 36.0.0.0 stub
area 36.0.0.0 authentication
area 36.0.0.0 default-cost 20
area 192.42.110.0 authentication
area 36.0.0.0 range 36.0.0.0 255.0.0.0
area 192.42.110.0 range 192.42.110.0 255.255.255.0
area 0 range 131.119.251.0 255.255.255.0
area 0 range 131.119.254.0 255.255.255.0

redistribute igrp 200 metric-type 2 metric 1 tag 200 subnets
redistribute rip metric-type 2 metric 1 tag 200
```

IGRP autonomous system 200 is on 131.119.0.0:

```
router igrp 200
network 131.119.0.0
!
! RIP for 192.42.110
!
```

```
router rip
network 192.42.110.0
redistribute igrp 200 metric 1
redistribute ospf 201 metric 1
```

BGP Route Advertisement and Redistribution Examples

The following examples illustrate configurations for advertising and redistributing BGP routes. The first example details the configuration for two neighboring routers that run IGRP within their respective autonomous systems and that are configured to advertise their respective BGP routes between each other. The second example illustrates route redistribution of BGP into IGRP and IGRP into BGP.

Example 1: Simple BGP Route Advertisement

This example provides the required configuration for two routers (R1 and R2) that are intended to advertise BGP routes to each other and to redistribute BGP into IGRP.

Configuration for Router R1

```
! Assumes autonomous system 1 has network number 131.108.0.0
router bgp 1
network 131.108.0.0
neighbor 192.5.10.1 remote-as 2
!
router igrp 1
network 131.108.0.0
network 192.5.10.0
redistribute bgp 1
! Note that IGRP is not redistributed into BGP
```

Configuration for Router R2

```
router bgp 2
network 150.136.0.0
neighbor 192.5.10.2 remote-as 1
!
router igrp 2
network 150.136.0.0
network 192.5.10.0
redistribute bgp 2
```

Example 2: Mutual Route Redistribution

The most complex redistribution case is one in which *mutual* redistribution is required between an IGP (in this case IGRP) and BGP.

Suppose that EGP is running on a router somewhere else in autonomous system 1, and that the EGP routes are injected into IGRP routing process 1. You must filter to ensure that the proper routes are advertised. The example configuration for router R1 illustrates use of access filters and a distribution list to filter routes advertised to BGP neighbors. This example also illustrates configuration commands for redistribution between BGP and IGRP. Only routes learned using the EBGP session with neighbors 192.5.10.1 and 192.5.10.24 are redistributed into IGRP.

Configuration for Router R1

```
router bgp 1
network 131.108.0.0
neighbor 192.5.10.1 remote-as 2
! External peer or neighbor
neighbor 192.5.10.15 remote-as 1
! Same AS; therefore internal neighbor
neighbor 192.5.10.24 remote-as 3
! A second External neighbor
redistribute igrp 1
distribute-list 1 out igrp 1
!
! All networks that should be
! advertised from R1 are
! controlled with access lists:
!
access-list 1 permit 131.108.0.0
access-list 1 permit 150.136.0.0
access-list 1 permit 128.125.0.0
!
router igrp 1
network 131.108.0.0
network 192.5.10.0
redistribute bgp 1
```

Default Metric Values Redistribution Example

The following example shows a router in autonomous system 109 using both the RIP and the IGRP routing protocols. The example advertises IGRP-derived routes using the RIP protocol and assigns the IGRP-derived routes a RIP metric of 10.

```
router rip
default-metric 10
redistribute igrp 109
```

Route-Map Examples

The examples in this section illustrate the use of redistribution, with and without route maps. Examples from both the IP and CLNS routing protocols are given.

The following example redistributes all OSPF routes into IGRP:

```
router igrp 109
redistribute ospf 110
```

The following example redistributes RIP routes with a hop count equal to 1 into OSPF. These routes will be redistributed into OSPF as external link state advertisements with a metric of 5, metric type of Type 1, and a tag equal to 1.

```
router ospf 109
redistribute rip route-map rip-to-ospf
!
route-map rip-to-ospf permit
match metric 1
set metric 5
set metric-type type1
set tag 1
```

The following example redistributes OSPF learned routes with tag 7 as a RIP metric of 15:

```
router rip
redistribute ospf 109 route-map 5
!
route-map 5 permit
match tag 7
set metric 15
```

The following example redistributes OSPF intra-area and interarea routes with next-hop routers on interface serial 0 into BGP with an INTER_AS metric of 5:

```
router bgp 109
redistribute ospf 109 route-map 10
!
route-map 10 permit
match route-type internal
match interface serial 0
set metric 5
```

The following example redistributes two types of routes into the integrated IS-IS routing table (supporting both IP and CLNS). The first are OSPF external IP routes with tag 5; these are inserted into Level 2 IS-IS LSPs with a metric of 5. The second are ISO-IGRP derived CLNS prefix routes that match CLNS access list 2000. These will be redistributed into IS-IS as Level 2 LSPs with a metric of 30.

```
router isis
redistribute ospf 109 route-map 2
redistribute iso-igrp nsfnet route-map 3
!
route-map 2 permit
match route-type external
match tag 5
set metric 5
set level level-2
!
route-map 3 permit
match address 2000
set metric 30
```

With the following configuration, OSPF external routes with tags 1, 2, 3, and 5 are redistributed into RIP with metrics of 1, 1, 5, and 5, respectively. The OSPF routes with a tag of 4 are not redistributed.

```
router rip
redistribute ospf 109 route-map 1
!
route-map 1 permit
match tag 1 2
set metric 1
!
route-map 1 permit
match tag 3
set metric 5
!
route-map 1 deny
match tag 4
!
route map 1 permit
match tag 5
set metric 5
```

The following configuration sets the condition that if there is an OSPF route to network 140.222.0.0, generate the default network 0.0.0.0 into RIP with a metric of 1:

```
router rip
redistribute ospf 109 route-map default
!
route-map default permit
match ip address 1
set metric 1
!
access-list 1 permit 140.222.0.0 0.0.255.255
access-list 2 permit 0.0.0.0 0.0.0.0
```

Given the following configuration, a RIP learned route for network 160.89.0.0 and an ISO-IGRP learned route with prefix 49.0001.0002 will be redistributed into an IS-IS Level 2 LSP with a metric of 5:

```
router isis
redistribute rip route-map 1
redistribute iso-igrp remote route-map 1
!
route-map 1 permit
match ip address 1
match clns address 2
set metric 5
set level level-2
!
access-list 1 permit 160.89.0.0 0.0.255.255
clns filter-set 2 permit 49.0001.0002...
```

The following configuration example illustrates how a route map is referenced by the **default-information** router configuration command. This is called *conditional default origination.* OSPF will originate the default route (network 0.0.0.0) with a Type 2 metric of 5 if 140.222.0.0, with network 0.0.0.0 in the routing table.

```
route-map ospf-default permit
match ip address 1
set metric 5
set metric-type type-2
!
access-list 1 140.222.0.0 0.0.255.255
!
router ospf 109
default-information originate route-map ospf-default
```

Using Route Maps with BGP

The following example shows how you can use route maps to modify incoming data from a neighbor. Any route received from 140.222.1.1 that matches the filter parameters set in autonomous system access list 200 will have its weight set to 200 and its local preference set to 250 and will be accepted.

```
router bgp 100
!
neighbor 140.222.1.1 route-map fix-weight in
neighbor 140.222.1.1 remote-as 1
!
route-map fix-weight permit 10
match as-path 200
set local-preference 250
set weight 200
!
```

```
ip as-path access-list 200 permit ^690$
ip as-path access-list 200 permit ^1800
```

The following example shows how you can use route maps to modify outbound data to a neighbor:

```
router bgp 100
neighbor 198.92.68.23 route-map oscar out
!
route-map oscar
set metric 150
match as-path 1
!
ip as-path access-list 1 permit ^2200_
```

In the following example, route map freddy marks all paths originating from autonomous system 690 with a multiple exit discriminator (MULTI_EXIT_DISC) metric attribute of 127. The second permit clause is required so that routes not matching autonomous system path list 1 will still be accepted from neighbor 1.1.1.1.

```
router bgp 100
neighbor 1.1.1.1 route-map freddy in
!
ip as-path access-list 1 permit ^690_
ip as-path access-list 2 permit .*
!
route-map freddy permit 10
match as-path 1
set metric 127
!
route-map freddy permit 20
match as-path 2
```

The following example shows how you can use route maps to modify incoming data from the IP forwarding table:

```
router bgp 100
redistribute igrp 109 route-map igrp2bgp
!
route-map igrp2bgp
match ip address 1
set local-preference 25
set metric 127
set weight 30000
set next-hop 192.92.68.24
set origin igp
!
access-list 1 permit 131.108.0.0 0.0.255.255
access-list 1 permit 160.89.0.0 0.0.255.255
access-list 1 permit 198.112.0.0 0.0.127.255
```

It is proper behavior to not accept any autonomous system path not matching the **match** clause of the route map. This means that you will not set the metric and the router will not accept the route. However, you can configure the router to accept autonomous system paths not matched in the **match** clause of the route map command by using multiple maps of the same name, some without accompanying **set** commands.

```
route-map fnord permit 10
match as-path 1
set local-preference 5
!
route-map fnord permit 20
match as-path 2
```

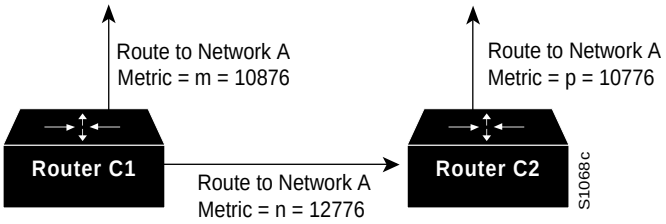
The following example shows how you can use route maps in a reverse operation to set the route tag (as defined by the BGP/OSPF interaction document, RFC 1403) when exporting routes from BGP into the main IP routing table:

```
router bgp 100
table-map set_ospf_tag
!
route-map set_ospf_tag
match as-path 1
set automatic-tag
!
ip as-path access-list 1 permit .*
```

IGRP Feasible Successor Relationship Example

In Figure 18-7, the assigned metrics meet the conditions required for a feasible successor relationship, so the paths in this example can be included in routing tables and used for load balancing.

Figure 18-7 Assigning Metrics for IGRP Path Feasibility



The feasibility test would work as follows:

Assume that Router C1 already has a route to Network A with metric m and has just received an update about Network A from C2. The best metric at C2 is p. The metric that C1 would use through C2 is n.

If both of the following two conditions are met, the route to network A through C2 will be included in C1's routing table:

- If *m* is greater than *p*.
- If the *multiplier* (value specified by the **variance** router configuration command) times *m* is greater than or equal to *n*.

The configuration for Router C1 would be as follows:

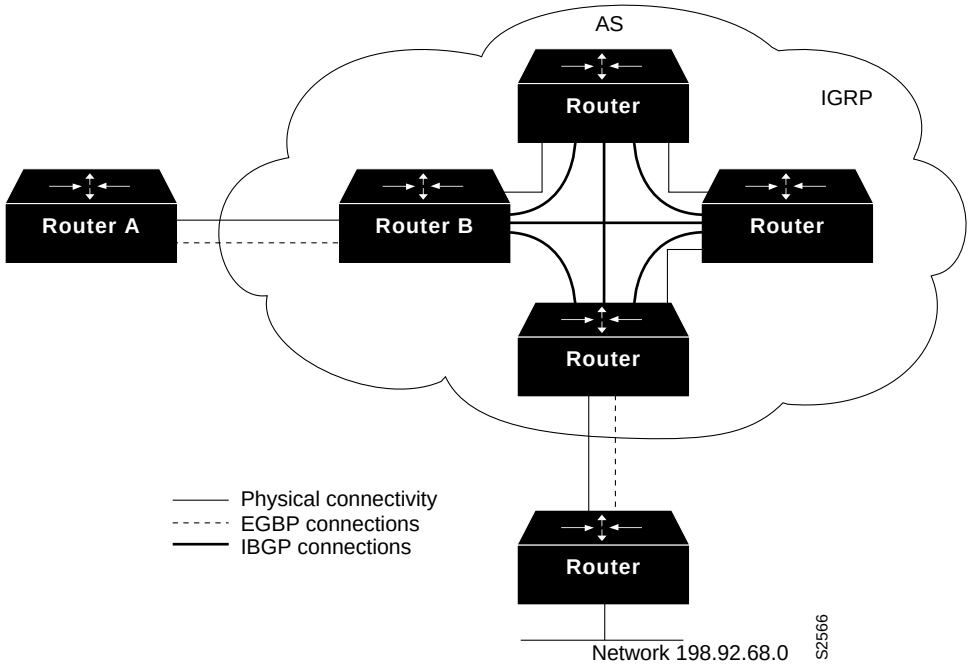
```
router igrp 109
variance 10
```

A maximum of four paths can be in the routing table for a single destination. If there are more than four feasible paths, the four best feasible paths are used.

BGP Synchronization Example

In the configuration shown in Figure 18-8, with synchronization on, Router B does not advertise network 10.0.0.0 to Router A until an IGRP route for network 10.0.0.0 exists. If you specify the **no synchronization** router configuration command, Router B advertises network 10.0.0.0 as soon as possible. However, because routing information still must be sent to interior peers, you must configure a full internal BGP mesh.

Figure 18-8 BGP Synchronization Configuration



BGP Basic Neighbor Specification Examples

The following example specifies that the router at the address 131.108.1.2 is a neighbor in autonomous system number 109.

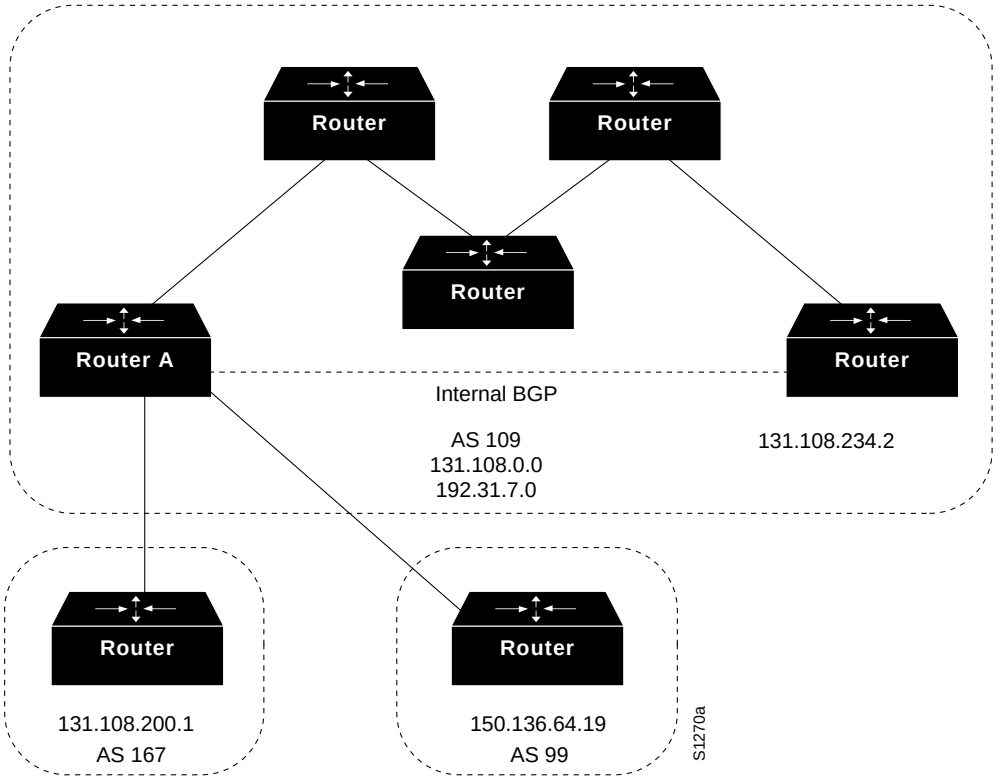
```
neighbor 131.108.1.2 remote-as 109
```

In the following example, a BGP router is assigned to autonomous system 109, and two networks are listed as originating in the autonomous system. Then the addresses of three remote routers (and their autonomous systems) are listed. The router being configured will share information about networks 131.108.0.0 and 192.31.7.0 with the neighbor routers. The first router listed is in the same Class B network, but in a different autonomous system; the second **neighbor** router configuration command illustrates specification of an internal neighbor (with the same autonomous system number) at address 131.108.234.2; and the last **neighbor** command specifies a neighbor on a different network.

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1 remote-as 167
neighbor 131.108.234.2 remote-as 109
neighbor 150.136.64.19 remote-as  99
```

In Figure 18-9, Router A is being configured. The internal BGP neighbor is not directly linked to Router A. External neighbors (in autonomous system 167 and autonomous system 99) must be linked directly to Router A.

Figure 18-9 Assigning Internal and External BGP Neighbors



Using Access Lists to Specify Neighbors

In the following example, the router is configured to allow connections from any router that has an IP address in access list 1; that is, any router with a 192.31.7.x address. Neighbors not explicitly specified as neighbors can connect to the router, but the router will not attempt to connect to them if the connection is broken. Continuing with the preceding sample configuration, the configuration is as follows:

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  remote-as 167
neighbor 131.108.234.2 remote-as 109
neighbor 150.136.64.19  remote-as  99
neighbor internal-ethernet neighbor-list 1
access-list 1 permit 192.31.7.0 0.0.0.255
```

BGP Aggregate Route Examples

The following examples show how you can use aggregate routes in BGP either by redistributing an aggregate route into BGP or by using the conditional aggregate routing feature.

In the following example, the **redistribute static** command is used to redistribute aggregate route 193.*.*.*:

```
ip route 193.0.0.0 255.0.0.0 null 0
!
router bgp 100
redistribute static
network 193.0.0.0 255.0.0.0
! this marks route as not incomplete
```

The following configuration creates an aggregate entry in the BGP routing table when there are specific routes that fall into the specified range. The aggregate route will be advertised as coming from your autonomous system and has the atomic aggregate attribute set to show that information might be missing. (By default, atomic aggregate is set unless you use the **as-set** keyword in the **aggregate-address** command.)

```
router bgp 100
aggregate-address 193.0.0.0 255.0.0.0
```

The following example creates an aggregate entry using the same rules as in the previous example, but the path advertised for this route will be an AS_SET consisting of all elements contained in all paths that are being summarized:

```
router bgp 100
aggregate-address 193.0.0.0 255.0.0.0 as-set
```

The following example not only creates the aggegate route for 193.*.*.*, but will also suppress advertisements of more specific routes to all neighbors:

```
router bgp 100
aggegate-address 193.0.0.0 255.0.0.0 summary-only
```

Third-Party EGP Support Example

In the following example configuration, the router is in autonomous system 110 communicating with an EGP neighbor in autonomous system 109 with address 131.108.6.5. Network 131.108.0.0 is advertised as originating within autonomous system 110. The configuration specifies that two routers, 131.108.6.99 and 131.108.6.100, should be advertised as third-party sources of routing information for those networks that are accessible through those routers. The global configuration commands also specify that those networks should be flagged as internal to autonomous system 110.

```
autonomous-system 110
router egp 109
network 131.108.0.0
neighbor 131.108.6.5
neighbor 131.108.6.5  third-party 131.108.6.99 internal
neighbor 131.108.6.5  third-party 131.108.6.100 internal
```

Backup EGP Router Example

The following example configuration illustrates a router that is in autonomous system 110 communicating with an EGP neighbor in autnomous system 109 with address 131.108.6.5. Network 131.108.0.0 is advertised with a distance of 1, and networks learned by RIP are being advertised with a distance of 5. Access list 3 filters which RIP-derived networks are allowed in outgoing EGP updates.

```
autonomous-system 110
router egp 109
network 131.108.0.0
neighbor 131.108.6.5
redistribute rip
default-metric 5
distribute-list 3 out rip
```

EGP Core Gateway Example

The following example illustrates how an EGP core gateway can be configured.

Figure 18-10 illustrates an environment with three routers (designated C1, C2, and C3) attached to a common X.25 network. The routers are intended to route information using EGP. With the following configuration (on the router designated Core), C1, C2, and C3 cannot route traffic directly to each other via the X.25 network:

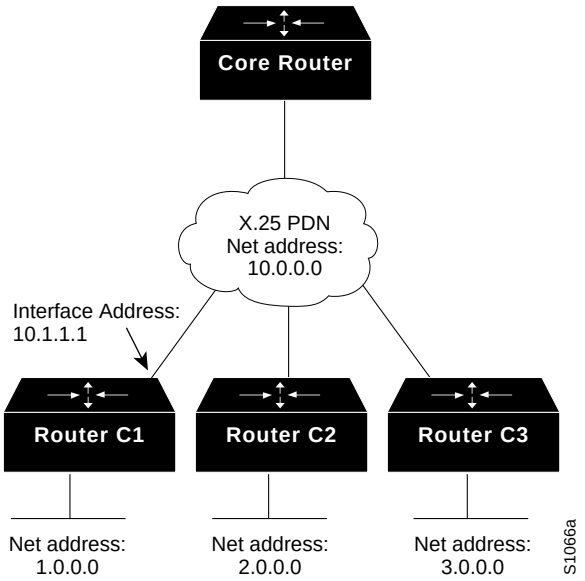
```
access-list 1 permit 10.0.0.0 0.255.255.255
! global access list assignment
router egp 0
neighbor any 1
```

This configuration specifies that an EGP process on any router on network 10.0.0.0 can act as a peer with the Core router. All traffic in this configuration will flow through the Core router.

Third-party advertisements allow traffic to bypass the Core router and go directly to the router that advertised reachability to the Core.

```
access-list 2 permit 10.0.0.0 0.255.255.255
! global access list assignment
router egp 0
neighbor any 2
neighbor any third-party 10.1.1.1
```

Figure 18-10 Core EGP Third-Party Update Configuration Example

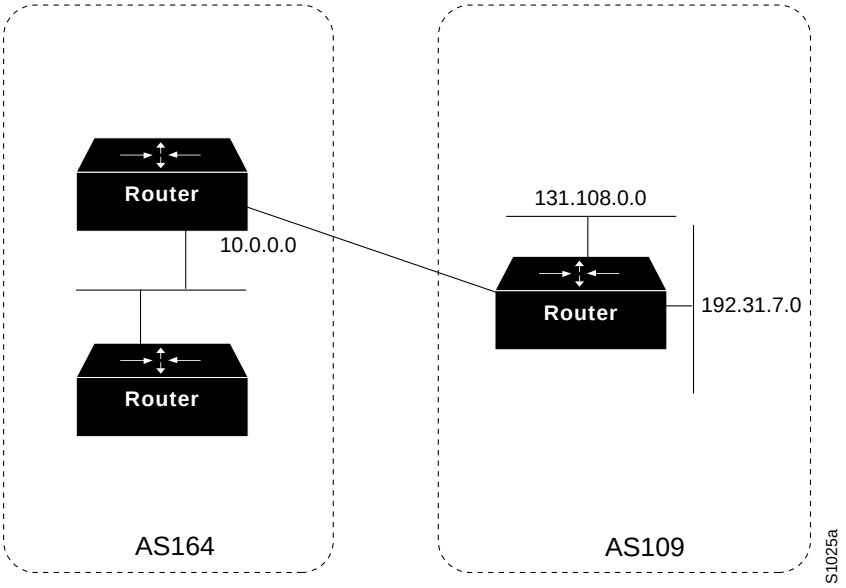


Autonomous System within EGP Example

The following example illustrates a typical configuration for an EGP router process. The router is in autonomous system 109 and is peering with routers in autonomous system 164, as shown in Figure 18-11. It will advertise the networks 131.108.0.0 and 192.31.7.0 to the router in autonomous system 164, 10.2.0.2. The information sent and received from peer routers can be filtered in various ways, including blocking information from certain routers and suppressing the advertisement of specific routes.

```
autonomous-system 109
router egp 164
network 131.108.0.0
network 192.31.7.0
neighbor 10.2.0.2
```

Figure 18-11 Router in AS 164 Peers with Router in AS 109

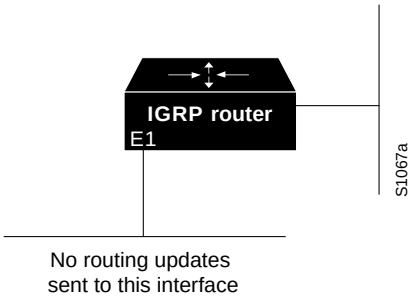


Passive Interface Examples

The following example sends IGRP updates to all interfaces on network 131.108.0.0 except interface Ethernet 1. Figure 18-12 shows this configuration.

```
router igrp 109
network 131.108.0.0
passive-interface ethernet 1
```

Figure 18-12 Filtering IGRP Updates



As in the first example, IGRP updates are sent to all interfaces on network 131.108.0.0 except interface Ethernet 1 in the following example. However, in this case a **neighbor** router configuration command is included. This command permits the sending of routing updates to specific neighbors. One copy of the routing update is generated per neighbor.

```
router igrp 109
network 131.108.0.0
passive-interface ethernet 1
neighbor 131.108.20.4
```

In OSPF, hello packets are not sent on an interface that is specified as passive. Hence, the router will not be able to discover any neighbors, and none of the OSPF neighbors will be able to see the router on that network. In effect, this interface will appear as a stub network to the OSPF domain. This is useful if you want to import routes associated with a connected network into the OSPF domain without any OSPF activity on that interface.

The **passive-interface** router configuration command typically is used when the wildcard specification on the **network** router configuration command configures more interfaces than is desirable. The following configuration causes OSPF to run on all subnets of 131.108.0.0:

```
interface Ethernet 0
ip address 131.108.1.1 255.255.255.0
interface Ethernet 1
ip address 131.108.2.1 255.255.255.0
interface Ethernet 2
ip address 131.108.3.1 255.255.255.0
!
router ospf 109
network 131.108.0.0 0.0.255.255 area 0
```

If you do not want OSPF to run on 131.108.3.0, enter the following commands:

```
router ospf 109
network 131.108.0.0 0.0.255.255 area 0
passive-interface Ethernet 2
```

Administrative Distance Examples

In the following example, the **router igrp** global configuration command sets up IGRP routing in autonomous system number 109. The **network** router configuration commands specify IGRP routing on networks 192.31.7.0 and 128.88.0.0. The first **distance** router configuration command sets the default administrative distance to 255, which instructs the router to ignore all routing updates from routers for which an explicit distance has not been set. The second **distance** command sets the administrative distance to 90 for all routers on the Class C network 192.31.7.0. The third **distance** command sets the administrative distance to 120 for the router with the address 128.88.1.3.

```
router igrp 109
network 192.31.7.0
network 128.88.0.0
distance 255
distance 90  192.31.7.0  0.0.0.255
distance 120  128.88.1.3  0.0.0.0
```

The following example assigns the router with the address 192.31.7.18 an administrative distance of 100, and all other routers on subnet 192.31.7.0 an administrative distance of 200:

```
distance 100 192.31.7.18  0.0.0.0
distance 200 192.31.7.0  0.0.0.255
```

However, if you reverse the order of these commands, all routers on subnet 192.31.7.0 are assigned an administrative distance of 200, including the router at address 192.31.7.18:

```
distance 200 192.31.7.0  0.0.0.255
distance 100 192.31.7.18  0.0.0.0
```

Assigning administrative distances is a problem unique to each network and is done in response to the greatest perceived threats to the connected network. Even when general guidelines exist, the network manager must ultimately determine a reasonable matrix of administrative distances for the network as a whole.

In the following example, the distance value for IP routes learned is 90. Preference is given to these IP routes rather than routes with the default administrative distance value of 110.

```
router isis
distance 90 ip
```

Split Horizon Examples

Two examples of configuring split horizon are provided.

Example 1

The following sample configuration illustrates a simple example of disabling split horizon on a serial link. In this example, the serial link is connected to an X.25 network.

```
interface serial 0
encapsulation x25
no ip split-horizon
```

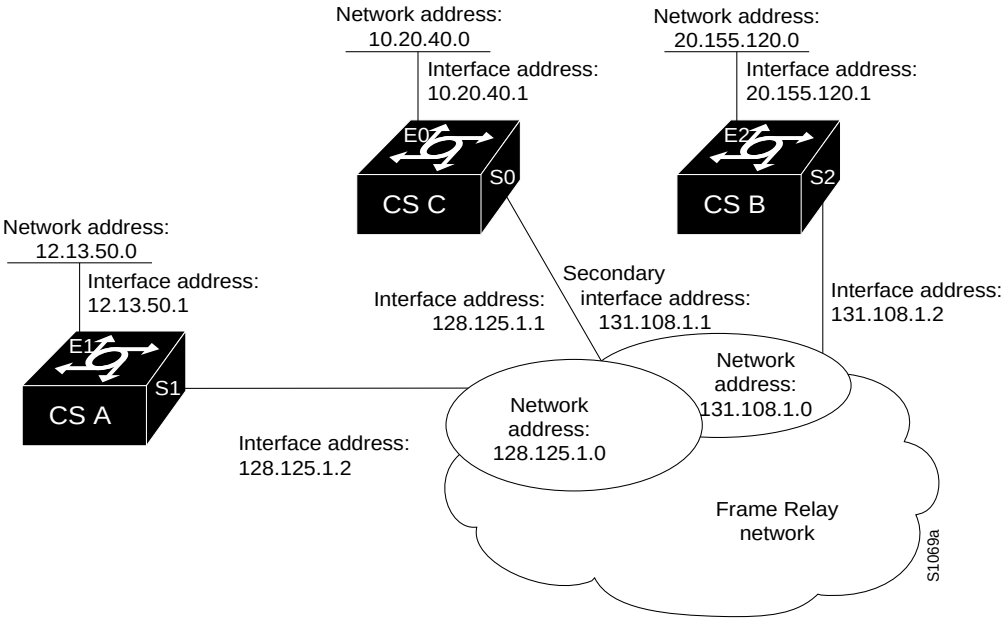
Example 2

In the next example, Figure 18-13 illustrates a typical situation in which the **no ip split-horizon** interface configuration command would be useful. This figure depicts two IP subnets that are both accessible via a serial interface on Router C (connected to Frame Relay network). In this example, the serial interface on Router C accommodates one of the subnets via the assignment of a secondary IP address.

The Ethernet interfaces for Router A, Router B, and Router C (connected to IP networks 12.13.50.0, 10.20.40.0, and 20.155.120.0) all have split horizon *enabled* by default, while the serial interfaces connected to networks 128.125.1.0 and 131.108.1.0 all have split horizon *disabled* by default. The partial interface configuration specifications for each router that follow Figure 18-13 illustrate that the **ip split-horizon** command is *not* explicitly configured under normal conditions for any of the interfaces.

In this example, split horizon must be disabled in order for network 128.125.0.0 to be advertised into network 131.108.0.0, and vice versa. These subnets overlap at Router C, interface S0. If split horizon were enabled on serial interface S0, it would not advertise a route back into the Frame Relay network for either of these networks.

Figure 18-13 Disabled Split Horizon Example for Frame Relay Network



Configuration for Router A

```
interface ethernet 1
ip address 12.13.50.1
!
interface serial 1
ip address 128.125.1.2
encapsulation frame-relay
```

Configuration for Router B

```
interface ethernet 2
ip address 20.155.120.1
!
interface serial 2
ip address 131.108.1.2
encapsulation frame-relay
```

Configuration for Router C

```
interface ethernet 0
ip address 10.20.40.1
!
interface serial 0
ip address 128.124.1.1
ip address 131.108.1.1 secondary
encapsulation frame-relay
```

CHAPTER 19

Configuring ISO CLNS

The International Organization for Standardization (ISO) Connectionless Network Service (CLNS) protocol is a standard for the network layer of the Open Systems Interconnection (OSI) model.

This chapter describes how to configure ISO CLNS. For a complete description of the commands in this chapter, refer to the "ISO CLNS Commands" chapter of the *Router Products Command Reference* publication. For historical background and a technical overview of ISO CLNS, see the *Internetworking Technology Overview* publication.

Cisco's Implementation of ISO CLNS

The Cisco routing software supports packet forwarding and routing for ISO CLNS on networks using a variety of data link layers: Ethernet, Token Ring, Fiber Distributed Data Interface (FDDI), and serial.

You can use CLNS routers on serial interfaces with High-Level Data Link Control (HDLC), Point-to-Point Protocol (PPP), Link Access Procedure, Balanced (LAPB), X.25, Switched Multimegabit Data Service (SMDS), or Frame Relay encapsulation. To use HDLC encapsulation, you must have a router at both ends of the link. If you use X.25 encapsulation, you must manually enter the NSAP-to-X.121 mapping. The LAPB, SMDS, Frame Relay, and X.25 encapsulations interoperate with other vendors.

Cisco's CLNS implementation also is compliant with the Government Open Systems Interconnection Profile (GOSIP) Version 2.

As part of its CLNS support, Cisco routers fully support these ISO and American National Standards Institute (ANSI) standards:

- ISO 9542—Documents the End System-to-Intermediate System (ES-IS) routing exchange protocol.
- ISO 8473—Documents the ISO Connectionless Network Protocol (CLNP).
- ISO 8348/Ad2—Documents network service access point (NSAP) addresses.
- ISO 10589—Documents Intermediate System-to-Intermediate System (IS-IS) Intra-domain Routing Exchange Protocol.

Both the ISO-developed IS-IS and Cisco's ISO Interior Gateway Routing Protocol (IGRP) dynamic routing protocols are supported for dynamic routing of ISO CLNS. In addition, static routing for ISO CLNS is supported.

ISO CLNS Configuration Task List

To configure ISO CLNS, complete the tasks in the following sections:

- Assign Domain Boundaries, NSAP Addresses, and Area Addresses
- Configure a Routing Process
- Configure ES-IS Hello Packet Parameters
- Create Packet-Forwarding Filters and Establish Adjacencies
- Configure CLNS over WANs
- Configure Miscellaneous Features
- Enhance ISO CLNS Performance
- Monitor and Maintain the ISO CLNS Network

See the end of this chapter for configuration examples.

Assign Domain Boundaries, NSAP Addresses, and Area Addresses

In the following section you will learn how to assign NETs, or addresses, for areas and domains. Addressing background material is provided first, followed by a description of how to assign these addresses. The topic of addressing includes the following sections:

- ISO CLNS Addressing Background
- Configure NETs for Domains and Areas
- Map NSAP Addresses to Media Addresses
- Specify Shortcut NSAP Addresses
- Use the IP Domain Name System to Discover ISO CLNS Addresses

ISO CLNS Addressing Background

Addresses in the ISO network architecture are referred to as network service access point (NSAP) addresses and network entity titles (NETs). Each node in an OSI network has one or more NETs. In addition, each node has many NSAP addresses. Each NSAP address differs from one of the NETs for that node in only the last byte (see Figure 19-1). This byte is called the *n-selector*. Its function is similar to the port number in other protocol suites.

Our implementation supports all NSAP address formats that are defined by ISO 8348/Ad2; however, we provide dynamic routing (ISO-IGRP or IS-IS routing) only for NSAP addresses that conform to the address constraints defined in the ISO standard for IS-IS (ISO 10589).

An NSAP address consists of two major fields:

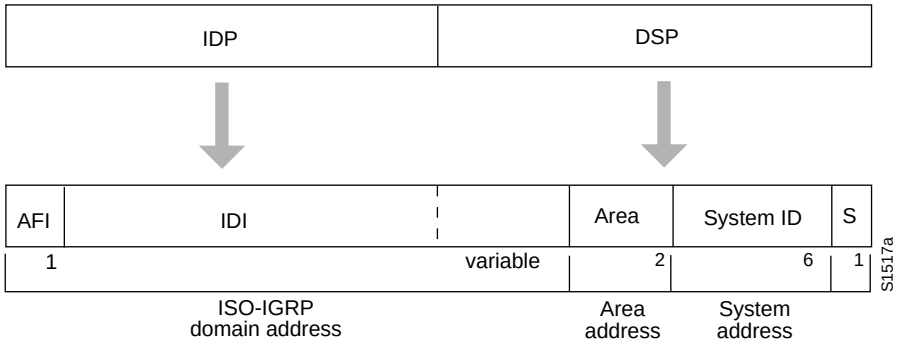
- The IDP is made up of 1-byte AFI and a variable length IDI. The length of the IDI and the encoding format for the DSP are based on the value of the AFI.
- The DSP is made up of a high-order DSP, an area ID, a system ID, and a 1-byte n-selector.

The key difference between the ISO-IGRP and IS-IS NSAP addressing schemes is in the definition of area addresses. Both use the system ID for Level 1 routing. However, they differ in the way addresses are specified for area routing. An ISO-IGRP NSAP address includes three separate levels

for routing: the *domain*, *area*, and *system ID*. An IS-IS address includes two fields: a single continuous *area* field comprising the domain and area fields defined for ISO-IGRP and the *system ID*.

Figure 19-1 illustrates the ISO-IGRP NSAP addressing structure.

Figure 19-1 ISO-IGRP NSAP Addressing Structure



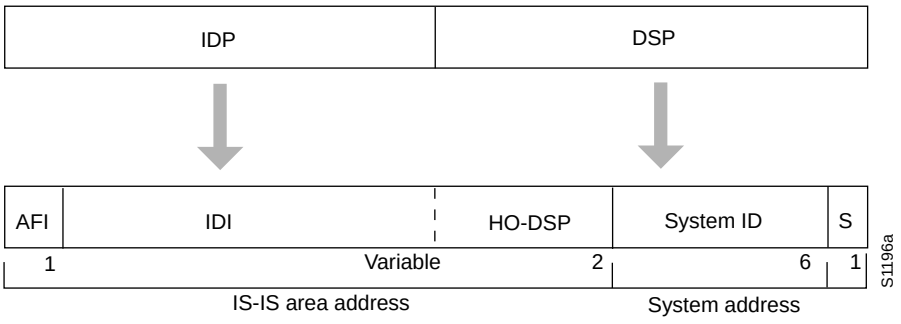
The ISO-IGRP NSAP address is divided into three parts: a domain part, an area address, and a system ID. Domain routing is performed on the domain part of the address. Area routing for a given domain uses the area address. System ID routing for a given area uses the system ID part. The NSAP address is laid out as follows:

- The domain part is of variable length and comes before the area address.
- The area address is the two bytes before the system ID.
- The system ID is the six bytes before the n-selector.
- The n-selector (S) is the last byte of the NSAP address.

Our ISO-IGRP routing implementation interprets the bytes from the AFI up to (but not including) the area field in the DSP as a *domain identifier*. The area field specifies the *area*, and the system ID specifies the *system*.

Figure 19-2 illustrates the IS-IS NSAP addressing structure.

Figure 19-2 IS-IS NSAP Addressing Structure



An IS-IS NSAP address is divided into two parts: an area address (AA) and a system ID. Level 2 routing uses the AA. Level 1 routing uses the system ID address. The NSAP address is laid out as follows:

- The n-selector (S) is the last byte of the NSAP address.
- The system ID is found between the area address and the n-selector byte.

- The area address is the NSAP address, not including the system ID and n-selector.

The IS-IS routing protocol interprets the bytes from the AFI up to (but not including) the system ID field in the DSP as an *area identifier*. The system ID specifies the *system*.

Addressing Rules

All NSAP addresses must obey the following constraints:

- No two nodes can have addresses with the same NET; that is, addresses that match all but the n-selector (S) field in the DSP.
- ISO-IGRP requires at least ten bytes of length; one for domain, two for area, six for system ID, and one for n-selector.
- Our implementation of IS-IS requires at least eight bytes; one for area, six for system ID, and one for n-selector.
- No two nodes residing within the same area can have addresses in which the system ID fields are the same.

The following are examples of OSI network and GOSIP NSAP addresses using the ISO-IGRP implementation.

The following is the OSI network NSAP address format:

```
47.0004.004D.0003.0000.0C00.62E6.00
|     Domain|Area|    System ID| S|
```

The following is an example of the GOSIP NSAP address structure. This structure is mandatory for addresses allocated from the International Code Designator (ICD) 0005 addressing domain. Refer to the GOSIP document, *U.S. Government Open Systems Interconnection Profile (GOSIP)*, Draft Version 2.0, April 1989, for more information.

```
|          Domain          |
 47.0005.80.ffff00.0000.ffff.0004.0000.0c00.62e6.00
  |   |    |    |    |    |    |           |      |
  |   |    |    |    |    |    |           |      |
AFI IDI  DFI  AAI  Resv  RD  Area     System ID  N-selector
```

Entering Routes

Routes are entered by specifying pairs (NSAP prefix and next-hop NET). NETs are similar in function to NSAP addresses. In the routing table, the best match means the longest NSAP prefix entry that matches the beginning of the destination NSAP address. In the following sample static routing table, Table 19-1, the next-hop NETs are listed for completeness, but are not necessary to understand the routing algorithm. Table 19-2 offers examples of how the longest matching NSAP prefix can be matched with routing table entries in Table 19-1.

Table 19-1 Sample Routing Table Entries

Entry	NSAP Address Prefix	Next-Hop NET
1	47.0005.000c.0001	47.0005.000c.0001.0000.1234.00
2	47.0004	47.0005.000c.0002.0000.0231.00
3	47.0005.0003	47.0005.000c.0001.0000.1234.00
4	47.0005.000c	47.0005.000c.0004.0000.0011.00
5	47.0005	47.0005.000c.0002.0000.0231.00

Table 19-2 Hierarchical Routing Examples

Datagram Destination NSAP Address	Table Entry Number Used
47.0005.000c.0001.0000.3456.01	1
47.0005.000c.0001.6789.2345.01	1
47.0004.1234.1234.1234.1234.01	2
47.0005.0003.4321.4321.4321.01	3
47.0005.000c.0004.5678.5678.01	4
47.0005.0001.0005.3456.3456.01	5

Octet boundaries must be used for the internal boundaries of NSAP addresses and NETs.

Configure NETs for Domains and Areas

The first task you have to perform in order to enable CLNS routing is to assign addresses or NETs for your domains and areas.

Note If you are going to configure a routing process, you might want to read the section "Configure a Routing Process," which appears after the discussion of addressing.

First, establish domains. The domain address uniquely identifies the routing domain. All routers within a given domain are given the same domain address.

Within each routing domain, you can set up one or more areas. Determine which routers are to be assigned to which areas. The area address uniquely identifies the routing area.

A router can have one or more area addresses. The concept of multiple area addresses is described in the section that follows, "Multihoming in IS-IS Areas."

Note ISO-IGRP and IS-IS should not be configured for the same area. Do *not* specify an NSAP address where all bytes up to (but not including) the system ID are the same when enabling both ISO-IGRP and IS-IS routing.

Assign domain and area addresses the same way, following the addressing rules described at the beginning of this section.

To configure NETs for the router, perform the following task in router configuration mode:

Task	Command
Configure NETs for a specified router.	**net** *network-entity-title*

See the "Basic Static Routing Examples" and "Static Intradomain Routing Example" sections at the end of this chapter for examples of configuring NETs.

Assign Multiple Area Addresses to IS-IS Areas

The area addressing scheme allowed in IS-IS routing supports assignment of multiple area addresses. This concept is referred to as *multihoming.* You must statically assign multiple area addresses on the router. All of the addresses must have the same system ID.

We currently support assignment of up to three area addresses for a given area. The number of areas allowed in a domain is unlimited.

Multihoming provides a mechanism for smoothly migrating network addresses:

- Splitting up an area—Stations within a given area can accumulate to a point that they are difficult to manage, cause excessive traffic, or threaten to exceed the usable address space for an area. Multiple area addresses can be assigned so that you can smoothly partition a network into separate areas without disrupting service.
- Merging areas—Use transitional area addresses to merge as many as three separate areas into a single area that share a common area address.
- Transition to a different address—You may need to change an area address for a particular group of stations. Use multiple area addresses to allow incoming traffic intended for an old area address to continue being routed to associated stations.

A router can dynamically learn about any adjacent router. As part of this process, the routers inform each other of their area addresses. If two routers share at least one area address, the set of area addresses of the two routers are merged. The merged set cannot contain more than three addresses. If there are more than three, the three addresses with the lowest numerical values are kept, and all others are dropped.

To configure multiple area addresses in IS-IS areas statically, perform the following task as many as three times for a specified router in router configuration mode:

Task	Command
Configure multiple area addresses statically.	**net** *network-entity-title*

Configure a Static NET Address for the Router

You must assign static addresses if you have configured the router to support ISO CLNS but you are not using a routing protocol.

A CLNP packet sent to any of the defined NSAP addresses or NETs will be received by the router. The router uses the following algorithm to select the NET to use when it sends a packet:

- If no dynamic routing protocol is running, use the NET defined for the outgoing interface if it exists; otherwise, use the NET defined for the router.
- If ISO-IGRP is running, use the NET of the routing process that is running on this interface.
- If IS-IS is running, use the NET of the IS-IS routing process that is running on this interface.

To assign an address to the router if you are not dynamically routing CLNS packets, perform the following task in global configuration mode:

Task	Command
Assign an address to the router when the router is not configured to dynamically route CLNS packets using ISO-IGRP or IS-IS.	**clns net** {*net-address* | *name*}

Map NSAP Addresses to Media Addresses

Conceptually, each end system (ES) lives in one area. It discovers the nearest intermediate system (IS) router by listening to ES-IS packets. Each ES must be able to communicate directly with an IS in its area.

When an ES wants to communicate with another ES, it sends the packet to any IS on the same medium. The IS looks up the destination NSAP address and forwards the packet along the best route. If the destination NSAP address is for an ES in another area, the Level 1 IS sends the packet to the nearest Level 2 IS. The Level 2 IS forwards the packet along the best path for the destination area until it gets to a Level 2 IS that is in the destination area. This IS then forwards the packet along the best path inside the area until it is delivered to the destination ES.

End systems need to know how to get to a Level 1 IS for their area, and Level 1 ISs need to know all of the ESs that are directly reachable through each of their interfaces. To provide this information, the routers support the ES-IS protocol. A router dynamically discovers all ESs running the ES-IS protocol. ESs that are not running the ES-IS protocol must be statically configured.

It is sometimes desirable for a router to have a neighbor entry statically configured rather than learned through ES-IS, ISO-IGRP, or IS-IS.

Perform the following tasks in interface configuration mode, as needed, to statically enter mapping information between the NSAP protocol addresses and the subnetwork point of attachment (SNPA) addresses for end systems or intermediate systems:

Task	Command
List all end systems that will be used when you manually specify the NSAP-to-SNPA mapping. In this case, the SNPAs are the MAC addresses.	**clns es-neighbor** *nsap snpa*
List all intermediate systems that will be used when you manually specify the NSAP-to-SNPA mapping. In this case, the SNPAs are the MAC addresses.	**clns is-neighbor** *nsap snpa*

Note It is necessary to use static mapping only for those end systems that do *not* support ES-IS. The router continues to dynamically discover those end systems that *do* support ES-IS.

If there are systems on the Ethernet that do not use ES-IS, or if X.25 is being used and no dynamic routing protocol is running over the X.25 network, you specify the NSAP/NET (protocol address) to SNPA (media address) mappings by performing the following tasks in interface configuration mode:

Task	Command
List all end systems that will be used when you manually specify the NSAP-to-SNPA mapping. In this case, the SNPAs are the X.25 network addresses (X.121 addresses).	**clns es-neighbor** *nsap snpa*
or	
List all intermediate systems that will be used when you manually specify the NSAP-to-SNPA mapping. In this case, the SNPAs are the X.25 network addresses (X.121 addresses).	**clns is-neighbor** *nsap snpa*
Establish an IP-to-X.121 address map.	**x25 map clns** *snpa X.25-facilities-info*[1]

1. This command is documented in the "X.25 and LAPB Commands" chapter of the *Router Products Command Reference* publication.

If you have configured interfaces for ISO-IGRP or IS-IS, the ES-IS routing software automatically turns ES-IS on for those interfaces.

Specify Shortcut NSAP Addresses

You can define a name-to-NSAP address mapping. This name can then be used in place of typing the long set of numbers associated with an NSAP address.

To define a name-to-NSAP address mapping, perform the following task in global configuration mode:

Task	Command
Define a name-to-NSAP address mapping.	**clns host** *name nsap*

The assigned NSAP name is displayed, where applicable, in **show** and **debug** EXEC commands.

There are some effects and requirements associated with using names to represent NETs and NSAP addresses, however; they include the following:

- Although using names as proxies for addresses is allowed with CLNS commands, they are never written out to NVRAM.
- The **clns host** global configuration command is generated after all other CLNS commands when the configuration file is parsed. As a result, you cannot edit the NVRAM version of the configuration to specifically change the address defined in the original **clns host** command. You must specifically change any commands that refer to the original address. This affects all commands that accept names.

The commands that are affected by these requirements include:

- **net** (router configuration command)
- **clns is-neighbor** (interface configuration command)
- **clns es-neighbor** (interface configuration command)
- **clns route** (global configuration command)

Use the IP Domain Name System to Discover ISO CLNS Addresses

If your router has both ISO CLNS and IP enabled, you can use the Domain Name System (DNS) to query ISO CLNS addresses by using the NSAP address type, as documented in RFC 1348. This feature is useful for the ISO CLNS **ping** EXEC command and when making Telnet connections. This feature is enabled by default.

To enable or disable DNS queries for ISO CLNS addresses, perform the following tasks in global configuration mode:

Task	Command
Allow DNS queries for CLNS addresses.	**ip domain-lookup nsap**
Disable DNS queries for CLNS addresses.	**no ip domain-lookup nsap**

Configure a Routing Process

The basic function of a router is to forward packets: to receive a packet in one interface and send it out another (or the same) interface to the proper destination. All routers do this by looking up the destination address in a table. The tables can be built either dynamically or statically. If you are configuring all of the entries in the table yourself, you are using *static* routing. If you have a routing process building the tables, you are using *dynamic* routing. It is possible, and sometimes necessary, to use both static and dynamic routing simultaneously.

When you configure only ISO CLNS and not routing protocols, the router only makes forwarding decisions. It does not perform other routing-related functions. In such a configuration, the router compiles a table of adjacency data, but does not advertise this information. The only information that is inserted into the routing table is the NSAP and network entity title (NET) addresses of this router, static routes, and adjacency information.

Static Routing

Static routing is used when it is not possible or desirable to use dynamic routing. The following are some instances of when you would use static routing:

- If you are using routers that do not support the same dynamic routing protocol, you must use static routing.
- If your network includes WAN links that involve paying for connect time or per packet, you would use static routing rather than paying to run a routing protocol over that link.
- If you want routers to advertise connectivity to external networks but you are not running an interdomain routing protocol, you *must* use static routes.
- If you must interoperate with another vendor's equipment that does not support any of the dynamic routing protocols that we support, you must use static routing.
- For operation over X.25, Frame Relay, or SMDS networks, static routing is generally preferable.

Note An interface that is configured for static routing cannot reroute *around* failed links.

Dynamic Routing

We support two dynamic routing protocols for CLNP networks:

- ISO-IGRP
- IS-IS

Both routing protocols support the concept of *areas*. Within an area, all routers know how to reach all of the system IDs. Between areas, routers know how to reach the proper area.

IS-IS supports two levels of routing: *station routing* (within an area) and *area routing* (between areas). ISO-IGRP supports three levels of routing: *system* routing, *area* routing, and *interdomain* routing. Routing across domains (interdomain routing) can be done either statically or dynamically with ISO-IGRP.

Intermediate Systems and End Systems

Some intermediate systems keep track of how to communicate with all of the end systems in their areas and thereby function as Level 1 routers (also referred to as local routers). Other intermediate systems keep track of how to communicate with other areas in the domain, functioning as Level 2 routers (sometimes referred to as area routers). Our routers are always Level 1 and Level 2 routers when routing ISO-IGRP; they can be configured to be Level 1 only, Level 2 only, or both Level 1 and Level 2 routers when routing IS-IS.

End systems communicate with intermediate systems using the ES-IS protocol. Level 1 and Level 2 intermediate systems communicate with each other using either ISO IS-IS or our ISO-IGRP protocol.

This section describes the tasks associated with each routing protocol. When dynamically routing, you can choose either ISO-IGRP or IS-IS, or you can route both routing protocols at the same time.

Configure CLNS Static Routing

You do not need to explicitly specify a routing process to use static routing facilities. If you choose static routing, the configuration process begins with enabling CLNS routing on the router. CLNS routing is enabled by default when you configure either of the routing protocols.

CLNS static routing is configured when you do the following:

Step 1 Configure CLNS on the router.

Step 2 Assign a static NET address for the router.

Step 3 Enable ISO CLNS for each interface.

Step 4 Enter a specific static route.

Step 5 Configure other, optional variations.

Each of these steps is described in the following sections.

See the “Basic Static Routing Examples,” “Static Intradomain Routing Example,” and “ Static Interdomain Routing Example” sections at the end of this chapter for examples of configuring static routes.

Configure CLNS on the Router

To configure CLNS on the router, perform the following task in global configuration mode:

Task	Command
Configure CLNS on the router.	**clns routing**

Assign a Static NET Address for the Router

If you have configured a router to support ISO CLNS but you have not configured it to route CLNS packets dynamically using ISO-IGRP or IS-IS, then you must assign an address to the router.

A CLNP packet sent to any of the defined NSAP addresses or NETs will be received by the router. The router uses the following algorithm to select the NET to use when it sends a packet:

- If no dynamic routing protocol is running, use the NET defined for the outgoing interface if it exists; otherwise, use the NET defined for the router.
- If ISO-IGRP is running, use the NET of the routing process that is running.

- If IS-IS is running, use the NET of the IS-IS routing process that is running.

To assign an address to the router, perform the following task in global configuration mode:

Task	Command
Assign an address to the router when the router is not configured to dynamically route CLNS packets using ISO-IGRP or IS-IS.	**clns net** {*net-address* | *name*}

Enable ISO CLNS for Each Interface

You also must enable ISO CLNS for each interface. This is done automatically when you configure IS-IS or ISO-IGRP routing on an interface; however, if you do not intend to perform any dynamic routing on an interface but intend to pass ISO CLNS packet traffic to end systems, you must enable CLNS yourself.

Enable ISO CLNS when you want to pass ISO CLNS packet traffic to end systems but do not want to perform any dynamic routing on an interface. Perform the following task in interface configuration mode:

Task	Command
Enable ISO CLNS for each interface.	**clns enable**

Enter a Specific Static Route

You can enter a specific static route and apply it globally even when you are dynamically routing. NSAP addresses that start with the NSAP prefix you specify are forwarded to the next-hop node.

To apply a specific static route globally, perform the following task in global configuration mode:

Task	Command
Enter a specific static route.	**clns route** *nsap-prefix* {*next-hop-net* | *name*}

Configure Variations of the Clns Route Command

You also can perform the following tasks that use variations of the **clns route** global configuration command:

- Bind the next hop to a specified interface and media address when you do not know the NSAP address of your neighbor. Note that this version of the **clns route** command is not literally *applied* to a specific interface.
- Tell a router to discard packets with the specified NSAP prefix.
- Specify a default prefix.

The following list shows how to perform each of these tasks. Perform these tasks in global configuration mode:

Task	Command
Enter a specific static route for a specific interface.	**clns route** *nsap-prefix interface-type* [*snpa-address*]
Explicitly tell a router to discard packets with the specified NSAP prefix.	**clns route** *nsap-prefix* **discard**
Configure a default prefix rather than specify an NSAP prefix.	**clns route default** *nsap-prefix interface-type*

Configure ISO-IGRP Dynamic Routing

CLNS routing is enabled by default on the router when you configure ISO-IGRP. All you need to do to specify an ISO-IGRP routing process is to enable the ISO-IGRP routing process, identify the address for the router, and specify the interfaces that are to route ISO-IGRP. Optionally, you can set a level for your routing updates when you configure the interfaces. You can specify up to ten ISO-IGRP processes.

To configure ISO-IGRP dynamic routing, perform the following tasks in the order listed:

Task		Command
Step 1	Enter global configuration mode.	**configure**
Step 2	Enable the ISO-IGRP routing process, which places you in router configuration mode.	**router iso-igrp** [*tag*]
Step 3	Configure the NET or address for the routing process.	**net** *network-entity-title*
Step 4	Enter interface configuration mode.	**interface** *type number*
Step 5	Enable ISO-IGRP on specified interfaces; also set the level type for routing updates.	**clns router iso-igrp** *tag* [**level 2**]

Although IS-IS allows you to configure multiple NETs, ISO-IGRP allows only one NET per process.

You can configure an interface to advertise Level 2 information only. This option reduces the amount of router-to-router traffic by telling the router to send out only Level 2 routing updates on certain interfaces. Level 1 information is not passed on the interfaces for which the Level 2 option is set.

The additional tasks that follow allow you to customize ISO-IGRP.

See the “Dynamic Routing within the Same Area Example,” “Dynamic Routing in More Than One Area Example,” and “Dynamic Routing in Overlapping Areas Example” sections at the end of this chapter for examples of configuring dynamic routing.

You can also configure the following ISO-IGRP parameters:

- Adjust ISO-IGRP metrics
- Adjust ISO-IGRP timers
- Enable or disable split horizon
- Redistribute routes into an ISO-IGRP domain
- Specify preferred routes

Adjust ISO-IGRP Metrics

You have the option of altering the default behavior of ISO-IGRP routing and metric computations. This allows, for example, tuning of system behavior to allow for transmissions via satellite. Although ISO-IGRP metric defaults were carefully selected to provide excellent operation in most networks, you can adjust the metric.

Note Adjusting the ISO-IGRP metric can dramatically affect network performance, so ensure that all metric adjustments are made carefully. Due to the complexity of this task, it is not recommended unless it is done with guidance from an experienced system designer.

You can use different metrics for the ISO-IGRP routing protocol on CLNS. By performing the following task, you can configure the metric constants used in the ISO-IGRP composite metric calculation of reliability and load. Perform this task in router configuration mode:

Task	Command
Adjust the ISO-IGRP metric.	**metric weights** *qos k1 k2 k3 k4 k5*

Two additional ISO-IGRP metrics can be configured. These are the bandwidth and delay associated with an interface. Refer to the "Interface Commands" chapter of the *Router Products Command Reference* publication for details about the **bandwidth** and **delay** interface configuration commands used to set these metrics, and to the "Configuring Interfaces" chapter of this manual for configuration information.

Note Using the **bandwidth** and **delay** commands to change the values of the ISO-IGRP metrics also will change the values of IP IGRP metrics.

Adjust ISO-IGRP Timers

The basic timing parameters for ISO-IGRP are adjustable. Because the ISO-IGRP routing protocol executes a distributed, asynchronous routing algorithm, it is important that these timers be the same for all routers in the network.

To adjust ISO-IGRP timing parameters, perform the following task in router configuration mode:

Task	Command
Adjust the ISO-IGRP timers.	**timers basic** *update-interval holddown-interval invalid-interval*

Enable or Disable Split Horizon

Split horizon blocks information about routes from being advertised out the interface from which that information originated. This feature usually optimizes communication among multiple routers, particularly when links are broken.

To either enable or disable split horizon for ISO-IGRP updates, perform the following tasks in interface configuration mode:

Task	Command
Enable split horizon for ISO-IGRP updates.	**clns split-horizon**
Disable split horizon for ISO-IGRP updates.	**no clns split-horizon**

The default for all LAN interfaces is for split horizon to be enabled; the default for WAN interfaces on X.25, Frame Relay, or SMDS networks is for split horizon to be disabled.

Redistribute Routes into an ISO-IGRP Domain

You can configure a router to do interdomain dynamic routing by putting it into two domains and configuring it to redistribute the routing information between the domains. Routers configured this way are referred to as *border* routers. If you have a router that is in two routing domains, you might want to redistribute routing information between the two domains.

Note It is not necessary to use redistribution between areas.

Also, you can conditionally control the redistribution of routes between routing domains by defining *route maps* between the two domains. Route maps allow you to use tags in routes to influence route redistribution. These methods of specifying route redistribution are listed in the following tables.

Static routes by default are redistributed into ISO-IGRP routing domains.

Perform the following tasks in router configuration mode:

Task	Command
Redistribute routes from one routing protocol into other routing domains.	**redistribute** *protocol* [*tag*] [**route-map** *map-tag*]
Redistribute static routes by causing the specified routing process to advertise static CLNS routes.	**redistribute static**

Perform the following task in global configuration mode:

Task	Command
Define any route maps needed to control redistribution.	**route-map** *map-tag* {**permit** | **deny**} *sequence-number*

Specify Preferred Routes

When multiple routing processes are running in the same router for CLNS, it is possible for the same route to be advertised by more than one routing process. The router always picks the route whose routing protocol has the lowest administrative distance.The lower the value of the distance, the more preferred the route.

Default administrative distances are already set. By default, the following administrative distances are assigned:

- Static routes—10
- ISO-IGRP routes—100
- IS-IS routes—110

However, if you need to change an administrative distance for a route, perform the following task in router configuration mode:

Task	Command
Specify preferred routes by setting the lowest administrative distance.	**distance** *value* [**clns**]

If you want an ISO-IGRP prefix route to override a static route, you must set the distance for the routing process to be lower than 10.

Configure IS-IS Dynamic Routing

CLNS routing is enabled by default on the router when you configure IS-IS dynamic routing. All you need to do to specify an IS-IS routing process is to enable the process, identify the address for the router, and specify the interfaces that are to route IS-IS. You can specify *only one* IS-IS process per router.

To configure IS-IS dynamic routing, perform the following required tasks in the order listed:

Task		Command
Step 1	Enter global configuration mode.	**configure**
Step 2	Enable IS-IS routing, which places you in router configuration mode.	**router isis** [*tag*]
Step 3	Configure NETs for the routing process; you can specify a name for a NET as well as an address	**net** *network-entity-title*
Step 4	Enter interface configuration mode.	**interface** *type number*.
Step 5	Specify the interfaces that should be actively routing IS-IS.	**clns router isis** [*tag*]

For IS-IS, multiple NETs per router are allowed, with a maximum of three. However, only one IS-IS process is allowed, whether you run it in integrated mode, ISO CLNS only, or IP only.

See the "IS-IS Routing Configuration Examples" section at the end of this chapter for examples of configuring IS-IS routing.

Configure IS-IS Interface Parameters

Our IS-IS implementation allows you to customize certain interface-specific IS-IS parameters. You can perform the following optional tasks:

- Configure IS-IS link-state metrics.
- Set the advertised hello interval.
- Set the advertised CSNP interval.
- Set the retransmission interval.
- Specify designated router election.
- Specify the interface circuit type.
- Configure IS-IS password authentication.

You are not required to alter any of these parameters, but some interface parameters must be consistent across all routers in an attached network. Therefore, be sure that if you do configure any of these parameters, the configurations for all routers on the network have compatible values.

Configure IS-IS Link-State Metrics

You can configure a cost for a specified interface. The default metric is used as a value for the IS-IS metric. This is the value assigned when there is no quality of service (QOS) routing performed. The only metric that is supported by the router and that you can configure is the *default-metric*, which you can configure for Level 1 and/or Level 2 routing.

To configure the link state metric, perform the following task in interface configuration mode:

Task	Command
Configure the metric (or cost) for the specified interface.	**isis metric** *default-metric delay-metric expense-metric error-metric* {**level-1** | **level-2**}

Set the Advertised Hello Interval

You can specify the length of time in seconds between hello packets that the router sends on the interface.

To set the advertised hello interval, perform the following task in interface configuration mode:

Task	Command
Specify the length of time, in seconds, between hello packets the router sends on the specified interface.	**isis hello-interval** *seconds* {**level-1** | **level-2**}

The hello interval can be configured independently for Level 1 and Level 2, except on serial point-to-point interfaces. (Because there is only a single type of hello packet sent on serial links, it is independent of Level 1 or Level 2.) Specify an optional level for X.25, SMDS, and Frame Relay multiaccess networks.

Set the Advertised CSNP Interval

Complete sequence number PDUs (CSNPs) are sent by the designated router to maintain database synchronization.

You can configure the IS-IS CSNP interval for the interface by performing the following task in interface configuration mode:

Task	Command
Configure the IS-IS CSNP interval for the specified interface.	**isis csnp-interval** *seconds* {**level-1** | **level-2**}

This feature does not apply to serial point-to-point interfaces. It does apply to WAN connections if the WAN is viewed as a multiaccess meshed network.

Set the Retransmission Interval

You can configure the number of seconds between retransmission of IS-IS link state PDUs (LSPs) for point-to-point links.

To set the retransmission level, perform the following task in interface configuration mode:

Task	Command
Configure the number of seconds between retransmission of IS-IS LSPs for point-to-point links.	**isis retransmit-interval** *seconds*

The value you specify should be an integer greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links.

Specify Designated Router Election

You can configure the priority to use for designated router election. Priorities can be configured for Level 1 and Level 2 individually.

To configure the priority to use for designated router election, perform the following task in interface configuration mode:

Task	Command
Configure the priority to use for designated router election.	**isis priority** *value* {**level-1** | **level-2**}

Specify the Interface Circuit Type

You can specify adjacency levels on a specified interface.

To configure the adjacency for neighbors on the specified interface, perform the following task in interface configuration mode:

Task	Command
Configure the type of adjacency desired for neighbors on the specified interface (specify the interface circuit type).	**isis circuit-type** {**level-1** | **level-1-2** | **level-2-only**}

If you specify Level 1, a Level 1 adjacency might be established if there is at least one area address common to both this system and its neighbors.

If you specify both Level 1 and Level 2, a Level 1 and 2 adjacency is established if the neighbor is also configured as both Level 1 and Level 2 and there is at least one area in common. If there is no area in common, a Level 2 adjacency is established. This is the default value.

If you specify Level 2, a Level 2 adjacency is established.

Note that it is seldom necessary to configure an interface as Level 1 only or Level 2 only—the protocols will determine the adjacency type automatically.

Configure IS-IS Password Authentication

You can assign different passwords for different routing levels. By default, authentication is disabled. Specifying Level 1 or Level 2 disables the password only for Level 1 or Level 2 routing, respectively. If you do not specify a level, the default is Level 1.

To configure an authentication password for an interface, perform the following task in interface configuration mode:

Task	Command
Configure the authentication password for an interface.	**isis password** *password* {**level-1** | **level-2**}

Configure IS-IS Parameters

You can configure the following IS-IS parameters:

- Redistribute routes into an IS-IS domain.
- Specify preferred routes.
- Specify router-level support.
- Configure IS-IS authentication passwords.

Redistribute Routes into an IS-IS Domain

If you have a router that is in two routing domains, you might want to redistribute routing information between the two domains. First, you specify the destination routing protocol, then you define the routing protocol that is to be redistributed into the destination routing protocol.

Redistribution only occurs for Level 2 routing.

You also can conditionally control the redistribution of routes between routing domains by defining route maps between them.

Additionally, you can cause the specified routing process to advertise static CLNS routes. Static routes are always redistributed into IS-IS unless you explicitly disable this feature.

The methods of specifying route redistribution are listed in the following tables.

Perform the following tasks in router configuration mode:

Task	Command
Redistribute routes from one routing protocol into other routing domains.	**redistribute** *protocol* [*tag*] [**route-map** *map-tag*]
Redistribute static routes by causing the specified routing process to advertise static CLNS routes.	**redistribute static** [**clns**]

Perform the following task in global configuration mode:

Task	Command
Define any route maps needed to control redistribution.	**route-map** *map-tag* {**permit** | **deny**} *sequence-number*

See the “Route-Map Examples” section at the end of this chapter for examples of configuring route maps.

Specify Preferred Routes

When multiple routing processes are running in the same router for CLNS, it is possible for the same route to be advertised by more than one routing process. The router always picks the route whose routing protocol has the lowest administrative distance.The lower the value of the distance, the more preferred the route.

By default the following administrative distances are assigned:

- Static routes—10
- ISO-IGRP routes—100
- IS-IS routes—110

However, if you need to change an administrative distance for a route, perform the following task in router configuration mode:

Task	Command
Specify preferred routes by setting the lowest administrative distance.	**distance** *value* [**clns**]

If you want an IS-IS prefix route to override a static route, you must set the distance for the routing process to be lower than 10.

Specify Router-Level Support

You can configure the router to act as a Level 1 (intra-area) router, as both a Level 1 router and a Level 2 (interarea) router, or as an interarea router only.

To configure the IS-IS level, perform the following task in router configuration mode:

Task	Command
Configure the IS-IS level at which the router is to operate.	**is-type** {**level-1** | **level-1-2** | **level-2-only**}

Note that it is seldom necessary to configure the IS type because the IS-IS protocol will automatically establish this.

Configure IS-IS Authentication Passwords

You can assign passwords to areas and domains. An area password is inserted in Level 1 (station router level) LSPs, CSNPs, and partial sequence number PDUs (PSNPs). A routing domain authentication password is inserted in Level 2 (the area router level) LSP, CSNP, and PSNP PDUs.

To configure area or domain passwords, perform the following tasks in router configuration mode:

Task	Command
Configure the area authentication password.	**area-password** *password*
Configure the routing domain authentication password.	**domain-password** *password*

Configure ES-IS Hello Packet Parameters

You can configure ES-IS parameters for communication between end systems and routers. In general, you should leave these parameters at their default values.

When configuring an ES-IS router, be aware of the following:

- ES-IS does not run over X.25 links unless the *broadcast* facility is enabled.
- ES hello (ESH) packets and IS hello (ISH) packets are sent without options. Options in received ESH and ISH packets are ignored.

ISs and ESs periodically send out hello packets to advertise their availability. The frequency of these hello packets can be configured.

The recipient of a hello packet creates an adjacency entry for the system that sent it. If the next hello packet is not received within the interval specified, the adjacency times out and the adjacent node is considered unreachable.

A default rate has been set for hello packets; however, you can change the default by performing the following task in global configuration mode:

Task	Command
Specify the rate at which ESH and ISH packets are sent.	**clns configuration-time** *seconds*

A default rate has been set for packet validity; however, you can change the default by performing the following task in global configuration mode:

Task	Command
Allow the sender of an ESH or ISH packet to specify the length of time you consider the information in these packets to be valid.	**clns holding-time** *seconds*

A default rate has been set for the ES Configuration Timer (ESCT) option; however, you can change the default by performing the following task in interface configuration mode:

Task	Command
Specify how often the end system should transmit ES Hello packet PDUs.	**clns esct-time** *seconds*

Create Packet-Forwarding Filters and Establish Adjacencies

You can build powerful CLNS filter expressions, or access lists, that can be used to control either the forwarding of frames through router interfaces or the establishment of adjacencies with or the application of filters to any combination of ES or IS neighbors, ISO-IGRP neighbors, or IS-IS neighbors.

CLNS filter expressions are complex logical combinations of CLNS filter sets. CLNS filter sets are lists of address templates against which CLNS addresses are matched. Address templates are CLNS address *patterns* that are either simple CLNS addresses that match just one address, or match multiple CLNS addresses through the use of wildcard characters, prefixes, and suffixes. Frequently used address templates can be given *aliases* for easier reference.

To establish CLNS filters, perform the following tasks in global configuration mode:

Task	Command
Create aliases for frequently used address templates.	**clns template-alias** *name template*
Build filter sets of multiple address template permit and deny conditions.	**clns filter-set** *sname* [**permit** | **deny**] *template*
Build filter expressions, combining the use of one or more filter sets.	**clns filter-expr** *ename term*

Perform the following tasks in interface configuration mode:

Task	Command
Apply a filter expression to frames forwarded in or out of router interfaces on a per-interface basis.	**clns access-group** *name* [**in** | **out**]
Apply a filter expression to the establishment of IS-IS adjacencies on a per-interface basis.	**isis adjacency-filter** *name* [**match-all**]
Apply a filter expression to the establishment of ISO-IGRP adjacencies on a per-interface basis.	**iso-igrp adjacency-filter** *name*
Apply a filter expression to the establishment of ES or IS adjacencies on a per-interface basis.	**clns adjacency-filter** {**es** | **is**} *name*

See the "CLNS Filter Examples" section at the end of this chapter for examples of configuring CLNS filters.

Configure CLNS over WANs

This section provides general information about running ISO CLNS over WANs. For more information, see the relevant chapters describing the specific types of encapsulation that might be used.

You can use CLNS routers on serial interfaces with HDLC, PPP, LAPB, X.25, Frame Relay, DDR, or SMDS encapsulation. To use HDLC encapsulation, you must have a router at both ends of the link. If you use X.25 encapsulation, and if IS-IS or ISO-IGRP is not used on an interface, you must manually enter the NSAP-to-X.121 mapping. The LAPB, SMDS, Frame Relay, and X.25 encapsulations interoperate with other vendors.

Both ISO-IGRP and IS-IS can be configured over WANs.

X.25 is not a broadcast medium; therefore, ES-IS generally is not used to automatically advertise and record mappings between NSAP/NET (protocol addresses) and subnetwork points of attachment (SNPA) (media addresses). (With X.25, the SNPAs are the X.25 network addresses [X.121 addresses]. These are usually assigned by the X.25 network provider.) If you use static routing, you must configure the NSAP-to-X.121 mapping.

Configuring a serial line to use CLNS over X.25 requires configuring the general X.25 information and the CLNS-specific information. First, configure the general X.25 information. Then, enter the static mapping information.

You can specify nondefault packet and window sizes, reverse charge information, and so on. The X.25 facilities information that can be specified is exactly the same as in the **x25 map** interface configuration command described in the "Configuring X.25 and LAPB" chapter.

See the "Configuring ISO CLNS over X.25 Example" section at the end of this chapter for an example of configuring CLNS over X.25.

Configure Miscellaneous Features

Perform the optional tasks in the following sections to configure miscellaneous features of an ISO CLNS network:

- Assign Static NSAP Addresses for an Interface
- Configure DECnet OSI or Phase V Cluster Aliases
- Configure Digital-Compatible Mode
- Allow Security-Option Packets to Pass

Assign Static NSAP Addresses for an Interface

You can assign an NSAP address for a specific interface. This allows the router to advertise different addresses on each interface. This is useful if you are doing static routing and need to control the source NET used by the router on each interface.

To assign an NSAP address for a specified interface, perform the following task in interface configuration mode:

Task	Command
Assign an NSAP address for a specific interface.	**clns net** {*nsap-address* | *name*}

Configure DECnet OSI or Phase V Cluster Aliases

DECnet Phase V *cluster aliasing* allows multiple systems to advertise the same system ID in end-system hello packets. The router does this by caching multiple ES adjacencies with the same NSAP address, but different SNPA addresses. When a packet is destined to the common NSAP address, the router splits the packet loads among the different SNPA addresses. A router that supports this capability forwards traffic to each system. You can do this on a per-interface basis.

To configure cluster aliases, perform the following task in interface configuration mode:

Task	Command
Allow multiple systems to advertise the same system ID in end-system Hello packets on a per-interface basis.	**clns cluster-alias**

If DECnet Phase V cluster aliases are disabled on an interface, ES Hello packet information is used to replace any existing adjacency information for the NSAP address. Otherwise, an additional adjacency (with a different SNPA) is created for the same NSAP address.

See the "Configuring DECnet Cluster Aliases Example" section at the end of this chapter for an example of configuring DECnet OSI cluster aliases.

Configure Digital-Compatible Mode

If you have an old DECnet implementation of ES-IS in which the NSAP address advertised in an ISH does not have the N-selector byte present, you may want to configure the router to allow ISHs sent and received to ignore the N-selector byte. The n-selector byte is the last byte of the NSAP address.

To enable Digital-compatible mode, perform the following task in interface configuration mode:

Task	Command
Allow ISHs sent and received to ignore the N-selector byte.	**clns dec-compatible**

Allow Security-Option Packets to Pass

By default, the router will discard any packets it sees as set with security options. You can disable this behavior; that is, allow such packets to pass through.

Perform the following task in global configuration mode:

Task	Command
Allow the router to accept any packets it sees as set with security options.	**clns security pass-through**

Header Options

The ISO CLNS routing software ignores the Record Route option, the Source Route option, and the QOS (quality of service) option other than congestion experienced. The security option causes a packet to be rejected with a bad option indication.

Enhance ISO CLNS Performance

Generally, you do not need to change the default settings for CLNS packet switching, but there are some modifications you can make when you decide to make changes in your network's performance. This section describes the following ISO CLNS parameters that you can change:

- MTU size
- Checksums
- Fast switching
- Congestion threshold
- Error protocol data units (ERPDUs)
- Redirect PDUs (RPDUs)
- Parameters for locally sourced packets

See the "Customizing Performance Parameters Example" section at the end of this chapter for examples of configuring various performance parameters.

Specify the MTU Size

All interfaces have a default maximum packet size. You can, however, set the maximum transmission unit (MTU) size of the packets sent on the interface to reduce fragmentation. The minimum value is 512; the default and maximum packet size depends on the interface type.

Changing the MTU value of packets sent on the interface can affect the CLNS MTU value. If the CLNS MTU is at its maximum given the interface MTU, then the CLNS MTU will change with the interface MTU. However, the reverse is not true: changing the CLNS MTU value has no effect on the **mtu** value of other packets sent on the interface.

To set the CLNS MTU packet size for a specified interface, perform the following task in interface configuration mode:

Task	Command
Set the MTU size of the packets sent on the interface.	**clns mtu** *size*

The CTR card does not support the switching of frames larger than 4472 bytes. Interoperability problems might occur if CTR cards are intermixed with other Token Ring cards on the same network. These problems can be minimized by lowering the CLNS MTU sizes to be the same on all devices on the network.

Disable Checksums

When the ISO CLNS routing software sources a CLNS packet, by default it generates checksums. You can disable this function.

Perform the following task in interface configuration mode:

Task	Command
Disable checksum generation.	**no clns checksum**

Note Enabling checksum generation has no effect on routing packets (ES-IS, ISO-IGRP, and IS-IS) sourced by the system. It applies to pings and trace-route packets.

Disable Fast Switching Through the Cache

Fast switching through the cache is enabled by default for all supported interfaces. You can disable fast switching by performing the following task in interface configuration mode:

Task	Command
Disable fast switching.	**no clns route-cache**

Note The cache still exists and is used after the **no clns route-cache** command is used; the software just does not do fast switching through the cache.

Set the Congestion Threshold

If a router configured for CLNS experiences congestion, it sets the congestion-experienced bit. You can set the congestion threshold on a per-interface basis. By setting this threshold, you cause the system to set the congestion-experienced bit if the output queue has more than the specified number of packets in it.

To set the congestion threshold, perform the following task in interface configuration mode:

Task	Command
Set the congestion threshold.	**clns congestion-threshold** *number*

Transmit ERPDUs

When a CLNS packet is received, the routing software looks in the routing table for the next hop. If it does not find one, the packet is discarded and an error protocol data unit (ERPDU) might be sent.

You can set an interval between ERPDUs. Doing so reduces bandwidth if this feature is disabled. When you determine the minimum interval between ERPDUs, the router does not send ERPDUs more frequently than one per interface per ten milliseconds.

To transmit ERPDUs, perform the following tasks in interface configuration mode:

Task	Command
Send an ERPDU when the routing software detects an error in a data PDU; this is enabled by default.	**clns send-erpdu**
Determine the minimum interval, in milliseconds, between ERPDUs.	**clns erpdu-interval** *milliseconds*

Control RDPDUs

If a packet is sent out the same interface it came in on, a redirect PDU (RDPDU) also can be sent to the sender of the packet. You can control RDPDUs in the following ways:

- You can allow CLNS to send RDPDUs when a better route for a given host is known; this is the default behavior. This reduces bandwidth if it is disabled.
- You can set interval times between RDPDUs.

To control RDPDUs, perform either of the following tasks in interface configuration mode:

Task	Command
Allow CLNS to send redirect PDUs when a better route for a given host is known.	**clns send-rdpdu**
Determine the minimum interval time, in milliseconds, between RDPDUs.	**clns rdpdu-interval** *milliseconds*

Note SNPA masks are never sent, and RDPDUs are ignored by the router when the router is acting as an IS.

Configure Parameters for Locally Sourced Packets

To configure parameters for packets sourced by a specified router, perform either of the following tasks in global configuration mode:

Task	Command
Globally specify in seconds the initial lifetime for locally generated packets for the specified router.	**clns packet-lifetime** *seconds*
Specify whether to request error PDUs on packets sourced by the router.	**clns want-erpdu**

It is a good idea to set the packet lifetime low in an internetwork that has frequent loops.

Note The **clns want-erpdu** command has no effect on routing packets (ES-IS, ISO-IGRP, and IS-IS) sourced by the system. It applies to pings and trace route packets.

Monitor and Maintain the ISO CLNS Network

Use the EXEC commands described in this section to monitor and maintain the ISO CLNS caches, tables, and databases.

Task	Command
Clear and reinitialize the CLNS routing cache.	**clear clns cache**
Remove ES neighbor information from the adjacency database.	**clear clns es-neighbors**
Remove IS neighbor information from the adjacency database.	**clear clns is-neighbors**
Remove CLNS neighbor information from the adjacency database.	**clear clns neighbors**
Remove all of the dynamically derived CLNS routing information.	**clear clns route**
Invoke a diagnostic tool for testing connectivity (privileged).	**ping clns** {*host* | *address*}
Test network node reachability using a simple ping facility (user).	**ping clns** {*host* | *address*}
Display information about the CLNS network.	**show clns**
Display the entries in the CLNS routing cache.	**show clns cache**
Display ES neighbor entries including the associated areas.	**show clns es-neighbors** [*interface-type unit*] [**detail**]
Display filter expressions.	**show clns filter-expr** [*name*] [**detail**]
Display filter sets.	**show clns filter-set** [*name*]
List the CLNS-specific or ES-IS information about each interface.	**show clns interface** [*interface-type unit*]
Display IS neighbor entries according to the area in which they are located.	**show clns is-neighbors** [*interface-type unit*] [**detail**]
Display both ES and IS neighbors.	**show clns neighbors** [*interface-type unit*] [**detail**]
List the protocol-specific information for each IS-IS or ISO-IGRP routing process in this router.	**show clns protocol** [*domain* | *area-tag*]
Display all the destinations to which this router knows how to route packets.	**show clns route** [*nsap*]
Display information about the CLNS packets this router has seen.	**show clns traffic**
Display the IS-IS link state database.	**show isis database** [**level-1**] [**level-2**] [**l1**] [**l2**] [**detail**] [*lspid*]
Display the IS-IS Level 1 routing table.	**show isis routes**
Display all route maps configured or only the one specified.	**show route-map** [*map-name*]
Discover the paths taken by packets in the network (privileged).	**trace**

Task	Command
Discover the paths taken by packets in the network (user).	**trace clns** *destination*
Display the routing table in which the specified CLNS destination is found.	**which-route** {*nsap-address* | *clns-name*}

ISO CLNS Configuration Examples

The following sections provide configuration examples of both intra- and interdomain static and dynamic routing using static, ISO-IGRP, and IS-IS routing techniques:

- Configuring NETs Examples
- Basic Static Routing Examples
- Static Intradomain Routing Example
- Static Interdomain Routing Example
- Dynamic Routing within the Same Area Example
- Dynamic Routing in More Than One Area Example
- Dynamic Routing in Overlapping Areas Example
- Dynamic Interdomain Routing Example
- IS-IS Routing Configuration Examples
- Configuring a Router in Two Areas Example
- Configuring ISO CLNS over X.25 Example
- Customizing Performance Parameters Example
- Configuring DECnet Cluster Aliases Example
- Route-Map Examples
- CLNS Filter Examples

Configuring NETs Examples

The following are simple examples of configuring NETs for both ISO-IGRP and IS-IS.

ISO-IGRP

The following example illustrates specifying an NET:

```
router iso-igrp Finance
net 47.0004.004d.0001.0000.0c11.1111.00
```

The following example illustrates using a name for an NET:

```
clns host NAME 39.0001.0000.0c00.1111.00
!
router iso-igrp Marketing
net NAME
```

The use of this **net** router configuration command configures the system ID, area address, and domain address. Only a single NET per routing process is allowed.

```
router iso-igrp local
net 49.0001.0000.0c00.1111.00
```

IS-IS

The following example illustrates specifying a single NET:

```
router isis Pieinthesky
net 47.0004.004d.0001.0000.0c11.1111.00
```

The following example illustrates using a name for an NET:

```
clns host NAME 39.0001.0000.0c00.1111.00
!
router isis
net NAME
!
```

The following example illustrates the assignment of three separate area addresses for a single router using **net** commands. Traffic received that includes an area address of 47.0004.004d.0001, 47.0004.004d.0002, or 47.0004.004d.0003, and that has the same system ID, is forwarded to this router.

```
router isis eng-area1
!   |    IS-IS Area  |  System ID   |S |
net 47.0004.004d.0001.0000.0C00.1111.00
net 47.0004.004d.0002.0000.0C00.1111.00
net 47.0004.004d.0003.0000.0C00.1111.00
```

Basic Static Routing Examples

Configuring FDDI, Ethernets, Token Rings, and serial lines for CLNS can be as simple as just enabling CLNS on the interfaces. This is all that is ever required on serial lines using HDLC encapsulation. If all systems on an Ethernet or Token Ring support ISO 9542 ES-IS, then nothing else is required.

Example 1

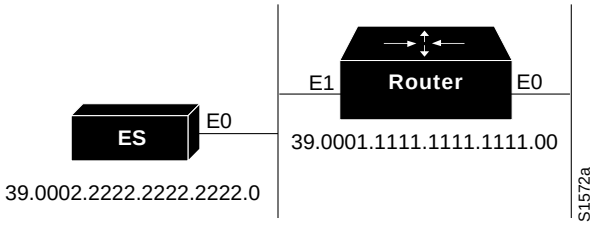
In the following example, an Ethernet and a serial line can be configured as follows:

```
! configure the following network entity title for the routing process
clns net 47.0004.004d.0055.0000.0C00.BF3B.00
! enables clns packets to be routed
clns routing
! pass ISO CLNS traffic on ethernet 0 to end systems without routing
interface ethernet 0
clns enable
interface serial 0
! pass ISO CLNS traffic on serial 0 to end systems without routing
clns enable
! creates an interface static route
clns route 47.0004.004d.0099 serial 0
clns route 47.0005 serial 0
```

Example 2

The following is a more complete example of CLNS static routing on a system with two Ethernet interfaces. After configuring routing, you define an NET and enable CLNS on the Ethernet 0 and Ethernet 1 interfaces. You must then define an ES neighbor and define a static route with the **clns route** global configuration command, as shown. In this situation, there is an ES on Ethernet 1 that does not support ES-IS. Figure 19-3 illustrates this network.

Figure 19-3 Static Routing Illustration

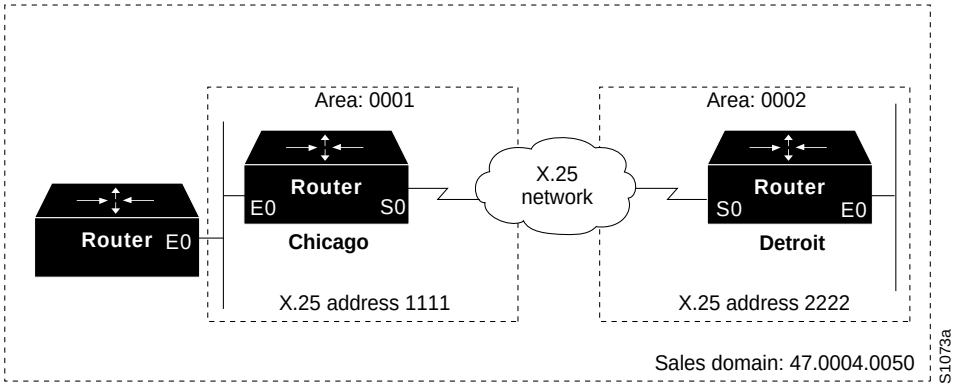



```
clns host foo 39.0001.1111.1111.1111.00
clns host bar 39.0002.2222.2222.2222.00
! assign a static address for the router
clns net foo
! enables CLNS packets to be routed
clns routing
!
interface Ethernet 0
! pass ISO CLNS packet traffic to end systems without routing them
clns enable
!
interface Ethernet 1
! pass ISO CLNS packet traffic to end systems without routing them
clns enable
! specify end system for static routing
clns es-neighbor bar 0000.0C00.62e7
! create an interface-static route to bar for packets with the following NSAP address
clns route 47.0004.000c bar
```

Static Intradomain Routing Example

Figure 19-4 and the configurations that follow demonstrate how to use static routing inside of a domain. Imagine a company with branch offices in Detroit and Chicago, connected with an X.25 link. These offices are both in the domain named Sales.

Figure 19-4 CLNS X.25 Intradomain Routing



The following example shows one way to configure the router in Chicago:

```
! defines the name chicago to be used in place of the following NSAP
clns host chicago 47.0004.0050.0001.0000.0c00.243b.00
! defines the name detroit to be used in place of the following NSAP
clns host detroit 47.0004.0050.0002.0000.0c00.1e12.00
! enable routing of CLNS packets
clns routing
router iso-igrp sales
! configure net chicago, as defined above
net chicago
!
interface ethernet 0
! specify iso-igrp routing using the previously specified tag sales
clns router iso-igrp sales
!
interface serial 0
! set the interface up as a DTE with X.25 encapsulation
encapsulation x25
x25 address 1111
x25 nvc 4
! specify iso-igrp routing using the previously specified tag sales
clns router iso-igrp sales
! define a static mapping between Detroit's nsap and its X.121 address
x25 map clns 2222 broadcast
```

This configuration brings up an X.25 virtual circuit between the router in Chicago and the router in Detroit. Routing updates will be sent across this link. This implies that the virtual circuit could be up continuously.

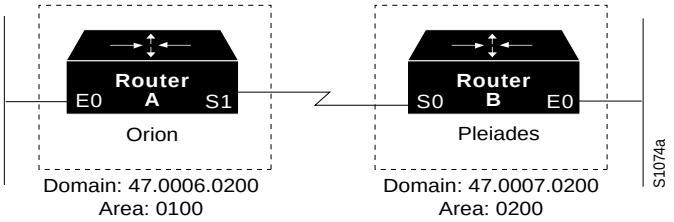
If the Chicago office should grow to contain multiple routers, it would be appropriate for each of those routers to know how to get to Detroit. Add the following command to redistribute information between routers in Chicago:

```
router iso-igrp sales
redistribute static
```

Static Interdomain Routing Example

Figure 19-5 and the example configurations that follow illustrate how to configure two routers that distribute information across domains. In this example, Router A (in domain Orion) and Router B (in domain Pleiades) communicate across a serial link.

Figure 19-5 CLNS Interdomain Static Routing



Router A

The following configuration shows how to configure Router A for static interdomain routing:

```
! defining tag orion for net 47.0006.0200.0100.0102.0304.0506.00
router iso-igrp orion
! configure the following network entity title for the routing process
net 47.0006.0200.0100.0102.0304.0506.00
! define the tag bar to be used in place of Router B's NSAP
clns host bar 47.0007.0200.0200.1112.1314.1516.00
!
interface ethernet 0
! specify iso-igrp routing using the previously specified tag orion
clns router iso-igrp orion
!
interface serial 1
! pass ISO CLNS traffic to end systems without routing
clns enable
! configure a static route to Router B
clns route 39.0001 bar
```

Router B

The following configuration shows how to configure Router B for static interdomain routing:

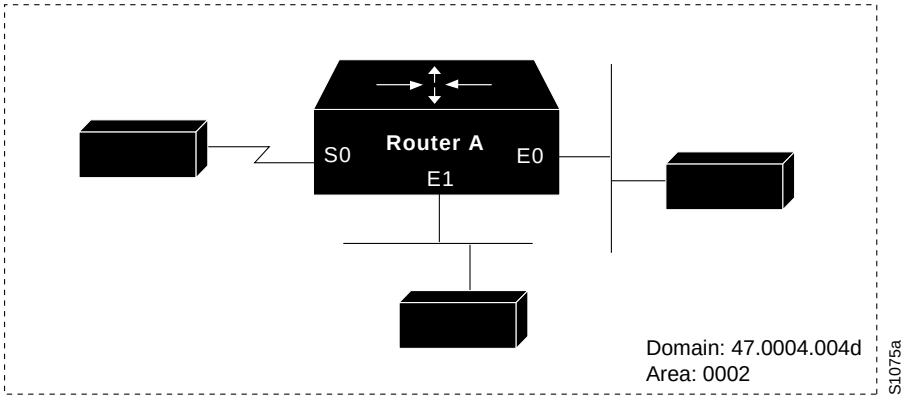
```
router iso-igrp pleiades
! configure the following network entity title for the routing process
net 47.0007.0200.0200.1112.1314.1516.00
! define the name foo to be used in place of Router A's NSAP
clns host foo 47.0006.0200.0100.0001.0102.0304.0506.00
!
interface ethernet 0
! specify iso-igrp routing using the previously specified tag pleiades
clns router iso-igrp pleiades
!
interface serial 0
! pass ISO CLNS traffic to end systems without routing
clns enable
! pass packets bound for foo in domain 47.0006.0200 through serial 0
clns route 47.0006.0200 foo
```

CLNS routing updates will not be sent on the serial link; however, CLNS packets will be sent and received over the serial link.

Dynamic Routing within the Same Area Example

Figure 19-6 and the example configuration that follows illustrate how to configure dynamic routing within a routing domain. The router can exist in one or more areas within the domain. The router named Router A exists in a single area.

Figure 19-6 CLNS Dynamic Routing within a Single Area

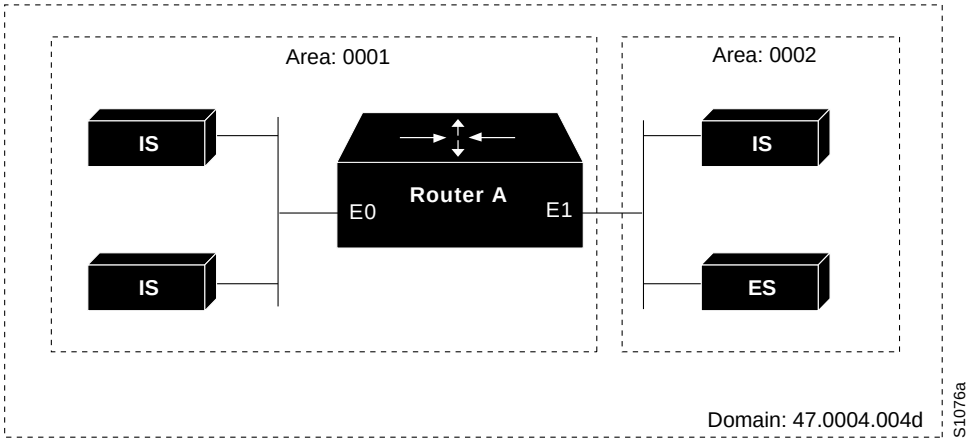



```
! enable clns packets to be routed
clns routing
! define a tag castor for the routing process
router iso-igrp castor
! configure the following net for the process in area 2, domain 47.0004.004d
net 47.0004.004d.0002.0000.0C00.0506.00
!
interface Ethernet 0
! specify iso-igrp routing using the previously specified tag castor
clns router iso-igrp castor
!
interface Ethernet 1
! specify iso-igrp routing using the previously specified tag castor
clns router iso-igrp castor
!
interface Serial 0
! specify iso-igrp routing using the previously specified tag castor
clns router iso-igrp castor
```

Dynamic Routing in More Than One Area Example

Figure 19-7 and the example configuration that follows illustrate how to configure a router named Router A that exists in two areas.

Figure 19-7 CLNS Dynamic Routing within Two Areas




```
! enable routing of clns packets
clns routing
```

```
! define a tag orion for the routing process
router iso-igrp orion
! configure the following net for the process in area 1, domain 47.0004.004d
net 47.0004.004d.0001.212223242526.00
!
interface ethernet 0
! specify iso-igrp routing using the previously specified tag orion
clns router iso-igrp orion
!
interface ethernet 1
! specify iso-igrp routing using the previously specified tag orion
clns router iso-igrp orion
```

Dynamic Routing in Overlapping Areas Example

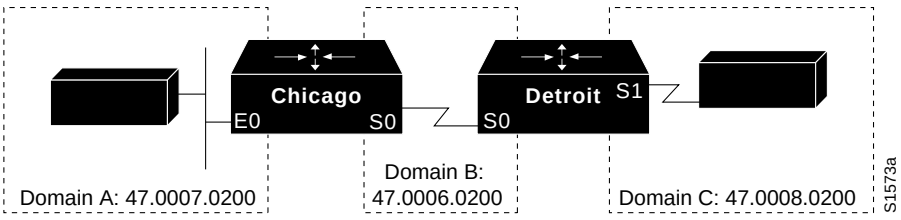
The example that follows illustrates how to configure a router with overlapping areas:

```
! enable routing of clns packets
clns routing
! define a tag capricorn for the routing process
router iso-igrp capricorn
! configure the following NET for the process in area 3, domain 47.0004.004d
net 47.0004.004d.0003.0000.0C00.0508.00
! define a tag cancer for the routing process
router iso-igrp cancer
! configure the following NET for the process in area 3, domain 47.0004.004d
net 47.0004.004d.0004.0000.0C00.0506.00
!
interface ethernet 0
! specify iso-igrp routing on interface ethernet 0 using the tag capricorn
clns router iso-igrp capricorn
!
interface ethernet 1
! specify iso-igrp routing on interface ethernet 1 using the tag capricorn
clns router iso-igrp capricorn
! specify iso-igrp routing on interface ethernet 1 using the tag cancer
clns router iso-igrp cancer
!
interface ethernet 2
! specify iso-igrp routing on interface ethernet 2 using the tag cancer
clns router iso-igrp cancer
```

Dynamic Interdomain Routing Example

Figure 19-8 and the configurations that follow illustrate how to configure three domains that are to be transparently connected.

Figure 19-8 CLNS Dynamic Interdomain Routing



Router Chicago

The following configuration shows how to configure Router Chicago for dynamic interdomain routing:

```
! enable routing of clns packets
clns routing
! define a tag A for the routing process
router iso-igrp A
! configure the following NET for the process in area 2, domain 47.0007.0200
net 47.0007.0200.0002.0102.0104.0506.00
! redistribute iso-igrp routing information throughout domain A
redistribute iso-igrp B
! define a tag B for the routing process
router iso-igrp B
! configure the following NET for the process in area 3, domain 47.0006.0200
net 47.0006.0200.0003.0102.0104.0506.00
! redistribute iso-igrp routing information throughout domain B
redistribute iso-igrp A
!
interface ethernet 0
! specify iso-igrp routing with the tag A
clns router iso-igrp A
!
interface serial 0
! specify iso-igrp routing with the tag B
clns router iso-igrp B
```

Router Detroit

The following configuration shows how to configure Router Detroit for dynamic interdomain routing. Comment lines have been eliminated from this example to avoid redundancy.

```
clns routing
router iso-igrp B
net 47.0006.0200.0004.0102.0104.0506.00
redistribute iso-igrp C
router iso-igrp C
net 47.0008.0200.0005.0102.01040.506.00
redistribute iso-igrp B
interface serial 0
clns router iso-igrp B
interface serial 1
clns router iso-igrp C
```

Chicago injects a prefix route for domain A into domain B. Domain B injects this prefix route and a prefix route for domain B into domain C.

You also can configure a border router between domain A and domain C.

IS-IS Routing Configuration Examples

The examples that follow illustrate the basic syntax and configuration command sequence for IS-IS routing.

Level 1 and Level 2 Routing

The following example illustrates using the IS-IS protocol to configure a single area address for Level 1 and Level 2 routing:

```
! enable routing of clns packets
clns routing
```

```
! route dynamically using the is-is protocol
router isis
! configure the following NET for the process in area 47.0004.004d.0001
net 47.0004.004d.0001.0000.0c00.1111.00
!
interface ethernet 0
! enable is-is routing on ethernet 0
clns router isis
!
interface ethernet 1
! enable is-is routing on ethernet 1
clns router isis
!
interface serial 0
! enable is-is routing on serial 0
clns router isis
```

Level 2 Routing Only

The following example illustrates a similar configuration, featuring a single area address being used for specification of Level 1 and Level 2 routing. However, in this case, interface serial interface 0 is configured for Level 2 routing only. Most comment lines have been eliminated from this example to avoid redundancy.

```
clns routing
router isis
net 47.0004.004d.0001.0000.0c00.1111.00
interface ethernet 0
clns router isis
interface ethernet 1
clns router isis
interface serial 0
clns router isis
! configure a level 2 adjacency only for interface serial 0
isis circuit-type level-2-only
```

OSI Configuration

The following example illustrates an OSI configuration example. In this example, IS-IS runs with two area addresses, metrics tailored, and different circuit types specified for each interface. Most comment lines have been eliminated from this example to avoid redundancy.

```
clns routing
! enable is-is routing in area 1
router isis area1
! Router is in areas 47.0004.004d.0001 and 47.0004.004d.0011
net 47.0004.004d.0001.0000.0c11.1111.00
net 47.0004.004d.0011.0000.0c11.1111.00
! enable the router to operate as a station router and an interarea router
is-type level-1-2
interface Ethernet 0
clns router isis area1
! specify a cost of 5 for the level-1 routes
isis metric 5 level-1
! establish a level-1 adjacency
isis circuit-type level-1
interface Ethernet 1
clns router isis area1
isis metric 2 level-2
isis circuit-type level-2-only
interface serial 0
clns router isis area1
```

```
isis circuit-type level-1-2
! set the priority for serial 0 to 3 for a level-1 adjacency
isis priority 3 level-1
isis priority 1 level-2
```

ISO CLNS Dynamic Route Redistribution

The following example illustrates route redistribution between IS-IS and ISO-IGRP domains. In this case, the IS-IS domain is on Ethernet interface 0; the ISO-IGRP domain is on serial interface 0. The IS-IS routing process is assigned a null tag; the ISO-IGRP routing process is assigned a tag of *remote-domain.* Most comment lines have been eliminated from this example to avoid redundancy.

```
router isis
net 39.0001.0001.0000.0c00.1111.00
! redistribute iso-igrp routing information throughout remote-domain
redistribute iso-igrp remote-domain

router iso-igrp remote-domain
net 39.0002.0001.0000.0c00.1111.00
! redistribute is-is routing information
redistribute isis

interface ethernet 0
clns router isis

interface serial 0
clns router iso-igrp remote
```

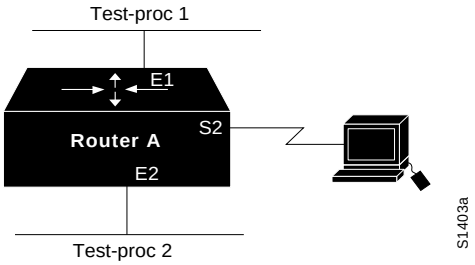
Configuring a Router in Two Areas Example

The following two examples show how to configure a router in two areas. The first example configures ISO-IGRP; the second configures IS-IS.

ISO-IGRP

In the following example, the router is in domain 49.0001 and has a system ID of aaaa.aaaa.aaaa. The router is in two areas: 31 and 40 (decimal). Figure 19-9 illustrates this configuration.

Figure 19-9 ISO-IGRP Configuration




```
clns routing
router iso-igrp test-proc1
! 001F in the following net is the hex value for area 31
net 49.0001.001F.aaaa.aaaa.aaaa.00
router iso-igrp test-proc2
! 0028 in the following net is the hex value for area 40
net 49.0001.0028.aaaa.aaaa.aaaa.00
interface ethernet 1
clns router iso-igrp test-proc1
```

```
interface s2
clns router iso-igrp test-proc1
interface ethernet 2
clns router iso-igrp test-proc2
```

IS-IS

To run IS-IS instead of ISO-IGRP, use this configuration. The illustration in Figure 19-9 still applies. Ethernet interface 2 is configured for IS-IS routing and is assigned the tag of test-proc2.

```
clns routing
router iso-igrp test-proc1
net 49.0002.0002.bbbb.bbbb.bbbb.00
router isis test-proc2
net 49.0001.0002.aaaa.aaaa.aaaa.00
interface ethernet 1
clns router iso-igrp test-proc1
interface serial 2
clns router iso-igrp test-proc1
interface ethernet 2
clns router is-is test-proc2
```

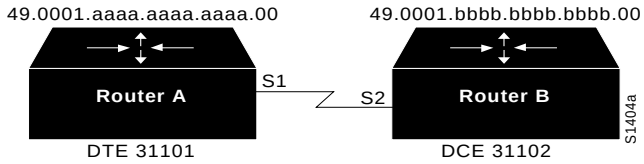
To allow CLNS packets only to blindly pass through an interface without routing updates, you could use a simple configuration. The following example shows such a configuration:

```
clns routing
interface serial 2
! permits serial 2 to pass CLNS packets without having CLNS routing turned on
clns enable
```

Configuring ISO CLNS over X.25 Example

In the following example, serial interface 1 on Router A acts as a DTE for X.25. It permits broadcasts to pass through. Router B is an IS, which has a CLNS address of 49.0001.bbbb.bbbb.bbbb.00 and an X.121 address of 31102. Router A has a CLNS address of 49.0001.aaaa.aaaa.aaaa.00 and an address of 31101. Figure 19-10 illustrates this configuration.

Figure 19-10 Routers Acting as DTEs and DCEs



Router A

```
clns routing
router iso-igrp test-proc
net 49.0001.aaaa.aaaa.aaaa.00
interface serial 1
clns router iso-igrp test-proc
! assume the host is a DTE and encapsulates x.25
encapsulation x25
! define the X.121 address of 31101 for serial 1
X25 address 31101
! set up an entry for the other side of the X.25 link (Router B)
```

```
x25 map clns 31101 broadcast
```

Router B

```
clns routing
router iso-igrp test-proc
net 49.0001.bbbb.bbbb.bbbb.00
interface serial 2
clns router iso-igrp test-proc
! configure this side as a DCE
encapsulation x25-dce
! define the X.121 address of 31102 for serial 2
X25 address 31102
! configure the NSAP of Router A and accept reverse charges
x25 map clns 31101 broadcast accept-reverse
```

Customizing Performance Parameters Example

The following example shows how to set ES hello packet (ESH) and IS hello packet (ISH) parameters in a simple ISO-IGRP configuration, as well as the MTU for a serial interface:

```
clns routing
router iso-igrp xavier
net 49.0001.004d.0002.0000.0C00.0506.00
! send IS/ES hellos every 45 seconds
clns configuration-time 45
! recipients of the hello packets keep info. in the hellos for 2 minutes
clns holding-time 120
interface serial 2
! specify an mtu of 978 bytes; generally, do not alter the default mtu value
clns mtu 978
```

Configuring DECnet Cluster Aliases Example

The following example enables cluster aliasing for CLNS:

```
clns routing
clns nsap 47.0004.004d.0001.0000.0C00.1111.00
router iso-igrp pleiades
!
interface ethernet 0
! enable cluster aliasing on interface ethernet 0
clns cluster-alias
!
interface ethernet 1
! enable cluster aliasing on interface ethernet 1
clns cluster-alias
```

Route-Map Examples

The following example redistributes two types of routes into the integrated IS-IS routing table (supporting both IP and CLNS). The first routes are OSPF external IP routes with tag 5, and these are inserted into level-2 IS-IS LSPs with a metric of 5. The second routes are ISO-IGRP derived CLNS prefix routes that match CLNS filter expression “osifilter.” These are redistributed into IS-IS as level-2 LSPs with a metric of 30.

```
router isis
redistribute ospf 109 route-map ipmap
redistribute iso-igrp nsfnet route-map osimap
```

```
route-map ipmap permit
match route-type external
match tag 5
set metric 5
set level level-2

route-map osimap permit
match clns address osifilter
set metric 30
clns filter-set osifilter permit 47.0005.80FF.FF00
```

Given the following configuration, a RIP learned route for network 160.89.0.0 and an ISO-IGRP learned route with prefix 49.0001.0002 will be redistributed into an IS-IS level-2 LSP with a metric of 5.

```
router isis
redistribute rip route-map ourmap
redistribute iso-igrp remote route-map ourmap

route-map ourmap permit
match ip address 1
match clns address ourprefix
set metric 5
set level level-2

access-list 1 permit 160.89.0.0 0.0.255.255
clns filter-set ourprefix permit 49.0001.0002...
```

CLNS Filter Examples

The following example returns a permit action if an address starts with either 47.0005 or 47.0023. It returns an implicit deny action on any other address.

```
clns filter-set US-OR-NORDUNET permit 47.0005...
clns filter-set US-OR-NORDUNET permit 47.0023...
```

The following example returns a deny action if an address starts with 39.840F, but returns a permit action for any other address:

```
clns filter-set NO-ANSI deny 38.840F...
clns filter-set NO-ANSI permit default
```

The following example builds a filter that accepts end system adjacencies with only two systems, based only on their system IDs:

```
clns filter-set ourfriends ...0000.0c00.1234.**
clns filter-set ourfriends ...0000.0c00.125a.**

interface ethernet 0
clns adjacency-filter es ourfriends
```

CHAPTER 20

Configuring Novell IPX

Novell Internet Packet Exchange (IPX) is derived from the Xerox Network Systems (XNS) Internet Datagram Protocol (IDP). IPX and XNS have the following differences:

- IPX and XNS do not always use the same Ethernet encapsulation format.
- IPX uses Novell's proprietary Service Advertisement Protocol (SAP) to advertise special network services. File servers and print servers are examples of services that are typically advertised.
- IPX uses ticks, while XNS uses hop count as the primary metric in determining the best path to a destination.

This chapter describes how to configure Novell IPX and provides configuration examples. For a complete description of the commands mentioned in this chapter, refer to the "Novell IPX Commands" chapter in the *Router Products Command Reference* publication. For historical background and a technical overview of Novell IPX, see the *Internetworking Technology Overview* publication.

Cisco's Implementation of Novell IPX

Cisco's implementation of Novell's IPX protocol has been certified as providing full IPX router functionality. A Cisco router connects Ethernet, Token Ring, and FDDI networks, either directly or through high-speed serial lines (56 kbps to T1 speeds), X.25, or Frame Relay. At this time, the Cisco X.25 and T1 support is not compatible with Novell. This means that our routers must be used on both ends of T1 and X.25 circuits.

Cisco supports the IPX MIB. The IPX Accounting group represents one of the local variables we support. This group provides access to the active database that is created and maintained if IPX accounting is enabled on a router.

Cisco routers also support IPX Enhanced IGRP, which provides the following features:

- Automatic redistribution. IPX RIP routes are automatically redistributed into Enhanced IGRP, and Enhanced IGRP routes are automatically redistributed into RIP. If desired, you can turn off redistribution. You also can completely turn off Enhanced IGRP and IPX RIP on the router or on individual interfaces.
- Increased network width. With IPX RIP, the largest possible width of your network is 15 hops. When Enhanced IGRP is enabled, the largest possible width is 224 hops. Because the Enhanced IGRP metric is large enough to support thousands of hops, the only barrier to expanding the network is the transport layer hop counter. Cisco works around this problem by incrementing the

transport control field only when an IPX packet has traversed 15 routers and the next hop to the destination was learned via Enhanced IGRP. When a RIP route is being used as the next hop to the destination, the transport control field is incremented as usual.

- Incremental SAP updates. Complete SAP updates are sent periodically on each interface until an Enhanced IGRP neighbor is found and thereafter only when there are changes to the SAP table. This procedure works by taking advantage of Enhanced IGRP's reliable transport mechanism, which means that an Enhanced IGRP peer must be present for incremental SAPs to be sent. If no peer exists on a particular interface, periodic SAPs will be sent on that interface until a peer is found. This functionality is automatic on serial interfaces and can be configured on LAN media.

IPX Addresses

An IPX network address consists of a network number and a node number expressed in the format *network.node*.

The network number identifies a physical network. It is a four-byte (32-bit) quantity that must be unique throughout the entire IPX internetwork. The network number is expressed as eight hexadecimal digits. Our router software does not require that you enter all eight digits: you can omit leading zeros.

The node number identifies a node on the network. It is a 48-bit quantity, represented by dotted triplets of four-digit hexadecimal numbers.

The following is an example of an IPX network address:

```
4a.0000.0c00.23fe
```

In this example, the network number is 4a (more specifically, it is 0000004a), and the node number is 0000.0c00.23fe. All digits in the address are hexadecimal.

IPX Configuration Task List

To configure IPX routing, complete the tasks in the following sections. At a minimum, you must enable IPX routing. The remaining tasks are optional.

- Enable IPX Routing
- Configure NLSP
- Configure IPX Enhanced IGRP
- Control Access to IPX Networks
- Tune IPX Network Performance
- Configure IPX Accounting
- Shut Down an IPX Network
- Configure IPX over WANs
- Monitor and Maintain the IPX Network

See the end of this chapter for configuration examples.

Enable IPX Routing

To enable IPX routing, you must perform the tasks described in the following sections:

- Enable IPX Routing on the Router
- Assign Network Numbers to Individual Interfaces

Enable IPX Routing on the Router

The first step in enabling IPX routing is to enable it on the router. If you do not specify the node number of the router, the router uses the hardware media access control (MAC) address currently assigned to it as its node address. This is the MAC address of the first Ethernet, Token Ring, or FDDI interface card.

To enable IPX routing on the router, perform the following global configuration task:

Task	Command
Enable IPX routing on the router.	**ipx routing** [*node*]

For an example of how to enable IPX routing, see the section "Enabling IPX Routing Example" later in this chapter.

Caution If you plan to use DECnet and IPX routing concurrently on the same interface, you should enable DECnet routing first, then enable IPX routing without specifying the optional Media Access Control (MAC) node number. If you enable IPX before enabling DECnet routing, routing for IPX will be disrupted because DECnet forces a change in the MAC-level node number.

Assign Network Numbers to Individual Interfaces

After you have enabled IPX routing on the router, you assign network numbers to individual interfaces. This has the effect of enabling IPX routing on those interfaces. When you enable IPX routing on an interface, you also can specify an encapsulation (frame type) to use for packets being transmitted on that network.

A single interface can support a single network or multiple logical networks. For a single network, you can configure any encapsulation type. Of course, it should match the encapsulation type of the servers and clients using that network number.

When assigning network numbers to an interface that supports multiple networks, you must specify a different encapsulation type for each network. Because multiple networks share the physical medium, this allows the router to determine which packets belong to which network. For example, you can configure up to four IPX networks on a single Ethernet cable, because four encapsulation types are supported for Ethernet. Again, the encapsulation type should match the servers and clients using the same network number.

The following sections describe how to enable IPX routing on interfaces that support a single network and those that support multiple networks.

Assign Network Numbers to Interfaces That Support a Single Network

To assign a network number to an interface that supports a single network, perform the following interface configuration task:

Task	Command
Enable IPX routing on an interface.	**ipx network** *network* [**encapsulation** *encapsulation-type*]

If you specify an encapsulation type, make sure you choose the one that matches that used by the servers and clients on that network.

For an example of how to enable IPX routing, see the section "Enabling IPX Routing Example."

Assign Network Numbers to Interfaces that Support Multiple Networks

To assign network numbers to interfaces that support multiple networks, you normally use subinterfaces. A subinterface is a mechanism that allows a single physical interface to support multiple logical interfaces or networks. That is, several logical interfaces or networks can be associated with a single hardware interface. Each subinterface must use a distinct encapsulation, and the encapsulation must match that of the clients and servers using the same network number. To run NLSP on multiple networks on the same physical LAN interface, you must configure subinterfaces.

Any interface configuration parameters that you specify on an individual subinterface are applied to that subinterface only.

To configure multiple IPX networks on a physical interface using subinterfaces, perform the following tasks starting in global configuration mode:

Task		Command
Step 1	Specify a subinterface.	**interface** *type interface-number.subinterface-number*
Step 2	Enable IPX routing, specifying the first encapsulation type.	**ipx network** *number* **encapsulation** *encapsulation-type*

To configure more than one subinterface, repeat these two steps.

Note When enabling NLSP and configuring multiple encapsulations on the same physical LAN interface, you must use subinterfaces. You cannot use secondary networks.

For examples of configuring multiple IPX networks on an interface, see the section "Enabling and Disabling IPX Routing on Multiple Networks Example" later in this chapter.

Table 20-1 lists the encapsulation types you can use on IEEE interfaces and shows the correspondence between the encapsulation type and the IPX frame type.

Table 20-1 Novell IPX Encapsulation Types on IEEE Interfaces

Interface Type	Encapsulation Type	IPX Frame Type
Ethernet	novell-ether (default)	Ethernet_802.3
	arpa	Ethernet_II
	sap	Ethernet_802.2
	snap	Ethernet_Snap
Token Ring	sap (default)	Token-Ring
	snap	Token-Ring_Snap
FDDI	snap (default)	Fddi_Snap
	sap	Fddi_802.2

When assigning network numbers to interfaces that support multiple networks, you can also configure primary and secondary networks. The first logical network you configure on an interface is considered the primary network. Any additional networks are considered secondary networks. Again, each network on an interface must use a distinct encapsulation and it should match that of the clients and servers using the same network number.

Any interface configuration parameters that you specify on this interface are applied to all the logical networks. For example, if you set the routing update timer to 120 seconds, this value is used on all four networks.

To use primary and secondary networks to configure multiple IPX networks on an interface, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Enable IPX routing on the primary network.	**ipx network** *network* **encapsulation** *encapsulation-type*
Step 2	Enable IPX routing on a secondary network.	**ipx network** *network* **encapsulation** *encapsulation-type* **secondary**

To configure more than one secondary network, repeat Step 2 as appropriate.

Note When enabling NLSP and configuring multiple encapsulations on the same physical LAN interface, you must use subinterfaces. You cannot use secondary networks.

Configure NLSP

The NetWare Link Services Protocol (NLSP) is a link-state routing protocol based on the Open Systems Interconnection (OSI) Intermediate System to Intermediate System (IS-IS) protocol.

NLSP is designed to be used in a hierarchical routing environment, in which networked systems are grouped into routing areas. Routing areas can then be grouped into routing domains, and domains can be grouped into an internetwork.

Level 1 routers are used to connect networked systems within a given routing area. Areas are connected to each other by Level 2 routers, and domains are connected by Level 3 routers. A Level 2 router also acts as a Level 1 router within its own area; likewise, a Level 3 router also acts as a Level 2 router within its own domain.

The current NLSP specification defines only Level 1 procedures, which allow operation within a routing area and routing to the nearest Level 2 router only.

The router at each level of the topology stores complete information for its level. For instance, Level 1 routers store complete link-state information about their entire area. This information includes a record of all the routers in the area, the links connecting them, the operational status of the routers and links, and other related parameters. For each point-to-point link, the database records the end-point routers and the state of the link. For each LAN, the database records which routers are connected to the LAN. Similarly, Level 2 routers would store information about all the areas in the routing domain, and Level 3 routers would store information about all the domains in the internetwork.

Our implementation of NLSP is based on revision 1.0 of the Novell NLSP specification, which specifies routing with a routing area (that is, Level 1 routing). Our implementation of NLSP also includes NLSP MIB variables.

NLSP is a link-state protocol. This means that every router in a routing area maintains an identical copy of the link-state database, which contains all information about the topology of the area. All routers synchronize their views of the databases among themselves to keep their copies of the link-state databases consistent. NLSP has three major databases:

- Adjacency—keeps track of the router's immediate neighbors and the operational status of the directly attached links by exchanging hello packets. Adjacencies are created upon receipt of periodic hello packets. If a link or router goes down, adjacencies time out and are deleted from the database.
- Link state—tracks the connectivity of an entire routing area by aggregating the immediate neighborhood information from all routers into link-state packets (LSPs). Link-state packets contain lists of adjacencies. They are flooded to all other routers via a reliable flooding algorithm every time a link state changes. LSPs are refreshed every two hours. To keep the size of the link-state database reasonable, NLSP uses fictitious pseudonodes, which represent the LAN as a whole, and designated routers, which originate LSPs on behalf of the pseudonode.
- Forwarding—calculated from the adjacency and link state databases using Dijkstra's shortest path first (SPF) algorithm.

To configure NLSP, you must have configured IPX routing on your router, as described earlier in this chapter. Then, you must perform the tasks described in the following sections:

- Define an Internal Network
- Enable NLSP Routing on the Router
- Configure NLSP on an Interface

You can optionally perform the tasks described in the following sections:

- Configure RIP and SAP Compatibility
- Configure Maximum Hop Count
- Configure the Link Delay and Throughput
- Configure the Metric Value
- Configure the Priority of the System for Designated Router Election
- Configure Default Routes
- Configure Transmission and Retransmission Intervals
- Modify Link-State Packet (LSP) Parameters

For an example of enabling NLSP, see the section "Enabling and Disabling IPX Routing Protocols Examples" later in this chapter.

Define an Internal Network

An internal network number is a number assigned to the router. In order for NLSP to operate, you must configure an internal network number for each router.

To enable IPX routing and define an internal network numbers, perform the following task in global configuration mode:

Task	Command
Enable IPX routing.	**ipx routing**
Define an internal network number.	**ipx internal-network** *network-number*

Enable NLSP Routing on the Router

To enable NLSP on the router, perform the following tasks starting in global configuration mode:

Task		Command
Step 1	Enable NLSP on the router.	**ipx router nlsp**
Step 2	Define a set of network numbers to be part of the current NLSP area.	**area-address** *address mask*

Configure NLSP on an Interface

You configure NLSP differently on LAN and WAN interfaces, as described in the following sections.

Configure NLSP on a LAN Interface

To configure NLSP on a LAN interface, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Enable IPX routing on an interface.	**ipx network** *number* [**encapsulation** *encapsulation-type*]
Step 2	Enable NLSP on the interface.	**ipx nlsp enable**

To configure multiple encapsulations on the same physical LAN interfaces, you must configure subinterfaces. Each subinterface must have a different encapsulation type. To do this, perform the following tasks starting in global configuration mode:

Task		Command
Step 1	Specify a subinterface.	**interface** *type interface-number.subinterface-number*
Step 2	Enable IPX routing, specifying the first encapsulation type.	**ipx network** *number* **encapsulation** *encapsulation-type*
Step 3	Enable NLSP on the subinterface.	**ipx nlsp enable**

Repeat these three steps for each subinterface.

Note When enabling NLSP and configuring multiple encapsulations on the same physical LAN interface, you must use subinterfaces. You cannot use secondary networks.

Configure NLSP on a WAN Interface

To configure NLSP on a WAN interface, perform the following tasks starting in global configuration mode:

Task		Command
Step 1	Specify a serial interface.	**interface serial** *number*
Step 2	Enable IPXWAN.	**ipx ipxwan** [*local-node* **unnumbered** *local-server-name retry-interval retry-limit*]

Task	Command
Step 3 Enable NLSP on the interface.	**ipx nlsp enable**

Configure RIP and SAP Compatibility

RIP and SAP are enabled by default on all interfaces configured for IPX, and these interfaces always respond to RIP and SAP requests. When you also enable NLSP on an interface, the interface, by default, generates and sends RIP and SAP periodic traffic only if another RIP router or SAP service is sending RIP or SAP traffic.

To modify the generation of periodic RIP updates on a network enabled for NLSP, perform one of the following tasks in interface configuration mode:

Task	Command
Never generate RIP periodic traffic.	**ipx nlsp rip off**
Always generate RIP periodic traffic.	**ipx nlsp rip on**
Send RIP periodic traffic only if another RIP router is sending periodic RIP traffic. (This is the default on interfaces configured for NLSP.)	**ipx nlsp rip auto**

To modify the generation of periodic SAP updates on a network enabled for NLSP, perform one of the following tasks in interface configuration mode:

Task	Command
Never generate SAP periodic traffic.	**ipx nlsp sap off**
Always generate SAP periodic traffic.	**ipx nlsp sap on**
Send SAP periodic traffic only if another SAP service is sending periodic SAP traffic. (This is the default on interfaces configured for NLSP.)	**ipx nlsp sap auto**

Configure Maximum Hop Count

By default, IPX packets whose hop count exceeds 15 are discarded. In larger internetworks, this may be insufficient. You can increase the hop count to a maximum of 254 hops. To modify the maximum hop count, perform the following task in global configuration mode:

Task	Command
Set the maximum hop count accepted from RIP update packet**s.**	**ipx maximum-hops** *hop*

Configure the Link Delay and Throughput

The delay and throughput of each link are used by NLSP as part of its route calculations. By default, these parameters are set to appropriate values or, in the case of IPXWAN, are dynamically measured.

The link delay and throughput you specify replaces the default value or overrides the value measured by IPXWAN when it starts. The value is also supplied to NLSP for use in metric calculations.

To change the link delay, perform the following task in interface configuration mode:

Task	Command
Specify the link delay.	**ipx link-delay** *microseconds*

To change the throughput, perform the following task in interface configuration mode:

Task	Command
Specify the throughput.	**ipx throughput** *bits-per-second*

Configure the Metric Value

NLSP assigns a default link cost (metric) based on the link throughput. If desired, you can set the link cost manually. To set the NLSP link cost for an interface, perform the following task in interface configuration mode:

Task	Command
Set the metric value for an interface.	**ipx nlsp metric** *metric-number*

Configure the Priority of the System for Designated Router Election

NLSP elects a designated router on each LAN interface. This router creates a virtual router called a pseudonode, which generates routing information on behalf of the LAN and transmits it to the rest of the routing area. The routing information generated includes adjacencies and RIP routes. The use of a designated router significantly reduces the number of entries in the adjacency database.

By default, electing a designated router is done automatically. However, you can manually affect the identity of the designated router by changing the priority of the system: the system with the highest priority is elected to be the designated router.

By default, the priority of the system is 44. To change it, perform the following task in interface configuration mode:

Task	Command
Configure the designated router election priority.	**ipx nlsp priority** *priority-number*

Configure Default Routes

The default route is used when a route to any destination network is unknown. By default, IPX treats network number –2 (0xFFFFFFFE) as the default route. To disable the use of this default route, perform the following task in global configuration mode:

Task	Command
Disable default route handling.	**ipx default-route**

Unless configured otherwise, all known routes are advertised out each interface. However, you can choose to advertise only the default route if it is known. This greatly reduces the CPU overhead when routing tables are large. Note that services are not considered to be reachable via the default route alone. A specific route to the destination network must be known before a service advertisement will be accepted. Therefore, advertise only the default route with caution if services are to be advertised via the interface.

To advertise only the default route via an interface, perform the following task in interface configuration mode:

Task	Command
Advertise only the default route.	**ipx advertise-default-route-only** *network*

Configure Transmission and Retransmission Intervals

You can configure the hello and CSNP transmission intervals, and the LSP retransmission interval.

To configure the hello transmission interval, perform the following task in interface configuration mode:

Task	Command
Configure the hello transmission interval.	**ipx nlsp hello-interval** *seconds*

To configure the CSNP transmission interval, perform the following task in interface configuration mode:

Task	Command
Configure the CSNP transmission interval.	**ipx nlsp csnp-interval** *seconds*

To configure the LSP retransmission interval, perform the following task in interface configuration mode:

Task	Command
Configure the LSP retransmission interval.	**ipx nlsp retransmit-interval** *seconds*

Modify Link-State Packet (LSP) Parameters

To modify LSP parameters, perform one or more of the following tasks in router configuration mode:

Task	Command
Set the minimum LSP generation interval.	**lsp-gen-interval** *seconds*
Set the maximum time the LSP persists.	**max-lsp-lifetime** *seconds*
Set the LSP refresh time.	**lsp-refresh-interval** *seconds*
Set the maximum size of a link-state packet.	**lsp-mtu** *bytes*
Set the minimum time between SPF calculations.	**spf-interval** *seconds*

Configure IPX Enhanced IGRP

Enhanced IGRP is an enhanced version of the Interior Gateway Routing Protocol (IGRP) developed by Cisco Systems, Inc. Enhanced IGRP uses the same distance vector algorithm and distance information as IGRP. However, the convergence properties and the operating efficiency of Enhanced IGRP have improved significantly over IGRP.

The convergence technology is based on research conducted at SRI International and employs an algorithm referred to as the Diffusing Update Algorithm (DUAL). This algorithm guarantees loop-free operation at every instant throughout a route computation and allows all routers involved in a topology change to synchronize at the same time. Routers that are not affected by topology changes are not involved in recomputations. The convergence time with DUAL rivals that of any other existing routing protocol.

Enhanced IGRP offers the following features:

- Fast convergence. The DUAL algorithm allows routing information to converge as quickly as any currently available routing protocol.
- Partial updates. Enhanced IGRP sends incremental updates when the state of a destination changes, instead of sending the entire contents of the routing table. This feature minimizes the bandwidth required for Enhanced IGRP packets.
- Less CPU usage than IGRP. This occurs because full update packets do not have to be processed each time they are received.
- Neighbor discovery mechanism. This is a simple hello mechanism used to learn about neighboring routers. It is protocol-independent.
- Scaling. Enhanced IGRP scales to large networks.

Enhanced IGRP has four basic components:

- Neighbor discovery/recovery
- Reliable transport protocol
- DUAL finite state machine
- Protocol-dependent modules

Neighbor discovery/recovery is the process that routers use to dynamically learn of other routers on their directly attached networks. Routers must also discover when their neighbors become unreachable or inoperative. Neighbor discovery/recovery is achieved with low overhead by periodically sending small hello packets. As long as hello packets are received, a router can determine that a neighbor is alive and functioning. Once this status is determined, the neighboring routers can exchange routing information.

The reliable transport protocol is responsible for guaranteed, ordered delivery of Enhanced IGRP packets to all neighbors. It supports intermixed transmission of multicast and unicast packets. Some Enhanced IGRP packets must be transmitted reliably and others need not be. For efficiency, reliability is provided only when necessary. For example, on a multiaccess network that has multicast capabilities, such as Ethernet, it is not necessary to send hellos reliably to all neighbors individually. Therefore, Enhanced IGRP sends a single multicast hello with an indication in the packet informing the receivers that the packet need not be acknowledged. Other types of packets, such as updates, require acknowledgment, and this is indicated in the packet. The reliable transport has a provision to send multicast packets quickly when there are unacknowledged packets pending. Doing so helps ensure that convergence time remains low in the presence of varying speed links.

The DUAL finite state machine embodies the decision process for all route computations. It tracks all routes advertised by all neighbors. DUAL uses the distance information, known as a metric, to select efficient, loop-free paths. DUAL selects routes to be inserted into a routing table based on feasible successors. A successor is a neighboring router used for packet forwarding that has a least-cost path to a destination that is guaranteed not to be part of a routing loop. When there are no feasible successors but there are neighbors advertising the destination, a recomputation must occur. This is the process whereby a new successor is determined. The amount of time it takes to recompute the route affects the convergence time. Even though the recomputation is not processor intensive,

it is advantageous to avoid recomputation if it is not necessary. When a topology change occurs, DUAL will test for feasible successors. If there are feasible successors, it will use any it finds in order to avoid unnecessary recomputation.

The protocol-dependent modules are responsible for network layer protocol-specific tasks. It is also responsible for parsing Enhanced IGRP packets and informing DUAL of the new information received. Enhanced IGRP asks DUAL to make routing decisions, but the results are stored in the IPX routing table. Also, Enhanced IGRP is responsible for redistributing routes learned by other IPX routing protocols.

IPX Enhanced IGRP Configuration Task List

tasks in the following sectionstasks. At a minimum, you must enable Enhanced IGRP. The remaining tasks are optional.

- Enable IPX Enhanced IGRP
- Configure Miscellaneous Enhanced IGRP Parameters

Enable IPX Enhanced IGRP

To create an IPX Enhanced IGRP routing process, perform the following tasks:

Task		Command
Step 1	Enable an Enhanced IGRP routing process in global configuration mode.	**ipx router eigrp** *autonomous-system-number*
Step 2	Enable Enhanced IGRP on a network in IPX router configuration mode.	**network** {*network-number* | **all**}

For an example of how to enable Enhanced IGRP, see the section “Enabling IPX Enhanced IGRP Example.”

To associate multiple networks with an Enhanced IGRP routing process, you can repeat Step 2.

Configure Miscellaneous Enhanced IGRP Parameters

To configure the following miscellaneous Enhanced IGRP parameters, perform one or more of the following tasks:

- Redistribute Routing Information
- Adjust the Interval between Hello Packets and the Hold Time
- Disable Split Horizon
- Control SAP Updates
- Control the Advertising of Routes in Routing Updates
- Control the Processing of Routing Updates
- Query the Backup Server

Redistribute Routing Information

By default, the router redistributes IPX RIP routes into Enhanced IGRP, and vice versa. When routes are redistributed, a RIP route to a destination with a hop count of 1 is always preferred over an Enhanced IGRP route with a hop count of 1. This ensures that the router always believes a Novell IPX server over a Cisco router for internal IPX networks. The only exception to this rule is if both the RIP and Enhanced IGRP updates were received from the same router. In this case, and in the case of all other RIP metrics (2 through 15), the Enhanced IGRP route always is preferred over the RIP route when the hop counts are the same.

Internal Enhanced IGRP routes are always preferred over external Enhanced IGRP routes. This means that if there are two Enhanced IGRP paths to a destination, the path that originated within the Enhanced IGRP autonomous sytem will always be preferred over the Enhanced IGRP path that originated from outside of the autonomous system, regardless of the metric. Redistributed RIP routes are alwaysadvertised in Enhanced IGRP as external.

To disable route redistribution, perform the following task in IPX router configuration mode:

Task	Command
Disable redistribution of RIP routes into Enhanced IGRP and Enhanced IGRP routes into RIP.	**no redistribute** {**rip** | **eigrp** *autonomous-system-number* | **connected** | **static**}

Adjust the Interval between Hello Packets and the Hold Time

You can adjust the interval between hello packets and the hold time.

Routers periodically send hello packets to each other to dynamically learn of other routers on their directly attached networks. Routers use this information to discover who their neighbors are and to discover when their neighbors become unreachable or inoperative. By default, hello packets are sent every 5 seconds.

You can configure the hold time, in seconds, on a specified interface for the Enhanced IGRP routing process designated by the autonomous system number. The hold time is advertised in hello packets and indicates to neighbors the length of time they should consider the sender valid. The default hold time is three times the hello interval, or 15 seconds.

To change the interval between hello packets, perform the following task in interface configuration mode:

Task	Command
Set the interval between hello packets.	**ipx hello-interval eigrp** *autonomous-system-number seconds*

On very congested and large networks, 15 seconds may not be sufficient time for all routers to receive hello packets from their neighbors. In this case, you may want to increase the hold time. To do this, perform the following task in interface configuration mode:

Task	Command
Set the hold time.	**ipx hold-time eigrp** *autonomous-system-number seconds*

Note Do not adjust the hold time without advising technical support.

Disable Split Horizon

Split horizon controls the sending of Enhanced IGRP update and query packets. If split horizon is enabled on an interface, these packets are not sent for destinations if this interface is the next hop to that destination.

By default, split horizon is enabled on all interfaces.

Split horizon blocks information about routes from being advertised by a router out any interface from which that information originated. This behavior usually optimizes communication among multiple routers, particularly when links are broken. However, with nonbroadcast networks, such as Frame Relay and SMDS, situations can arise for which this behavior is less than ideal. For these situations, you may wish to disable split horizon.

To disable split horizon, perform the following task in interface configuration mode:

Task	Command
Disable split horizon.	**no ipx split-horizon eigrp** *autonomous-system-number*

Control SAP Updates

If IPX Enhanced IGRP peers are found on an interface, you can configure the router to send SAP updates either periodically or when a change occurs in the SAP table. When no IPX Enhanced IGRP peer is present on the interface, periodic SAPs are always sent.

On serial lines, by default, if an Enhanced IGRP neighbor is present, the router sends SAP updates only when the SAP table changes. On Ethernet, Token Ring, and FDDI interfaces, by default, the router sends SAP updates periodically. To reduce the amount of bandwidth required to send SAP updates, you might want to disable the periodic sending of SAP updates on LAN interfaces. Do this only when all nodes out this interface are Enhanced IGRP peers; otherwise, loss of SAP information on the other nodes will result.

To send SAP updates only when a change occurs in the SAP table, perform the following task in interface configuration mode:

Task	Command
Send SAP updates only when a change in the SAP table occurs, and send SAP changes only.	**ipx sap-incremental eigrp** *autonomous-system-number* **rsup-only**

To send periodic SAP updates, perform the following task in interface configuration mode:

Task	Command
Send SAP updates periodically.	**no ipx sap-incremental eigrp** *autonomous-system-number*

For an example of how to configure SAP updates, see the section “Enhanced IGRP SAP Update Examples” later in this chapter.

Control the Advertising of Routes in Routing Updates

To control which routers learn about routes, you can control the advertising of routesin routing updates. To do this, perform the following task in router configuration mode:

Task	Command
Control the advertising of routes in routing updates.	**distribute-list** *access-list-number* **out** [*interface-name* | *routing-process*]

Control the Processing of Routing Updates

To control the processing of routes listed in incoming updates, perform the following task in router configuration mode:

Task	Command
Control which incoming route updates are processes.	**distribute-list** *access-list-number* **in** [*interface-name*]

Query the Backup Server

The backup server table is a table kept for each Enhanced IGRP peer. It lists the IPX servers that have been advertised by that peer. If a server is removed from the main server table at any time and for any reason, the router examines the backup server table to see if this just-removed server is known by any of the Enhanced IGRP peers. If it is, the information from that peer is advertised back into the main server table just as if that peer had readvertised the server information to this router. Using this method to allow the router to keep the backup server table consistent with what is advertised by each peer means that only changes to the table need to be advertised between Enhanced IGRP routers; full periodic updates do not need to be sent.

By default, the router queries its own copy of each Enhanced IGRP neighbor's backup server table every 15 seconds. To change this interval, perform the following global configuration task:

Task	Command
Specify the minimum period of time between successive queries of a neighbor's backup server table.	**ipx backup-server-query-interval** *interval*

Control Access to IPX Networks

To control access to IPX networks, you create access lists and then apply them with filters to individual interfaces.

There are four types of IPX access lists that you can use to filter various kinds of traffic:

- Standard access list—Restricts traffic based on the source network number. You can further restrict traffic by specifying a destination address and a source and destination address mask. Standard IPX access lists have numbers from 800 to 899.
- Extended access list—Restricts traffic based on the IPX protocol type. You can further restrict traffic by specifying source and destination addresses and address masks, and source and destination sockets. Extended IPX access lists have numbers from 900 to 999.
- SAP access list—Restricts traffic based on the IPX Service Advertisement Protocol (SAP) type. These lists are used for SAP filters and Get Nearest Server (GNS) response filters. Novell SAP access lists have numbers from 1000 to 1099.

- IPX NetBIOS access list—Restricts IPX NetBIOS traffic based on NetBIOS names, not numbers.

There are 13 different IPX filters that you can define for IPX interfaces. They fall into five groups:

- Generic output filters—Control which packets are routed out an interface based on the packet's source and destination addresses and IPX protocol type.
- Routing table filters—Control which Routing Information Protocol (RIP) updates are accepted and advertised by the router and which routers the local router accepts RIP updates from.
- SAP filters—Control which SAP services the router accepts and advertises and which Get Nearest Server (GNS) response messages it sends out.
- IPX NetBIOS filters—Control incoming and outgoing IPX NetBIOS packets.
- Broadcast filters—Control which broadcast packets are forwarded.

Table 20-2 summarizes the filters and the commands you use to define them. Use the **show ipx interfaces** command to display the filters defined on an interface.

Table 20-2 IPX Filters

Filter Type	Command Used to Define Filter
Generic filters	
Filter outbound packets based on protocol, address and address mask, and socket.	**ipx access-group** *access-list-number*
Routing table filters	
Control which networks are added to the routing table.	**ipx input-network-filter** *access-list-number*
Control which networks are advertised in routing updates.	**ipx output-network-filter** *access-list-number*
Control the routers from which updates are accepted.	**ipx router-filter** *access-list-number*
SAP filters	
Filter incoming service advertisements.	**ipx input-sap-filter** *access-list-number*
Filter outgoing service advertisements.	**ipx output-sap-filter** *access-list-number*
Control the routers from which SAP updates are accepted.	**ipx router-sap-filter** *access-list-number*
Filter list of servers in GNS response messages.	**ipx output-gns-filter** *access-list-number*
IPX NetBIOS filters	
Filter incoming packets by node name.	**ipx netbios input-access-filter host** *name*
Filter incoming packets by byte pattern.	**ipx netbios input-access-filter bytes** *name*
Filter outgoing packets by node name.	**ipx netbios output-access-filter host** *name*
Filter outgoing packets by byte pattern.	**ipx netbios output-access-filter bytes** *name*
Broadcast filters	
Control which broadcast packets are forwarded.	**ipx helper-list** *access-list-number*

Keep the following in mind when configuring IPX network access control:

- Access lists entries are scanned in the order you enter them. The first matching entry is used. To improve performance, it is recommended that you place the most commonly used entries near the beginning of the access list.
- An implicit *deny everything* entry is 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 re-enter it with the new entries.
- Take care not to set up conditions that result in packets getting lost. One way this can happen is when a router or interface is configured to advertise services on a network that has access lists that deny these packets.

You perform the tasks in one or more of the following sections to control access to IPX networks:

- Create Access Lists
- Create Generic Filters
- Create Filters for Updating the Routing Table
- Create SAP Filters
- Create GNS Response Filters
- Create IPX NetBIOS Filters
- Create Broadcast Message Filters

Create Access Lists

To create access lists, you can perform one or more of the following tasks in global configuration mode:

Task	Command
Create a standard IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-node*[*source-node-mask*]] [*destination-network*[.*destination-node* [*destination-node-mask*]]]
Create an extended IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[**.***source-node* [*source-network-mask***.***source-node-mask*]] *source-socket* [*destination-network* [**.***destination-node* [*destination-network-mask***.***destination-node-mask*]] *destination-socket*]]
Create an IPX access list for SAP filters.	**access-list** *access-list-number* {**deny** | **permit**} *network*[**.***node*] [*network-mask node-mask*] [*service-type* [*server-name*]]
Create an access list for filtering IPX NetBIOS packets by node name.	**netbios access-list host** *name* {**deny** | **permit**} *string*
Create an access list for filtering IPX NetBIOS packets by arbitrary byte pattern.	**netbios access-list bytes** *name* {**deny** | **permit**} *offset byte-pattern*

Once you have created an access list, apply it to a filter on the appropriate interfaces as described in the sections that follow. This activates the access list.

Create Generic Filters

Generic filters determine which packets to send out an interface based on the packet's source and destination addresses, IPX protocol type, and source and destination socket numbers.

To create generic filters, perform the following tasks:

Step 1 Create a standard or an extended access list.

Step 2 Apply a filter to an interface.

To create an access list, perform one of the following tasks in global configuration mode:

Task	Command
Create a standard IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-node*[*source-node-mask*]] [*destination-network*[.*destination-node* [*destination-node-mask*]]]
Create an extended IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[**.***source-node* [*source-network-mask***.***source-node-mask*]] *source-socket* [*destination-network* [**.***destination-node* [*destination-network-mask***.***destination-node-mask*]] *destination-socket*]]

To apply a generic filter to an interface, perform the following task in interface configuration mode:

Task	Command
Apply a generic filter to an interface.	**ipx access-group** *access-list-number*

For an example of creating a generic filter, see the section "IPX Network Access Example."

Create Filters for Updating the Routing Table

Routing table update filters control the entries that the router accepts for its routing table and the networks that it advertises in its routing updates.

To create filters to control updating of the routing table, perform the following tasks:

Step 1 Create a standard or an extended access list.

Step 2 Apply one or more routing filters to an interface.

To create an access list, perform one of the following tasks in global configuration mode:

Task	Command
Create a standard IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-node*[*source-node-mask*]] [*destination-network*[.*destination-node* [*destination-node-mask*]]]

Task	Command
Create an extended IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[**.***source-node* [*source-network-mask***.***source-node-mask*]] *source-socket* [*destination-network* [**.***destination-node* [*destination-network-mask***.***destination-node-mask*]] *destination-socket*]]

To apply routing table update filters to an interface, perform one or more of the following tasks in interface configuration mode:

Task	Command
Control which networks are added to the routing table when IPX routing updates are received.	**ipx input-network-filter** *access-list-number*
Control which networks are advertised in routing updates sent out by the router.	**ipx output-network-filter** *access-list-number*
Control the routers from which routing updates are accepted.	**ipx router-filter** *access-list-number*

You can apply one of each of these filters to each interface.

Create SAP Filters

A common source of traffic on Novell networks is SAP messages, which are generated by NetWare servers and our routers when they broadcast their available services. To control how SAP messages from network segments or specific servers are routed among IPX networks, perform the following steps:

Step 1 Create a SAP access list.

Step 2 Apply one or more filters to an interface.

To create a SAP access list, perform the following task in global configuration mode:

Task	Command
Create a SAP access list.	**access-list** *access-list-number* {**deny** | **permit**} *network*[*.node*] [*network.node-mask*] [*service-type* [*server-name*]]

To apply SAP filters to an interface, perform one or more of the following tasks in interface configuration mode:

Task	Command
Filter incoming service advertisements.	**ipx input-sap-filter** *access-list-number*
Filter outgoing service advertisements.	**ipx output-sap-filter** *access-list-number*
Filter service advertisements received from a particular router.	**ipx router-sap-filter** *access-list-number*

You can apply one of each SAP filter to each interface.

For examples of creating and applying SAP filters, see the sections "SAP Input Filter Example" and "SAP Output Filter Example " later in this chapter.

Create GNS Response Filters

To create filters for controlling which servers are included in the GNS responses sent by the router, perform the following tasks:

Step 1 Create a SAP access list.

Step 2 Apply a GNS filter to an interface.

To create a SAP access list, perform the following task in global configuration mode:

Task	Command
Create a SAP access list.	**access-list** *access-list-number* {**deny** | **permit**} *network*[.*node*] [*network.node-mask*] [*service-type* [*server-name*]]

To apply a GNS filter to an interface, perform the following task in interface configuration mode:

Task	Command
Filter the list of servers in GNS response messages.	**ipx output-gns-filter** *access-list-number*

Create IPX NetBIOS Filters

Novell's IPX NetBIOS allows messages to be exchanged between nodes using alphanumeric names as well as node addresses. Therefore, the router lets you filter incoming and outgoing NetBIOS packets by the node name or by an arbitrary byte pattern (such as the node address) in the packet.

Note These filters apply to IPX NetBIOS packets only. They have no effect on LLC2 NetBIOS packets.

Keep the following in mind when configuring IPX NetBIOS access control:

- Host (node) names are case sensitive.
- Host and byte access lists can have the same names because the two types of lists are independent of each other.
- When filtering by node name, the names in the access lists are compared with the destination name field for IPX NetBIOS "find name" requests.
- Access filters that filter by byte offset can have a significant impact on the packet transmission rate because each packet must be examined. You should use these access lists only when absolutely necessary.
- If a node name is not found in an access list, the default action is to deny access.

To create filters for controlling IPX NetBIOS access, perform the following tasks:

Step 1 Create a NetBIOS access list.

Step 2 Apply the access list to an interface.

To create one or more NetBIOS access lists, perform one or both of the following tasks in global configuration mode:

Task	Command
Create an access list for filtering IPX NetBIOS packets by node name.	**netbios access-list host** *name* {**deny** | **permit**} *string*
Create an access list for filtering IPX NetBIOS packets by arbitrary byte pattern.	**netbios access-list bytes** *name* {**deny** | **permit**} *offset byte-pattern*

To apply a NetBIOS access list to an interface, perform one or more of the following tasks in interface configuration mode:

Task	Command
Filter incoming packets by node name.	**ipx netbios input-access-filter host** *name*
Filter incoming packets by byte pattern.	**ipx netbios input-access-filter bytes** *name*
Filter outgoing packets by node name.	**ipx netbios output-access-filter host** *name*
Filter outgoing packets by byte pattern.	**ipx netbios output-access-filter bytes** *name*

You can apply one of each of these four filters to each interface.

Create Broadcast Message Filters

Routers normally block all broadcast requests and do not forward them to other network segments. This is done to prevent the degradation of performance inherent in broadcast traffic over the entire network. You can define which broadcast messages get forwarded to other networks by applying a broadcast message filter to an interface.

To create filters for controlling broadcast messages, perform the following tasks:

Step 1 Create an access list.

Step 2 Apply a broadcast message filter to an interface.

To create an access list, perform one of the following tasks in global configuration mode:

Task	Command
Create a standard IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-node*[*source-node-mask*]] [*destination-network*[.*destination-node* [*destination-node-mask*]]]
Create an extended IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[**.***source-node* [*source-network-mask***.***source-node-mask*]] *source-socket* [*destination-network* [**.***destination-node* [*destination-network-mask***.***destination-node-mask*]] *destination-socket*]]

To apply a broadcast message filter to an interface, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Specify a helper address for forwarding broadcast messages.	**ipx helper-address** *network.node*
Step 2	Apply a broadcast message filter to an interface.	**ipx helper-list** *access-list-number*

Note A broadcast message filter has no effect unless you have issued an **ipx helper-address** and/or an **ipx type-20-propagation** command on the interface to enable and control the forwarding of broadcast messages. These commands are discussed later in this chapter.

For examples of creating and applying broadcast message filters, see the section "Helper Facilities to Control Broadcasts Examples " later in this chapter.

Tune IPX Network Performance

To tune IPX network performance, perform the tasks in one of more of the following sections:

- Control Novell IPX Compliance
- Configure Static Routes
- Adjust RIP Update Timers
- Configure RIP Update Packet Size
- Configure Static SAP Table Entries
- Configure the Queue Length for SAP Requests
- Adjust SAP Update Timers
- Configure SAP Update Packet Size
- Set Maximum Paths
- Control Responses to GNS Requests
- Use Helper Addresses to Forward Broadcast Messages
- Control the Forwarding of Type 20 Packets
- Disable IPX Fast Switching
- Enable Autonomous Switching
- Enable SSE Switching
- Pad Odd-Length Packets
- Repair Corrupted Network Numbers

Control Novell IPX Compliance

Our routers' implementation of Novell's IPX protocol has been certified as providing full IPX router functionality, as defined in the Novell IPX Router Specification.

To control specific aspects of IPX compliance, you can use a combination of global configuration and interface configuration commands. You can perform one or more of the following tasks in global configuration mode:

Task	Command
Restrict the acceptance of IPX type 20 propagation packets.	**ipx type-20-input-checks**
Restrict the forwarding of IPX type 20 propagation packets.	**ipx type-20-output-checks**

You can perform one or more of the following tasks in interface configuration mode:

Task	Command
Set the tick count, which is used in the IPX Routing Information Protocol (RIP) delay field.	**ipx delay** *number*
Administratively bring down an IPX network on an interface. This removes the network from the interface.	**ipx down** *network*
Set the delay between multiple-packet routing updates.	**ipx output-rip-delay** *delay*
Set the delay between packets sent in multiple-packet SAP updates.	**ipx output-sap-delay** *delay*
Forward IPX type 20 propagation packets to other network segments.	**ipx type-20-propagation**

To achieve full compliance, issue the following interface configuration commands on each interface configured for IPX:

Task		Command
Step 1	Set the delay between multiple-packet routing updates to 55 ms.	**ipx output-rip-delay 55**
Step 2	Set the delay between packets sent in multiple-packet SAP updates to 55 ms.	**ipx output-sap-delay 55**
Step 3	Optionally enable type 20 packet propagation if you want to forward type 20 broadcast traffic across the router.	**ipx type-20-propagation**

Configure Static Routes

IPX uses RIP, Enhanced IGRP, or NLSP to determine the best path when several paths to a destination exist. The routing protocol then dynamically updates the routing table. However, you might want to add static routes to the routing table to explicitly specify paths to certain destinations. Static routes always override any dynamically learned paths.

Be careful when assigning static routes. When links associated with static routes are lost, traffic may stop being forwarded or traffic may be forwarded to a nonexistent destination, even though an alternative path might be available.

To add a static route to the router's routing table, perform the following task in global configuration mode:

Task	Command
Add a static route to the routing table.	**ipx route** [*network* | **default**] *network.node*

You can configure static routes that can be overridden by dynamically learned routes. These are referred to a floating static routes. You can use a floating static route to create a path of last resort that is used only when no dynamic routing information is available.

Note that by default, floating static routes are not redistributed into other dynamic protocols.

To add a floating static route to the router's routing table, perform the following task in global configuration mode:

Task	Command
Add a floating static route to the routing table.	**ipx route** *network network.node* **floating-static**

Adjust RIP Update Timers

You can set the interval between IPX RIP updates on a per-interface basis. You also can specify that a delay be inserted between the packets of a multiple-packet update.

You can set RIP update times only in a configuration in which all routers are our routers or in which the IPX routers allow configurable timers. The timers for all routers connected to the same network segment should be the same. The RIP update value you choose affects internal IPX timers as follows:

- IPX routes are marked invalid if no routing updates are heard within three times the value of the update interval (3 * *interval*) and are advertised with a metric of infinity.
- IPX routes are removed from the routing table if no routing updates are heard within four times the value of the update interval (4 * *interval*).
- If you define an update timer for more than one interface in a router, the granularity of the update timer is determined by the lowest value defined for one of the interfaces in the router. The router "wakes up" at this granularity interval and sends out updates as appropriate. For more information about granularity, see the "Novell IPX Commands" chapter in the *Router Products Command Reference* publication.

You might want to set a delay between the packets in a multiple-packet update if there are some slower PCs on the network.

To adjust RIP update times on the router, perform one or both of the following tasks in interface configuration mode:

Task	Command
Adjust the RIP update interval.	**ipx update-time** *interval*
Adjust the delay between multiple-packet routing updates.	**ipx output-rip-delay** *delay*

By default, a network's or server's RIP entry ages out at an interval equal to three times the RIP update interval. To configure the multiplier that controls the interval, perform the following task in interface configuration mode:

Task	Command
Configure the interval at which a network RIP entry ages out.	**ipx rip-multiplier** *multiplier*

Configure RIP Update Packet Size

By default, the maximum size of RIP updates sent out an interface is 432 bytes. This size allows for 50 routes at 8 bytes each plus a 32-byte IPX RIP header.To modify the maximum packet size, perform the following task in interface configuration mode:

Task	Command
Configure the maximum packet size of RIP updates sent out an interface.	**ipx rip-max-packetsize** *bytes*

Configure Static SAP Table Entries

Servers use SAP to advertise their services via broadcast packets. Routers store this information in the SAP table, also known as the Server Information Table (SIT). This table is updated dynamically. You might want to explicitly add an entry to the SIT so that clients always use the services of a particular server. Static SAP assignments always override any identical entries in the SAP table that are learned dynamically, regardless of hop count. If a dynamic route that is associated with a static SAP entry is lost or deleted, the router will not announce the static SAP entry until it relearns the route.

To add a static entry to the router's SAP table, perform the following task in global configuration mode:

Task	Command
Specify a static SAP table entry.	**ipx sap** *service-type name network.node socket hop-count*

Configure the Queue Length for SAP Requests

The router maintains a list of SAP requests to process, including all pending GNS queries from clients attempting to reach servers. When the network is restarted, the router can be inundated with hundreds of requests for servers. Typically, many of these are repeated requests from the same clients. You can configure the maximum length allowed for the pending SAP requests queue. SAP requests received when the queue is full are dropped, and the client must resend them.

To set the queue length for SAP requests, perform the following task in global configuration mode:

Task	Command
Configure the maximum SAP queue length.	**ipx sap-queue-maximum** *number*

Adjust SAP Update Timers

You can adjust the interval at which SAP updates are sent, and you can set the delay between packets sent in multipacket SAP updates.

Changing the interval at which SAP updates are sent is most useful on limited-bandwidth, point-to-point links or on X.25 interfaces. You should ensure that all Novell servers and routers on a given network have the same SAP interval. Otherwise, they might decide that a server is down when it is really up.

Adjusting the delay between packets sent in a multipacket SAP update is useful when the IPX network has slow IPX servers and/or routers. Setting a delay between packets in a multipacket SAP update forces our router interface to slow its output of SAP packets.

To modify the SAP timers used by the router, perform one or both of the following tasks in interface configuration mode:

Task	Command
Adjust the interval at which SAP updates are sent.	**ipx sap-interval** *interval*
Adjust the delay between packets sent in multiple-packet SAP updates.	**ipx output-sap-delay** *delay*

By default, a network's or server's SAP entry ages out at an interval equal to three times the SAP update interval. To configure the multiplier that controls the interval, perform the following task in interface configuration mode:

Task	Command
Configure the interval at which a network's or server's SAP entry ages out.	**ipx sap-sap-multiplier** *multiplier*

Configure SAP Update Packet Size

By default, the maximum size of SAP updates sent out an interface is 480 bytes. This size allows for 7 servers (64 bytes each) plus a 32-byte IPX SAP header.To modify the maximum packet size, perform the following task in interface configuration mode:

Task	Command
Configure the maximum packet size of SAP updates sent out an interface.	**ipx sap-max-packetsize** *bytes*

Set Maximum Paths

You can set the maximum number of equal-cost, parallel paths to a destination. (Note that when paths have differing costs, the router chooses lower-cost routes in preference to higher-cost routes.) The router then distributes output on a packet-by-packet basis in round-robin fashion. That is, the first packet is sent along the first path, the second packet along the second path, and so on. When the final path is reached, the next packet is sent to the first path, the next to the second path, and so on. This round-robin scheme is used whether or not fast switching is enabled.

The cost of a path is determined by ticks, with hop count used as a tie breaker.

Limiting the number of equal-cost paths can save memory on routers with limited memory or very large configurations. Additionally, in networks with a large number of multiple paths and systems with limited ability to cache out-of-sequence packets, performance might suffer when traffic is split between many paths.

To set the maximum number of paths on the router, perform the following task in global configuration mode:

Task	Command
Set the maximum number of equal-cost paths to a destination.	**ipx maximum-paths** *paths*

Control Responses to GNS Requests

You can set the method in which the router responds to SAP GNS requests, and you can set the delay time in responding to these requests. By default, the router responds to GNS requests. You can disable this.

The default method of responding to GNS requests is to respond with the server whose availability was learned most recently.

To control responses to GNS requests, perform one or more of the following tasks in global configuration mode:

Task	Command
Respond to GNS requests using a round-robin selection method.	**ipx gns-round-robin**
Set the delay when responding to GNS requests.	**ipx gns-response-delay** [*milliseconds*]
Disable the sending of replies to GNS queries.	**ipx gns-reply-disable**

Use Helper Addresses to Forward Broadcast Messages

Routers normally block all broadcast requests and do not forward them to other network segments. This is done to prevent the degradation of performance over the entire network. You can enable the forwarding of broadcast messages (except type 20 broadcasts) to other networks and forward all other unrecognized broadcast messages. These are non-RIP and non-SAP packets that are not addressed to the local network. Forwarding broadcast messages is sometimes useful when a network segment does not have an end-host capable of servicing a particular type of broadcast request. You can specify the address of a server, network, or networks that can process the broadcast messages.

Our routers support all-networks flooded broadcasts (sometimes referred to as *all-nets flooding*). These are broadcast messages that are forwarded to all networks. Use all-nets flooding carefully and only when necessary, because the receiving networks may be overwhelmed to the point that no other traffic can traverse them.

Use the **ipx helper-list** command, described earlier in this chapter, to define access lists that control which broadcast packets get forwarded.

To specify a helper address for forwarding broadcast messages, perform the following task in interface configuration mode:

Task	Command
Specify a helper address for forwarding broadcast messages.	**ipx helper-address** *network.node*

You can specify multiple helper addresses on an interface.

For an example of using helper addresses to forward broadcast messages, see the section “Helper Facilities to Control Broadcasts Examples ” later in this chapter.

Control the Forwarding of Type 20 Packets

NetBIOS over IPX uses type 20 propagation broadcast packets flooded to all networks to get information about the named nodes on the network. NetBIOS uses a broadcast mechanism to get this information, because it does not implement a network layer.

Routers normally block all broadcast requests. By enabling type 20 packet propagation, IPX interfaces on the router may accept and forward type 20 propagation packets. Before forwarding (flooding) the packets, the router performs loop detection as described by the IPX router specification.

You can configure the router to apply extra checks to type 20 propagation packets above and beyond the loop detection described in the IPX specification. These checks are the same ones that are applied to helpered all-nets broadcast packets. They can limit unnecessary duplication of type 20 broadcast packets. The extra helper checks are as follows:

- Accept type 20 propagation packets only on the primary network, which is the network that is the primary path back to the source network.
- Forward type 20 propagation packets only via networks that do not lead back to the source network.

While this extra checking increases the robustness of type 20 propagation packet handling by decreasing the amount of unnecessary packet replication, it has two side effects:

- If type 20 packet propagation is not configured on all interfaces, these packets might be blocked when the primary interface changes.
- It might be impossible to configure an arbitrary, manual spanning tree for type 20 packet propagation.

You can enable the forwarding of type 20 packets on individual interfaces, and you can restrict the acceptance and forwarding of type 20 packets. The tasks to do this are described in the following sections.

Enable the Forwarding of Type 20 Packets

By default, type 20 propagation packets are dropped by the router. You can configure the router to receive type 20 propagation broadcast packets and forward (flood) them to other network segments, subject to loop detection.

To enable the receipt and forwarding of type 20 packets, perform the following task in interface configuration mode:

Task	Command
Forward IPX type 20 propagation packet broadcasts to other network segments.	**ipx type-20-propagation**

Restrict the Acceptance of Incoming Type 20 Packets

For incoming type 20 propagation packets, the router is configured by default to accept packets on all interfaces enabled to receive type 20 propagation packets. You can configure the router to accept packets only from the single network that is the primary route back to the source network. This means that similar packets from the same source that are received via other networks will be dropped.

Checking of incoming type 20 propagation broadcast packets is done only if the interface is configured to receive and forward type 20 packets.

To impose restrictions on the receipt of incoming type 20 propagation packets in addition to the checks defined in the IPX specification, perform the following global configuration task:

Task	Command
Restrict the acceptance of IPX type 20 propagation packets.	**ipx type-20-input-checks**

Restrict the Forwarding of Outgoing Type 20 Packets

For outgoing type 20 propagation packets, the router is configured by default to send packets on all interfaces enabled to send type 20 propagation packets, subject to loop detection. You can configure the router to send these packets only to networks that are not routes back to the source network. (The router uses the current routing table to determine routes.)

Checking of outgoing type 20 propagation broadcast packets is done only if the interface is configured to receive and forward type 20 packets.

To impose restrictions on the transmission of type 20 propagation packets and to forward these packets to all networks using only the checks defined in the IPX specification, perform the following global configuration task:

Task	Command
Restrict the forwarding of IPX type 20 propagation packets.	**ipx type-20-output-checks**

Disable IPX Fast Switching

Fast switching allows higher throughput by switching a packet using a cache created by previous packets. Fast switching is enabled by default on all interfaces.

Packet transfer performance is generally better when fast switching is enabled. However, you might want to disable fast switching in order to save memory space on interface cards and to help avoid congestion when high-bandwidth interfaces are writing large amounts of information to low-bandwidth interfaces.

To disable IPX fast switching, perform the following task in interface configuration mode:

Task	Command
Disable IPX fast switching.	**no ipx route-cache**

Enable Autonomous Switching

Autonomous switching provides faster packet switching by allowing the ciscoBus controller to switch packets independently without having to interrupt the system processor. It is available only in Cisco 7000 systems and in AGS+ systems with high-speed network controller cards, such as the CSC-HSCI, CSC-MEC, CSC-FCI, CSC-C2FCIT, and CSC-C2CTR, and with a CSC-CCTL2 ciscoBus controller running Microcode Version 11.0 or later. Autonomous switching is disabled by default on all interfaces.

To enable autonomous switching, perform the following task in interface configuration mode:

Task	Command
Enable autonomous switching.	**ipx route-cache cbus**

Enable SSE Switching

The silicon switching engine (SSE) is on the Silicon Switch Processor (SSP) board in the Cisco 7000 series. SSE switching contributes to very fast packet processing by allowing the SSE to perform switching independently of the system processor.

To enable SSE switching, perform the following task in interface configuration mode:

Task	Command
Enable the SSE switching cache.	**ipx route-cache sse**

Pad Odd-Length Packets

Some IPX end hosts reject Ethernet packets that are not padded to be an even length. Certain topologies can result in such packets being forwarded onto a remote Ethernet network. Under specific conditions, you can use padding on intermediate media as a temporary workaround for this problem.

To enable the padding of odd-length packets, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Disable fast switching.	**no ipx route-cache**
Step 2	Enable the padding of odd-length packets.	**ipx pad-process-switched-packets**

Repair Corrupted Network Numbers

To repair corrupted network numbers on an interface, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Disable fast switching.	**no ipx route-cache**
Step 2	Repair corrupted network numbers.	**ipx source-network-update**

Caution The **ipx source-network-update** interface configuration command interferes with the proper working of OS/2 Requestors. Do not use this command in a network that has OS/2 Requestors.

Caution Do not use the **ipx source-network-update** interface configuration command on interfaces on which NetWare servers are using internal network numbers (that is, all 3.1*x* and 4.0 servers).

Configure IPX Accounting

IPX accounting allows you to collect information about IPX packets and the number of bytes that are switched through the router. You collect information based on the source and destination IPX address. Accounting tracks only IPX traffic that is routed through the router; it does not track traffic generated by or terminating at the router.

IPX accounting statistics are accurate even if IPX fast switching is enabled or if IPX access lists are being used. However, IPX accounting does not keep statistics if autonomous switching is enabled.

The router software maintains two accounting databases: an active database and a checkpointed database.

To enable IPX accounting, perform the following task in interface configuration mode:

Task	Command
Enable IPX accounting.	**ipx accounting**

To control IPX accounting on the router, perform one or more of the following tasks in global configuration mode:

Task	Command
Set the maximum number of accounting entries.	**ipx accounting-threshold** *threshold*
Set the maximum number of transit entries.	**ipx accounting-transits** *count*
Filter the networks for which IPX accounting information is kept.	**ipx accounting-list** *number mask*

Shut Down an IPX Network

You can administratively shut down an IPX network in two ways. In the first way, the network still exists in the configuration, but is not active. When shutting down, the network sends out update packets informing its neighbors that it is shutting down. This allows the neighboring systems to update their routing, SAP, and other tables without having to wait for routes and services learned via this network to time out.

To shut down an IPX network such that the network still exists in the configuration, perform the following task in interface configuration mode:

Task	Command
Shut down an IPX network but have the network still exist in the configuration.	**ipx down** *network*

In the second way, you shut down an IPX network and remove it from the configuration. To do this, perform one of the following tasks in interface configuration mode:

Task	Command
Shut down an IPX network and remove it from the configuration.	**no ipx network**
When multiple networks are configured on an interface, shut down all networks and remove them from the interface.	**no ipx network** *network* (where *network* is 1, the primary interface)
When multiple networks are configured on an interface, shut down one of the secondary networks and remove it from the interface.	**no ipx network** *network* (where *network* is the number of the secondary interface [not 1])

When multiple networks are configured on an interface and you want shut down one of the secondary networks and remove it from the interface, perform the second task in the previous table specifying the network number of one of the secondary networks.

For an example of shutting down an IPX network, see the section “Enabling IPX Routing Example” later in this chapter.

Configure IPX over WANs

You can configure IPX over dial-on-demand routing (DDR), Frame Relay, Point-to-Point Protocol (PPP), Switched Multimegabit Data Service (SMDS), and X.25 networks. To do this, you configure address mappings as described in the appropriate chapter. You can also configure IPX over Point-to-Point Protocol (PPP); address maps are not necessary for this protocol. You can fast switch IPX serial interfaces configured for Frame Relay, and you can fast switch SNAP-encapsulated packets over interfaces configured for ATM.

Additionally, you can configure the IPXWAN protocol.

Configure IPX over DDR

IPX sends periodic watchdog (keepalive) packets. These are keepalive packets that are sent from servers to clients after a client session has been idle for approximately 5 minutes. On a DDR link, this means that a call would be made every 5 minutes, regardless of whether there were data packets to send. You can prevent these calls from being made by configuring the router to respond to the server’s watchdog packets on a remote client’s behalf. This is sometimes referred to as “spoofing the server.”

When configuring IPX over DDR, you might want to disable the generation of these packets so that a call is not made very 5 minutes. This is not an issue for the other WAN protocols, because they establish dedicated connections rather than establishing connections only as needed.

To keep the serial interface idle when only watchdog packets are being sent, refer to the tasks described in the “Configuring DDR” chapter. For an example of configuring IPX over DDR, see the section “IPX over DDR Example” later in this chapter.

Configure the IPXWAN Protocol

Our routers support the IPXWAN protocol, as defined in RFC 1362. IPXWAN allows a router that is running IPX routing to connect via a serial link to another router, possibly from another manufacturer, that is also routing IPX and using IPXWAN.

IPXWAN is a connection startup protocol. Once a link has been established, IPXWAN incurs little or no overhead.

You can use the IPXWAN protocol over PPP. You can also use it over HDLC; however, the routers at both ends of the serial link must be our routers.

To configure IPXWAN, perform the following tasks in interface configuration mode on a serial interface:

Task		Command
Step 1	Ensure that you have not configured an IPX network number on the interface.	**no ipx network**
Step 2	Enable PPP.	**encapsulation ppp**[1]
Step 3	Enable IPXWAN.	**ipx ipxwan** [*local-node* {*network-number* | **unnumbered**} *local-server-name retry-interval retry-limit*]

Task		Command
Step 4	Optionally, define how to handle IPXWAN when a serial link fails.	**ipx ipxwan error** [**reset** | **resume** | **shutdown**]
Step 5	Optionally, enable static routing with IPXWAN.	**ipx ipxwan static**

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

Monitor and Maintain the IPX Network

To monitor and maintain a Novell IPX network, perform one or more of the following tasks at the EXEC prompt:

Task	Command
Delete all entries in the IPX accounting or accounting checkpoint database.	**clear ipx accounting** [**checkpoint**]
Delete all entries in the IPX fast-switching cache.	**clear ipx cache**
Delete all NLSP adjacencies from the router's adjacency database.	**clear ipx nlsp neighbors**
Delete entries in the IPX routing table.	**clear ipx route** [*network* | *****]
Have the Cisco 7000 route processor recompute the IPX SSE fast-switching cache.	**clear ipx sse**
Reinitialize the route processor on the Cisco 7000.	**clear sse**
List the entries in the IPX accounting or accounting checkpoint database.	**show ipx accounting** [**checkpoint**]
List the entries in the IPX fast-switching cache.	**show ipx cache**
List the neighbors discovered by Enhanced IGRP.	**show ipx eigrp neighbors** [**servers**] [*autonomous-system-number* | *interface*]
Display the contents of the Enhanced IGRP topology table.	**show ipx eigrp topology** [*network-number*]
Display the status of the IPX interfaces configured in the router and the parameters configured on each interface.	**show ipx interface** [*type number*]
Display the entries in the link-state packet (LSP) database.	**show ipx nlsp database** [*lspid*] [**detail**]
Display the router's NLSP neighbors and their states.	**show ipx nlsp neighbors** [*interface*] [**detail**]
List the entries in the IPX routing table.	**show ipx route** [*network*] [**default**] [**detailed**]
List the servers discovered through SAP advertisements.	**show ipx servers** [unsorted | **sorted** [**name** | **net** | **type**]]
Display information about the number and type of IPX packets transmitted and received.	**show ipx traffic**
Display a summary of SSP statistics	**show sse summary**

The router can transmit Cisco pings or standard Novell pings as defined in the NLSP specification. By default, the router generates Cisco pings. To choose the ping type, perform the following task in global configuration mode:

Task	Command
Select the ping type.	**ipx ping-default** {**cisco** | **novell**}

To initiate a ping, perform one of the following tasks in EXEC mode:

Task	Command
Diagnose basic IPX network connectivity (user-level command).	**ping ipx** *network.node*
Diagnose basic IPX network connectivity (privileged command).	**ping** [**ipx**] [*network.node*]

Configuration Examples

This section provides configuration examples for the following IPX configuration situations:

- Enabling IPX Routing Example
- Enabling and Disabling IPX Routing on Multiple Networks Example
- Enabling and Disabling IPX Routing Protocols Examples
- Enabling IPX over a WAN Interface Example
- IPX over DDR Example
- IPX Network Access Example
- SAP Input Filter Example
- SAP Output Filter Example
- IPX NetBIOS Filter Examples
- Helper Facilities to Control Broadcasts Examples
- IPX Accounting Example
- Enabling IPX Enhanced IGRP Example
- Enhanced IGRP SAP Update Examples

Enabling IPX Routing Example

The following configuration commands enable IPX routing, defaulting the IPX host address to that of the first IEEE-conformance interface (in this example, Ethernet 0). Routing is then enabled on Ethernet 0 and Ethernet 1 for IPX networks 2abc and 1def, respectively.

```
ipx routing
interface ethernet 0
ipx network 2abc
interface ethernet 1
ipx network 1def
```

Enabling and Disabling IPX Routing on Multiple Networks Example

The following example uses subinterfaces to create four logical networks on Ethernet interface 0. Each subinterface has a different encapsulation. Any interface configuration parameters that you specify on an individual subinterface are applied to that subinterface only.

```
ipx routing
interface ethernet 0.1
ipx network 1 encapsulation novell-ether
interface ethernet 0.2
ipx network 2 encapsulation snap
interface ethernet 0.3
ipx network 3 encapsulation arpa
interface ethernet 0.4
ipx network 4 encapsulation sap
```

Note When enabling NLSP and configuring multiple encapsulations on the same physical LAN interface, you must use subinterfaces. You cannot use secondary networks.

You can administratively bring down each of the four subinterfaces separately by using the **shutdown** interface configuration command for each subinterface. For example, the following commands administratively shut down a subinterface:

```
interface ethernet 0.3
shutdown
```

To bring down network 1, use the following commands:

```
interface ethernet 0.1
ipx down 1
```

To bring network 1 back up, use the following commands:

```
interface ethernet 0.1
no ipx down 1
```

To remove all the networks on the interface, use the following interface configuration commands:

```
interface ethernet 0.1
no ipx network
interface ethernet 0.2
no ipx network
interface ethernet 0.3
no ipx network
interface ethernet 0.4
no ipx network
```

The following example uses primary and secondary networks to create the same four logical networks as shown earlier in this section. Any interface configuration parameters that you specify on this interface are applied to all the logical networks. For example, if you set the routing update timer to 120 seconds, this value is used on all four networks.

```
ipx routing
interface ethernet 0
ipx network 1 encapsulation novell-ether
ipx network 2 encapsulation snap secondary
ipx network 3 encapsulation arpa secondary
ipx network 4 encapsulation sap secondary
```

Using this method to configure logical networks, if you administratively bring down Ethernet interface 0 using the **shutdown** interface configuration command, all four logical networks are shut down. You cannot bring down each logical network independently using the **shut** command; however, you can do this using the **ipx down** command.

To bring down network 1, use the following command:

```
interface ethernet 0
ipx down 1
```

To bring the network back up, use the following command:

```
interface ethernet 0
no ipx down 1
```

To shut down all four networks on the interface and remove all the networks on the interface, use one of the following interface configuration commands:

```
no ipx network

no ipx network 1
```

To remove one of the secondary networks on the interface (in this case, network 2), use the following interface configuration command:

```
no ipx network 2
```

Enabling and Disabling IPX Routing Protocols Examples

Three routing protocols can run over interfaces configured for IPX: RIP, Enhanced IGRP, and NLSP. This section provides examples of how to enable and disable various combinations of routing protocols.

When you enable IPX routing with the **ipx routing** global configuration command, the RIP routing protocol is automatically enabled. The following example enables RIP on networks 1 and 2:

```
ipx routing
!
interface ethernet 0
ipx network 1
!
interface ethernet 1
ipx network 2
```

The following example enables RIP on networks 1 and 2 and Enhanced IGRP on network 1:

```
ipx routing
!
interface ethernet 0
ipx network 1
!
interface ethernet 1
ipx network 2
!
ipx router eigrp 100
network 1
```

The following example enables RIP on network 2 and Enhanced IGRP on network 1:

```
ipx routing
!
interface ethernet 0
ipx network 1
!
```

```
interface ethernet 1
ipx network 2
!
ipx router eigrp 100
ipx network 1
!
ipx router rip
no ipx network 1
```

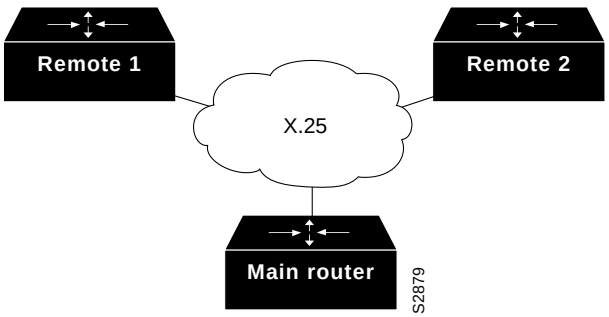
The following example configures NLSP on two of a router's Ethernet interfaces. Note that RIP is automatically enabled on both of these interfaces. This example assumes that the encapsulation type is 802.2.

```
ipx routing
ipx internal-network 3
!
ipx router nlsp
area-address 0 0
!
interface ethernet 0
ipx network e0 encapsulation sap
ipx nlsp enable
!
interface ethernet 1
ipx network e1 encapsulation sap
ipx nlsp enable
```

Enabling IPX over a WAN Interface Example

When you configure the router to transport IPX packets over a serial interface that is running a WAN protocol such as X.25 or PPP, you specify how the packet will be encapsulated for transport. This encapsulation is not the same as the encapsulation used on an IPX LAN interface. Figure 20-1 illustrates IPX over a WAN interface.

Figure 20-1 IPX over a WAN Interface



The following examples configure a serial interface for X.25 encapsulation and for several IPX subinterfaces used in a nonmeshed topology.

Configuration for Main Router

```
hostname Main
!
no ip routing
novell routing 0000.0c17.d726
!
```

```
interface Ethernet0
no ip address
Novell network 100
media-type 10BaseT
!
interface Serial0
no ip address
shutdown
!
interface Serial1
no ip address
encapsulation x25
x25 address 33333
x25 htc 28
!
interface Serial1.1 point-to-point
no ip address
novell network 2
x25 map novell 2.0000.0c03.a4ad 11111 BROADCAST
!
interface Serial1.2 point-to-point
no ip address
novell network 3
x25 map novell 3.0000.0c07.5e26 55555 BROADCAST
```

Configuration for Router 1

```
hostname Remote1
!
no ip routing
novell routing 0000.0c03.a4ad
!
interface Ethernet0
no ip address
novell network 1
!
interface Serial0
no ip address
encapsulation x25
novell network 2
x25 address 11111
x25 htc 28
x25 map novell 2.0000.0c17.d726 33333 BROADCAST
```

Configuration for Router 2

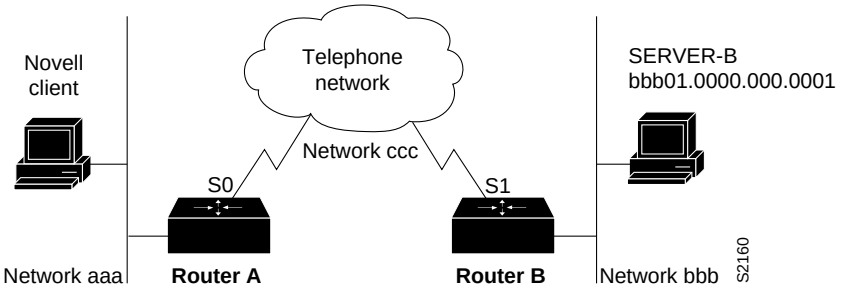
```
hostname Remote2
!
no ip routing
novell routing 0000.0c07.5e26
!
interface Ethernet0
no ip address
novell network 4
media-type 10BaseT
!
interface Serial0
no ip address
shutdown
!
interface Serial1
no ip address
encapsulation x25
```

```
novell network 3
x25 address 55555
x25 htc 28
x25 map novell 3.0000.0c17.d726 33333 BROADCAST
```

IPX over DDR Example

In the configuration shown in Figure 20-2, an IPX client is separated from its server by a DDR telephone line.

Figure 20-2 IPX over DDR Configuration



Routing and service information is sent every minute. The output RIP and SAP filters defined in this example filter these updates, preventing them from being sent between Routers A and B. If you were to forward these packets, the two routers would each have to telephone the other once a minute. On a serial link that charges based on the number of packets transmitted, this is generally not desirable. This might not be an issue on a dedicated serial line.

Once the server and client have established contact, the server will send keepalive (watchdog) packets regularly. The purpose of these packets is to ensure that the connection between the server and the client is still functional; these packets contain no other information. Servers send watchdog packets approximately every 5 minutes. If you were to allow Router B to forward the server's keepalive packets to Router A and the client, Router B would have to telephone Router A every 5 minutes just to send these packets. Again, on a serial link that charges based on the number of packets transmitted, this is generally not desirable. Instead of having Router B telephone Router A only to send keepalive packets, you can enable watchdog spoofing on Router B. This way, when the server connected to this router sends keepalive packets, Router B will respond on behalf of the remote client (the client connected to Router A).

Configuration for Router A

```
access-list 1000 permit -1 7
access-list 1000 deny -1
!
!configure the router to which the client is connected
ipx routing 0000.0c00.59e8
ipx sap 4 SERVER-B BBB01.0000.0000.0001.4212
!
interface ethernet 0
ipx network aaa
!
interface serial 0
no keepalive
dialer in-band
dialer string 8986
```

```
ipx network ccc
pulse-time 1
dialer-group 1
ipx output-sap-filter 1000
!
ipx route bbb ccc.0000.0c01.d877
ipx route bbb01 ccc.0000.0c01.d877
!
access-list 800 permit ffffffff bbb01.0000.0000.0001
access-list 800 deny -1
access-list 1000 permit ffffffff bbb01.0000.0000.0001
access-list 1000 deny -1
dialer-list 1 list 800
```

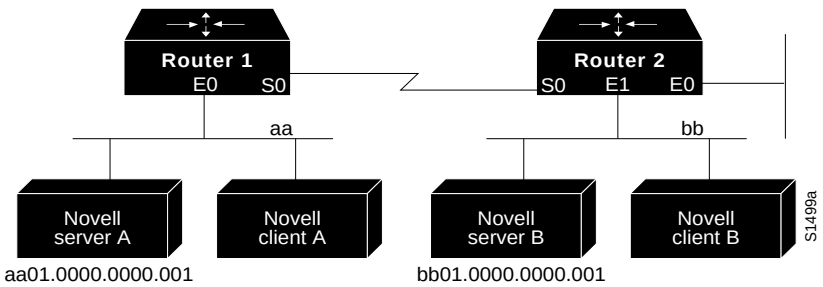
Configuration for Router B

```
!configure the router to which the server is attached
ipx routing 0000.0x01.d877
!
interface ethernet 0
ipx network bbb
!
interface serial 1
no ip address
bandwidth 56
no keepalive
ipx output-sap-filter 1000
dialer in-band
ipx network bbb
pulse-time 1
no ipx route-cache
no ipx-route-cache cbus
!enable watchdog spoofing on the server's router
ipx watchdog-spoof
!
ipx route aaa ccc.0000.0c00.59e8
access-list 1000 permit 4 bbb01.0000.0001
access-list 1000 deny -1
```

IPX Network Access Example

Using access lists to manage traffic routing can be a powerful tool in overall network control. However, it requires a certain amount of planning and the appropriate application of several related commands. Figure 20-3 illustrates a network featuring two routers on two network segments.

Figure 20-3 Novell IPX Servers Requiring Access Control



Suppose you want to prevent clients and servers on Network aa from using the services on Network bb, but you want to allow the clients and servers on Network bb to use the services on Network aa. To do this, you would need an access list on Ethernet interface 1 on Router 2 that blocks all packets coming from Network aa and destined for Network bb. You would not need any access list on Ethernet interface 0 on Router 1.

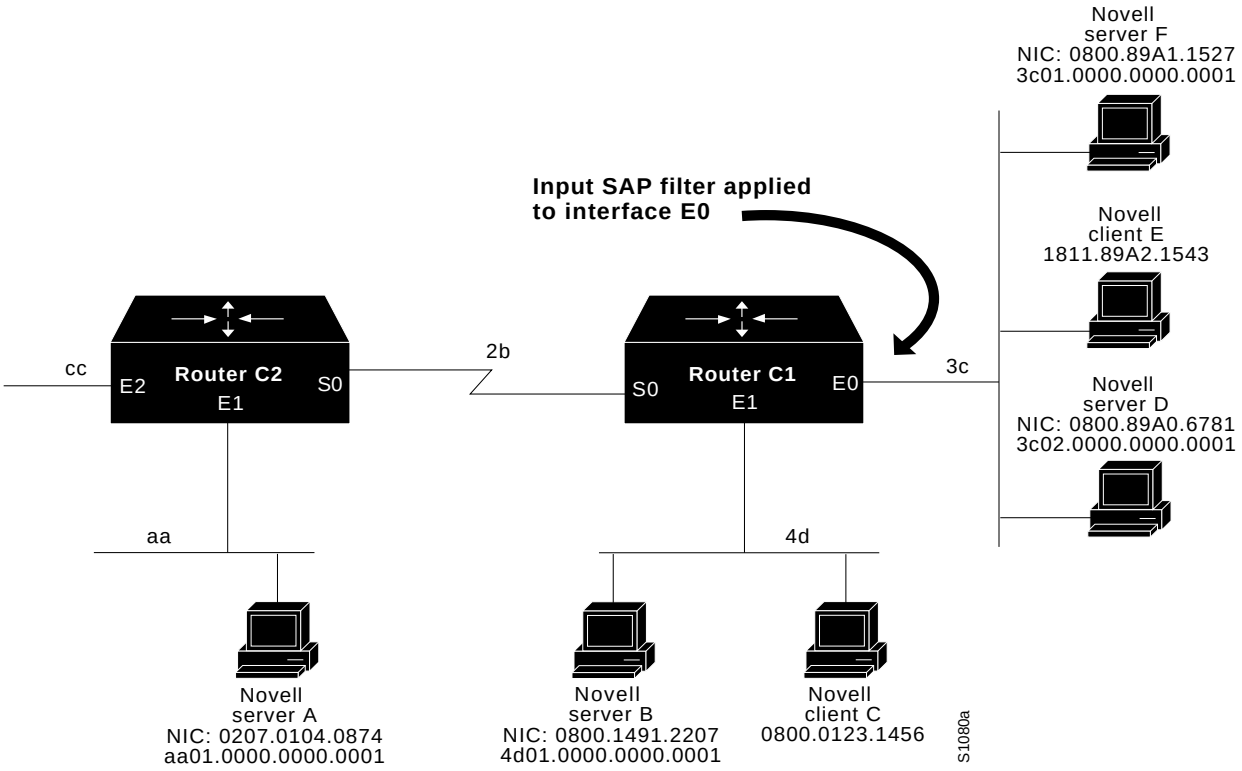
You would configure serial interface 0 on Router 2 with the following commands:

```
ipx routing
access-list 800 deny aa bb01
access-list 800 permit -1 -1
interface serial 0
ipx network bb
ipx access-group 800
```

SAP Input Filter Example

SAP input filters allow a router to determine whether or not to accept information about a service. Router C1, illustrated in Figure 20-4, will not accept and, consequently not advertise, any information about Novell server F. However, Router C1 will accept information about all other servers on the network 3c. Router C2 receives information about servers D and B.

Figure 20-4 SAP Input Filter



The following example configures Router C1. The first line denies server F, and the second line accepts all other servers.

```
access-list 1000 deny 3c01.0000.0000.0001
access-list 1000 permit -1
```

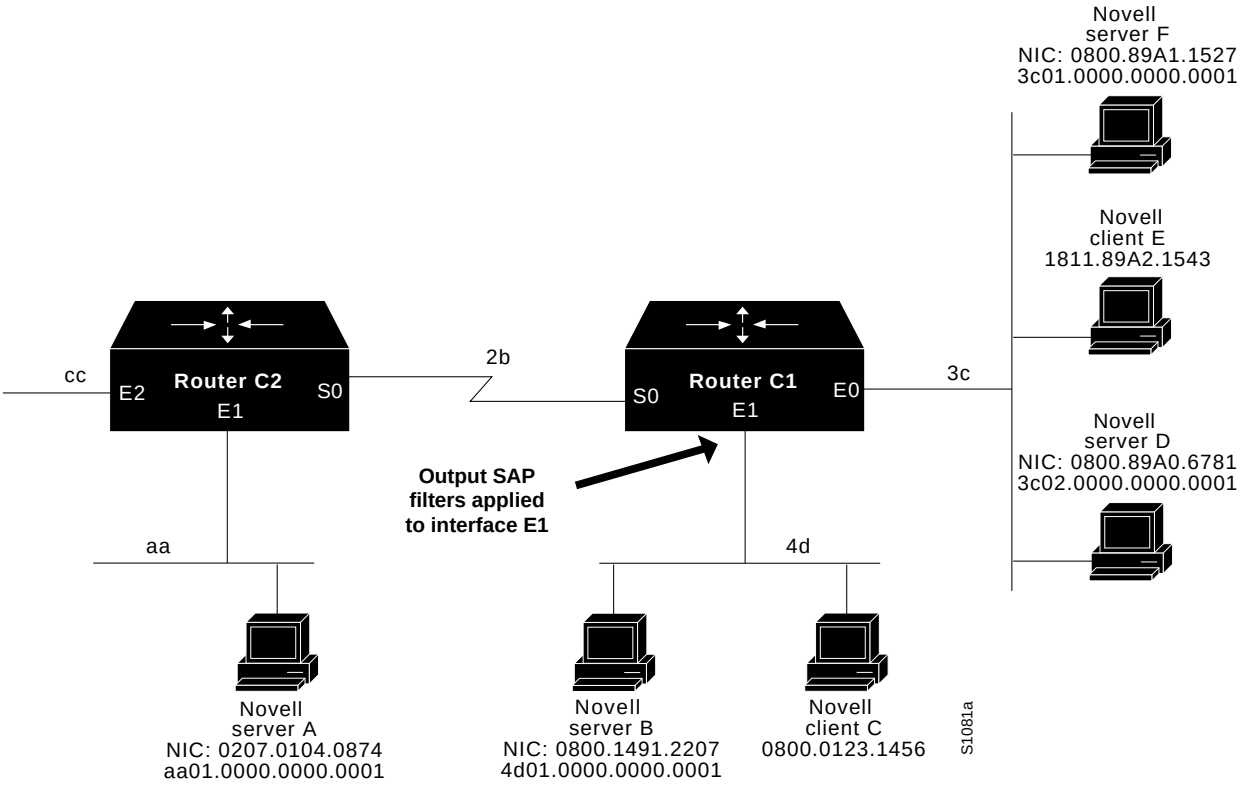
```
interface ethernet 0
ipx network 3c
ipx input-sap-filter 1000
interface ethernet 1
ipx network 4d
interface serial 0
ipx network 2b
```

Note NetWare Versions 3.11 and later use an internal network and node number as their address for access list commands (the first configuration command in this example).

SAP Output Filter Example

SAP output filters are applied prior to the router sending information out a specific interface. In the example that follows, Router C1 (illustrated in Figure 20-5) is prevented from advertising information about Novell server A out interface Ethernet 1, but can advertise server A on network 3c.

Figure 20-5 SAP Output Filter



The following example refers to Router C1. The first line denies server A. All other servers are permitted.

```
access-list 1000 deny aa01.0000.0000.0001
access-list 1000 permit -1
interface ethernet 0
novell net 3c
```

```
interface ethernet 1
ipx network 4d
ipx output-sap-filter 1000
interface serial 0
ipx network 2b
```

IPX NetBIOS Filter Examples

The following is an example of using a NetBIOS host name to filter IPX NetBIOS frames. The example denies all outgoing IPX NetBIOS frames with a NetBIOS host name of Boojum on Ethernet interface 0:

```
netbios access-list host token deny Boojum
netbios access-list host token permit *
!
ipx routing 0000.0c17.d45d
!
interface Ethernet 0
ipx network 155 encapsulation ARPA
ipx output-rip-delay 60
ipx output-sap-delay 60
ipx type-20-propagation
ipx netbios output-access-filter host token
no mop enabled
!
interface Ethernet 1
no ip address
ipx network 105
!
interface Fddi 0
no ip address
no keepalive
ipx network 305 encapsulation SAP
!
interface Serial 0
no ip address
shutdown
!
interface Serial 1
no ip address
no keepalive
ipx network 600
ipx output-rip-delay 60
ipx output-sap-delay 60
ipx type-20-propagation
```

The following is an example of using a byte pattern to filter IPX NetBIOS frames. This example permits IPX NetBIOS frames fro IPX network numbers that end in 05. This means that all IPX NetBIOS frames from Ethernet interface 1 (network 105) and FDDI interface 0 (network 305) will be forwarded by serial interface 0, but this interface will filter out and not forward all frames from Ethernet interface 0 (network 155).

```
netbios access-list bytes finigan permit 2 **05
!
ipx routing 0000.0c17.d45d
!
interface ethernet 0
ipx network 155 encapsulation ARPA
ipx output-rip-delay 60
ipx output-sap-delay 60
ipx type-20-propagation
media-type 10BaseT
!
```

```
interface ethernet 1
no ip address
ipx network 105
media-type 10BaseT
!
interface Fddi 0
no ip address
no keepalive
ipx network 305 encapsulation SAP
!
interface serial 0
no ip address
shutdown
!
interface Serial 1
no ip address
no keepalive
ipx network 600
ipx output-rip-delay 60
ipx output-sap-delay 60
ipx type-20-propagation
ipx netbios input-access-filter bytes finigan
```

Helper Facilities to Control Broadcasts Examples

The following examples illustrate how to control broadcast messages on IPX networks. Note that in the following examples, packet type 2 is used. This type has been chosen arbitrarily; the actual type to use depends on the specific application.

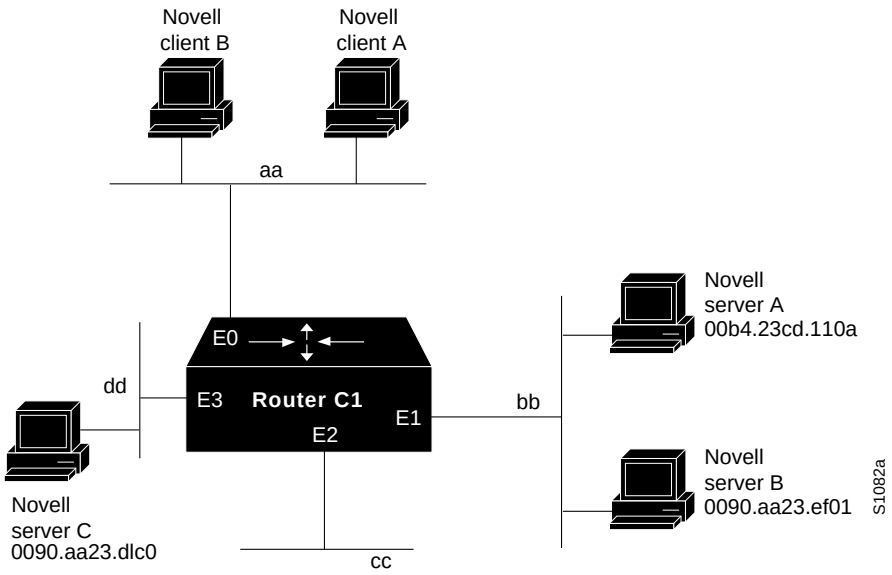
Forwarding to an Address Example

All broadcast packets are normally blocked by the router. However, type 20 propagation packets may be forwarded, subject to certain loop-prevention checks. Other broadcasts may be directed to a set of networks or a specific host (node) on a segment. The following examples illustrate these options.

Figure 20-6 shows a router (C1) connected to several Ethernet interfaces. In this environment, all IPX clients are attached to segment aa, while all servers are attached to segments bb and dd. In controlling broadcasts, the following conditions are to be applied:

- Only type 2 and type 20 broadcasts are to be forwarded.
- The IPX clients on network aa are allowed to broadcast via type 2 to any server on networks bb and dd.
- The IPX clients are allowed to broadcast via type 20 to any server on network dd.

Figure 20-6 IPX Clients Requiring Server Access through a Router



The following example configures the router shown in Figure 20-6. The first line permits broadcast traffic of type 2 from network aa. The interface and network commands configure each specific interface. The **ipx helper-address** commands permit broadcast forwarding from network aa to bb and from network aa to dd. The helper list allows type 2 broadcasts to be forwarded. (Note that type 2 broadcasts are chosen as an example only. The actual type to use depends on the application.) The **ipx type-20-propagation** command acts as a specific permission to allow type 20 broadcasts to be forwarded between networks aa and dd is also required.

```
access-list 900 permit 2 aa
interface ethernet 0
ipx network aa
ipx type-20-propagation
ipx helper-address bb.ffff.ffff.ffff
ipx helper-address dd.ffff.ffff.ffff
ipx helper-list 900
interface ethernet 1
ipx network bb
interface ethernet 3
ipx network dd
ipx type-20-propagation
```

This configuration means that any network that is downstream from network aa (for example, some arbitrary network aa1) will not be able to broadcast (type 2) to network bb through Router C1 unless the routers partitioning networks aa and aa1 are configured to forward these broadcasts with a series of configuration entries analogous to the example provided for Figure 19-5. These entries must be applied to the input interface and be set to forward broadcasts between directly connected networks. In this way, such traffic can be passed along in a directed manner from network to network. A similar situation exists for type 20 packets.

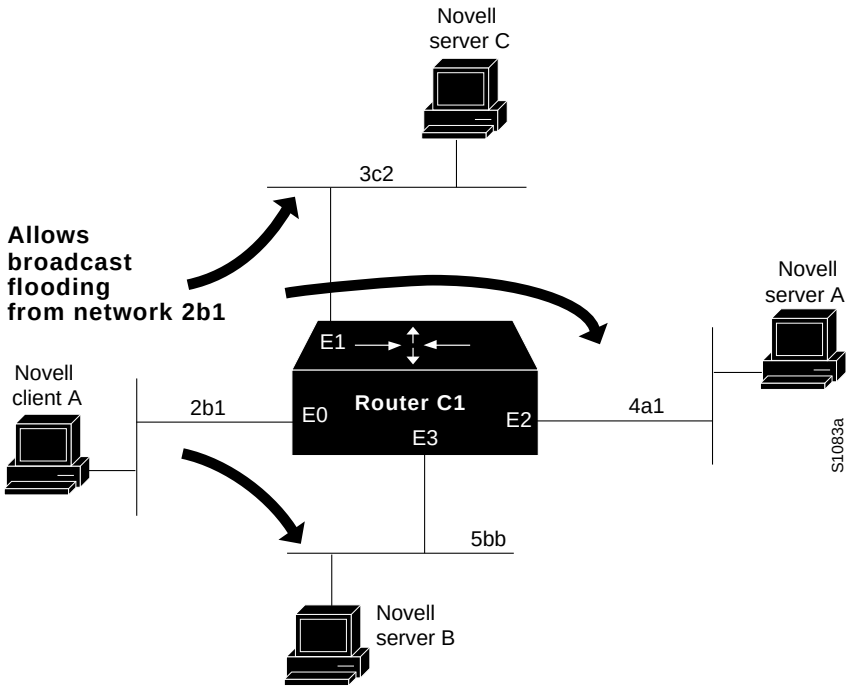
The following example rewrites the **ipx helper-address** interface configuration command line to direct broadcasts to server A:

```
ipx helper-address bb.00b4.23cd.110a
! Permits node-specific broadcast forwarding to
! Server A at address 00b4.23cd.110a on network bb
```

Forwarding to All Networks Example

In some networks, it might be necessary to allow client nodes to broadcast to servers on multiple networks. If you configure your router to forward broadcasts to all attached networks, you are flooding the interfaces. In the environment illustrated in Figure 20-7, client nodes on network 2b1 must obtain services from IPX servers on networks 3c2, 4a1, and 5bb through Router C1. To support this requirement, use the flooding address (-1.ffff.ffff.ffff) in your **ipx helper-address** interface configuration command specifications.

Figure 20-7 Type 2 Broadcast Flooding



In the following example, the first line permits traffic of type 2 from network 2b1. Then the first interface is configured with a network number. The all-nets helper address is defined and the helper list limits forwarding to type 2 traffic.Type 2 broadcasts from network 2b1 are forwarded to all directly connected networks. All other broadcasts, including type 20, are blocked. To permit broadcasts, delete the **ipx helper-list** entry. To allow type 20 broadcast, enable the **ipx type-20-propagation** interface configuration command on all interfaces.

```
access-list 901 permit 2 2b1
interface ethernet 0
ipx network 2b1
ipx helper-address -1.ffff.ffff.ffff
ipx helper-list 901
interface ethernet 1
ipx network 3c2
interface ethernet 2
ipx network 4a1
interface ethernet 3
ipx network 5bb
```

All-Nets Flooded Broadcast Example

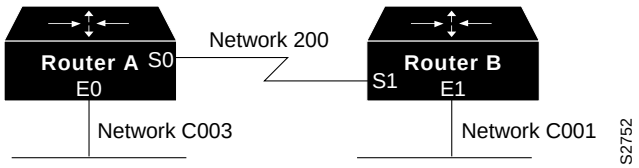
The following example configures all-nets flooding on an interface. As a result of this configuration, Ethernet interface 0 will forward all broadcast messages (except type 20) to all the networks it knows how to reach. This flooding of broadcast messages might overwhelm these networks with so much broadcast traffic that no other traffic may be able to pass on them.

```
interface ethernet 0
ipx network 23
ipx helper-address -1.FFFF.FFFF.FFFF
```

IPX Accounting Example

The following example configures two Ethernet network segments that are connected via a serial link. (See Figure 20-8.) On Router A, IPX accounting is enabled on both the input and output interfaces (that is, on Ethernet interface 0 and serial interface 0). This means that statistics are gathered for traffic traveling in both directions (that is, out to the Ethernet network and out the serial link). However, on Router B, IPX accounting is enabled only on the serial interface and not on the Ethernet interface. This means that statistics are gathered only for traffic that passes out the router on the serial link.

Figure 20-8 IPX Accounting Example



Configuration for Router A

```
ipx routing
interface ethernet 0
no ip address
ipx network C003
ipx accounting
interface serial 0
no ip address
ipx network 200
ipx accounting
```

Configuration for Router B

```
ipx routing
interface ethernet 1
no ip address
no keepalive
ipx network C001
no mop enabled
interface serial 1
no ip address
ipx network 200
ipx accounting
```

Enabling IPX Enhanced IGRP Example

The following example configures two interfaces for Enhanced IGRP routing in autonomous system 1:

```
ipx routing
!
interface ethernet 0
ipx network 10
!
interface serial 0
ipx network 20
!
ipx router eigrp 1
network 10
network 20
```

Enhanced IGRP SAP Update Examples

If an Ethernet interface has neighbors that are all configured for Enhanced IGRP, you might want to reduce the bandwidth used by SAP packets by sending SAP updates incrementally. To do this, you would configure the interface as follows:

```
ipx routing
!
interface ethernet 0
ipx network 10
ipx sap-incremental eigrp 1
!
interface serial 0
ipx network 20
!
ipx router eigrp 1
network 10
network 20
```

If you want to send periodic SAP updates on a serial line that is configured for Enhanced IGRP and that has an Enhanced IGRP peer on the other sides, use the following commands:

```
ipx routing
!
interface ethernet 0
ipx network 10
!
interface serial 0
ipx network 20
no ipx sap-incremental eigrp 1
!
ipx router eigrp 1
network 10
network 20
```

CHAPTER 21

Configuring XNS

The Xerox Network Systems (XNS) protocols, which were developed by the Xerox Corporation, are designed to be used across a variety of communication media, processors, and office applications. Ungermann-Bass, Inc. (now a part of Tandem Computers) adopted XNS in developing its Net/One XNS routing protocol. Standard XNS routing uses the Routing Information Protocol (RIP) update packets and the hop count metric. Ungermann-Bass Net/One uses hello packets and a path-delay metric.

This chapter describes how to configure standard XNS routing and Ungermann-Bass Net/One XNS routing. It also provides configuration examples. For a complete description of the commands discussed in this chapter, refer to the "XNS Commands" chapter in the *Router Products Command Reference* publication. For historical background and a technical overview of XNS, see the *Internetworking Technology Overview* publication.

Cisco's Implementation of XNS

Cisco provides a subset of the XNS protocol stack to support XNS routing. XNS traffic can be routed over Ethernet, FDDI, and Token Ring local area networks (LANs), as well as over point-to-point serial lines running High-Level Data Link Control (HDLC), Link Access Protocol, Balanced (LAPB), X.25, Frame Relay, or Switched Multimegabit Data Service (SMDS).

Ungermann-Bass Net/One Environments

Some of the tasks described in this chapter explain how to configure Ungermann-Bass Net/One XNS routing. Net/One uses XNS at the network layer. However, Net/One as a whole is not equivalent to standard XNS. When using our routers in Net/One environments, keep in mind the following differences between Net/One and standard XNS environments:

- Net/One routers use a proprietary routing protocol instead of the standard XNS RIP. Although they generate both Ungermann-Bass and standard RIP update packets, Net/One routers listen only to Net/One updates. We support both the reception and the generation of Net/One routing packets. Also, our routers can interoperate with Ungermann-Bass routers.
- Net/One routers send periodic hello packets, which are used by end nodes in discovering routers to be used when sending packets to destinations that are not on the local cable. Standard XNS end hosts use RIP for this purpose. Our routers can be configured to generate Net/One hello packets.
- Net/One equipment uses a non-XNS booting protocol for downloading network software. During the downloading process, XNS network numbers are embedded in this protocol's packets. Ungermann-Bass routers pass the booting protocol from network to network and modify the embedded network numbers. Our equipment does not understand the Net/One booting protocol

and does not modify the embedded network numbers. For network booting to work through our routers, Net/One Network Management Consoles must be specially configured. Contact Ungermann-Bass for information about how to do this.

- The Net/One Network Resource Monitor (a network management and monitoring tool) uses XNS packets whose destination host addresses are specific nodes, but whose destination network addresses are the broadcast network (network address of –1). These packets are sent as Media Access Control (MAC)-layer broadcasts and are expected to be flooded throughout the XNS internetwork. On our router, you enable the flooding of these packets as described in the section "Control Broadcast Messages" later in this chapter.
- Net/One equipment uses proprietary network management protocols. Our routers do not participate in these protocols.

Net/One uses a distance-vector routing protocol, similar to standard XNS RIP. The major difference between the two protocols is the metrics. Standard RIP uses hop count to determine the best route to a remote network, while the Net/One protocol uses a path-delay metric. The standard RIP protocol maintains information only about hop counts, while the Net/One protocol maintains information about both hop count and its own metrics.

Ungermann-Bass routers generate standard RIP updates by extracting the hop-count values from the Ungermann-Bass routing protocol. When configured in Ungermann-Bass emulation mode, our routers participate in this protocol and behave insofar as routing protocols are concerned like Ungermann-Bass routers.

Our routers also can be configured to listen to standard RIP updates when in Ungermann-Bass Net/One emulation mode. When our router in Ungermann-Bass emulation mode receives a RIP packet, each route in that packet is treated as though it had come from an Ungermann-Bass routing packet. The hop count used is the actual hop count from the RIP packet. The delay metric used is computed by assuming that each hop is the longest-delay link used by Ungermann-Bass, which is a 9.6-kbps serial link. Information from RIP packets is used in creating outgoing Ungermann-Bass updates, and vice versa.

Note Older versions of our software implemented a restricted version of the Ungermann-Bass routing protocol. Using that software in certain configurations could create routing instability and forwarding loops. If you are planning to use Software Releases 8.3 and earlier in Ungermann-Bass environments, consult the Release 8.3 documentation for information about the restrictions.

XNS Addresses

An XNS address consists of a network number and a host number expressed in the format *network.host*.

The network number identifies a physical network. It is a 32-bit quantity that must be unique throughout the entire XNS internetwork. The network number is expressed in decimal. A network number of zero identifies the local network. The XNS network number is expressed in decimal format in our configuration files and routing tables. However, the router internally converts the network number into hexadecimal. This means, for instance, that a network analyzer will display the network number in hexadecimal.

The host number is a 48-bit quantity that is unique across all hosts ever manufactured. It is represented by dotted triplets of four-digit hexadecimal numbers.

The following is an example of an XNS address:

```
47.0000.0x00.23fe
```

When XNS routing is enabled, the address used is either the IEEE-compliant address specified in the XNS routing configuration command or the first IEEE-compliant address in the system. The address also is used as the node address for non-LAN media, notably serial links.

Configuration Task List

To configure XNS routing, complete the tasks in the following sections. At a minimum, you must enable routing.

- Enable XNS Routing
- Control Access to the XNS Network
- Tune XNS Network Performance
- Configure XNS over WANs
- Monitor the XNS Network

See the end of this chapter for configuration examples.

Enable XNS Routing

When enabling XNS routing, you can enable either standard XNS routing or Ungermann-Bass Net/One routing. You use standard routing for XNS networks that do not have any Ungermann-Bass systems. You use Net/One routing for networks that do have Ungermann-Bass systems.

Standard XNS routing uses the standard XNS RIP protocol, while Net/One uses an Ungermann-Bass proprietary routing protocol. Net/One routers generate both Ungermann-Bass update packets and standard RIP update packets; however, they listen only to Ungermann-Bass updates. The standard XNS RIP uses a hop count to determine the best route to a distant network, while the Ungermann-Bass protocol uses a path-delay metric. Our router supports both the reception and the generation of Ungermann-Bass routing packets.

Enable Standard XNS Routing

To enable standard XNS routing, perform the following tasks:

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Enable XNS routing on the router.	**xns routing** [*address*]
Step 3	Enter interface configuration mode.	**interface** *type number*
Step 4	Enable XNS routing on an interface.	**xns network** *number*

For an example of how to enable XNS routing, see the section “XNS Configuration Examples.”

If you omit the address from the **xns routing** command, the router uses the address of the first IEEE-compliant (Token Ring, FDDI, or Ethernet) interface MAC address it finds in its interface list. The router uses the address 0123.4567.abcd for non-IEEE–compliant interfaces.

To forward XNS packets across a Token Ring interface, you must specify an XNS encapsulation type to use on the interface. To do this, perform one of the following tasks in interface configuration mode:

Task	Command
Encapsulate XNS packets being forwarded across an IBM Token Ring network.	**xns encapsulation snap**
Encapsulate XNS packets being forwarded across an Ungermann-Bass Token Ring network.	**xns encapsulation ub**
Encapsulate XNS packets being forwarded across an older 3Com Token Ring network.	**xns encapsulation 3com**

Enable Ungermann-Bass Net/One Routing

To enable Ungermann-Bass Net/One routing, perform Step 1 and Step 2, and optionally perform Step 3 and Step 4:

Task		Command
Step 1	Enter global configuration mode.	See Table 2-1.
Step 2	Enable Ungermann-Bass Net/One routing on the router.	**xns ub-emulation**
Step 3	Enter interface configuration mode.	**interface** *type number*
Step 4	Enable the receipt of RIP updates on the interface.	**xns hear-rip** [*access-list-number*]

For an example of how to enable Ungermann-Bass Net/One routing, see the section "Enabling and Configuring Net/One Routing Configuration Example."

Control Access to the XNS Network

To control access to XNS networks, you create access lists and then apply them with filters to individual interfaces.

There are two types of XNS access lists that you can use to filter routed traffic:

- Standard access list—Restricts traffic based on the source network number. You can further restrict traffic by specifying a destination address and a source and destination address mask. Standard XNS access lists have numbers from 400 to 499.
- Extended access—Restricts traffic based on the XNS protocol type. You can further restrict traffic by specifying source and destination addresses and address masks, and source and destination socket numbers and masks. Extended XNS access lists have numbers from 500 to 599.

There are two different types of filters you can define for XNS interfaces, and you can apply one of each type to each interface:

- Generic filters—These filters control which packets are sent out an interface based on the packet's source and destination addresses, source and destination socket numbers, and XNS protocol type.
- Routing table filters—These filters control which routing (RIP) updates are accepted and advertised by the router and which routers the local router accepts RIP updates from.

Table 21-1 summarizes the types of filters and the commands you use to define them. Use the **show xns interface** command to display the filters defined on an interface.

Table 21-1 XNS Filters

Filter Type	Command Used to Define Filter
Generic filters	
Filter based on protocol, address and address mask, port and port mask.	**xns access-group** *access-list-number*
Routing table filters	
Filter which networks are added to the routing table.	**xns input-network-filter** *access-list-number*
Filter which networks are advertised in routing updates.	**xns output-network-filter** *access-list-number*
Control the routers from which updates are accepted.	**xns router-filter** *access-list-number*

You perform one or more of the tasks in the following sections to control access to XNS networks:

- Create Access Lists
- Create Generic Filters
- Create Filters for Updating the Routing Table

Keep the following in mind when configuring XNS network access control:

- Access lists entries are evaluated in the order you enter them, and the first matching entry is used. To improve performance, place the most commonly matched entries near the beginning of the access list.
- An implicit *deny everything* entry is 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.

Create Access Lists

To create access lists, perform one or more of the following tasks in global configuration mode:

Task	Command
Create a standard XNS access list.	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-address* [*source-address-mask*]] *[destination-network*[.*destination-address* [*destination-address-mask*]]]
Create an extended XNS access list.	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[.*source-host* [*source-network-mask.*]*source-host-mask*] *source-socket* [*destination-network* [.*destination-host* [*destination-network-mask.destination-host-mask*] [*destination-socket/*[*PEP*]]]

Once you have created an access list, you apply it to a filter on the appropriate interfaces as described in the sections that follow. This activates the access list.

Create Generic Filters

Generic filters determine which packets to send out an interface based on the packet's source and destination addresses, XNS protocol type, and source and destination socket numbers.

To create generic filters, perform the following tasks:

Step 1 Create a standard or extended access list.

Step 2 Apply a filter to an interface.

To create an access list, perform one of the following tasks in global configuration mode:

Task	Command
Create a standard XNS access list.	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-address* [*source-address-mask*]] *[destination-network*[.*destination-address* [*destination-address-mask*]]]
Create an extended XNS access list.	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[.*source-host* [*source-network-mask.*]*source-host-mask*] *source-socket* [*destination-network* [.*destination-host* [*destination-network-mask.destination-host-mask*] [*destination-socket/*[*PEP*]]]

To apply a generic filter to an interface, perform the following task in interface configuration mode:

Task	Command
Apply a generic filter to an interface.	**xns access-group** *access-list-number*

For an example of creating a generic access list, see the section "3Com Access List Example."

Create Filters for Updating the Routing Table

Routing table update filters control the entries that the router accepts for its routing table and the networks that it advertises in its routing updates.

To create filters to control updating of the routing table, perform the following tasks:

Step 1 Create a standard or an extended access list.

Step 2 Apply one or more filters to an interface.

To create an access list, perform one of the following tasks in global configuration mode:

Task	Command
Create a standard XNS access list.	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-address* [*source-address-mask*]] [*destination-network*[.*destination-address* [*destination-address-mask*]]]

Task	Command
Create an extended XNS access list.	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[*.source-host* [*source-network-mask.*]*source-host-mask*] *source-socket* [*destination-network* [*.destination-host* [*destination-network-mask.destination-host-mask*] [*destination-socket/*[*PEP*]]]

Standard access list numbers can be from 400 to 499. Extended access list numbers can be from 500 to 599.

To assign routing table update filters to an interface, perform one of the following tasks in interface configuration mode. You can apply one of each of the following filters to each interface.

Task	Command
Control which networks are added to the routing table when XNS routing updates are received.	**xns input-network-filter** *access-list-number*
Control which networks are advertised in routing updates sent out by the router.	**xns output-network-filter** *access-list-number*
Control the routers from which routing updates are accepted.	**xns router-filter** *access-list-number*

Tune XNS Network Performance

To tune XNS network performance, perform the tasks in one or more of the following sections:

- Configure Static Routes
- Set Routing Table Update Timers
- Set Maximum Paths
- Control Broadcast Messages
- Disable XNS Fast Switching

Configure Static Routes

XNS uses the (RIP) to determine the best path when several paths to a destination exist. RIP then dynamically updates the routing table. Static routes usually are not used in XNS environments because nearly all XNS routers support dynamic routing with RIP. However, you might want to add static routes to the routing table to explicitly specify paths to certain destinations. Static routes always override any dynamically learned paths.

Be careful when assigning static routes. When links associated with static routes are lost, traffic might stop being forwarded, even though an alternative path might be available.

To add a static route to the router's routing table, perform the following task in global configuration mode:

Task	Command
Add a static route to the routing table.	**xns route** *network network.host*

Set Routing Table Update Timers

You can set the delay between XNS RIP updates. Normally, RIP sends routing table updates every 30 seconds.

You can set RIP timers only in a configuration in which all routers are our routers, because the timers for all routers connected to the same network segment should be the same and because you cannot set the timer for systems running the Ungermann-Bass routing protocol.

The RIP update value you choose affects internal XNS timers as follows:

- XNS routes are marked invalid if no routing updates are heard within three times the value of *the update interval* ($3 \times$ *interval*) and are advertised with a metric of infinity.
- XNS routes are removed from the routing table if no routing updates are heard within six times the value of *the update interval* ($6 \times$ *interval*).

To set the RIP update timers, perform the following task in interface configuration mode:

Task	Command
Set the RIP update timer.	**xns update-time** *interval*

For an example of setting the RIP update timer, see the section "Routing Update Timers Example."

Set Maximum Paths

You can set the maximum number of equal-cost, parallel paths to a destination. (Note that when paths have differing costs, the router chooses lower-cost routes in preference to higher-cost routes.) The router distributes output on a packet-by-packet basis in round-robin fashion. That is, the first packet is sent along the first path, the second packet along the second path, and so on. If the final path is reached before all packets are sent, the next packet is sent to the first path, the next to the second path, and so on. This round-robin scheme is used regardless of whether fast switching is enabled.

Limiting the number of equal-cost paths can save memory on routers with limited memory or very large configurations. Additionally, in networks with a large number of multiple paths and systems with limited ability to cache out-of-sequence packets, performance might suffer when traffic is split between many paths.

To set the maximum number of paths on the router, perform the following task in global configuration mode:

Task	Command
Set the maximum number of equal-cost paths to a destination.	**xns maximum-paths** *number*

Control Broadcast Messages

Network end nodes often send broadcast messages to discover services: a request is broadcast to many or all nodes in the internetwork, and one or more of the nodes that can offer that service reply. Both end nodes and routers sometimes send broadcast messages to contain data that must be received by many other nodes. An example is RIP routing updates.

Although broadcast messages can be very useful, they are not without costs. Every node on a physical network must receive and process all broadcasts sent on that network, even if the processing consists of ignoring the broadcasts. If many nodes answer the broadcast, network load might increase dramatically for a short period of time. Also, if the broadcast is propagated to more than one physical network, there is extra load on the networks and the intervening routers.

The following are the types of broadcasts and how each is handled:

- A *local* broadcast is one that is intended only for nodes on the physical network (typically one Ethernet or Token Ring LAN) on which the packet is originally sent. XNS networks usually denote local broadcasts with a specific network number in the packet's destination field. If a node does not know the number of the local XNS network (common when booting), it can use a network number of zero to denote a local broadcast.
- An *all-nets* broadcast is one that is intended for all nodes throughout the XNS internet. XNS networks usually denote all-nets broadcasts with a destination network field of all ones (typically written as –1 or as FFFFFFFF).
- A *directed* broadcast is one that is intended for all nodes on a specific remote network. Directed broadcasts are denoted by the use of a specific remote network number in the destination field.

All these broadcast types use the host address FFFF.FFFF.FFFF in the packet's destination host field. The destination MAC address used in the underlying LAN frame is the broadcast address. Directed broadcasts intended for remote networks can be sent directly to the MAC address of a router that provides the path to their ultimate destination, and physically broadcast only when they reach it.

Some implementations expect all broadcasts to be treated as local broadcasts. Others expect broadcasts with destination network fields of zero to be treated as all-nets broadcasts. Some do not support directed broadcasts. In addition, some implementations expect packets with destination network fields of all ones, but with destination node fields that correspond to specific hosts, to be flooded throughout the internet as MAC-layer broadcasts. This way, nodes can be located without knowledge of which physical networks they are connected to. We support all these models by using helpering and flooding features.

Helpering, which is typically used for service discovery broadcasts, sends the broadcasts to user-specified candidate servers on remote networks. When a packet is helpered, the router changes its destination address to be the configured helper address, and the packet then is routed toward that address. The host at the helper address is expected to process the packet and (usually) to reply to the packet's sender. A helper address can be a directed broadcast address, in which case the helpered packet will be forwarded to a remote network and rebroadcast there.

Flooding sends packets throughout the entire XNS internet. Flooded packets are not modified, except for the hop count field. Flooding is useful when many nodes throughout the internet need to receive a packet or when a service that can be anywhere in the internet must be discovered. You should avoid flooding in large, slow, or heavily loaded networks because the load on the routers, links, and end nodes by heavy flooded traffic is large.

Many broadcast messages are sent when a host first becomes active on the network. A host will generate a broadcast packet when it does not know the current address of the host that is supposed to receive its next packet—the local server, for instance. Generally, it is not a good idea to place a router between users and the servers that carry their primary applications; you should minimize internet traffic. However, if you need that server configuration for some other reason, you need to ensure that users can broadcast between networks without cluttering the internetwork with unnecessary traffic.

Whenever our router receives an XNS broadcast packet, it processes it as follows:

- If the packet is a routing update or requests services that are offered by the router itself, the packet is processed by the router and is not forwarded any further.
- If a helper address is set on the interface on which the packet arrived and the packet's protocol type appears in the **xns forward-protocol** list, the packet is forwarded to the helper address. The helper address can be a directed broadcast address.

Forward Broadcast Messages to Specified Hosts

To configure helpering, which forwards broadcast messages to the specified host or hosts, perform the following task in interface configuration mode:

Task	Command
Forward broadcast messages to the specified host.	**xns helper-address** *network.host*

You can specify multiple **xns helper-address** commands on a given interface.

For an example of forwarding broadcast messages, see the section "Helpering Example."

Specify XNS Protocol Types for Forwarding Broadcast Messages

When considering which packets will be forwarded via helpering, you can forward packets that have been generated by a specific XNS protocol. To do this, perform the following task in global configuration mode:

Task	Command
Forward packets of a specific XNS protocol to a helper address.	**xns forward-protocol** *protocol*

Configure Flooding

Different XNS implementations require different flooding behavior. By default, our routers do no flooding. You can, however, configure interfaces on the router to apply flooding to the packets *received* on an interface.

Whenever our router receives an XNS broadcast packet, it processes it for flooding. An *all-nets* broadcast is one that is forwarded to all networks. XNS networks usually indicate all-nets broadcasts by setting the destination network address to all ones (typically written as –1 or FFFFFFFF). Packets with –1 destination networks and specific destination hosts are sent as MAC-layer broadcasts so that they can be picked up and further flooded by other routers. Flooding is applied to packets *received* on the interfaces.

Our router chooses the interfaces through which flooded packets are sent using rules designed to avoid packet looping and most packet duplication. The underlying principle of these rules is that packets should be flooded *away* from their sources, never *toward* them. Packets that the router is configured to flood are sent out through all interfaces, except in the following cases:

- Packets that would ordinarily be flooded are ignored unless they are received via the interface that would be used to route a unicast packet to the flooded packet's *source* network. If there are multiple paths to the source network, packets received only on the primary path (the first path the router learned) are flooded. If a packet is received on an interface that fails this rule, the interface that passes it will receive another copy of that packet.

- Packets are never flooded out of the router through any interface that is one of the router's paths back toward the packet's source. A copy of the flooded packet will appear on the network connected to such an interface via some other path.
- If the router has no route to a packet's source network, the packet is not flooded. This is to prevent odd behavior during routing convergence after network topology changes.
- Packets that fail the access lists applied to outgoing interfaces are not flooded through those interfaces.
- Packets with destination networks and specific destination hosts are sent as MAC-layer broadcasts so that they can be picked up and further flooded by other routers.
- For backward compatibility, any attempt to set a helper address of –1.FFFF.FFFF.FFFF on an interface will result in that interface's having no helper address set, but having **xns flood broadcast allnets** enabled.

To define an interface's flooding behavior, perform one of the following tasks in interface configuration mode:

Task	Command
Flood packets whose destination address is –1.FFFF.FFFF.FFFF.	**xns flood broadcast allnets**
Flood packets whose destination address is 0.FFFF.FFFF.FFFF.	**xns flood broadcast net-zero**
Flood packets with destinations of –1.*specific-host.*	**xns flood specific allnets**

It is most closely in accordance with the XNS specification to flood packets with destinations of –1.FFFF.FFFF.FFFF and destinations of –1.*specific-host*, but not to flood packets with destinations of 0.FFFF.FFFF.FFFF. However, 3Com environments often require flooding of broadcast whose network address is all zeros.

Disable XNS Fast Switching

Fast switching allows higher throughput by switching a packet using a cache created by previous packets. Fast switching is enabled by default on all interfaces.

Packet transfer performance is generally better when fast switching is enabled. However, you may want to disable fast switching in order to save memory space on interface cards and to help avoid congestion when high-bandwidth interfaces are writing large amounts of information to low-bandwidth interfaces.

To disable XNS fast switching on an interface, perform the following task in interface configuration mode:

Task	Command
Disable XNS fast switching.	**no xns route-cache**

Configure XNS over WANs

You can configure XNS over X.25, Frame Relay, and SMDS networks. To do this, configure the appropriate address mappings as described in the "Configuring X.25 and LAPB," "Configuring Frame Relay," and "Configuring SMDS" chapters.

Monitor the XNS Network

To monitor an XNS network, perform one or more of the following tasks at the EXEC prompt:

Task	Command
List the entries in the XNS fast-switching cache.	**show xns cache**
Display the status of the XNS interfaces configured in the router and the parameters configured on each interface.	**show xns interface** [*type number*]
List the entries in the XNS routing table.	**show xns route** [*network*]
Display information about the number and type of XNS packets transmitted and received.	**show xns traffic**
Diagnose basic XNS network connectivity (user-level command).	**ping xns** *address*
Diagnose basic XNS network connectivity (privileged command).	**ping**

XNS Configuration Examples

Use the following configuration examples to help in configuring XNS routing on your router:

- Enabling XNS Routing Configuration Example
- Enabling and Configuring Net/One Routing Configuration Example
- Routing Update Timers Example
- 3Com Access List Example
- Extended Access List with Network Mask Option Example
- Helpering Example

Enabling XNS Routing Configuration Example

The following example enables XNS routing on the router and then enables XNS on three interfaces. On the Ethernet interfaces, the router uses the preassigned MAC-level addresses as XNS host addresses. On the serial interface, the router uses the MAC address associated with the first IEEE 802 interface found on the router.

```
xns routing
interface ethernet 0
xns network 20
interface ethernet 1
xns network 21
interface serial 1
xns network 24
```

Enabling and Configuring Net/One Routing Configuration Example

The following example enables Ungermann-Bass Net/One routing. Serial interface 0 is connected to a non-Net/One portion of the XNS internet, so the **xns hear-rip** command is issued to allow the learning of routes from the standard RIP updates used by the remote routers. There are Ungermann-Bass nodes connected to interface Token Ring 0, so the encapsulation on that interface is set to Ungermann-Bass. Broadcast flooding is configured to match the expectations of Net/One.

```
xns routing
xns ub-emulation
!
interface tokenring 0
xns network 23
xns flood broadcast allnets
xns encapsulation ub
xns flood specific allnets
!
interface ethernet 0
xns network 20
xns flood broadcast allnets
xns flood specific allnets
!
interface ethernet 1
xns network 21
xns flood broadcast allnets
xns flood specific allnets
!
interface serial 0
xns network 24
xns hear-rip
xns flood broadcast allnets
xns flood specific allnets
```

Routing Update Timers Example

The following example creates a routing process that specifies a specific address for use on serial lines and other non-802.x interfaces. It also sets the RIP routing update timers for the three interfaces.

```
xns routing 0000.0C53.4679
!
interface ethernet 0
xns network 20
xns update-time 20
!
interface serial 0
xns network 24
xns update-time 20
!
interface ethernet 1
xns network 21
xns update-time 20
```

3Com Access List Example

The following partial example controls specific services between networks 1002 and 1006 in a 3Com network. Echo and error packets, as well as all Sequenced Packet Protocol (SPP) and Packet Exchange Protocol (PEP) (that is, normal data traffic) can go from network 1002 to network 1006. However, all NetBIOS requests are denied. The final three lines are blanket permissions for RIP, SPP, and PEP packets.

```
access-list 524 permit 2 1002 0x0000 1006 0x0000
!  permit Echo from 1002 to 1006
access-list 524 permit 3 1002 0x0000 1006 0x0000
!  permit Error from 1002 to 1006
access-list 524 deny 5 -1 0x0000 -1 0x046B
!  deny all NetBIOS
access-list 524 permit 4 1002 0x0000 1006 0x0000
!  permit PEP from 1002 to 1006
access-list 524 permit 5 1002 0x0000 1006 0x0000
!  permit SPP from 1002 to 1006
access-list 524 permit 1
!  permit all RIP
!
!These are needed if you want PEP and SPP to be permitted from
!networks other than 1002
access-list 524 permit 4
!  permit all PEP
access-list 524 permit 5
!  permit all SPP
```

Extended Access List with Network Mask Option Example

The following example allows protocol type 20 on any socket, from a certain make of machine (Cisco Ethernet), on network 10 to access any hosts on networks 1000 through 1015 on any socket.

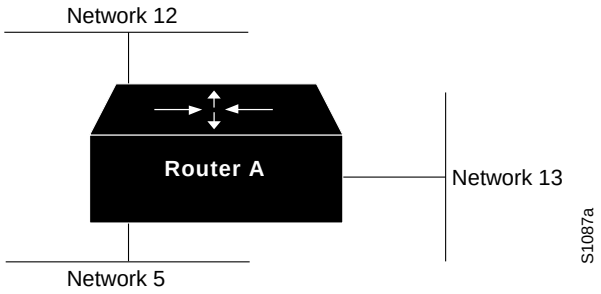
```
access-list 505 permit 20 10.0000.0C00.0000 0000.0000.FFFF 0
1000.0000.0000.0000 15.FFFF.FFFF.FFFF 0
```

Helpering Example

The following commands set up helpering for the configuration shown in Figure 21-1. The router forwards packets of protocol type 1 only. Ethernet interface 0 has a helper address set, with the helper on network 12 available through the Ethernet interface 2.

```
xns routing
xns forward-protocol 1
interface ethernet 0
xns network 5
xns helper address 12.FFFF.FFFF.FFFF
interface ethernet 1
xns network 13
interface ethernet 2
xns network 12
```

Figure 21-1 Helper Addresses



In this example, the following actions will be taken on broadcast packets:

- Broadcast packets with a network address of 5 are forwarded to the helper address on network 12.
- Broadcast packets addressed to network 0 also are forwarded to the helper address on network 12.
- Broadcast packets addressed to network 13 are directed broadcasts and are sent through the E1 interface directly to network 13. They are not sent to the helper address.
- Broadcast packets of protocol 1 that are destined for network 5 are sent to the helper address.
- Broadcast packets not of protocol 1 that are destined for network 5 are discarded because they do not match the specified protocol.
- Broadcast packets of protocol 1 that are destined for network 0 are sent to the helper address.
- Broadcast packets not of protocol 1 that are destined for network 0 are discarded.

Broadcast packets destined for network 12 are directed broadcasts and are broadcast on Ethernet interface 2 to network 12. This has nothing to do with helpering or protocol forwarding.

CHAPTER 22

Configuring Transparent Bridging

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

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

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

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

Transparent Bridging Features

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

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

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

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

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

Source-Route Transparent Bridging Features

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

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

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

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

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

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

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

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

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

Transparent and SRT Bridging Configuration Task List

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

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

See the "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** *bridge-group* **protocol** {**ieee** | **dec**}

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

Assign Each Network Interface to a Bridge Group

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

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

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

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

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

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

Choose the OUI for Ethernet Type II Frames

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

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

Configure Transparently Bridged Virtual LANs

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 virtual LAN feature permits a bridge group to extend outside the router to identify traffic switched within the bridge group.

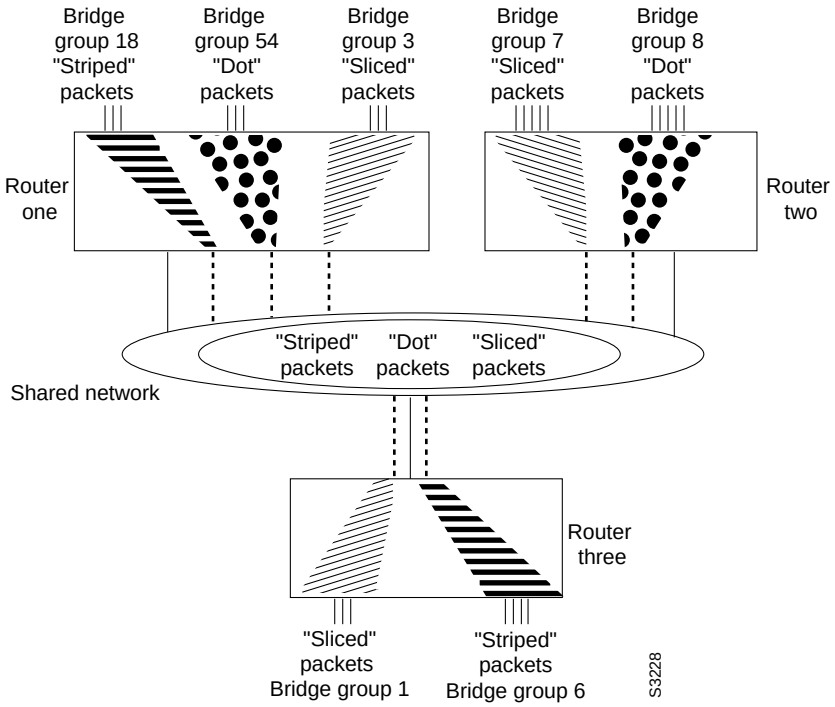
While bridge groups remain internal organizations of network interfaces functioning as distinct bridges within a router, transparent bridging on subinterfaces permits bridge groups to be used to multiplex different streams of bridged traffic on a LAN or HDLC serial interface. In this way, bridged traffic 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 virtual LAN. This approach can be extend to impose logical topologies upon transparently bridged networks.

The primary application of transparently bridged virtual LANs 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 virtual LANs composed of collections of bridge groups on several routers. These virtual LANs improve performance because they reduce the propagation of locally bridged traffic, and they improve security benefits because they completely separate traffic.

In Figure 22-1, different bridge groups on different routers are configured into three virtual LANs that span the bridged network. Each bridge group consists of conventionally bridged local interfaces and a sub-interface on the backbone FDDI LAN. Bridged traffic on the sub-interface is encapsulated and “colored” with a virtual LAN identifier known as a Security Association Identifier common to all bridge groups participating in the virtual LAN. In addition, bridges only accept packets bearing Security Association IDs for which they have a configured subinterface. Thus a bridge group is configured to participate in a virtual LAN if it contains a sub-interface configured with the virtual LANs characteristic Security Association ID. See the section “Transparently Bridged Virtual LAN Configuration Example” later in this chapter for an example configuration of the topology shown in Figure 22-1.

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 maximum transmission unit (MTU) size of the LAN from which the packet originated. In order 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 22-1 Transparently Bridged Virtual LANs on an FDDI Backbone



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

Task	Command
Specify a subinterface.	**interface** *interface-type/ interface-number.subinterface-number*
Specify the IEEE 802.10 Security Data Exchange (SDE) Security Association ID. (In other words, specify the "color.")	**encapsulation sde** *said*
Associate the subinterface with an existing bridge group.	**bridge-group** *group*

Note Transparently bridged virtual LANs are supported only in conjunction with the IEEE spanning-tree protocol. In logically segmenting a transparently bridged network into virtual LANs, each virtual LAN computes its own spanning-tree topology. This prevents traffic bridged within one virtual LAN from being adversely affected by physical topology changes occurring within another virtual LAN elsewhere in the network. Configuring each virtual LAN to compute its own spanning-tree topology provides much greater stability than running a single spanning tree throughout

Configure Transparent Bridging over WANs

We support transparent bridging over the following types of networks:

- X.25

- Frame Relay
- SMDS
- PPP

This section describes how to configure bridging on these networks.

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

Configure X.25 Transparent Bridging

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

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

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

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

Configure Frame Relay Transparent Bridging

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

Bridging in a Frame Relay Network with No Multicasts

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

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

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

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

Bridging in a Frame Relay Network with Multicasts

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

Configure SMDS Transparent Bridging

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

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

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

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

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

Configure Transparent Bridging Options

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

- Disable IP Routing
- Enable Autonomous Bridging
- Configure LAT Compression
- Establish Multiple Spanning-Tree Domains
- Prevent the Forwarding of Dynamically Determined Stations
- Forward Multicast Addresses
- 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 all routers/bridges. You can enable IP routing when you decide to route IP packets. To disable or enable IP routing, perform one of the following tasks in global configuration mode:

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

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

Enable Autonomous Bridging

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

- MEC
- FCIT transparent
- HSSI HDLC

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

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

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

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

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

Configure LAT Compression

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

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

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

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

Establish Multiple Spanning-Tree Domains

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

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

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

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

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

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

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

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

Prevent the Forwarding of Dynamically Determined Stations

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

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

Forward Multicast Addresses

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

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

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

Configure Bridge Table Aging Time

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

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

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

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

Filter Transparently Bridged Packets

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

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

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

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

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

Filter by MAC-Level Address

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

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

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

- Define and apply extended access lists

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

Filter by Specific MAC Address

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

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

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

Filter by Vendor Code

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

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

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

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

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

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

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

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

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

Filter by Protocol Type

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Define and Apply Extended Access Lists

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

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

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

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

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

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

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

Filter LAT Service Announcements

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

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

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

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

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

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

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

Enable LAT Group Code Service Filtering

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

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

Specify Deny or Permit Conditions for LAT Group Codes on Input

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

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

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

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

Specify Deny or Permit Conditions for LAT Group Codes on Output

You can specify the group codes by which to deny or permit access upon output. Specifying deny 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** *bridge-group* **output-lat-service-deny** *group-list*
Specify the group codes with which to permit access upon output.	**bridge-group** *bridge-group* **output-lat-service-permit** *group-list*

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

Adjust Spanning-Tree Parameters

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

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

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

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

Set the Bridge Priority

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

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

Set an Interface Priority

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

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

Assign Path Costs

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

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

Adjust BPDU Intervals

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

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

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

Adjust the Interval between Hello BPDUs

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

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

Define the Forward Delay Interval

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

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

Define the Maximum Idle Interval

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

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

Disable the Spanning Tree on an Interface

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

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

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

Tune the Transparently Bridged Network

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

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

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

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

Step 1 Define a circuit group.

Step 2 Optionally, configure a transmission pause interval.

Step 3 Modify the load distribution strategy.

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

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

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

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

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

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

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

Monitor and Maintain the Transparent Bridge Network

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

Task	Command
Remove any learned entries from the forwarding database and clear the transmit and receive counts for any statically configured forwarding entries.	**clear bridge** *bridge-group*
Reinitialize the Silicon Switch Processor (SSP) on the Cisco 7000 series.	**clear sse**

Task	Command
Display classes of entries in the bridge forwarding database.	**show bridge** [*bridge-group*] [*interface*] [*address*] [*mask*] [**verbose**]
Display the interfaces configured in each circuit group and show whether they are participating in load distribution.	**show bridge** [*bridge-group*] **circuit-group** [*circuit-group*] [*src-mac-address*] [*dst-mac-address*]
Display IEEE 802.10 transparently bridged virtual LAN configuration.	**show bridge vlan**
Display information about configured bridge groups.	**show bridge group** [**verbose**]
Display the spanning-tree topology known to the router/bridge, including whether or not filtering is in effect.	**show span**
Display a summary of SSP statistics.	**show sse summary**

Transparent and SRT Bridging Configuration Examples

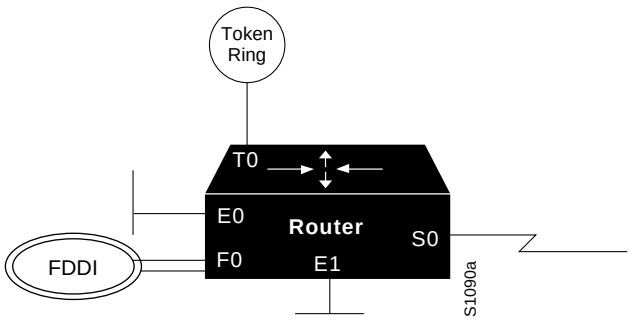
The following sections provide example configurations that you can use as a guide to configuring your bridging environment:

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

Basic Bridging Example

Figure 22-2 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 22-2 Example of Basic Bridging



The configuration file for the router/bridge depicted in Figure 22-2 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
```

Transparently Bridged Virtual LANs Configuration Example

The following example shows the configuration for the topology shown in Figure 22-1. The "striped" virtual LAN is identified as Security Association ID 45; the "dot" virtual LAN is identified as Security Association ID 1008; the "sliced" virtual LAN is identified as Security Association ID 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 ID across bridges.

Router One

```
bridge 18 protocol ieee
int e 0/1
bridge-group 18
int e 0/2
bridge group 18
int e 0/3
bridge-group 18
int f 4/0.8
encapsulation sde 45
bridge-group 18
```

```
bridge 54 protocol ieee
int e 1/1
bridge group 54
int e 1/2
bridge group 54
int e 1/3
bridge group 54
int f 4/0.13
encapsulation sde 1008
bridge group 54

bridge 3 protocol ieee
int e 2/1
bridge-group 3
int e 2/2
bridge-group 3
int e 2/3
bridge-group 3
int f 4/0.30
encapsulation sde 4321
bridge-group 3
```

Router Two

```
bridge 7 protocol ieee
int e 0/1
bridge-group 7
int e 0/2
bridge-group 7
int e 0/3
bridge-group 7
int e 0/4
bridge-group 7
int f 2/0.11
encapsulation sde 4321
bridge-group 7

bridge 8 protocol ieee
int e 1/1
bridge-group 8
int e 1/2
bridge-group 8
int e 1/3
bridge-group 8
int e 1/4
bridge-group 8
int f 2/0.14
encapsulation sde 1008
bridge-group 8
```

Router Three

```
bridge 1 protocol ieee
int e 0/1
bridge-group 1
int e 0/2
bridge-group 1
int e 0/3
bridge-group 1
int f 2/0.5
encapsulation sde 4321
bridge-group 1
```

```
bridge 6 protocol ieee
int e 1/1
bridge-group 6
int e 1/2
bridge-group 6
int e 1/3
bridge-group 6
int f 2/0.3
encapsulation sde 45
bridge-group 6
```

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
```

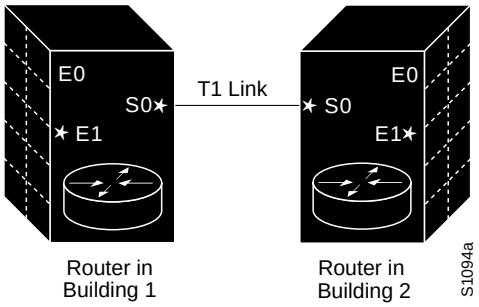
Ethernet Bridging Example

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 22-3 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 22-3 Ethernet Bridging Configuration Example



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

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

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

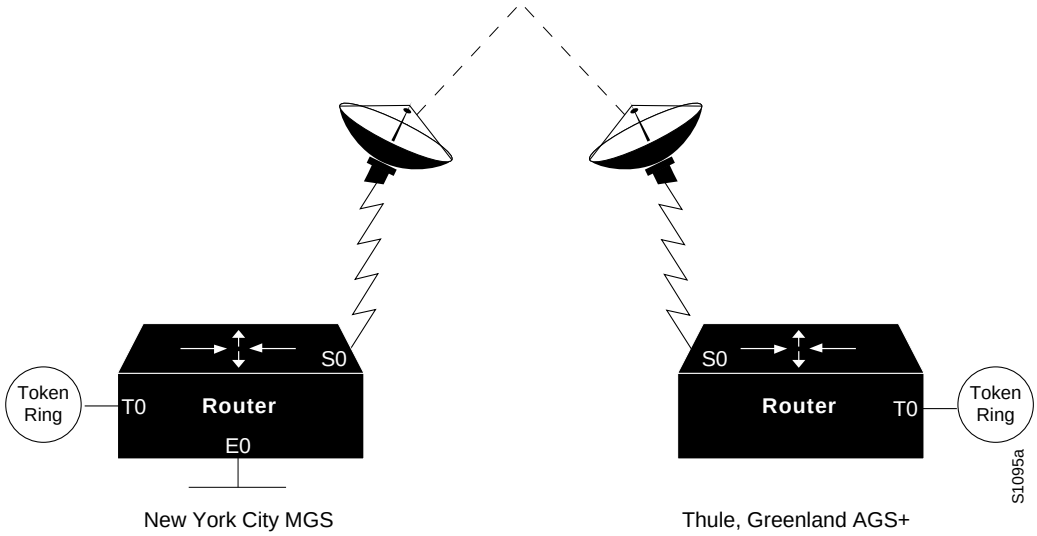
```
decnet address 3.56
!
interface ethernet 0
ip address 128.88.11.9 255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.2 255.255.255.0
bridge-group 1
decnet cost 10
!
interface ethernet 1
ip address 128.88.16.8 255.255.255.0
bridge-group 1
decnet cost 10
!
bridge 1 protocol dec
```

SRT Bridging Example

In Figure 22-4, 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 22-4 Example Network Configuration



Configuration for the New York City Router

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

Configuration for the Thule, Greenland Router

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

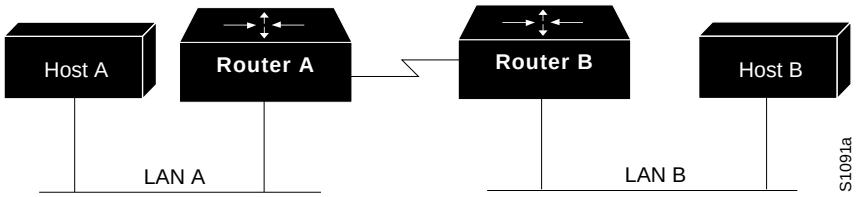
```
!
bridge 1 protocol ieee
```

Multicast or Broadcast Packets Bridging Example

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

Assume you are bridging IP in your network as illustrated in Figure 22-5.

Figure 22-5 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
```

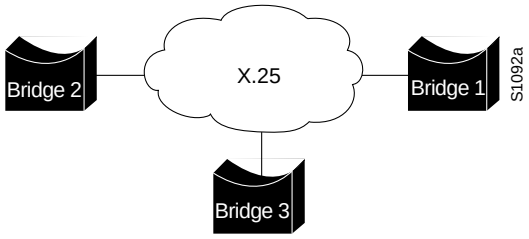
The correct access list would be as follows:

```
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 22-6 is an example configuration illustrating three bridges connected to each other through an X.25 network.

Figure 22-6 X.25 Bridging Examples



Following are the configuration commands for each of the bridges depicted in Figure 22-6.

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
```

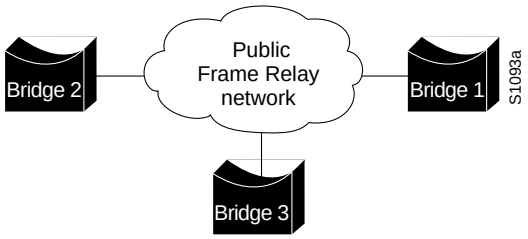
Configuration for Bridge 3

```
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 22-7 illustrates three bridges connected to each other through a Frame Relay network.

Figure 22-7 Frame Relay Bridging Example



Bridging in a Frame Relay Network with No Multicasts

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

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

Configuration for Bridge 1

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

Configuration for Bridge 2

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

Configuration for Bridge 3

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

Bridging in a Frame Relay Network with Multicasts

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

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

Configuration for Bridge 1

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

Configuration for Bridge 2

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

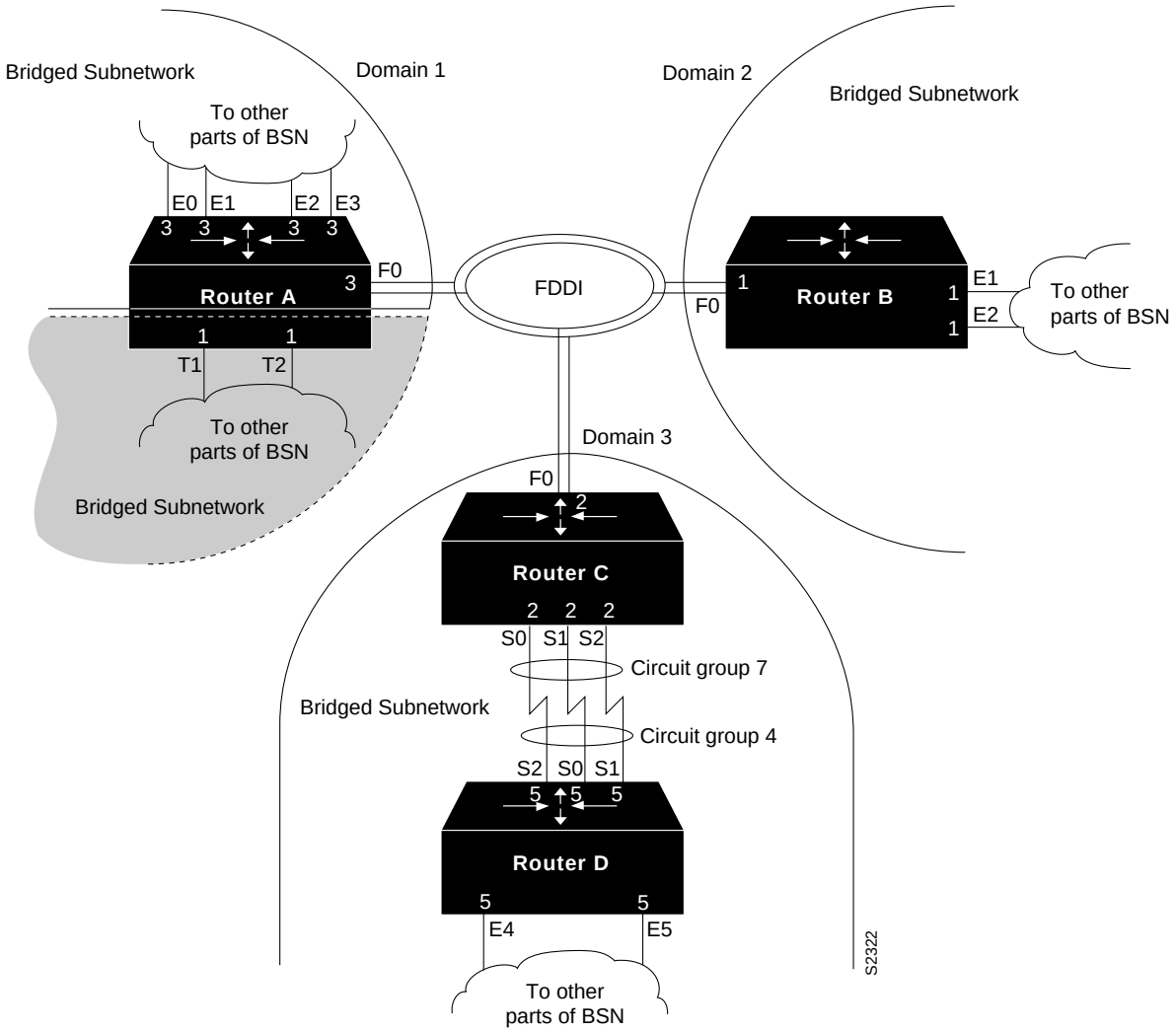
Configuration for Bridge 3

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

Complex Transparent Bridging Network Topology Example

Figure 22-8 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 22-8 Bridged Subnetworks with Domains



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

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

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

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

Configuration for Router A

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

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

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

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

Configuration for Router B

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

```
interface Ethernet1
bridge-group 1
!
interface Ethernet2
bridge-group 1
```

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

Configuration for Router C

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

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

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

Configuration for Router D

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

CHAPTER 23

Configuring Source-Route Bridging

Our bridging software includes source-route bridging (SRB) capability. A source-route bridge connects multiple physical Token Rings into one logical network segment. When the network segment bridges only Token Ring media to provide connectivity, it is called source-route bridging. When the network bridges Token Ring, and some sort of non-Token Ring media is introduced into the bridged network segment, it is called remote source-route bridging (RSRB).

The source-route bridging feature enables our router/bridge to simultaneously act as a Level 3 router and a Level 2 source-route bridge. Thus, protocols such as Novell's Internetwork Packet Exchange (IPX) or Xerox Network Systems (XNS) can be routed on Token Rings, while other protocols such as Systems Network Architecture (SNA) or NetBIOS are source-route bridged.

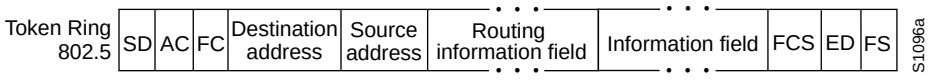
For a complete description of the commands mentioned in this chapter, refer to the chapter "Source-Route Bridging Commands" in the *Router Products Command Reference* publication. For historical background and a technical overview of source-route bridging, see the *Internetworking Technology Overview* publication.

Source-Route Bridging Overview

Source-route bridging technology is a combination of bridging and routing functions. A source-route bridge is allowed to make routing decisions based upon the contents of the Media Access Control (MAC) frame header. Keeping the routing function at the MAC, or Level 2, layer allows the higher-layer protocols to execute their tasks more efficiently and allows the local-area network (LAN) to be expanded without the knowledge of the higher-layer protocols.

As designed by IBM and the IEEE 802.5 committee, source-route bridges connect extended Token Ring LANs. A source-route bridge uses the routing information field (RIF) in the IEEE 802.5 MAC header of a datagram (see Figure 23-1) to determine which rings or Token Ring network segments the packet must transit. The source station inserts the RIF into the MAC header immediately following the source address field in every frame, giving this style of bridging its name. The destination station reverses the routing field to reach the originating station.

Figure 23-1 IEEE 802.5 Token Ring Frame Format



The information in a RIF is derived from explorer packets generated by the source node. These explorer packets traverse the entire source-route bridge network, gathering information on the possible paths the source node might use to send packets to the destination.

Unlike transparent spanning-tree bridging, which requires time to recompute topology in the event of failures, source-route bridging allows multiple, active paths through the network, which provides for more timely switches to alternate routes in the event of failure. Most importantly, source-route bridging places the burden of transmitting frames with the end stations by allowing them to determine the routes the frames take.

Cisco's Implementation of Source-Route Bridging

Cisco's source-route bridging software implementation includes the following features:

- Provides configurable fast-switching software for source-route bridging.
- Provides for a local source-route bridge that connects two or more Token Ring networks.
- Provides *ring groups* to configure a source-route bridge with more than two network interfaces. A ring group is a collection of Token Ring interfaces in one or more routers that are collectively treated as a *virtual ring*.
- Provides two types of explorer packets to collect RIF information—an *all-routes* explorer packet, which follows all possible paths to a destination ring, and a *spanning-tree* explorer packet, which follows a statically configured limited route (spanning tree) when looking for paths.
- For RSRB, provides for multiple router/bridges separated by non-Token Ring segments. Three options are available:
  - Encapsulate the Token Ring traffic inside Internet Protocol (IP) datagrams passed over a Transmission Control Protocol (TCP) connection between two router/bridges.
  - Use Fast Sequenced Transport (FST) to transport remote source-route bridging packets to their peers without TCP or User Datagram Protocol (UDP) header or processor overhead.
  - Use MAC-layer encapsulations over a single serial line, Ethernet, Token Ring, or FDDI ring connected between two routers attached to Token Ring networks.
- Provides for configurable limits to the size of the TCP backup queue.
- Provides a dynamically determined RIF cache based on the protocol; also allows you to manually add entries to the RIF cache.
- Provides for filtering by MAC address, link service access point (LSAP) header, and protocol type.
- Provides for filtering of NetBIOS frames either by station name or by a packet byte offset.
- Provides for translation into transparently bridged frames to allow source-route stations to communicate with nonsource-route stations (typically on Ethernet).
- Provides support for the SRB Management Information Base (MIB) variables as described in the IETF draft "Bridge MIB" document, "Definition of Managed Objects for Bridges," by E. Decker, P. Langille, A. Rijsinghani, and K. McCloghrie, June 1991. Only the SRB component of the Bridge MIB is supported.
- Provides support for the Token Ring MIB variables as described in RFC 1231, "IEEE 802.5 Token Ring MIB," by K. McCloghrie, R. Fox, and E. Decker, May 1991. Cisco implements the mandatory tables (Interface Table and Statistics Table) but not the optional table (Timer Table)

of the Token Ring MIB. The Token Ring MIB has been implemented for the 4/16-Mb Token Ring cards that can be user adjusted for either 4- or 16-Mb transmission speeds (CSC-1R, CSC-2R, CSC-R16M, or CSC-C2CTR).

SRB Configuration Task List

You can perform the tasks in the following sections to configure source-route bridging:

- Configure Source-Route Bridging
- Configure Remote Source-Route Bridging
- Configure Bridging of Routed Protocols
- Configure Translation between SRB and Transparent Bridging Environments
- Configure NetBIOS Support
- Configure LAN Network Manager Support
- Secure the SRB Network
- Tune the SRB Network
- Establish SRB Interoperability with Specific Token Ring Implementations
- Monitor and Maintain the SRB Network

See the end of this chapter for “SRB Configuration Examples”.

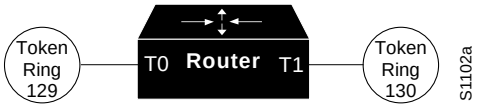
Configure Source-Route Bridging

Our implementation of source-route bridging enables you to connect two or more Token Ring networks using either Token Ring or Fiber Distributed Data Interface (FDDI) media.

As designed by IBM and the IEEE 802.5 committee, when a router is configured as a source-route bridge, bridged traffic does not pass across non-Token Ring media, and only those protocols that are not being routed are source-route bridged. For example, if IPX routing is enabled on the router that is configured for source-route bridging, IPX datagrams will not be source-route bridged. However, datagrams for other nonrouted protocols will be source-route bridged. Our implementation of source-route bridging extends this definition.

A dual-port bridge is the simplest possible source-route bridging configuration. When configured as a dual-port bridge, the router serves to connect two Token Ring LANs. One LAN is connected through one port (Token Ring interface), and the other LAN is connected through the other port (also a Token Ring interface). Figure 23-2 shows a dual-port bridge.

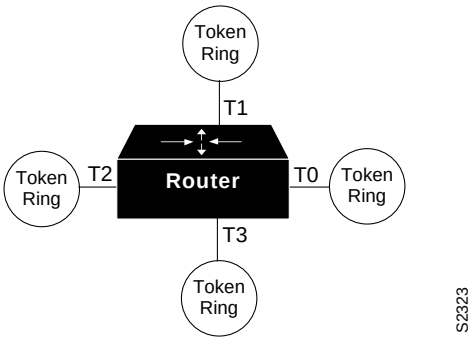
Figure 23-2 Dual-Port Bridge



A dual-port bridge is a limitation imposed by IBM Token Ring chips; they can only process two ring numbers. If you have a router with two or more Token Ring interfaces, you can work around the two-ring number limitation. You can configure your router as multiple dual-port bridges or as a multiport bridge using a virtual ring.

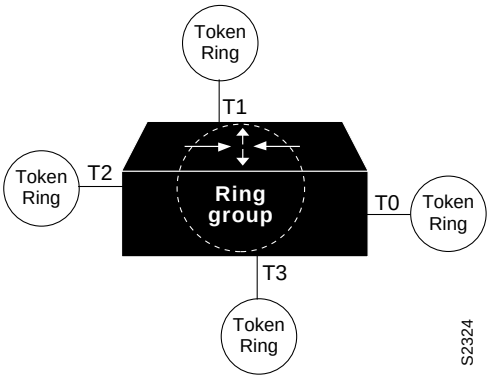
You can define several separate dual-port bridges in the same router. However, the devices on the LANs cannot have any-to-any connectivity; that is, they cannot connect to every other device on the bridged LANs. Only the devices connected to the dual-port bridge can communicate with one another. Figure 23-3 shows two separate dual-port bridges (T0-T2 and T1-T3) configured on the same router.

Figure 23-3 Multiple Dual-Port Bridges



A better solution for overcoming the two-ring number limitation of IBM Token Ring chips is to configure a multiport bridge using a virtual ring. A virtual ring on a multiport bridge allows the router to interconnect three or more LANs with any-to-any connectivity; that is, connectivity between any of the devices on each of the three LANs is allowed. A virtual ring creates a logical Token Ring internal to the router that causes all the Token Rings connected to the router to be treated as if they are all on the same Token Ring. The virtual ring is called a *ring group.* Figure 23-4 shows a multiport bridge using a virtual ring.

Figure 23-4 Multiport Bridge Using a Virtual Ring

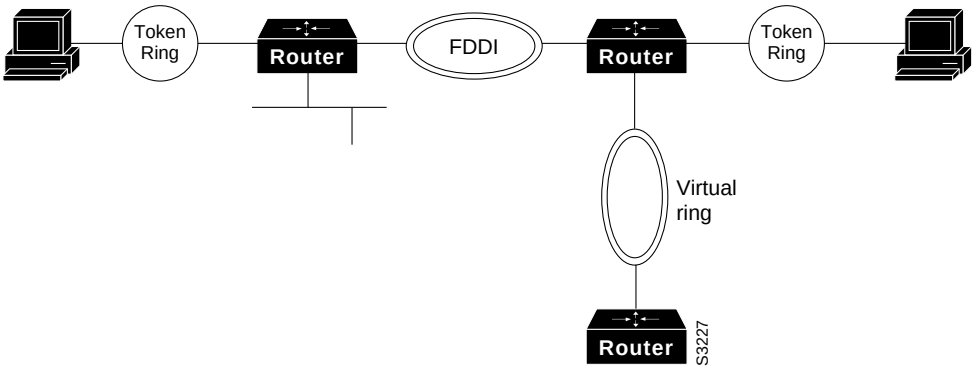


To take advantage of this virtual ring feature, each Token Ring interface on the router must be configured to belong to the same ring group. For information about configuring a multiport bridge using a virtual ring, see the “Configure a Multiport Bridge Using a Virtual Ring” section later in this chapter.

Our implementation of SRB expands the basic functionality to allow autonomous switching of SRB network traffic for FDDI interfaces, adding counters to SRB accounting statistics, and implementing process-level switching of SRB over FDDI. This functionality provides a significant increase in performance for Token Rings interconnected across an FDDI backbone.

Note Autonomous FDDI SRB is supported on Cisco 7000 series routers where autonomous switching is possible.

Figure 23-5 Autonomous FDDI SRB



You can configure the router for source-route bridging by performing the tasks in one of the first three sections and optionally, the tasks in the last section:

- Configure a Dual-Port Bridge
- Configure a Multiport Bridge Using a Virtual Ring
- Configure Autonomous FDDI SRB
- Enable the Forwarding and Blocking of Spanning-Tree Explorers
- Enable the Automatic Spanning-Tree Function

Configure a Dual-Port Bridge

A router equipped with Token Ring cards is by default a Token Ring host, and SRB is disabled by default. To configure a dual-port bridge that connects two Token Rings, you must enable source-route bridging on each of the Token Ring interfaces that connect to the two Token Rings. To enable source-route bridging, perform the following task in interface configuration mode for each of the Token Ring interfaces:

Task	Command
Enable local source-route-bridging on a Token Ring interface.	**source-bridge** *local-ring bridge-number target-ring*

For multiple dual-port source-route bridges, you would repeat this task for each Token Ring interface that is part of a dual-port bridge. If you wanted your network to use only source-route bridging, you could connect as many of these routers via Token Rings as you needed. Remember, to use source-route bridging requires you bridge only Token Ring media.

Note Ring numbers need to be unique across interfaces and networks, so that when you enable source-route bridging over an interface, the local and target rings are defined. Each node on the network will know if it is the target of explorer packets sent on the network.

Configure a Multiport Bridge Using a Virtual Ring

To configure a source-route bridge to have more than two network interfaces, you must perform the following tasks in the specified order:

- Define a *ring group*.
- Enable source-route-bridging and assign a ring group to a Token Ring interface.

Once you have completed these tasks, the router acts as a multiport bridge not as a dual-port bridge.

Note Ring numbers need to be unique across interfaces and networks.

Define a Ring Group in SRB Context

Because all IBM Token Ring chips can only process two ring numbers, we have implemented the concept of a ring group or virtual ring. A ring group is a collection of Token Ring interfaces in one or more routers that share the same ring number. This ring number is used just like a physical ring number, showing up in any route descriptors contained in packets being bridged. Within the context of a multiport bridge that uses source-route bridging rather than remote source-route bridging (RSRB), the ring group resides in the same router. See the "Configure Remote Source-Route Bridging" section to compare ring groups in the SRB and RSRB context.

A ring group must be assigned a ring number that is unique throughout the network. It is possible to assign different Token Ring interfaces on the same router to different ring groups, if, for example, you plan to administer them as interfaces in separate domains.

To define or remove a ring group, perform one of the following tasks in global configuration mode:

Task	Command
Define a ring group.	**source-bridge ring-group** *ring-group*
Remove a ring group.	**no source-bridge ring-group** *ring-group*

Enable SRB and Assign a Ring Group to an Interface

After you have defined a ring group, you must assign that ring group to those interfaces you plan to include in that ring group. An interface can only be assigned to one ring group. To enable any-to-any connectivity among the end stations connected through this multiport bridge, you must assign the same target ring number to all Token Ring interfaces on the router.

To enable SRB and assign a ring group to an interface, perform the following task in interface configuration mode:

Task	Command
Enable source-route-bridging and assign a ring group to a Token Ring interface.	**source-bridge** *local-ring bridge-number target-ring*

Configure Autonomous FDDI SRB

To configure autonomous FDDI SRB, perform the following tasks, beginning in global configuration mode:

Task	Command
Configure an FDDI interface.	**interface fddi** *slot/port*
Enable source-route bridging.	**source-bridge** *local-ring bridge-number target-ring*
Enable autonomous switching.	**source-bridge route-cache cbus**

Note The **multiring** command and the LAN Net Manager are not supported on FDDI.

Enable the Forwarding and Blocking of Spanning-Tree Explorers

When trying to determine the location of remote destinations on a source-route bridge, the source device will need to send explorer packets. Explorer packets are used to collect RIF information. The source device can send spanning-tree explorers or all-routes explorers. Note that some older IBM devices only generate all-routes explorer packets, but many newer IBM devices are capable of generating spanning-tree explorer packets.

A spanning-tree explorer packet is an explorer packet that is sent to a defined group of nodes that comprise a statically configured spanning tree in the network. In contrast, an all-routes explorer packet is an explorer packet that is sent to every node in the network on every path.

Forwarding all-routes explorer packets is the default. However, in complicated source-route bridging topologies, using this default can generate an exponentially large number of explorers that are traversing the network. The number of explorer packets becomes quite large because duplicate explorer packets are sent across the network to every node on every path. Eventually each explorer packet will reach the destination device. The destination device will respond to each of these explorer packets. It is from these responses that the source device will collect the RIF and determine which route it will use to communicate with the destination device. Usually, the route contained in the first returned response will be used.

The number of explorer packets traversing the network can be reduced by sending spanning-tree explorer packets. Spanning-tree explorer packets are sent to specific nodes; that is, to only the nodes on the spanning tree, not to all nodes in the network. You must manually configure the spanning-tree topology over which the spanning-tree explorers are sent. You do this by configuring which interfaces on the routers will forward spanning-tree explorers and which interfaces will block them.

To enable forwarding of spanning-tree explorers on an outgoing interface, perform the following task in interface configuration mode:

Task	Command
Enable the forwarding of spanning-tree explorer packets on an interface.	**source-bridge spanning**

Note While enabling the forwarding of spanning-tree explorer packets is not an absolute requirement, it is strongly recommended in complex topologies. Configuring an interface to block or forward spanning-tree explorers has no effect on how that interface handles all-routes explorer packets. All-routes explorers can always traverse the network.

To block forwarding of spanning tree explorers on an outgoing interface, perform the following task in interface configuration mode:

Task	Command
Block spanning-tree explorer packets on an interface.	**no source-bridge spanning**

Enable the Automatic Spanning-Tree Function

The automatic spanning tree function supports automatic resolution of spanning trees in SRB networks, which provides a single path for spanning explorer frames to traverse from a given node in the network to another. Spanning explorer frames have a single-route broadcast indicator set in the routing information field. Port identifiers consist of ring numbers and bridge numbers associated with the ports. The spanning tree algorithm for SRB does not support Topology Change Notification BDPU.

Note Although the automatic spanning tree function can be configured with SR/TLB, the SRB domain and TB domain have separate spanning trees. Each Token Ring interface can belong to only one spanning tree. Only one bridge group can run the automatic spanning tree function in a router at a time.

To create a bridge group that runs an automatic spanning-tree function compatible with the IBM SRB spanning-tree implementation, perform the following task in global configuration mode:

Task	Command
Create a bridge group that runs the automatic spanning-tree function.	**bridge** *bridge-group* **protocol ibm**

To enable the automatic spanning-tree function for a specified group of bridged interfaces, perform the following task in interface configuration mode:

Task	Command
Enable the automatic spanning-tree function on a group of bridged interfaces.	**source-bridge spanning** *bridge-group*

To assign a path cost for a specified interface, perform the following task in interface configuration mode:

Task	Command
Assign a path cost for a specified group of bridged interfaces.	**source-bridge spanning** *bridge-group* **path-cost** *path-cost*

Note Ports running IEEE and IBM protocols will form a spanning tree together on the LAN, but they will not mix in the router itself. Make sure the configurations are correct and that each LAN runs only one protocol.

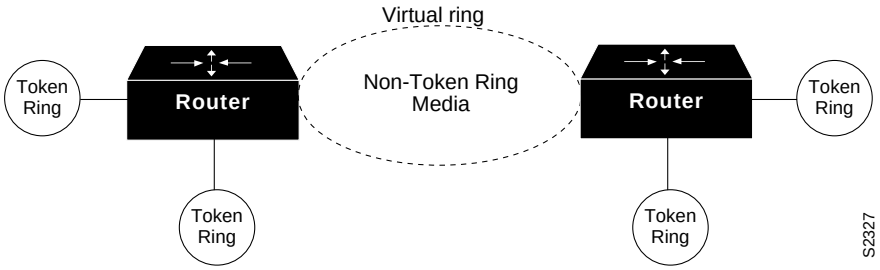
See the end of this chapter for an example of source-route bridging with the automatic spanning-tree function enabled.

Configure Remote Source-Route Bridging

While source-route bridging involves bridging between Token Ring media only, remote source-route bridging (RSRB) involves multiple router/bridges separated by non-Token Ring network segments. Figure 23-6 shows an RSRB topology. The virtual ring can extend across any non-Token Ring media supported by the RSRB such as serial, Ethernet, FDDI, and WANs. The type of media you select will determine the way you set up RSRB.

Note If you will be bridging across Token Ring media, it is recommended that you do not use RSRB. Use SRB instead.

Figure 23-6 Remote Source-Route Bridged Topology



To set up RSRB, perform the tasks in one of the following sections:

- Configure RSRB Using Direct Encapsulation
- Configure RSRB Using IP Encapsulation over an FST Connection
- Configure RSRB Using IP Encapsulation over a TCP Connection
- Configure RSRB Using IP Encapsulation over a Fast-Switched TCP Connection
- Configure RSRB Using TCP and LLC2 Local Acknowledgment
- Configure Direct Frame Relay Encapsulation between RSRB Peers

After you configure RSRB, you can establish SAP prioritization by performing the tasks described in the “Establish SAP Prioritization” section later in this chapter.

Note Only use IP encapsulation over a TCP connection within complex meshed networks to support connections between peers that are separated by multiple hops and have the potential of using multiple paths and where performance is not an issue. Use direct encapsulation in point-to-point connections. In such point-to-point configuration, using TCP would add too much unnecessary processing overhead.

Configure RSRB Using Direct Encapsulation

Configuring RSRB using the direct encapsulation method uses an HDLC-like encapsulation to pass frames over a single physical network connection between two routers attached to Token Rings. Use this method when you are running source-route bridge traffic over point-to-point, single-hop, serial or LAN media. Although this method does not have the flexibility of the TCP approach, it provides the best performance of the three methods because it involves less overhead.To configure a remote source-route bridge to use a point-to-point serial line or a single Ethernet, or a single FDDI hop, you must perform the following tasks:

- Define a ring group in RSRB context.
- Identify the remote peers.
- Enable source-route bridging on the appropriate interfaces.

Define a Ring Group in RSRB Context

In our implementation of RSRB, whenever you connect Token Rings using non-Token Ring media, you must treat that non-Token Ring media as a virtual ring by assigning it to a ring group. Every router/bridge with which you wish to exchange Token Ring traffic must be a member of this same ring group. These other router/bridges are referred to as remote peer bridges. The ring group is therefore made up of interfaces that reside on separate routers.

A ring group must be assigned a ring number that is unique throughout the network. It is possible to assign different interfaces on the same router to different ring groups, if, for example, you plan to administer them as interfaces in separate domains.

To define or remove a ring group, perform one of the following tasks in global configuration mode:

Task	Command
Define a ring group.	**source-bridge ring-group** *ring-group*
Remove a ring group.	**no source-bridge ring-group** *ring-group*

Identify the Remote Peers (Direct Encapsulation)

The interfaces that you identify as remote peer bridges must be serial, Ethernet, FDDI, or Token Ring interfaces. On a serial interface, you must use HDLC encapsulation. To identify remote-peer bridges, perform the following task in global configuration mode:

Task	Command
Define the ring group and identify the interface over which to send SRB traffic to another router/bridge in the ring group.	**source-bridge remote-peer** *ring-group* **interface** *interface-name* [*mac-address*] [**lf** *size*] [**version** *number*]

You must specify one **source bridge remote peer** command for each peer router that is part of the virtual ring. You must also specify one **source bridge remote peer** command to identify the IP address of the local router. If you specify a MAC address, be sure that it is the MAC address on the remote interface that is directly connected to the system that is being configured. It should not be the MAC address of the Token Ring interface on the remote peer.

You can assign a keepalive interval to the remote source-bridging peer. Perform the following task in interface configuration mode:

Task	Command
Define the keepalive interval of the remote source-bridging peer.	**source-bridge keepalive** *seconds*

Enable SRB on the Appropriate Interfaces

You must enable source-route bridging on each of the interfaces through which source-route bridging traffic will pass. The value you specify in the target ring parameter should be the ring group number you have assigned to the interface. To enable SRB on an interface, perform the following task in interface configuration mode:

Task	Command
Enable source-route bridging on an interface.	**source-bridge** *local-ring bridge-number target-ring*

Configure RSRB Using IP Encapsulation over an FST Connection

Encapsulating the source-route bridged traffic inside IP datagrams passed over a Fast Sequenced Transport (FST) connection between two router/bridges is not as fast as direct encapsulation, but it outperforms IP encapsulation over a TCP connection because it has lower overhead. However, this method does not allow for local acknowledgment, nor is it suitable for use in networks that tend to reorder frame sequences. To configure a remote source-route bridge to use IP encapsulation over an FST connection, you must perform the following tasks:

- Set up an FST peer name and assign an IP address.
- Identify the remote peers.
- Enable SRB on the appropriate interfaces.

Note FST encapsulation preserves the dynamic media-independent nature of IP routing to support SNA and NetBIOS applications.

For an example of how to configure RSRB over an FST connection, see the "RSRB Using IP Encapsulation over a TCP Connection Example" section later in this chapter.

Set Up an FST Peer Name and Assign an IP Address

To set up an FST peer name and provide an IP address to the local router, perform the following task in global configuration mode:

Task	Command
Set up an FST peer name and provide the local router with an IP address.	**source-bridge fst-peername** *local-interface-address*

In our implementation of RSRB, whenever you connect Token Rings using non-Token Ring media, you must treat that non-Token Ring media as a virtual ring by assigning it to a ring group. Every router/bridge with which you wish to exchange Token Ring traffic must be a member of this same

ring group. Therefore, after you set up an FST peer name, define a ring group. For more information about defining a ring group, see the "Define a Ring Group in SRB Context" section earlier in this chapter.

Identify the Remote Peers (FST Connection)

All of the router/bridges with which you want to exchange Token Ring traffic are referred to as remote peer bridges. The remote peers can be at the other end of an FST connection. To identify the remote peers, perform the following task in global configuration mode:

Task	Command
Identify your peers and specify an FST connection.	**source-bridge remote-peer** *ring-group* **fst** *ip-address* [**lf** *size*]

You must specify one **source bridge remote peer** command for each peer router that is part of the virtual ring. You must also specify one **source bridge remote peer** command to identify the IP address of the local router. The IP address you specify should be the IP address the router tries to reach.

You can assign a keepalive interval to the RSRB peer. Perform the following task in interface configuration mode:

Task	Command
Define the keepalive interval of the RSRB peer.	**source-bridge keepalive** *seconds*

Enable SRB on the Appropriate Interfaces

You must enable SRB on each of the interfaces through which SRB traffic will pass. The value of the target ring parameter you specify should be the ring group number you have assigned to the interface. To enable SRB on an interface, perform the following task in interface configuration mode:

Task	Command
Enable local SRB on a Token Ring interface.	**source-bridge** *local-ring bridge-number target-ring*

Performance Considerations When Using FST in a Redundant Network Topology

FST is fast switched when it receives or sends frames from Ethernet, Token Ring or FDDI interfaces. It is also fast switched when sending and receiving from serial interfaces configured with the HDLC encapsulation. In all other cases, FST is slow switched.

In cases where FST is fast switched, in either the routers configured for FST or in the routers contained within the IP "cloud" between a pair of FST peers, only one path will be used at a given time between the two FST peers. This provides an extremely high likelihood that frames will not arrive at one peer in a different sequence than are sent from a remote peer. In the very rare cases where this can happen, the FST code on the receiving peer will discard the out-of-order frame. As such, the Token Ring end hosts will rarely lose a frame over the FST router cloud, and performance levels will remain adequate.

The same conditions hold true for any slow-switched topology that provides only a single path between the peers. For example, a single X.25 network cloud would fall under this category. Similarly, if two slow-switched paths are of very different costs such that one always will be chosen over the other, the chances of having frames received out of sequence are also rare.

However, if two or more slow-switched paths of equal cost exist between the two routers (such as two parallel X.25 networks), the routers alternate in sending packets between the two or more equal-cost paths. This results in a high probability of frames arriving out of sequence at the receiver. In such cases, the FST code will dispose of every out-of-sequence packet, leading to a large number of drops at the router. This requires that the end hosts retransmit frames, greatly reducing overall throughput.

When such parallel paths exist, it is strongly recommended that one be chosen over the other as the preferred path. This can be done by specifying a higher bandwidth for the path that contains the direct connections to the two or more parallel paths on the router.

Do not use FST when the probability routinely exists for frames to lose their ordering in your network. If you have a network where frames are routinely reordered, it is far better to use the TCP protocol for remote source-route bridging, because it provides the overhead necessary to bring frames back in order on the receiving router. FST, to remain fast, does not provide for such a mechanism, and will toss out-of-order frames.

Configure RSRB Using IP Encapsulation over a TCP Connection

Encapsulating the source-route bridged traffic inside IP datagrams passed over a TCP connection between two router/bridges offers lower performance than the other two methods, but is the appropriate method to use under the following conditions:

- You plan to connect Token Ring networks across arbitrary media including Ethernet, FDDI, serial interfaces, X.25 networks, and so forth.
- You plan to connect Token Ring networks across a multiprotocol backbone network.
- You plan to load balance over multiple, redundant paths. Using this topology, when a path fails there is no need for hosts to retransmit explorer packets. IP routing handles the network reconfiguration transparently to the Token Ring hosts.

To configure a remote source-route bridge to use IP encapsulation over a TCP connection, you must perform the following tasks:

- Identify the remote peers.
- Enable SRB on the appropriate interfaces.

Identify the Remote Peer (TCP Connection)

In our implementation, whenever you connect Token Rings using non-Token Ring media, you must treat that non-Token Ring media as a virtual ring by assigning it to a ring group. Every router/bridge with which you wish to exchange Token Ring traffic must be a member of this same ring group. For more information about defining a ring group, see the “Define a Ring Group in SRB Context” section earlier in this chapter.

To identify the remote peers, perform the following task in global configuration mode:

Task	Command
Identify the IP address of a peer in the ring group with which to exchange source-bridge traffic using TCP.	**source-bridge remote-peer** *ring-group* **tcp** *ip-address* [**lf** *size*] [**tcp-receive-window** *wsize*] [**local-ack**] [**priority**]

You must specify one **source bridge remote peer** command for each peer router that is part of the virtual ring. You must also specify one **source bridge remote peer** command to identify the IP address of the local router.

You can assign a keepalive interval to the remote source-bridging peer. Perform the following task in interface configuration mode:

Task	Command
Define the keepalive interval of the remote source-bridging peer.	**source-bridge keepalive** *seconds*

Enable SRB on the Appropriate Interfaces

To enable SRB on an interface, perform the following task in interface configuration mode:

Task	Command
Enable local source-route-bridging on a Token Ring interface.	**source-bridge** *local-ring bridge-number target-ring*

The value of the target ring parameter you specify should be the ring group number.

Configure RSRB Using IP Encapsulation over a Fast-Switched TCP Connection

The fast-switched TCP (FTCP) encapsulation type speeds up Token Ring-to-Token Ring RSRB over TCP by fast-switching Token Ring frames to and from the TCP pipe. FTCP encapsulation supports the same options as TCP, with the exception of priority queueing.

In our implementation, whenever you connect Token Rings using non-Token Ring media, you must treat that non-Token Ring media as a virtual ring by assigning it to a ring group. Every router/bridge with which you wish to exchange Token Ring traffic must be a member of this same ring group. For more information about defining a ring group, see the "Define a Ring Group in SRB Context" section earlier in this chapter.

To configure RSRB fast switching, you must perform the tasks in the following sections:

- Identify the Remote Peer (TCP Connection)
- Enable SRB on the Appropriate Interfaces

Identify the Remote Peer (TCP Connection)

You must identify the remote peer with which the router will communicate. To identify the remote peers, perform the following task in global configuration mode:

Task	Command
Identify the IP address of a peer in the ring group with which to exchange source-bridge traffic using TCP.	**source-bridge remote-peer** *ring-group* **tcp** *ip-address* [**lf** *size*] [**tcp-receive-window** *wsize*] [**local-ack**] [**priority**]

You must identify a remote peer for each interface you configure for remote source-route bridging. The IP address you specify is the IP address the router tries to reach.

You can assign a keepalive interval to the remote peer. Perform the following task in interface configuration mode:

Task	Command
Define the keepalive interval of the remote peer.	**source-bridge keepalive** *seconds*

Enable SRB on the Appropriate Interfaces

To enable SRB on an interface, perform the following task in interface configuration mode:

Task	Command
Enable local SRB on a Token Ring interface.	**source-bridge** *local-ring bridge-number target-ring*

The value of the target ring parameter you specify should be the ring group number.

Configure RSRB Using TCP and LLC2 Local Acknowledgment

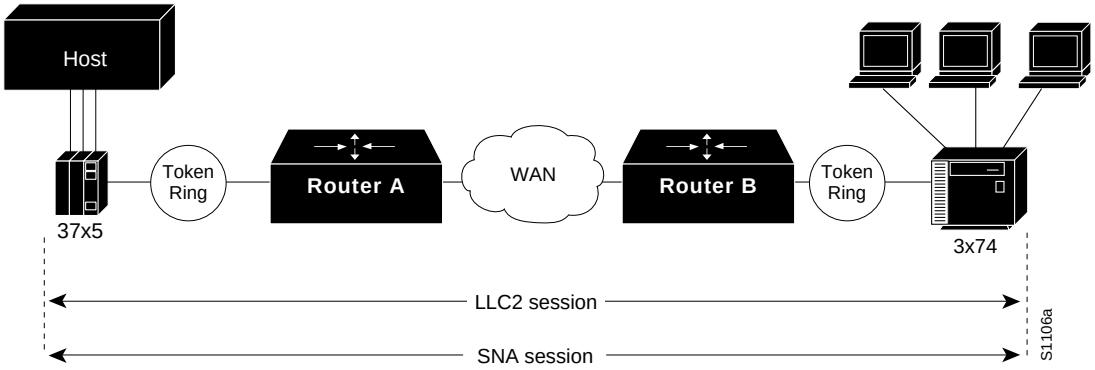
Encapsulating the source-route bridged traffic inside IP datagrams passed over a TCP connection between two router/bridges with local acknowledgment enabled is appropriate when you have LANs separated by wide geographic distances and you want to avoid time delays, multiple retransmissions, or loss of user sessions.

Logical Link Control–Type 2 (LLC2) is an ISO standard data link level protocol used in Token Ring networks. LLC2 was designed to ensure reliable transmission of data across LAN media with minimal or predictable time delays. With the advent of RSRB and wide area network (WAN) backbones, LANs are now separated by wide, geographic distances spanning countries and continents. As a result, these LANs have time delays that are longer than LLC2 allows for bidirectional communication between hosts. The local acknowledgment capability in router/bridges supporting remote source-route bridging addresses the problem of unpredictable time delays, multiple retransmissions, and loss of user sessions.

In a typical LLC2 session, when host A sends a frame to host B, the sending host A expects host B to respond positively or negatively in a certain amount of predefined time commonly called the *T1 time*. If host A does not receive an acknowledgment of the frame it sent to host B within the T1 time, it will retry a few number of times (normally 8 to 10). If there is still no response from host B, host A will drop the session.

Figure 23-7 illustrates an LLC2 session. A 37x5 on a LAN segment can communicate with a 3x74 on a different LAN segment separated via a wide-area backbone network. Frames are transported between Router A and Router B using RSRB. However, the LLC2 session between the 37x5 and the 3x74 is still end-to-end; that is, every frame generated by the 37x5 traverses the backbone network to the 3x74, and the 3x74, on receipt of the frame, acknowledges it.

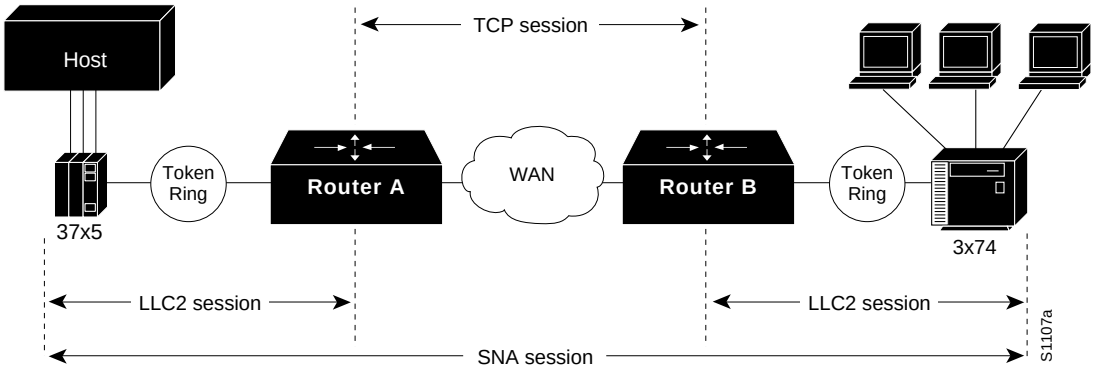
Figure 23-7 LLC2 Session without Local Acknowledgment



On backbone networks consisting of slow serial links, the T1 timer on end hosts could expire before the frames have a chance to reach the remote hosts, causing the end host to retransmit. This results in duplicate frames reaching the remote host at the same time as the first frame also reached the remote host, albeit slowly. These frame duplications break the LLC2 protocol, resulting in the loss of sessions between the two IBM machines.

One way to solve this time delay problem is to increase the timeout value on the end nodes to account for the maximum transit time between the two end machines. However, in networks consisting of hundreds or even thousands of nodes, every machine would need to be reconfigured with new values. With local acknowledgment for LLC2 turned on, the LLC2 session between the two end nodes would not be end-to-end, but instead, terminates at two local routers. Figure 23-8 shows the LLC2 session with the 37x5 ending at Router A and the LLC2 session with the 3x74 ending at Router B. Both Router A and Router B execute the full LLC2 protocol as part of local acknowledgment for LLC2.

Figure 23-8 LLC2 Session with Local Acknowledgment



With local acknowledgment for LLC2 enabled in both routers, Router A acknowledges frames received from the 37x5. The 37x5 still thinks that the acknowledgments it receives are from the 3x74. Router A looks like the 3x74 to the 37x5. Similarly, Router B acknowledges frames received from the 3x74. The 3x74 thinks that the acknowledgments it receives are from the 37x5. Router B looks like the 3x74 to 37x5. Because the frames no longer have to travel the WAN backbone networks to be acknowledged, but instead are locally acknowledged by routers, the end machines do not time out, resulting in no loss of sessions.

The advantages of enabling local acknowledgment for LLC2 include the following:

- Local acknowledgment for LLC2 solves the T1 timer problem without having to change any configuration on the end nodes. The end nodes are unaware that the sessions are being locally acknowledged. In networks consisting of hundreds or even thousands of machines, this is a definite advantage. All the frames acknowledged by the router appear to the end hosts to be coming from the remote IBM machine. In fact, by looking at a trace from a protocol analyzer, one cannot say whether a frame was acknowledged by the local router or by a remote IBM machine. The MAC addresses and the RIFs generated by the routers are identical to those generated by the remote IBM machine. The only way to find out whether a session is locally acknowledged is to use either a **show local-ack** command or a **show source-bridge** command on the router.
- All the supervisory (RR, RNR, REJ) frames that are locally acknowledged go no farther than the router. Without local acknowledgment for LLC2, *every* frame traverses the backbone. With local acknowledgment, only data (I-frames) traverse the backbone, resulting in less traffic on the backbone network. For installations in which customers pay for the amount of traffic passing through the backbone, this could be a definite cost-saving measure. A simple protocol exists between the two *peers* to bring up or down a TCP session.

To configure a remote source-route bridge to use IP encapsulation over a TCP connection, you must perform the following tasks:

- Enable LLC2 local acknowledgment between two remote peers.
- Enable SRB on the appropriate interfaces.

Enable LLC2 Local Acknowledgment between Two Remote Peer Bridges

In our implementation, whenever you connect Token Rings using non-Token Ring media, you must treat that non-Token Ring media as a virtual ring by assigning it to a ring group. Every router/bridge with which you wish to exchange Token Ring traffic must be a member of this same ring group. For more information about defining a ring group, see the "Define a Ring Group in SRB Context" section earlier in this chapter.

To enable LLC2 local acknowledgment, perform the following task in global configuration mode:

Task	Command
Enable LLC2 local acknowledgment on a per-remote-peer basis.	**source-bridge remote-peer** *ring-group* **tcp** *ip-address* **local-ack**

You must use one instance of the **source bridge remote peer** command for each interface you configure for RSRB.

Enable SRB on the Appropriate Interfaces

To enable SRB on an interface, perform the following task in interface configuration mode:

Task	Command
Enable local SRB on a Token Ring interface.	**source-bridge** *local-ring bridge-number target-ring*

The value of the target ring parameter you specify should be the ring group number.

For an example of how to configure RSRB with local acknowledgment, see the "Example of RSRB with Local Acknowledgment" section later in this chapter.

Enable Local Acknowledgment and Passthrough

To configure some sessions on a few rings to be locally acknowledged while the remaining sessions are passed through, perform the following task in global configuration mode:

Task	Command
Configure a router for passthrough.	**source-bridge passthrough** *ring-group*

Notes on Using LLC2 Local Acknowledgment

LLC2 local acknowledgment can only be enabled with TCP remote peers (as opposed to LAN or direct serial interface remote peers) because the routers need the reliable transmission of TCP to provide the same reliability that an LLC2 LAN end-to-end connection provides. Therefore, the direct media encapsulation options for the **source-bridge remote-peer** command cannot be used.

If the LLC2 session between the local host and the router terminates in either router, the other will be informed to terminate its connection to its local host.

If the TCP queue length of the connection between the two routers reaches 90 percent of its limit, the routers will send Receiver-not-Ready (RNR) messages to the local hosts until the queue limit is reduced to below this limit.

The configuration of the LLC2 parameters for the local Token Ring interfaces can affect overall performance. Refer to the chapter "Configuring LLC2 and SDLC Parameters" in this manual for more details about fine-tuning your network through the LLC2 parameters.

Note As previously stated, local acknowledgment for LLC2 is meant only for extreme cases in which communication is not possible otherwise. Because the router must maintain a full LLC2 session, the number of simultaneous sessions it can support before performance degrades depends on the mix of other protocols and their loads also running in it.

The routers at each end of the LLC2 session execute the full LLC2 protocol, which could result in some overhead. The decision to turn on local acknowledgment for LLC2 should be based on the speed of the backbone network in relation to the Token Ring speed. For LAN segments separated by slow-speed serial links (for example, 56 kbps), the T1 timer problem could occur more frequently. In such cases, it may be wise to turn on local acknowledgment for LLC2. For LAN segments separated by a FDDI backbone, backbone delays will be minimal; in such cases, local acknowledgment for LLC2 should not be turned on. Speed mismatch between the LAN segments and the backbone network is one criterion to be used in the decision to use local acknowledgment for LLC2.

There are some situations (such as host B dying between the time host A sends data and the time host B receives it), when host A would believe, *at the LLC2 layer*, that data was received that actually was not, because the router acknowledges that it received data from host A before it knows that host B can actually receive the data. But because both NetBIOS and SNA have error recovery in situations where an end device goes down, these higher-level protocols will resend any missing or lost data. These transaction request/confirmation protocols exist above LLC2, so they are not affected by tight timers, as is LLC2. They also are transparent to local acknowledgment.

If you are using NetBIOS applications, note that there are two NetBIOS timers—one at the link level and one at the next higher level. Local acknowledgment for LLC2 is designed to solve session timeouts at the link level only. If you are experiencing NetBIOS session timeouts, you have two options:

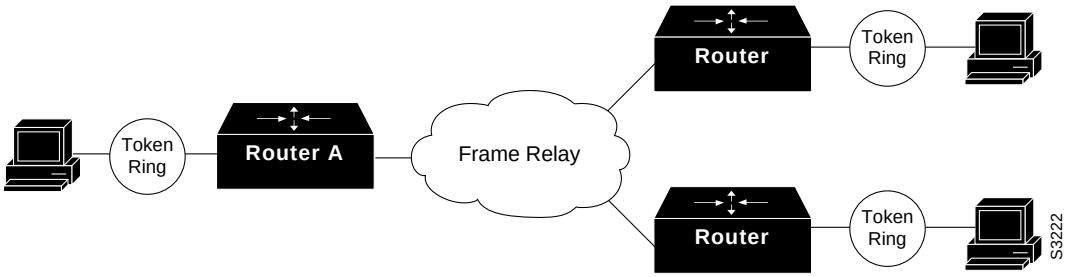
- Experiment with increasing your NetBIOS timers.
- Avoid using NetBIOS applications on slow serial lines.

Configure Direct Frame Relay Encapsulation between RSRB Peers

You can configure direct Frame Relay encapsulation to allow the RSRB peers to send RSRB protocol packets on a Frame Relay PVC. This eliminates the overhead introduced by Transmission Control Protocol/Internet Protocol (TCP/IP)-encapsulated Frame Relay packets as in the current implementation.

Figure 23-9 illustrates direct Frame Relay encapsulation between RSRB peers.

Figure 23-9 RSRB Direct Frame Relay Encapsulation



The RSRB direct encapsulation design can use RFC 1490 format or Cisco Frame Relay encapsulation for routed packets.

To configure RSRB direct Frame Relay encapsulation, perform the following tasks, starting in interface configuration mode:

Task	Command
Specify the serial interface on which Frame Relay is configured.	**source-bridge remote-peer** *bridge-group* **interface serial** [*port* **dlci** *dlci*]
Specify the DLCI number onto which the RSRB traffic is to be mapped.	**frame-relay map rsrb** *dlci*

Establish SAP Prioritization

The SAP prioritization feature allows you to use SAP priority lists and filters to specify the priority of one protocol over another across an RSRB or SDLLC WAN.

Define a SAP Priority List

To establish a SAP priority list, perform the following tasks:

Task	Command
Define the priority list.	**sap-priority-list** *number queue-keyword* [**dsap** *ds*] [**ssap** *ss*] [**dmac** *dm*] [**smac** *sm*]
Define the priority on an interface.	**sap-priority** *number*

Task	Command
Apply the priority list to an interface.	**priority-group** *list*

Define SAP Filters

You can define SAP filters and NetBIOS filters on Token Ring and Ethernet interfaces.

To filter by LSAP address on the RSRB WAN interface, perform the following global configuration tasks as appropriate:

Task	Command
Filter by LSAP address (TCP encapsulation).	**rsrb remote-peer** *ring-group* **tcp** *ip-address* **lsap-output-list** *access-list-number*
Filter by LSAP address (FST encapsulation).	**rsrb remote-peer** *ring-group* **fst** *ip-address* **lsap-output-list** *access-list-number*
Filter by LSAP address (direct encapsulation).	**rsrb remote-peer** *ring-group* **interface** *interface-name* **lsap-output-list** *access-list-number*

To filter packets by NetBIOS station name on an RSRB WAN interface, perform one of the following global configuration tasks as appropriate:

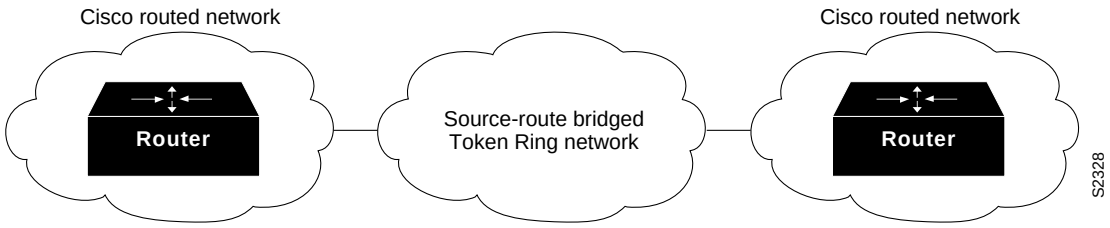
Task	Command
Filter by NetBIOS station name (TCP encapsulation).	**rsrb remote-peer** *ring-group* **tcp** *ip-address* **netbios-output-list** *name*
Filter by NetBIOS station name (FST encapsulation).	**rsrb remote-peer** *ring-group* **fst** *ip-address* **lsap-output-list** *name*
Filter by NetBIOS station name (direct encapsulation).	**rsrb remote-peer** *ring-group* **interface** *ip-address* **lsap-output-list** *host*

Configure Bridging of Routed Protocols

Source-route bridges use MAC information, specifically the information contained in the routing information field (RIF), to bridge packets. A RIF contains a series of ring and bridge numbers that represent the possible paths the source node might use to send packets to the destination. Each ring number in the RIF represents a single Token Ring in the source-route bridged network and is designated by a unique 12-bit ring number. Each bridge number represents a bridge that is between two Token Rings in the SRB network and is designated by a unique 4-bit bridge number. The information in a RIF is derived from explorer packets traversing the source-route bridged network. Without the RIF information, a packet could not be bridged across a source-route bridged network. For more information about RIFs and their format, refer to the *Internetworking Technology Overview* publication.

Unlike source-route bridges, Level 3 routers use protocol-specific information (for example Novell IPX or XNS headers) rather than MAC information to route datagrams. As a result, the router software default for routed protocols is to not collect RIF information and to not be able to bridge routed protocols. However, if you want the router to bridge routed protocols across a source-route bridged network, the router must be able to collect and use RIF information to bridge packets across a source-route bridged network. You can configure the router to append RIF information to routed protocols so that routed protocols can be bridged. Figure 23-10 shows a network topology in which you would want to use this feature.

Figure 23-10 Topology for Bridging Routed Protocols across a Source-Route Bridged Network



To configure the router to bridge routed protocols, you must perform the task in the first section, and optionally, one or both of the tasks in the other sections as follows:

- Enable Use of the RIF
- Configure a Static RIF Entry
- Configure the RIF Timeout Interval

Enable Use of the RIF

You can configure the router so that it will append RIF information to the routed protocols. This allows routed protocols to be bridged across a source-route bridged network. The routed protocols that you can bridge are as follows:

- Apollo Domain
- AppleTalk
- ISO CLNS
- DECnet
- IP
- IPX
- VINES
- XNS

Enable use of the RIF only on Token Ring interfaces on the router.

To configure the router to append RIF information, perform the following task in interface configuration mode:

Task	Command
Enable collection and use of RIF information.	**multiring** {*protocol-keyword* | **all** | **other**}

For an example of how to configure the router to bridge routed protocols, see the "SRB and Routing Certain Protocols Example" section later in this chapter.

Configure a Static RIF Entry

If a Token Ring host does not support the use of IEEE 802.2 TEST or XID datagrams as explorer packets, you might need to add static information to the RIF cache of the router/bridge.

To configure a static RIF entry, perform the following task in global configuration mode:

Task	Command
Enter static source-route information into the RIF cache.	**rif** *mac-address rif-string* {*interface-name* | **ring-group** *ring*}

Configure the RIF Timeout Interval

RIF information that can be used to bridge routed protocols is maintained in a cache whose entries are aged.

To configure the number of minutes an inactive RIF entry is kept in the cache, perform the following task in global configuration mode:

Task	Command
Specify the number of minutes an inactive RIF entry is kept.	**rif timeout** *minutes*

Configure Translation between SRB and Transparent Bridging Environments

Source-route translational bridging (SR/TLB) is a router software feature that allows you to combine SRB and transparent bridging networks without the need to convert all of your existing source-route bridges to source-route transparent (SRT) nodes. As such, it provides a cost-effective connectivity path between Ethernets and Token Rings, for example.

Note When you are translationally bridging, you will have to route routed protocols and translationally bridge all others, such as LAT.

Overview of SR/TLB

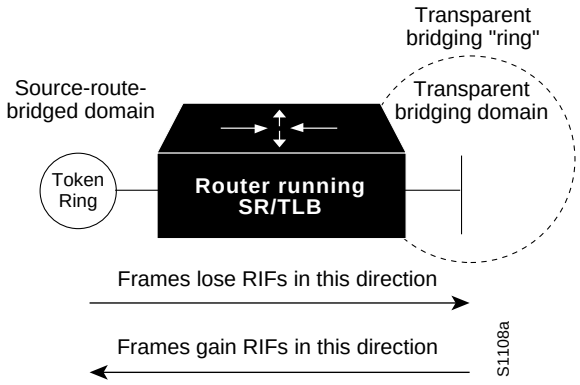
You can bridge packets between an SRB domain and a transparent bridging domain. Using this feature, a software "bridge" is created between a specified virtual ring group and a transparent bridge group. To the source-route station, this bridge looks like a standard source-route bridge. There is a ring number and a bridge number associated with a ring that actually represents the entire transparent bridging domain. To the transparent bridging station, the bridge represents just another port in the bridge group.

When bridging from the SRB (typically, Token Ring) domain to the transparent bridging (typically, Ethernet) domain, the source-route fields of the frames are removed. The RIFs are cached for use by subsequent return traffic.

When bridging from the transparent bridging domain to the SRB domain, the router/bridge checks the packet to see if it has a multicast or broadcast destination or a unicast (single host) destination. If it is multicast, the packet is sent as a spanning-tree explorer. If it is a unicast destination, the router/bridge looks up the path to the destination in the RIF cache. If a path is found, it will be used; otherwise, the router/bridge will send the packet as a spanning-tree explorer.

An example of a simple topology is shown in Figure 23-11.

Figure 23-11 Example of a Simple SR/TLB Topology



Note The spanning-tree protocol messages used to prevent loops in the transparent bridging domain are *not* passed between the SRB domain and the transparent bridging domain. Therefore, you must not set up multiple paths between the SRB and transparent bridging domains.

The following notes and caveats apply to all uses of SR/TLB:

- Multiple paths cannot exist between the source-route bridged domain and the transparent bridged domain. Such paths can lead to data loops in the network, because the spanning-tree packets used to avoid these loops in transparent bridging networks do not traverse the SRB network.
- Some devices, notably PS/2s under certain configurations running OS/2 Extended Edition Version 1.3, do not correctly implement the “largest frame” processing on RIFs received from remote source-route bridged hosts. The maximum Ethernet frame size is smaller than that allowed for Token Ring. As such, bridges allowing for communication between Ethernet and Token Ring will tell the Token Ring hosts, through the RIF on frames destined to the Token Ring, that hosts on the Ethernet cannot receive frames larger than a specified maximum, typically 1472 bytes. Some machines ignore this run-time limit specification and send frames larger than the Ethernet can accept. The router and any other Token Ring/Ethernet bridge has no choice but to drop these frames. To allow such hosts to successfully communicate across or to an Ethernet, you must configure their maximum frame sizes manually. For the PS/2, this can be done through Communications Manager.
- Any access filters applied on any frames apply to the frames as they appear on the media to which the interface with the access filter applies. This is important because in the most common use of SR/TLB (Ethernet and Token Ring connectivity), the bit ordering of the MAC addresses in the frame is swapped. Refer to the SR/TLB examples in the “SRB Configuration Examples” section of this chapter.

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

Ethernets or between Token Ring and FDDI LANs.

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

To enable SR/TLB, you must perform the task in the following section:

- Enable Bridging between Transparent Bridging and SRB

In addition, you can also perform the tasks in the following sections:

- Enable Translation Compatibility with IBM 8209 Bridges
- Enable Token Ring LLC2-to-Ethernet Conversion

Enable Bridging between Transparent Bridging and SRB

Before enabling bridging, you must have completely configured your router using multiport SRB and transparent bridging. Once you have done this, establish bridging between transparent bridging and source-route bridging by performing the following task in global configuration mode:

Task	Command
Enable bridging between transparent bridging and SRB.	**source-bridge transparent** *ring-group pseudo-ring bridge-num tb-group* [*oui*]

Enable Translation Compatibility with IBM 8209 Bridges

To transfer data between IBM 8209 Ethernet/Token Ring bridges and routers running the SR/TLB software (to create a Token Ring backbone to connect Ethernets), perform the following task on each Token Ring interface in interface configuration mode:

Task	Command
Move data between IBM 8209 Ethernet/Token Ring bridges and routers running translational bridging software.	**ethernet-transit-oui standard**

Enable Token Ring LLC2-to-Ethernet Conversion

The routers support the following types of Token Ring to Ethernet frame conversions:

- Token Ring LLC2 to Ethernet Type II (0x80d5 processing)
- Token Ring LLC2 to Ethernet 802.3 LLC2 (standard)

For most non-IBM hosts, Token Ring LLC2 frames can be translated in a straightforward manner into Ethernet 802.3 LLC2 frames. This is the default conversion on routers.

However, many Ethernet-attached IBM devices use nonstandard encapsulation of LLC2 on Ethernet. Such IBM devices, including PS/2s running OS/2 Extended Edition and RT-PCs, do not place their LLC2 data inside an 802.3 format frame, but rather place it into an Ethernet Type 2 frame whose type is specified as *0x80d5*. This nonstandard format is called 0x80d5, named after the type

of frame. This format is also sometimes called *RT-PC Ethernet format* because these frames were first widely seen on the RT-PC. Hosts using this nonstandard 0x80d5 format cannot read the standard Token Ring LLC2 to Ethernet 802.2 LLC frames.

The format of all these frames is given in the *Internetworking Technology Overview* publication.

To enable Token Ring LLC2 to Ethernet LLC2 conversion, you can perform one or both of the following tasks:

- Enable 0x80d5 processing.
- Enable Standard Token Ring LLC2 to Ethernet LLC2 conversion.

Enable 0x80d5 Processing

You can change the router's default translation behavior of translating Token Ring LLC to Ethernet 802.3 LLC to translate Token Ring LLC2 frames into Ethernet *0x80d5 format* frames. To enable this nonstandard conversion, perform the following task in global configuration mode:

Task	Command
Change the router's Ethernet/Token Ring translation behavior to translate Token Ring LLC2 frames into Ethernet *0x80d5 format* frames.	**source-bridge enable-80d5**

Enable Standard Token Ring LLC2-to-Ethernet LLC2 Conversion

After you change the router's translation behavior to perform Token Ring LLC2 frames into Ethernet 80d5 format frames, some of the non-IBM hosts in your network topology might use the standard Token Ring conversion of Token Ring LLC2 to 802.3 LLC2 frames. If this is the case, you can change the translation method of those hosts to use the standard translation method on a per-DSAP basis. The translation method for all the IBM hosts would still remain as Token Ring LLC2 to Ethernet 0x80d5 translation.

To define non-IBM hosts in your network topology to use the standard translation method while the IBM hosts use the nonstandard method, perform the following task in global configuration mode:

Task	Command
Allow some other devices to use normal LLC2/IEEE 802.3 translation on a per-DSAP basis.	**source-bridge sap-80d5** *dsap*

Configure NetBIOS Support

NetBIOS is a nonroutable protocol that was originally designed to transmit messages between stations, typically IBM PCs, on a Token Ring network. NetBIOS allows messages to be exchanged between the stations using a name rather than a station address. Each station knows its name and is responsible for knowing the names of other stations on the network.

Note In addition to this type of NetBIOS, which runs over LLC2, we have implemented another type of NetBIOS that runs over IPX. For information on the IPX type of NetBIOS, refer to the chapter "Configuring Novell IPX" in this manual.

NetBIOS name caching allows the router to maintain a cache of NetBIOS names, which avoids the high overhead of transmitting many of the broadcasts used between client and server NetBIOS PCs (IBM PCs or PS/2s) in an SRB environment.

When NetBIOS name caching is enabled, the router performs the following actions:

- Notices when any hosts send a series of duplicated "query" frames and reduces them to one frame per period. The time period is configurable.
- Keeps a cache of mappings between NetBIOS server and client names and their MAC addresses. By watching NAME_QUERY and NAME_RECOGNIZED request and response traffic between clients and servers, the router can forward broadcast requests sent by clients to find servers (and by servers in reply to their clients) directly to their needed destinations, rather than forwarding them for broadcast across the entire bridged network.

The router will time out the entries in the NetBIOS name cache after a specific interval of their initial storage. The timeout value is a user-configurable value. You can configure the timeout value for a particular Token Ring if the NetBIOS name cache is enabled on the interface connecting to that Token Ring. In addition, you can configure static name cache entries that never time out for frequently accessed servers whose locations or paths typically do not change. Static RIF entries are also specified for such hosts.

Generally, NetBIOS name caching is most useful when a large amount of NetBIOS broadcast traffic creates bottlenecks on WAN media connecting distant locations, and the WAN media is overwhelmed with this traffic. However, when two high-speed LAN segments are directly interconnected, the packet savings of NetBIOS name caching is probably not worth the router processor overhead associated with it.

Note NetBIOS name caching is not recommended to be turned on in backbone routers, particularly if you have it enabled in all the routers connected to the backbone. NetBIOS caching should be distributed among multiple routers. NetBIOS name caching can be used only between routers that are running Software Release 9.1 or later.

To enable NetBIOS name caching, you must perform the tasks in the following sections:

- Enable the Proxy Explorers Feature on the Appropriate Interface
- Specify Timeout and Enable NetBIOS Name Caching

In addition, you can configure NetBIOS name caching as described in the following sections:

- Configure the NetBIOS Cache Name Length
- Enable NetBIOS Proxying
- Create Static Entries in the NetBIOS Name Cache
- Specify Dead-Time Intervals for NetBIOS Packets

Enable the Proxy Explorers Feature on the Appropriate Interface

In order to enable NetBIOS name caching on an interface, the proxy explorers feature must first be enabled on that interface. This feature must either be enabled for response to all explorer packets or for response to NetBIOS packets only.

To determine whether the proxy explorers feature has been enabled, perform the following task in EXEC mode:

Task	Command
Determine whether or not the proxy explorers feature has been enabled	**show configuration**[1]

1. This command is documented in the "System Image, Microcode Image, and Configuration File Load Commands" chapter of the *Router Products Command Reference* publication.

To determine whether proxy explorers has been configured for response to all explorer packets, look in the router's configuration file for the **source-bridge proxy-explorer** entry for the appropriate interface. For example, if the appropriate interface is Token Ring 0, look for an entry similar to the following:

```
interface tokenring 0
source-bridge proxy-explorer
```

If that entry does not exist, look for the **source-bridge proxy-netbios-only** entry for the appropriate interface.

If neither entry exists, proxy explorers has not yet been enabled for the appropriate interface. To enable proxy explorers for response to all explorer packets, refer to the section "Configure Proxy Explorers" later in this chapter.

Otherwise, enable proxy explorers only for the NetBIOS name caching function by performing the following task in interface configuration mode:

Task	Command
Enable use of proxy explorers only for the NetBIOS name caching function and not for their general local response to explorers.	**source-bridge proxy-netbios-only**

Specify Timeout and Enable NetBIOS Name Caching

After you have ensured that the proxy explorers feature has been enabled for the appropriate interface, you can specify a cache timeout and enable NetBIOS name caching. To do this, perform the following tasks:

Task	Command
Specify the timeout for entries in the router's NetBIOS name cache.	**netbios name-cache timeout** *minutes*
Enable NetBIOS name caching for the appropriate interfaces.	**netbios enable-name-cache**

Configure the NetBIOS Cache Name Length

To specify how many characters of the NetBIOS type name that the name cache will validate, perform the following global configuration task:

Task	Command
Specify the number of characters of the NetBIOS type name to cache.	**netbios name-cache name-len** *length*

Enable NetBIOS Proxying

The router can act as a proxy and send NetBIOS datagram type frames. To enable this capability, perform the following global configuration task:

Task	Command
Enable NetBIOS proxying.	**netbios name-cache proxy-datagram** *seconds*

To define the validation time when the router is acting as a proxy for NetBIOS NAME_QUERY command or for explorer frames, perform the following global configuration task:

Task	Command
Define validation time.	**rif validate-age** *seconds*

Create Static Entries in the NetBIOS Name Cache

If the router communicates with one or more NetBIOS stations on a regular basis, adding static entries to the NetBIOS name cache for these stations can reduce network traffic and router overhead. You can define a static NetBIOS name cache entry that associates the server with the NetBIOS name and the MAC address. If the router acts as a NetBIOS server, you can specify that the static NetBIOS name cache is available locally through a particular interface. If a remote router acts as the NetBIOS server, you can specify that the NetBIOS name cache is available remotely. To do this, perform one of the following tasks in global configuration mode:

Task	Command
Define a static NetBIOS name cache entry and specify that it is available locally through a particular interface.	**netbios name-cache** *mac-address netbios-name interface-name*
Define a static NetBIOS name cache entry and specify that it is available remotely.	**netbios name-cache** *mac-address netbios-name* **ring-group** *group-number*

If you have defined a NetBIOS name cache entry, you must also define a RIF entry. For an example of how to configure a static NetBIOS entry, see the "Example of NetBIOS Support with a Static NetBIOS Cache Entry" section later in this chapter.

Specify Dead-Time Intervals for NetBIOS Packets

When NetBIOS name caching is enabled and default parameters are set on the router (as well as the NetBIOS name server and the NetBIOS name client), approximately 20 broadcast packets per logon are kept on the local ring where they are generated. The broadcast packets are of the type ADD_NAME_QUERY, ADD_GROUP_NAME, and STATUS_QUERY.

The router also converts pairs of FIND_NAME and NAME_RECOGNIZED packets received from explorers, which traverse all rings, to specific route frames that are sent only between the two machines that need to see these packets.

You can specify a query-timeout, or "dead-time" interval to prevent repeat or duplicate broadcast of these type of packets for the duration of the interval.

To specify dead time intervals, perform one or both of the following tasks in global configuration mode:

Task	Command
Specify a dead time interval during which the router drops any broadcast (NetBIOS ADD_NAME_QUERY, ADD_GROUP_NAME, or STATUS_QUERY) frames if they are duplicate frames sent by the same host.	**netbios name-cache query-timeout** *seconds*
Specify a dead time interval during which the router drops FIND_NAME and NAME_RECOGNIZED frames if they are duplicate frames sent by the same host.	**netbios name-cache recognized-timeout** *seconds*

Configure LAN Network Manager Support

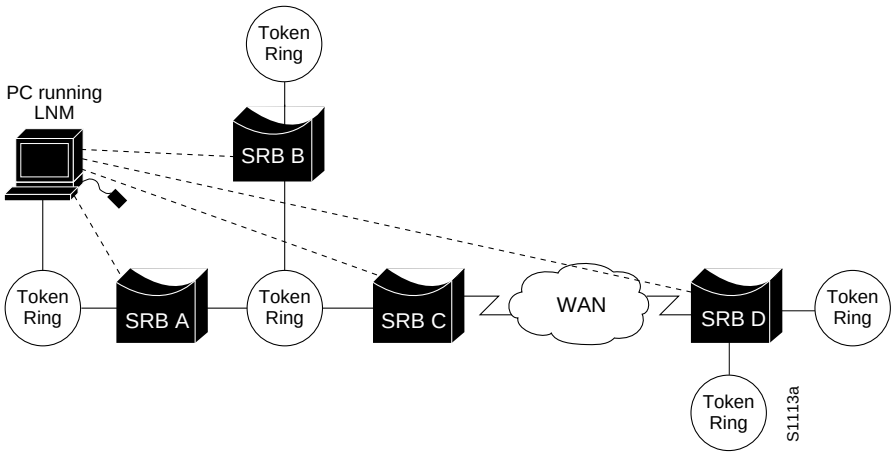
LAN Network Manager (LNM), formerly called LAN Manager, is an IBM product for managing a collection of source-route bridges. Using either a proprietary protocol or the Simple Network Management Protocol (SNMP), LNM allows you to monitor the entire collection of Token Rings that comprise your source-route bridged network. You can use LNM to manage the configuration of source-route bridges, monitor Token Ring errors, and gather information from Token Ring parameter servers.

Note LNM is supported on the 4/16-Mb Token Ring cards that can be configured for either 4- or 16-Mb transmission speeds. LNM support is not provided on CSC-R16M cards with SBEMON 2.0.

LNM is not limited to managing locally attached Token Ring networks; it also can manage any other Token Rings in your source-route bridged network that are connected through non-Token Ring media. To accomplish this task, LNM works in conjunction with the IBM Bridge Program. The IBM Bridge Program gathers data about the local Token Ring network and relays it back to LNM. In this manner, the bridge program becomes a proxy for information about its local Token Ring. Without this ability, you would require direct access to a device on every Token Ring in the network. This process would make managing an SRB environment awkward and cumbersome.

Figure 23-12 shows some Token Rings attached through a cloud and one LNM linking to a source-route bridge on each local ring.

Figure 23-12 LNM Linking to a Source-Route Bridge on Each Local Ring



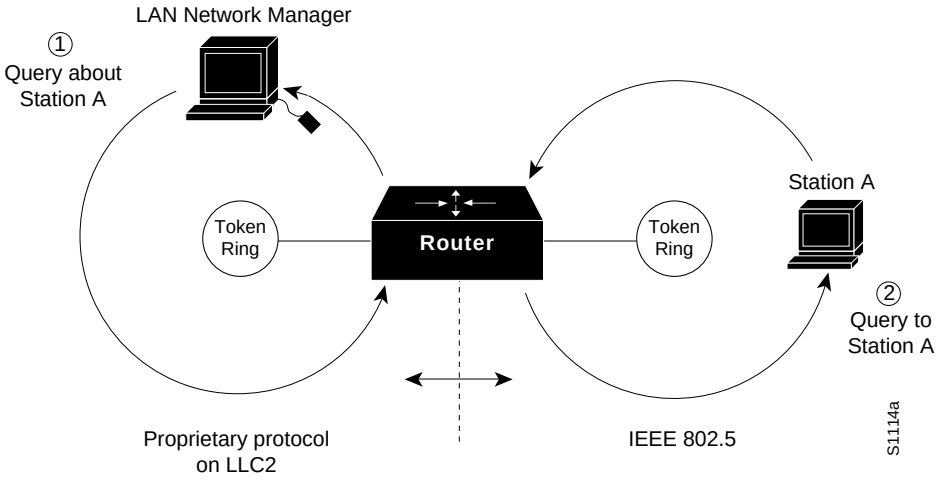
If LNM requires information about a station somewhere on a Token Ring, it uses a proprietary IBM protocol to query to one of the source-route bridges connected to that ring. If the bridge can provide the requested information, it simply responds directly to LNM. If the bridge does not have the necessary information, it queries the station using a protocol published in the IEEE 802.5 specification. In either case, the bridge uses the proprietary protocol to send a valid response back to LNM, using the proprietary protocol.

As an analogy, consider a language translator who sits between a French-speaking diplomat and a German-speaking diplomat. If the French diplomat asks the translator a question in French for the German diplomat and the translator knows the answer, he or she simply responds without translating the original question into German. If the French diplomat asks a question the translator does not know how to answer, the translator must first translate the question to German, wait for the German diplomat to answer, and then translate the answer back to French.

Similarly, if LNM queries a source-route bridge in the proprietary protocol and the bridge knows the answer, it responds directly using the same protocol. If the bridge does not know the answer, it must first translate the question to the IEEE 802.5 protocol, query the station on the ring, and then translate the response back to the proprietary protocol to send to LNM.

Figure 23-13 illustrates requests from the LNM originating in an IBM proprietary protocol and then translated into IEEE 802.5 MAC-level frames.

Figure 23-13 LAN Network Manager Monitoring and Translating



Notice that the proprietary protocol LNM uses to communicate with the source-route bridge is an LLC2 connection. Although its protocol cannot be routed, LNM can monitor or manage anything within the SRB network.

How the Router Works with LNM

As of Software Release 9.0, our routers using 4/16-Mbps Token Ring interfaces configured for SRB support the proprietary protocol that LNM uses. These routers provide all functions the IBM Bridge Program currently provides. Thus LNM can communicate with a router as if it were an IBM source-route bridge, such as the IBM 8209, and can manage or monitor any Token Ring connected to the router.

Through IBM Bridge support, LNM provides three basic services for the SRB network:

- The Configuration Report Server (CRS) monitors the current logical configuration of a Token Ring and reports any changes to LNM. CRS also reports various other events, such as the change of an active monitor on a Token Ring.
- The Ring Error Monitor (REM) monitors errors reported by any station on the ring. In addition, REM monitors whether the ring is in a functional or a failure state.
- The Ring Parameter Server (RPS) reports to LNM when any new station joins a Token Ring and ensures that all stations on a ring are using a consistent set of reporting parameters.

IBM Bridge support for LNM also allows asynchronous notification of some events that can occur on a Token Ring. Examples of these events include notification of a new station joining the Token Ring or of the ring entering failure mode, known as *beaconing*. Support is also provided for LNM to change the operating parameters in the bridge. For a complete description of LNM, refer to the IBM product manual supplied with the LNM program.

LNM support in our source-route bridges is a powerful tool for managing SRB networks. Through the ability to communicate with LNM and to provide the functionality of the IBM Bridge Program, our device appears as part of the IBM network. You therefore gain from the interconnectivity of our products without having to learn a new management product or interface.

When SRB is enabled on the router, configuring the router to perform the functions of an IBM Bridge for communication with LNM occurs automatically. Therefore, if SRB has been enabled on the router, you do not need to perform any tasks to enable LNM support. However, the LNM software residing on a management station on a Token Ring on the network should be configured to properly communicate with the router.

There are several options for modifying LNM parameters in the router, but none are required for basic functionality. For example, because users can now modify the operation of the router through SNMP as well as through LNM, there is an option to exclude a user from modifying the router configuration through LNM. You also can specify which of the three LNM services (CRS, REM, RPS) the source-route bridge will perform.

To configure LNM support, perform the tasks in the following sections:

- Configure LNM Software on the Management Stations to Communicate with the Router
- Prevent LNM Stations from Modifying Router Parameters
- Enable Other LRMs to Change Router/Bridge Parameters
- Apply a Password to an LNM Reporting Link
- Enable LNM Servers
- Change Reporting Thresholds
- Change an LNM Reporting Interval
- Monitor LNM Operation

Configure LNM Software on the Management Stations to Communicate with the Router

Because configuring an LNM station is a fairly simple task and is well covered in the LNM documentation, it is not covered in depth here. However, it is important to mention that you must enter the MAC addresses of the interfaces comprising the ports of the bridges as adapter addresses. When you configure the router as a multiport bridge, configuring an LNM station is complicated by the virtual ring that is involved. The basic problem extends from the fact that LNM is designed to only understand the concept of a two-port bridge, and the router with a virtual ring is a *multiport* bridge. The solution is to configure a virtual ring into the LNM Manager station as a series of dual-port bridges.

Prevent LNM Stations from Modifying Router Parameters

Because there is now more than one way to remotely change parameters in a router (either using SNMP or the proprietary IBM protocol), some method is needed to prevent such changes from detrimentally interacting with each other.You can prevent any LNM station from modifying parameters in the router. It does not affect the ability of LNM to monitor events, only to change parameters in the router.

To prevent the modification of router parameters by LNM station, perform the following task in global configuration mode:

Task	Command
Prevent LNM stations from modifying LNM parameters in the router.	**lnm snmp-only**

Enable Other LRMs to Change Router/Bridge Parameters

LNM has a concept of reporting links and reporting link numbers. A reporting link is simply a connection (or potential connection) between a LAN Reporting Manager (LRM) and a bridge. A reporting link number is a unique number used to identify a reporting link. An IBM bridge allows four simultaneous reporting links numbered 0 through 3. Only the LRM attached on the lowest-numbered connection is allowed to change LNM parameters in the router, and then only when that connection number falls below a certain configurable number. In the default configuration, the LRM connected through link 0 is the only LRM that can change LNM parameters in the router.

To enable other LRMs to change router/bridge parameters, perform the following task in interface configuration mode:

Task	Command
Enable a LRM other than that connected through link 0 to change router/bridge parameters.	**lnm alternate** *number*

Apply a Password to an LNM Reporting Link

Each reporting link has its own password that is used not only to prevent unauthorized access from an LRM to a bridge but to control access to the different reporting links. This is important because it is possible to change parameters through some reporting links.

To apply a password to an LNM reporting link, perform the following task in interface configuration mode:

Task	Command
Apply a password to an LNM reporting link.	**lnm password** *number string*

Enable LNM Servers

As in an IBM bridge, the router provides several functions that gather information from a local Token Ring. All of these functions are enabled by default, but also can be disabled. The LNM servers are explained in the section “How the Router Works with LNM” earlier in this chapter.

To enable LNM servers, perform one or more of the following tasks in interface configuration mode:

Task	Command
Enable the LNM Configuration Report Server (CRS).	**lnm crs**
Enable the LNM Ring Error Monitor (REM).	**lnm rem**
Enable the LNM Ring Parameter Server (RPS).	**lnm rps**

Change Reporting Thresholds

The router sends a message to all attached LNMs whenever it begins to drop frames. The threshold at which this report is generated is based on a percentage of frames dropped compared with those forwarded. This threshold is configurable, and defaults to a value of 0.10 percent. You can configure the threshold by entering a single number, expressing the percentage loss rate in hundredths of a percent. The valid range is 0 to 9999.

To change reporting thresholds, perform the following task in interface configuration mode:

Task	Command
Change the threshold at which the router reports the frames-lost percentage to LNM.	**lnm loss-threshold** *number*

Change an LNM Reporting Interval

All stations on a Token Ring notify the Ring Error Monitor (REM) when they detect errors on the ring. In order to prevent excessive messages, error reports are not sent immediately, but are accumulated for a short interval and then reported. A station learns the duration of this interval from a router (configured as a source-route bridge) when it first enters the ring. This value is expressed in tens of milliseconds between error messages. The default is 200, or 2 seconds. The valid range is 0 to 65535.

To change an LNM reporting interval, perform the following task in interface configuration mode:

Task	Command
Set the time interval during which stations report ring errors to the Ring Error Monitor (REM).	**lnm softerr** *milliseconds*

Monitor LNM Operation

Once LNM support is enabled, you can monitor LNM operation. To observe the configuration of the LNM bridge and its operating parameters, perform the following tasks in the EXEC mode:

Task	Command
Display all configured bridges and their global parameters.	**show lnm bridge**
Display the logical configuration of all bridges configured in the router.	**show lnm config**
Display LNM information for an interface or all interfaces of the router.	**show lnm interface** [*interface*]
Display LNM information about a Token Ring or all Token Rings on the network.	**show lnm ring** [*ring-number*]
Display LNM information about a station or all stations on the network.	**show lnm station** [*address*]

Secure the SRB Network

This section describes how to configure three features that are used primarily to provide network security: NetBIOS access filters, administrative filters, and access expressions that can be combined with administrative filters. In addition, these features can be used to increase network performance because they reduce the number of packets that traverse the backbone network.

Configure NetBIOS Access Filters

NetBIOS packets can be filtered when transmitted across a Token Ring bridge. Two types of filters can be configured: one for source and destination station names and one for arbitrary byte patterns in the packet itself.

As you configure NetBIOS access filters, keep the following issues in mind:

- The access lists that apply filters to an interface are scanned in the order they are entered.
- There is no way to put a new access list entry in the middle of an access list. All new additions to existing NetBIOS access lists are placed at the end of the existing list.
- Access list arguments are case sensitive. The software makes a literal translation, so that a lowercase “a” is different from an uppercase “A.” (Most nodes are named in uppercase letters.)
- A host NetBIOS access list and byte NetBIOS access list can each use the same name. The two lists are identified as unique and bear no relationship to each other.
- The station names included in the access lists are compared with the source name field for NetBIOS commands 00 and 01 (ADD_GROUP_NAME_QUERY and ADD_NAME_QUERY), as well as the destination name field for NetBIOS commands 08 and 0A (DATAGRAM and NAME_QUERY).
- If an access list does not contain a particular station name, the default action is to deny the access to that station.

In order to minimize any performance degradation, NetBIOS access filters do not examine all packets. Rather, they examine certain packets that are used to establish and maintain NetBIOS client/server connections, thereby effectively stopping new access and load across the router. However, applying a new access filter does not terminate existing sessions immediately. All new sessions will be filtered, but existing sessions could continue for some time.

There are two ways you can configure NetBIOS access filters:

- Configure NetBIOS access filters using station names
- Configure NetBIOS access filters using a byte offset

Configure NetBIOS Access Filters Using Station Names

To configure access filters using station names, you must do the following:

Step 1 Assign the station access list name.

Step 2 Specify the direction of the message to be filtered on the interface.

The NetBIOS station access list contains the station name to match, along with a permit or deny condition. You must assign the name of the access list to a station or set of stations on the network.

To assign a station access list name, perform the following task in global configuration mode:

Task	Command
Assign the name of an access list to a station or set of stations on the network.	**netbios access-list host** *name* {**permit** | **deny**} *pattern*

When filtering by station name, you can choose to filter either incoming or outgoing messages on the interface. To specify the direction, perform the one of the following tasks in interface configuration mode:

Task	Command
Define an access list filter for incoming messages.	**netbios input-access-filter host** *name*
Define an access list filter for outgoing messages.	**netbios output-access-filter host** *name*

Configure Access Filters Using a Byte Offset

To configure access filters you must do the following:

Step 1 Assign a byte offset access list name.

Step 2 Specify the direction of the message to be filtered on the interface.

Keep the following notes in mind while configuring access filters using a byte offset:

- When an access list entry has an offset plus the length of the pattern that is larger than the packet's length, the entry will not make a match for that packet.
- Because these access lists allow arbitrary byte offsets into packets, these access filters can have a significant impact on the amount of packets per second transiting across the bridge. They should be used only when situations absolutely dictate their use.

The NetBIOS byte offset access list contains a series of offsets and hexadecimal patterns with which to match byte offsets in NetBIOS packets. To assign a byte offset access list name, perform the following task in global configuration mode:

Task	Command
Define the byte offsets and patterns within NetBIOS messages to match with access list parameters.	**netbios access-list bytes** *name* {**permit** | **deny**} *offset pattern*

Note Using NetBIOS Byte Offset access filters disables the autonomous or fast switching of source-route bridging frames.

When filtering by byte offset, you can filter either incoming or outgoing messages on the interface. To specify the direction, perform one of the following tasks in interface configuration mode:

Task	Command
Specify a byte-based access filter on incoming messages.	**netbios input-access-filter bytes** *name*

Task	Command
Specify a byte-based access filter on outgoing messages.	**netbios output-access-filter bytes** *name*

Configure Administrative Filters for Token Ring Traffic

Source-route bridges normally filter frames according to the routing information contained in the frame. That is, a bridge will not forward a frame back to its originating network segment or any other network segment that the frame has already traversed. This section describes how to configure another type of filter—the administrative filter.

Administrative filters can filter frames based on the following methods:

- Protocol type—IEEE 802 or Subnetwork Access Protocol (SNAP)
- Token Ring vendor code
- Source address
- Destination address

Whereas filtering by Token Ring address or vendor code causes no significant performance penalty, filtering by protocol type significantly affects performance. A list of SNAP (Ethernet) type codes is provided in the "Ethernet Type Codes" appendix in the *Router Products Command Reference* publication.

Filter Frames by Protocol Type

You can configure administrative filters by protocol type by specifying protocol type codes in an access list. You then apply that access list to either IEEE 802.2 encapsulated packets or to SNAP-encapsulated packets on the appropriate interface.

The order in which you specify these elements 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 If a single condition is to be denied, there must be an **access-list** command that permits everything as well, or all access is denied.

To filter frames by protocol type, perform the following task in global configuration mode:

Task	Command
Create an access list for filtering frames by protocol type.	**access-list** *access-list-number* {**permit** | **deny**} {*type-code wild-mask* | *address mask*}

You can filter IEEE 802-encapsulated packets on either input or output. The access list you specify is the one you created that includes the protocol type codes.

To enable filtering on input or output, perform one of the following tasks in interface configuration mode:

Task	Command
Enable filtering of IEEE 802-encapsulated packets on input by type code.	**source-bridge input-lsap-list** *access-list-number*
Enable filtering of IEEE 802-encapsulated packets on output by type code.	**source-bridge output-lsap-list** *access-list-number*

You can filter SNAP-encapsulated packets on either input or output. The access list you specify is the one you created that includes the protocol type codes.

To enable filtering on input or output, perform one of the following tasks in interface configuration mode:

Task	Command
Filter SNAP-encapsulated packets on input by type code.	**source-bridge input-type-list** *access-list-number*
Filter SNAP-encapsulated frames on output by type code.	**source-bridge output-type-list** *access-list-number*

Filter Frames by Vendor Code

To configure administrative filters by vendor code or address, define access lists that look for Token Ring addresses or for particular vendor codes for administrative filtering. To do so, perform the following task in global configuration mode:

Task	Command
Configure vendor code access lists.	**access-list** *access-list-number* {**permit** | **deny**} *address mask*

Filter Input by Source Addresses

To configure filtering on IEEE 802 source addresses, assign an access list to a particular input interface for filtering the Token Ring or IEEE 802 source addresses. To do so, perform the following task in interface configuration mode:

Task	Command
Enable filtering on IEEE 802 source addresses.	**source-bridge input-address-list** *access-list-number*

Filter Output by Source Addresses

To configure output filtering on IEEE 802 source addresses, assign an access list to a particular output interface. To do so, perform the following task in interface configuration mode:

Task	Command
Enable filtering on IEEE 802 destination addresses.	**source-bridge output-address-list** *access-list-number*

Configure Access Expressions that Combine Administrative Filters

You can use access expressions to combine access filters to establish complex conditions under which bridged frames can enter or leave an interface. Using access expressions, you can achieve levels of control on the forwarding of frames that otherwise would be impossible when using only simple access filters.

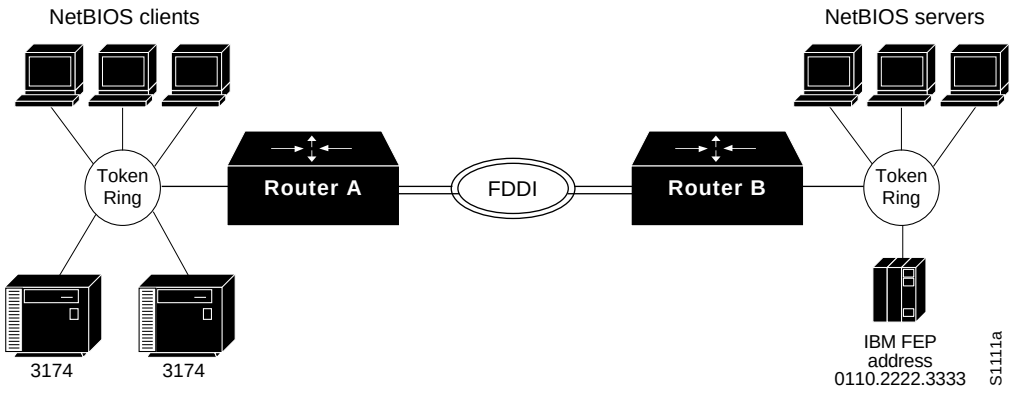
Access expressions are constructed from individual access lists that define administrative filters for the following fields in packets:

- LSAP and SNAP type codes
- MAC addresses
- NetBIOS station names
- NetBIOS arbitrary byte values

Note For any given router interface, an access expression cannot be used if an access list has been defined for a given direction. For example, if an input access list is defined for MAC addresses on an interface, no access expression can be specified for the input side of that interface.

Figure 23-14 shows how access expressions can be useful.

Figure 23-14 Access Expression Example



In Figure 23-14, two routers each connect a Token Ring to an FDDI backbone. On both Token Rings, SNA and NetBIOS bridging support is required. On Token Ring A, NetBIOS clients must communicate with any NetBIOS server off Token Ring B or any other, unpictured router. However, the 3174s off Token Ring A must only communicate with the one FEP off of Token Ring B, located at MAC address 0110.2222.3333.

Without access expressions, this scenario cannot be achieved. A filter on Router A that restricted access to only the FEP would also restrict access of the NetBIOS clients to the FEP. What is needed is an access *expression* that would state "If it is a NetBIOS frame, pass through, but if it is an SNA frame, allow only access to address 0110.2222.3333."

Note Using access-expressions that combine access filters disables the autonomous or fast switching of source-route bridging frames.

Configure Access Expressions

To configure an access expression perform the following tasks:

- Design the access expression.
- Configure the access lists used by the expression.
- Configure the access expression into the router.

When designing an access expression, you must create some phrase that indicates, in its entirety, all the frames that will *pass* the access expression. This access expression is designed to apply on frames coming from the Token Ring interface on Router A in Figure 23-14:

"Pass the frame if it is a NetBIOS frame or if it is an SNA frame destined to address 0110.2222.3333."

In Boolean form, this phrase can be written as follows:

"Pass if "NetBIOS or (SNA and destined to 0110.2222.3333).""

The preceding statement requires three access lists to be configured:

- An access list that passes a frame if it is a NetBIOS frame (SAP = 0xF0F0)
- An access list that passes a frame if it is an SNA frame (SAP = 0x0404)
- An access list that passes a MAC address of 0110.2222.3333

The following configuration allows for all these conditions:

```
! Access list 201 passes NetBIOS frames (command or response)
access-list 201 permit 0xF0F0 0x0001
!
access-list 202 permit 0x0404 0x0001 ! Permits SNA frames (command or response)
access-list 202 permit 0x0004 0x0001 ! Permits SNA Explorers with NULL DSAP
!
! Access list 701 will permit the FEP MAC address
! of 0110.2222.3333
access-list 701 permit 0110.2222.3333
```

The 0x0001 mask allows command and response frames to pass equally.

Apply the access expression to the appropriate interface by performing the following task in interface configuration mode:

Task	Command
Define a per-interface access expression.	**access-expression** {**in** | **out**} *expression*

Optimize Access Expressions

It is possible combine access expressions. Suppose you wanted to transmit SNA traffic through to a single address, but allow other traffic through the router without restriction. The phrase could be written as follows:

"Allow access if the frame is not an SNA frame, or if it is going to host 0110.2222.3333."

More tersely this would be:

"Not SNA or destined to 0110.2222.3333."

The access lists defined in the previous section create the following configuration:

```
interface tokenring 0
access-expression in ~lsap(202) | dmac(701)
!
access-list 202 permit 0x0404 0x0001 ! Permits SNA frames (command or response)
access-list 202 permit 0x0004 0x0001 ! Permits SNA Explorers with NULL DSAP
!
! Access list 701 will permit the FEP MAC address
! of 0110.2222.3333
access-list 701 permit 0110.2222.3333
```

This is a better and simpler access list than the one originally introduced and will probably result in better run-time execution as a result. Therefore, it is best to simplify your access expressions as much as possible before configuring them into the router.

Alter Access Lists Used in Access Expressions

Because access expressions are composed of access lists, special care must be taken when deleting and adding access lists that are referenced in these access expressions.

If an access list that is referenced in an access expression is deleted, the access expression merely ignores the deleted access list. However, if you want to redefine an access list, you can create a new access list with the appropriate definition and use the same name as the old access list. The newly defined access list replaces the old one of the same name.

For example, if you want to redefine the NetBIOS access list named MIS that was used in the preceding example, you would enter the following sequence of configuration commands:

```
!  Replace the NetBIOS access list
interface tokenring 0
access-expression in (smac(701) & netbios-host(accept))
no netbios access-list host accept permit CISCO*
```

Tune the SRB Network

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

- Prioritize Traffic Based on SNA Local LU Addresses
- Enable Class of Service
- Assign a Priority Group to an Input Interface
- Enable or Disable the Source-Route Fast-Switching Cache
- Enable or Disable the Source-Route Autonomous-Switching Cache
- Enable or Disable the SSE
- Optimize Explorer Processing
- Configure Proxy Explorers
- Configure the Largest Frame Size

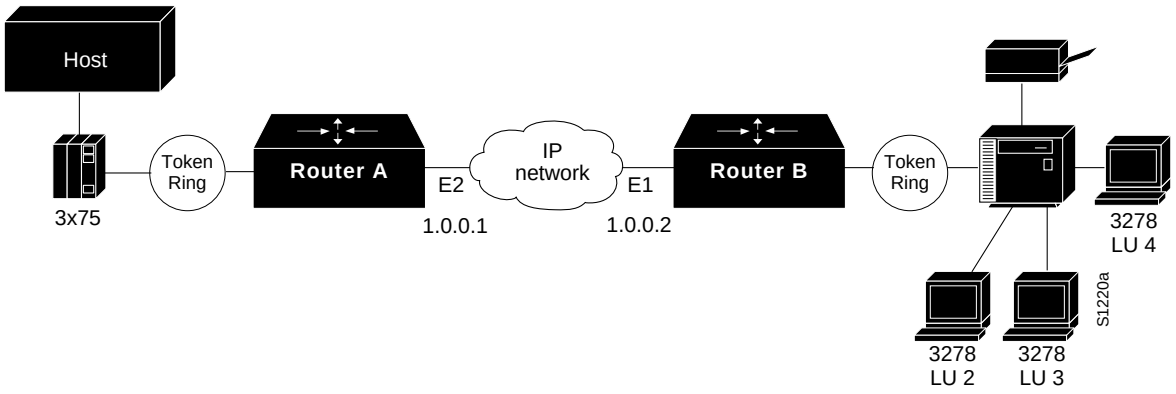
Note In some situations, you might discover that default settings for LLC2 configurations are not acceptable. In such a case, you can configure LLC2 for optimal use. The chapter "Configuring LLC2 and SDLC Parameters" in this manual describes how you can use them to optimize your network performance.

Prioritize Traffic Based on SNA Local LU Addresses

You can prioritize SNA traffic on an interface configured for either serial tunnel (STUN) or RSRB communication. The SNA local LU address prioritization feature allows SNA traffic to be prioritized according to the address of the logical units (LU) on the FID2 transmission headers. Currently, only dependent LUs are supported. The prioritization takes place on LU-LU traffic between an SNA Node type 5 or Node type 4, and Node type 2.

Figure 23-15 shows how SNA local address prioritization can be used.

Figure 23-15 SNA Local Address Prioritization



In Figure 23-15, the IBM mainframe is channel-attached to a 3x75 FEP, which is connected to a cluster controller via RSRb. Multiple 3270 terminals and printers, each with a unique local LU address, are then attached to the cluster controller. By applying SNA local LU address prioritization, each LU associated with a terminal or printer can be assigned a priority; that is, certain users can have terminals that have better response time than others, and printers can have lowest priority.

Note Both Local Acknowledgment and TCP priority features for STUN or RSRB must be turned on for SNA local address prioritization to take effect.

With the SNA local LU address prioritization feature, you can establish queuing priorities based on the address of the logical unit. To prioritize traffic, perform both of the following tasks in global configuration mode:

Task		Command
Step 1	Map LUs to TCP port numbers.	**locaddr-priority-list** *list-number address-number queue-keyword* [*dsap ds*] [*dmac dm*]
Step 2	Set the priority of TCP port numbers.	**priority-list** *list-number* **protocol** *protocol-name priority* **tcp** *port-number*

Enable Class of Service

To prioritize SNA traffic across the SNA backbone network, you can enable the class of service feature. This feature is useful only between FEP-to-FEP (PU4-to-PU4) communication across the non-SNA backbone. It allows important FEP traffic to flow on high-priority queues.

To enable class of service, IP encapsulation over a TCP connection and LLC2 local acknowledgment must be enabled.

To enable class of service, perform the following task in global configuration mode:

Task	Command
Enable class-of-service.	**source-bridge cos-enable**

Assign a Priority Group to an Input Interface

You can assign a priority group to an input interface. To do so, perform the following task in interface configuration mode:

Task	Command
Assign a priority group to an input interface.	**locaddr-priority** *list-number*

Enable or Disable the Source-Route Fast-Switching Cache

Rather than processing packets at the process level, the fast-switching feature enables the router to process packets at the interrupt level. Each packet is transferred from the input interface to the output interface without copying the entire packet to main system memory. Fast switching allows for faster implementations of local SRB between 4/16-Mb Token Ring cards in the same router/bridge, or between two router/bridges using the 4/16-Mb Token Ring cards and direct encapsulation.

By default, fast-switching software is enabled when SRB is enabled. To enable or disable source-route fast-switching, perform one of the following tasks in interface configuration mode:

Task	Command
Enable fast-switching.	**source-bridge route-cache**
Disable fast-switching.	**no source-bridge route-cache**

Note Using either NetBIOS Byte Offset access filters or access expressions that combine access filters disables the fast switching of source-route bridging frames.

Enable or Disable the Source-Route Autonomous-Switching Cache

Autonomous switching is a feature that enables the router to transmit packets from the input ciscoBus card to the output ciscoBus card without any involvement on the part of the router processor.

Autonomous switching is available for local SRB between ciscoBus Token Ring (CTR) cards in the same router/bridge. Autonomous switching provides higher switching rates than does fast switching between 4/16-Mb Token Ring cards. Autonomous switching works for both two-port bridges and multiport bridges that use ciscoBus Token Ring cards.

In a virtual ring that includes both ciscoBus Token Ring and 4/16-Mb Token Ring interfaces, frames that flow from one CTR interface to another are autonomously switched, and the remainder of the frames are fast switched. The switching that occurs on the CTR interface takes advantage of the high-speed ciscoBus controller processor.

To enable or disable source-route autonomous switching, perform one of the following tasks in interface configuration mode:

Task	Command
Enable autonomous switching.	**source-bridge route-cache cbus**
Disable autonomous switching.	**no source-bridge route-cache cbus**

Note Using either NetBIOS Byte Offset access filters or access-expressions that combine access filters disables the autonomous switching of SRB frames.

Enable or Disable the SSE

The Silicon Switch Engine (SSE) acts as a programmable cache to speed the switching of packets. To enable or disable the SSE, perform one of the following task in interface configuration mode:

Task	Command
Enable the SSE function.	**source-bridge route-cache sse**
Disable the SSE function.	**no source-bridge route-cache sse**

Optimize Explorer Processing

Efficient explorer processing is vital to the operation of SRB. The default configuration is satisfactory for most situations. However, there might be circumstances that create unexpected broadcast storms. You can optimize the handling of explorer frames, thus reducing processor overhead and increasing explorer packet throughput. This will enable the router to perform substantially better during explorer broadcast storms.

To optimize explorer processing, perform the following tasks in global configuration mode:

Task	Command
Set the maximum explorer queue depth.	**source-bridge explorerq-depth** *depth*
Set the maximum byte rate of explorers per ring.	**source-bridge explorer-maxrate** *maxrate*

You must also disable explorer fast-switching which is, by default, enabled. To disable explorer fast-switching, perform the following task in global configuration mode:

Task	Command
Disable explorer fast switching.	**no source-bridge explorer-fastswitch**

To enable explorer fast-switching after it has been disabled, perform the following task in global configuration mode:

Task	Command
Enable explorer fast switching.	**source-bridge explorer-fastswitch**

Configure Proxy Explorers

You can use the proxy explorers feature to limit the amount of explorer traffic propagating through the source-bridge network.

To configure proxy explorers, perform the following task in interface configuration mode:

Task	Command
Enable the interface to respond to any explorer packets that meet certain conditions necessary for a proxy response to occur.	**source-bridge proxy-explorer**

The router does not propagate proxy responses for a station. Instead, the router obtains the RIF path from the RIF cache, changes the explorer to a specific router frame, and forwards this frame to the destination. If the router does not receive a response before the validation timer expires, the RIF entry is marked as invalid. The invalid RIF entry is flushed from the cache table when another explorer for this station is received, and an explorer is forwarded to discover a path to this station.

Configure the Largest Frame Size

You can configure the largest frame size that is used to communicate with any peers in the ring group.

Generally, the router and the LLC2 device with which it communicates should support the same maximum SDLC I-frame size. The larger this value, the more efficiently the line is used, thus increasing performance.

Faster screen updates to 3278-style terminals often result by configuring the Token Ring FEP to send as large an I-frame as possible and then allowing the router to segment the frame into multiple SDLC I-frames.

After the Token Ring FEP has been configured to send the largest possible I-frame, you should configure the router to support the same maximum I-frame size. The default is 516 bytes. The maximum value the router can support is 8144 bytes.

To configure the largest frame size, perform the following task in global configuration mode:

Task	Command
Specify the largest frame size used to communicate with any peers in the ring group.	**source-bridge largest-frame** *ring-group size*

Establish SRB Interoperability with Specific Token Ring Implementations

This section describes how you can establish interoperability between router/bridges and specific Token Ring implementations. It includes the following sections:

- Establish SRB Interoperability with IBM PC/3270 Emulation Software
- Establish SRB Interoperability with TI MAC Firmware
- Reporting Spurious Frame-Copied Errors

Establish SRB Interoperability with IBM PC/3270 Emulation Software

You can establish interoperability with the IBM PC/3270 emulation program Version 3.0, even though it does not properly send packets over a source-route bridge.

Our implementation rewrites the RIF headers of the explorer packets that the PC/3270 emulation program sends to go beyond the local ring, thus confusing the IBM implementation into not looking beyond the local ring for the remote host.

To rewrite RIF headers, perform the following task in interface configuration mode:

Task	Command
Rewrite the RIF headers of explorer packets send by the PC/3270 emulation program to go beyond the local ring.	**source-bridge old-sna**

Establish SRB Interoperability with TI MAC Firmware

You can use a workaround to establish interoperability with Texas Instruments (TI) MAC firmware.

There is a known defect in earlier versions of the TI Token Ring MAC firmware. This implementation is used by Proteon, Apollo, and IBM RTs. A host using a MAC address whose first two bytes are zeros (such as a Cisco router/bridge) will not properly communicate with hosts using that version of TI firmware.

There are two solutions. The first involves installing a static RIF entry for every faulty node with which the router communicates. If there are many such nodes on the ring, this may not be practical.

You also can set the MAC address of our Token Ring to a value that works around the problem. Resetting the MAC address forces the use of a different MAC address on the specified interface, thereby avoiding the TI MAC firmware problem. However, you must ensure that no other host on the network is using that MAC address.

To reset the MAC address, perform the following task in interface configuration mode:

Task	Command
Reset the MAC address of the Token Ring interface to a value that provides a workaround to a problem in TI Token Ring MAC firmware.	**mac-address** *ieee-address*

Reporting Spurious Frame-Copied Errors

An IBM 3174 controller can be configured to report frame-copied errors to IBM LAN Network Manager software. These errors indicate that another host is responding to the MAC address of the 3174 controller. Both the 3174 and the IBM LAN Network Manager software can be configured to ignore frame-copied errors.

Monitor and Maintain the SRB Network

You can display a variety of information about the SRB network. To display the information you require, perform one or more of the following tasks in EXEC mode.

Task	Command
Display internal state information about the Token Ring interfaces in the system.	**show controllers token**
Provide high-level statistics about the state of source bridging for a particular interface.	**show interfaces**
Display all currently configured bridges and all parameters that are related to the bridge as a whole and not to one of its interfaces.	**show lnm bridge**
Display the logical (multiport bridge) configuration of the router.	**show lnm config**
Display all LNM-relevant information about a specific interface.	**show lnm interface** [*interface*]
Display all LNM-relevant information about a specific router ring number.	**show lnm ring** [*ring-number*]
Display all LNM-relevant information about a specific station or about all known stations on the ring.	**show lnm station** [*address*]
Show the current state of any current local acknowledgment for both LLC2 and SDLLC connections.	**show local-ack**
Display the contents of the NetBIOS cache.	**show netbios-cache**
Display the contents of the RIF cache.	**show rif**
Display the current source bridge configuration and miscellaneous statistics.	**show source-bridge**
Display the spanning-tree topology for the router.	**show span**
Display a summary of Silicon Switch Processor (SSP) statistics.	**show sse summary**

To maintain the SRB network, perform any of the following tasks in privileged EXEC mode:

Task	Command
Clear the entries of all dynamically learned NetBIOS names.	**clear netbios-cache**
Clear the entire RIF cache.	**clear rif-cache**
Clear the SRB statistical counters.	**clear source-bridge**
Reinitialize the SSP on the Cisco 7000 series.	**clear sse**

In addition to the EXEC-mode tasks to maintain the SRB network, you can perform the following task in global configuration mode:

Task	Command
Limit the size of the backup queue for RSRB to control the number of packets that can wait for transmission to a remote ring before they start being thrown away.	**source-bridge tcp-queue-max** *number*

SRB Configuration Examples

The following sections provide SRB configuration examples:

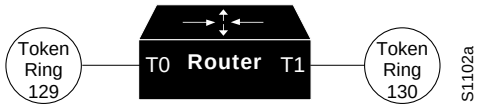
- Basic SRB with Spanning-Tree Explorers Example
- SRB with Automatic Spanning-Tree Function Configuration Example
- Optimized Explorer Processing Configuration Example
- SRB Only Example
- SRB and Routing Certain Protocols Example
- Multiport SRB Example
- SRB with Multiple Virtual Ring Groups Example
- Autonomous FDDI SRB Configuration Example
- RSRB Direct Frame Relay Encapsulation Example
- RSRB Using IP Encapsulation over a TCP Connection Example
- RSRB/TCP Fast Switching Configuration Example
- RSRB Using IP Encapsulation over an FST Connection Example
- RSRB Using All Types of Transport Methods Example
- RSRB with Local Acknowledgment Example
- RSRB with Local Acknowledgment and Passthrough Example
- Local Acknowledgment for LLC2 Example
- IP for Load Sharing over RSRB Example
- Adding a Static RIF Cache Entry Example
- Adding a Static RIF Cache Entry for a Two-Hop Path Example
- SR/TLB for a Simple Network Example
- SR/TLB with Access Filtering Example
- NetBIOS Support with a Static NetBIOS Cache Entry Example
- LNM for a Simple Network Example
- LNM for a More Complex Network Example
- NetBIOS Access Filters Example
- Filtering Bridged Token Ring Packets to IBM Machines Example
- Administrative Access Filters—Filtering SNAP Frames on Output Example

- Creating Access Expressions Example
- Access Expressions Example
- Configuring Priority for Locally Terminated Token Ring Interfaces in RSRB Example
- Fast Switching Example
- Autonomous Switching Example
- SNA Traffic Prioritization by LU Address Example

Basic SRB with Spanning-Tree Explorers Example

Figure 23-16 illustrates a simple two-port bridge configuration.Token Rings 129 and 130 are connected through the router/bridge.

Figure 23-16 Dual Port Source-Route Bridge Configuration



The example that follows routes IP, but source-route bridges all other protocols using spanning-tree explorers:

```
interface tokenring 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
multiring all
!
interface tokenring 1
ip address 131.108.130.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
! use RIFs, as necessary, with IP routing software
multiring all
```

SRB with Automatic Spanning-Tree Function Configuration Example

The following example of a Cisco series 7000 router configuration illustrates how to enable the automatic spanning tree function on an SRB network.

```
source-bridge ring-group 100

interface TokenRing 0/0
no ip address
ring-speed 16
multiring all
source-bridge active 1 10 100
source-bridge spanning 1
!
interface TokenRing 0/1
no ip address
ring-speed 16
multiring all
source-bridge active 2 10 100
source-bridge spanning 1
```

```
!
bridge 1 protocol ibm
```

Optimized Explorer Processing Configuration Example

The following configuration example improves the handling of explorer frames, enabling the router to perform substantially better during explorer broadcast storms. In this configuration, the maximum byte rate of explorers is set to 100000.

```
source-bridge explorer-maxrate 100000
source-bridge explorerQ-depth 100
no source-bridge explorer-fastswitch
```

SRB Only Example

The following example shows that all protocols are bridged, including IP. Because IP is being bridged, the system has only one IP address.

```
no ip routing
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
source-bridge 129 1 130
source-bridge spanning
!
interface TokenRing 1
ip address 131.108.129.2 255.255.255.0
source-bridge 130 1 129
source-bridge spanning
!
interface Ethernet 0
ip address 131.108.129.2 255.255.255.0
```

SRB and Routing Certain Protocols Example

In the following configuration, IP, XNS, and IPX are routed, while all other protocols are bridged between rings. While not strictly necessary, the Novell IPX and XNS network numbers are set consistently with the IP subnetwork numbers. This makes the network easier to maintain.

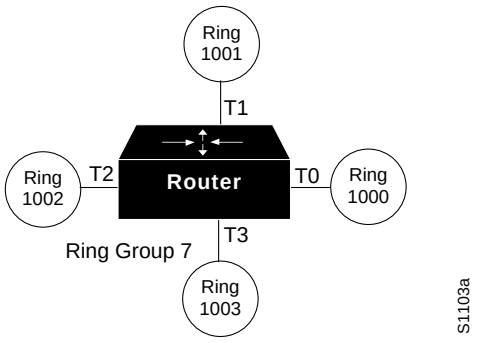
```
xns routing 0000.0C00.02C3
!
novell routing 0000.0C00.02C3
!
interface TokenRing 0
ip address 131.108.129.2 255.255.255.0
xns network 129
novell network 129
source-bridge 129 1 130
source-bridge spanning
multiring all
!
interface TokenRing 1
ip address 131.108.130.2 255.255.255.0
xns network 130
novell network 130
source-bridge 130 1 129
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 131.108.2.68 255.255.255.0
```

```
xns network 2
novell network 2
```

Multiport SRB Example

Figure 23-17 shows an example configuration of a four-port Token Ring source-route bridge. Rings 1000, 1001, 1002, and 1003 are all source-route bridged to each other across ring group 7.

Figure 23-17 Four-Port Source-Route Bridge



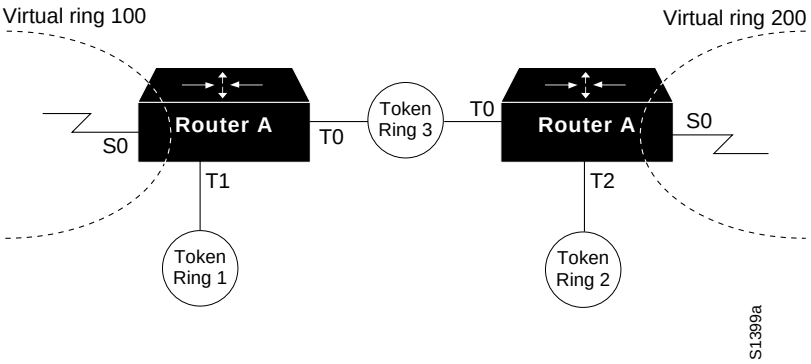
The following is a sample configuration file:

```
source-bridge ring-group 7
!
interface tokenring 0
source-bridge 1000 1 7
source-bridge spanning
!
interface tokenring 1
source-bridge 1001 1 7
source-bridge spanning
!
interface tokenring 2
source-bridge 1002 1 7
source-bridge spanning
!
interface tokenring 3
source-bridge 1003 1 7
source-bridge spanning
```

SRB with Multiple Virtual Ring Groups Example

Two virtual ring groups can only be connected through an actual Token Ring. Figure 23-18 shows Virtual Rings 100 and 200 connected through Token Ring 3.

Figure 23-18 Two Virtual Rings Connected by an Actual Token Ring



Configuration for Router A

```
source-bridge ring-group 100
!
interface tokenring 0
source-bridge 3 4 100
source-bridge spanning
interface tokenring 1
source-bridge 1 4 100
source-bridge spanning
```

Configuration for Router B

```
source-bridge ring-group 200
!
interface tokenring0
source-bridge 3 1 200
source-bridge spanning
interface tokenring 2
source-bridge 2 1 200
source-bridge spanning
```

Autonomous FDDI SRB Configuration Example

The following configuration for a Cisco 7000 series router illustrates how to enable SRB over FDDI:

```
interface fddi 1/0
source-bridge 1 10 100
source-bridge spanning
source-bridge route-cache cbus
```

RSRB Direct Frame Relay Encapsulation Example

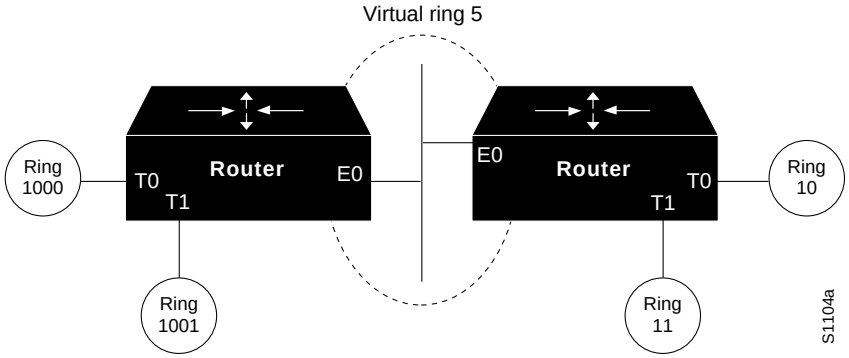
The following is the configuration file for direct Frame Relay encapsulation between RSRB peers:

```
source-bridge ring-group 200
source-bridge remote-peer 200 interface Serial0 dlci 30
!
interface serial 0
mtu 3000
no ip address
encapsulation frame-relay
clockrate 56000
frame-relay lmi-type ansi
frame-relay map rsrb 30
!
!
interface TokenRing 0
ip address 10.10.10.1 255.255.255.0
ring-speed 16
multiring all
source-bridge active 102 1 200
source-bridge spanning
```

RSRB Using IP Encapsulation over a TCP Connection Example

Figure 23-19 illustrates a configuration of two router/bridges configured for RSRB using TCP as a transport. Each router has two Token Rings. They are connected by an Ethernet segment over which the source-route bridged traffic will pass. The first router configuration is a source-route bridge at address 131.108.2.29.

Figure 23-19 RSRB Using TCP as a Transport



Using TCP as the transport, the configuration for the source-route bridge at address 131.108.2.29 as depicted in Figure 23-19 is as follows:

```
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 131.108.2.29
source-bridge remote-peer 5 tcp 131.108.1.27
!
interface ethernet 0
ip address 131.108.4.4 255.255.255.0
!
interface tokenring 0
ip address 131.108.2.29 255.255.255.0
source-bridge 1000 1 5
source-bridge spanning
```

```
!
interface tokenring 1
ip address 131.108.128.1 255.255.255.0
source-bridge 1001 1 5
source-bridge spanning
```

The configuration of the source-route bridge at 131.108.1.27 is as follows:

```
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 131.108.2.29
source-bridge remote-peer 5 tcp 131.108.1.27
!
interface ethernet 0
ip address 131.108.4.5 255.255.255.0
!
interface tokenring 0
ip address 131.108.1.27 255.255.255.0
source-bridge 10 1 5
source-bridge spanning
!
interface tokenring 1
ip address 131.108.131.1 255.255.255.0
source-bridge 11 1 5
source-bridge spanning
```

RSRB/TCP Fast Switching Configuration Example

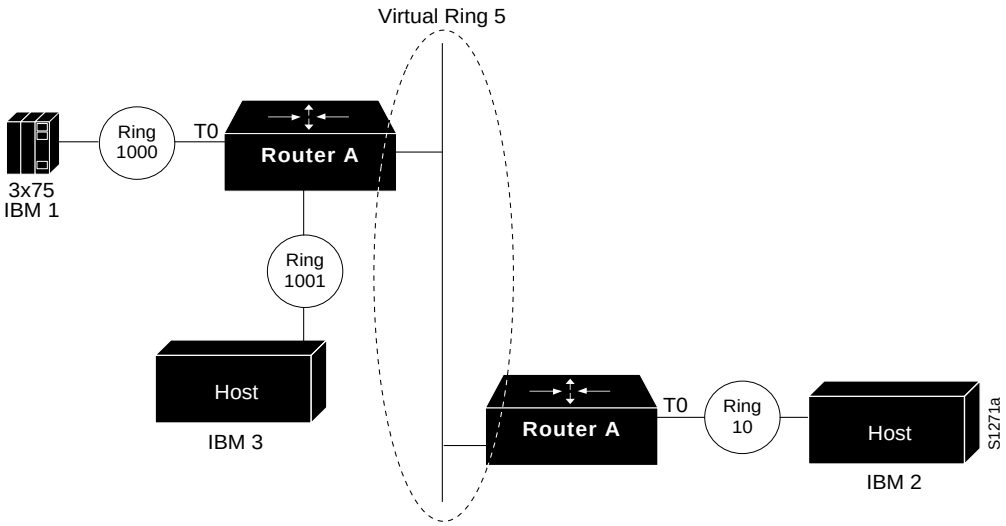
The following configuration enables RSRB/TCP fast switching:

```
source-bridge ring group 100
source-bridge remote-peer 100 ftcp 198.92.88.138
source-bridge remote-peer 100 ftcp 198.92.88.145
```

RSRB Using IP Encapsulation over an FST Connection Example

Figure 23-20 shows two routers connecting IBM hosts on Token Rings through an Ethernet backbone.

Figure 23-20 RSRB Using FST as a Transport



This example configuration enables IP encapsulation over an FST connection. In this configuration, the **source-bridge fst-peername** global configuration command is used to provide an IP address for the local router, the **source-bridge ring-group** global configuration command is used to define a ring group, and the **source-bridge remote peer** command with the **fst** option is used to associate the remote peer's IP address with the router's ring group and specify the remote peer's remote source-route bridging protocol version number. Because all FST peers support version 2 RSRB, the **version** keyword is always specified.

The configuration of the source-route bridge at 131.108.2.29 is as follows:

```
source-bridge fst-peername 131.108.2.29
source-bridge ring-group 5
source-bridge remote-peer 5 fst 131.108.1.27
!
interface ethernet 0
ip address 131.108.4.4 255.255.255.0
!
interface tokenring 0
ip address 131.108.2.29 255.255.255.0
source-bridge 1000 1 5
source-bridge spanning
!
interface tokenring 1
ip address 131.108.128.1 255.255.255.0
source-bridge 1001 1 5
source-bridge spanning
```

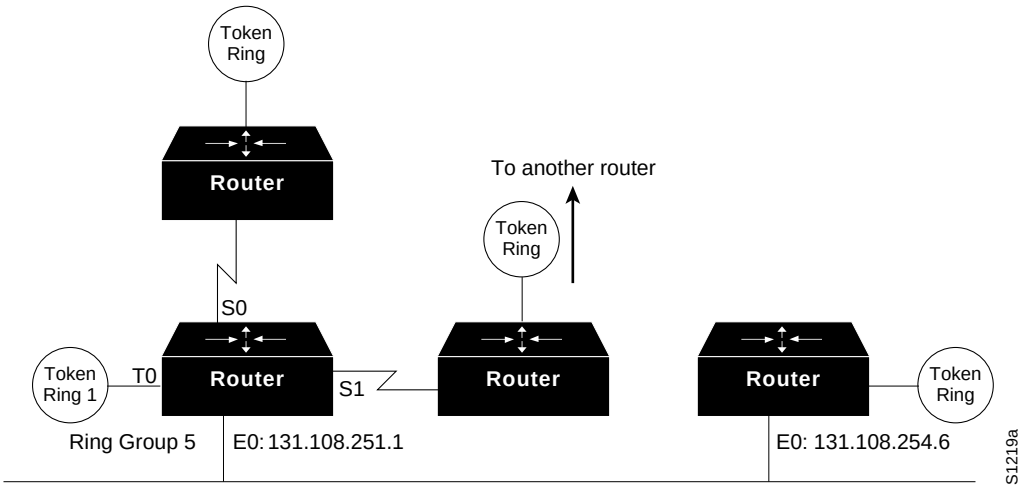
The configuration of the source-route bridge at 131.108.1.27 is as follows:

```
source-bridge fst-peername 131.108.1.27
source-bridge ring-group 5
source-bridge remote-peer 5 fst 131.108.2.29
!
interface ethernet 0
ip address 131.108.4.5 255.255.255.0
!
interface tokenring 0
ip address 131.108.1.27 255.255.255.0
source-bridge 10 1 5
source-bridge spanning
!
interface tokenring 1
ip address 131.108.131.1 255.255.255.0
source-bridge 11 1 5
source-bridge spanning
```

RSRB Using All Types of Transport Methods Example

Figure 23-21 shows a router/bridge configured for RSRB using all types of transport methods.

Figure 23-21 RSRB Using All Types of Transport Methods



The configuration for the network in Figure 23-21 is as follows:

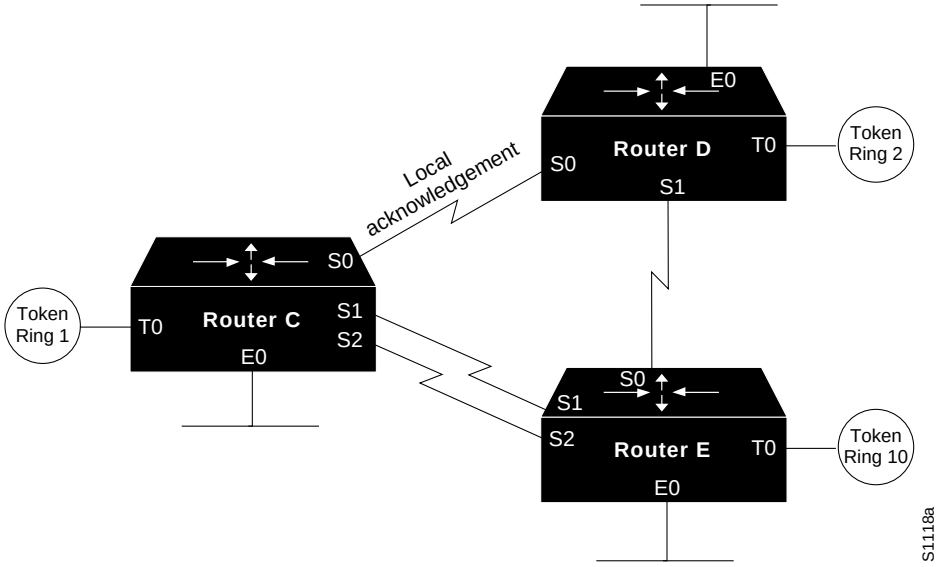
```
source-bridge fst-peername 131.108.251.1
source-bridge ring-group 5
source-bridge remote-peer 5 interface serial0
source-bridge remote-peer 5 interface serial1
source-bridge remote-peer 5 interface Ethernet0 0000.0c00.1234
source-bridge remote-peer 5 tcp 131.108.251.1
source-bridge remote-peer 5 fst 131.108.252.4
source-bridge remote-peer 5 tcp 131.108.253.5
!
interface tokenring 0
source-bridge 1 1 5
source-bridge spanning
!
interface ethernet 0
ip address 131.108.251.1 255.255.255.0
```

The two peers using the serial transport method will only function correctly if there are router/bridges at the other end of the serial line that have been configured to use the serial transport. The peers must also belong to the same ring group.

RSRB with Local Acknowledgment Example

In Figure 23-22, a triangular configuration is used to provide the maximum reliability with minimal cost. In addition, one of the links is doubled to gain better bandwidth. In addition to IP and SRB traffic, AppleTalk is also being routed between all the sites. Note that in this configuration, all the sessions between Router C and Router D are locally acknowledged. All the sessions between Router C and Router E are not locally acknowledged and are configured for normal remote source-route bridging. This example shows that not every peer must be locally acknowledged, but rather, that local acknowledgment can be turned on or off at the customer's discretion.

Figure 23-22 RSRB with Local Acknowledgment—Simple Configuration



The configuration files that enable this configuration follow.

Configuration for Router C

```
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1
source-bridge remote-peer 5 tcp 132.21.2.6 local-ack
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.1.1 255.255.255.0
source-bridge 1 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.4.25 255.255.255.0
appletalk address 4.25
appletalk zone Twilight
!
interface Serial 0
ip address 132.21.16.1 255.255.255.0
appletalk address 16.1
appletalk zone Twilight
!
interface Serial 1
ip address 132.21.17.1 255.255.255.0
appletalk address 17.1
appletalk zone Twilight
!
interface Serial 2
ip address 132.21.18.1 255.255.255.0
appletalk address 18.1
appletalk zone Twilight
!
router igrp 109
```

```
network 132.21.0.0
!
hostname RouterC
```

Configuration for Router D

```
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1 local-ack
source-bridge remote-peer 5 tcp 132.21.2.6
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.2.6 255.255.255.0
source-bridge 2 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.5.1 255.255.255.0
appletalk address 5.1
appletalk zone Twilight
!
interface Serial 0
ip address 132.21.16.2 255.255.255.0
appletalk address 16.2
appletalk zone Twilight
!
interface Serial 1
ip address 132.21.19.1 255.255.255.0
appletalk address 19.1
appletalk zone Twilight
!
router igrp 109
network 132.21.0.0
!
hostname RouterD
```

Configuration for Router E

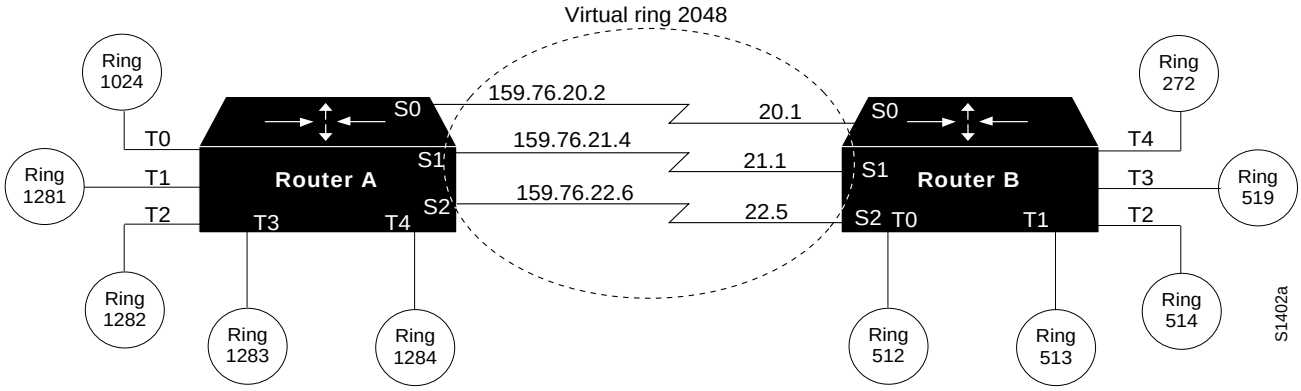
```
appletalk routing
!
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 132.21.1.1
source-bridge remote-peer 5 tcp 132.21.2.6
source-bridge remote-peer 5 tcp 132.21.10.200
!
interface TokenRing 0
ip address 132.21.10.200 255.255.255.0
source-bridge 10 1 5
source-bridge spanning
multiring all
!
interface Ethernet 0
ip address 132.21.7.1 255.255.255.0
appletalk address 7.1
appletalk zone Twilight
!
interface Serial 0
ip address 132.21.19.2 255.255.255.0
appletalk address 19.2
appletalk zone Twilight
```

```
!
interface Serial 1
ip address 132.21.17.2 255.255.255.0
appletalk address 17.2
appletalk zone Twilight
!
interface Serial 2
ip address 132.21.18.2 255.255.255.0
appletalk address 18.2
appletalk zone Twilight
!
router igrp 109
network 132.21.0.0
!
hostname RouterE
```

RSRB with Local Acknowledgment and Passthrough Example

Figure 23-23 shows two routers configured for remote source-route bridging with local acknowledgment and passthrough over the three serial lines that connect these routers. In turn, five Token Rings connect to each of these routers.

Figure 23-23 Network Topology for RSRB with Local Acknowledgment and Passthrough



The configuration files for each of these routers follows.

Configuration for Router A

```
source-bridge ring-group 2048
source-bridge remote-peer 2048 tcp 159.76.1.250 local-ack version 2
source-bridge remote-peer 2048 tcp 159.76.7.250 version 2
source-bridge passthrough 1281
source-bridge passthrough 1282
source-bridge passthrough 1283
source-bridge passthrough 1284
!
interface tokenring 0
ip address 159.76.7.250 255.255.255.0
llc2 ack-max 1
llc2 t1-time 1800
llc2 idle-time 29000
llc2 ack-delay-time 5
source-bridge 1024 1 2048
source-bridge spanning
```

```
early-token-release
multiring all
!
interface tokenring 1
ip address 159.76.8.250 255.255.255.0
clns-speed 4
clns mtu 464
source-bridge 1281 1 2048
source-bridge spanning
multiring all
interface tokenring 2
ip address 159.76.9.250 255.255.255.0
ring-speed 4
clns mtu 4464
source-bridge 1282 1 2048
source-bridge spanning
multiring all
interface tokenring 3
ip address 159.76.10.250 255.255.255.0
ring speed 4
clns mtu 4464
source-bridge 1283 1 2048
source-bridge spanning
multiring all
!
interface tokenring 4
ip address 159.78.11.250 255.255.255.0
ring speed 4
clns mtu 4464
source-bridge 1284 1 2048
source-bridge spanning
multiring all
!
interface serial 0
ip address 159.76.20.2 255.255.255.0
!
interface serial 1
ip address 159.76.21.4 255.255.255.0
!
interface serial 2
ip address 159.76.22.6 255.255.255.0
shutdown
!
interface serial 3
no ip address
shutdown
```

Configuration for Router B

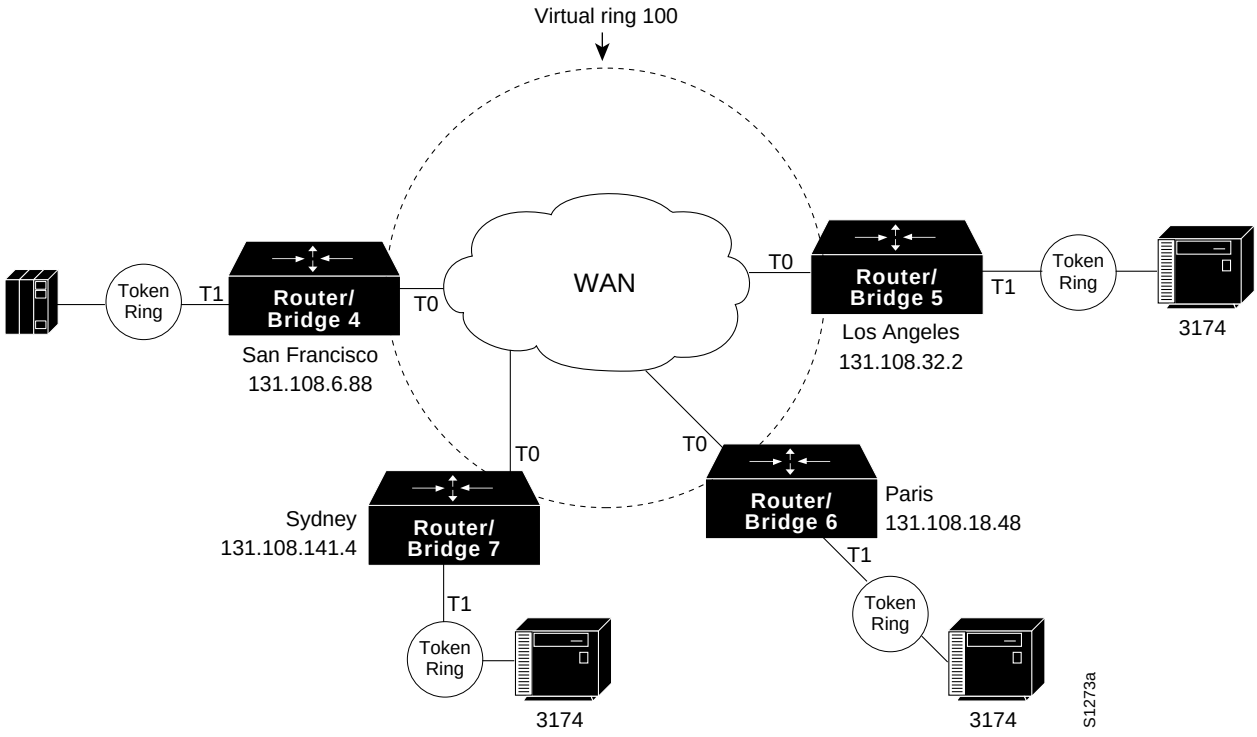
```
source-bridge ring-group 2048
source-bridge remote-peer 2048 tcp 159.76.1.250 version 2
source-bridge remote-peer 2048 tcp 159.76.7.250 local-ack version 2
!
interface tokenring 0
ip address 159.76.1.250 255.255.255.0
llc2 ack-max 2
llc2 t1-time 1900
llc2 idle-time 29000
llc2 ack-delay-time 5
source-bridge 512 1 2048
source-bridge spanning
early-token-release
multiring all
!
```

```
interface tokenring 1
ip address 159.76.2.250 255.255.255.0
ring-speed 16
clns mtu 8136
!
source-bridge 513 1 2048
source-bridge spanning
early-token-release
multiring all
!
interface tokenring 2
ip address 159.76.3.250 255.255.255.0
ring speed 16
clns mtu 8136
source-bridge 514 1 2048
source-bridge spanning
early-token-release
multiring all
!
interface tokenring 3
ip address 159.76.4.250 255.255.255.0
ring-speed 4
clns mtu 4464
source-bridge 519 2 2043
source-bridge spanning
multiring all
!
interface tokenring 4
ip address 159.76.5.250 255.255.255.0
ring-speed 4
clns mtu 4464
source-bridge 272 2 2048
source-bridge spanning
multiring all
!
interface tokenring
!
interface serial 0
ip address 159.76.20.1 255.255.255.0
!
interface serial 1
ip address 159.76.21.3 255.255.255.0
!
interface serial 2
ip address 159.76.22.5 255.255.255.0
!
interface serial 3
no ip address
shutdown
```

Local Acknowledgment for LLC2 Example

Figure 23-24 shows an IBM FEP located in San Francisco communicating with 3174 hosts in Sydney, Paris, and Los Angeles. The session between the FEP and the 3174 system in Los Angeles is not locally terminated, because the distance is great enough to cause timeouts on the line. However, the sessions to Paris and Sydney are locally terminated.

Figure 23-24 RSRB with Local Acknowledgment— Complex Configuration



The configuration described in this example is represented in the following sample configuration files.

Configuration for Router/Bridge 4 in San Francisco

```
source-bridge ring-group 100
! use direct encapsulation across serial link to Los Angeles
 source-bridge remote-peer 100 direct 131.108.32.2
 ! use fast sequenced transport with local termination to Paris
 source-bridge remote-peer 100 fst 131.108.18.48 local-ack
 ! use tcp encapsulation with local termination to Sydney
 source-bridge remote-peer 100 tcp 131.108.141.4 local-ack
 !
 interface tokenring 0
  ! source ring 1, bridge 4, destination ring 100
 source-bridge 1 4 100
 ! receive up to seven frames before sending an acknowledgment
 llc2 ack-max 7
 ! allow a 30 msec delay before I-frames must be acknowledged
 llc2 ack-delay-time 30
 !
 interface tokenring 1
```

```
 ! source ring 100, bridge 4, destination ring 1
 source-bridge 100 4 1
```

Configuration for Router/Bridge 7 in Sydney

```
source-bridge ring-group 100
 ! use tcp encapsulation with local termination from Sydney
 source-bridge remote-peer 100 tcp 131.108.6.88 local-ack
 interface tokenring 0
 ! source ring 1, bridge 7, destination ring 100
 source-bridge 1 7 100
 ! receive up to seven frames before sending an acknowledgment
 llc2 ack-max 7
 ! allow a 30 msec delay before I-frames must be acknowledged
 llc2 ack-delay-time 30
 !
 interface tokenring 1
 ! source ring 100, bridge 7, destination ring 1
 source-bridge 100 7 1
```

Configuration for Router/Bridge 6 in Paris

```
source-bridge ring-group 100
 ! use fast sequenced transport with local termination from Paris
 source-bridge remote-peer 100 fst 131.108.6.88 local-ack
 interface tokenring 0
 ! source ring 1, bridge 6, destination ring 100
 source-bridge 1 6 100
 ! receive up to seven frames before sending an acknowledgment
 llc2 ack-max 7
 ! allow a 30 msec delay before I-frames must be acknowledged
 llc2 ack-delay-time 30
 !
 interface tokenring 1
 ! source ring 100, bridge 6, destination ring 1
 source-bridge 100 6 1
```

Configuration for Router/Bridge 5 in Los Angeles

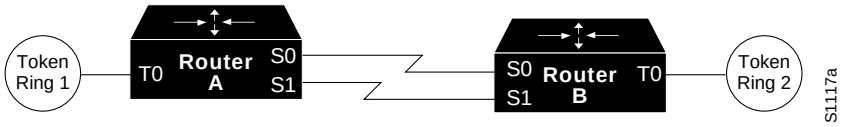
```
source-bridge ring-group 100
 ! use direct encapsulation across serial link from Los Angeles
 source-bridge remote-peer 100 direct 131.108.6.88

 interface tokenring 0
 ! source ring 1, bridge 5, destination ring 100
 source-bridge 1 5 100
 ! receive up to seven frames before sending an acknowledgment
 llc2 ack-max 7
 ! allow a 30 msec delay before I-frames must be acknowledged
 llc2 ack-delay-time 30
 !
 interface tokenring 1
 ! source ring 100, bridge 5, destination ring 1
 source-bridge 100 5 1
```

Note Both peers need to be configured for LLC2 local acknowledgment. If only one is so configured, unpredictable (and undesirable) results will occur.

IP for Load Sharing over RSRB Example

As Figure 23-25 shows, two routers are connected by two serial lines. Each has been configured as a basic remote dual-port bridge, but extended to include both reliability and IP load sharing. When both serial lines are up, traffic is split between them, effectively combining the bandwidth of the connections. If either one of the serial lines goes down, all traffic is routed to the remaining line with no disruption. This happens transparently with respect to the end connections, unlike other source-route bridges that would abort those connections.

Figure 23-25 RSRB—Simple Reliability



The sample configuration files that enable this configuration follow.

Configuration for Router/Bridge A

```
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 204.31.7.1
source-bridge remote-peer 5 tcp 204.31.8.1
!
interface TokenRing 0
ip address 204.31.7.1 255.255.255.0
source-bridge 1 1 5
source-bridge spanning
multiring all
!
interface Serial 0
ip address 204.31.9.1 255.255.255.0
!
interface Serial 1
ip address 204.31.10.1 255.255.255.0
!
router igrp 109
network 204.31.7.0
network 204.31.9.0
network 204.31.10.0
!
hostname RouterA
```

Configuration for Router/Bridge B

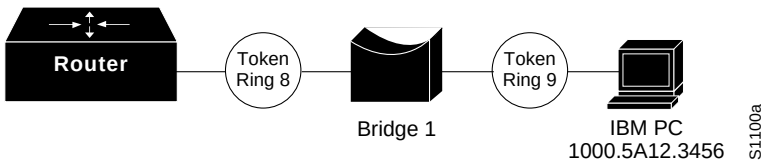
```
source-bridge ring-group 5
source-bridge remote-peer 5 tcp 204.31.7.1
source-bridge remote-peer 5 tcp 204.31.8.1
!
interface TokenRing 0
ip address 204.31.8.1 255.255.255.0
source-bridge 2 1 5
source-bridge spanning
multiring all
!
interface Serial 0
ip address 204.31.9.2 255.255.255.0
!
```

```
interface Serial 1
ip address 204.31.10.2 255.255.255.0
!
router igrp 109
network 204.31.8.0
network 204.31.9.0
network 204.31.10.0
!
hostname RouterB
```

Adding a Static RIF Cache Entry Example

In the example configuration in Figure 23-26, the path between rings 8 and 9 connected via SRB 1 is described by the route descriptor 0081.0090. A full RIF, including the route control field, would be 0630.0081.0090.

Figure 23-26 Assigning a RIF to a Source-Route Bridge



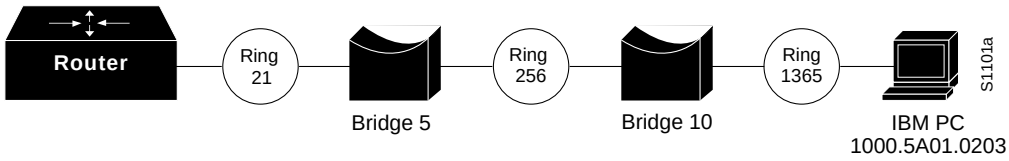
The static RIF entry would be submitted to the leftmost router as follows:

```
rif 1000.5A12.3456 0630.0081.0090
```

Adding a Static RIF Cache Entry for a Two-Hop Path Example

In Figure 23-27, assume that a datagram was sent from a router/bridge on ring 21 (15 hexadecimal), across bridge 5 to ring 256 (100 hexadecimal), and then across bridge 10 (A hexadecimal) to ring 1365 (555 hexadecimal) for delivery to a destination host on that ring.

Figure 23-27 Assigning a RIF to a Two-Hop Path



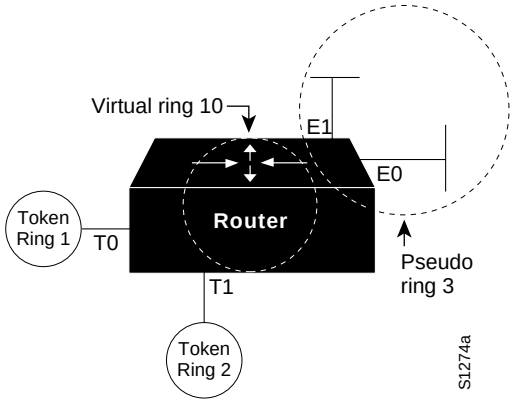
The RIF in the router on the left describing this two-hop path is 0830.0155.100a.5550 and is entered as follows:

```
rif 1000.5A01.0203 0830.0155.100a.5550
```

SR/TLB for a Simple Network Example

In the simple example illustrated in Figure 23-28, a four-port router with two Ethernets and two Token Rings is used to connect transparent bridging on the Ethernets to SRB on the Token Rings.

Figure 23-28 Example of a Simple SR/TLB Configuration



Assume that the following configuration for SRB and transparent bridging existed before you wanted to enable SR/TLB:

```
interface tokenring 0
source-bridge 1 1 2
!
interface tokenring 1
source-bridge 2 1 1
!
interface Ethernet 0
bridge-group 1
!
interface Ethernet 0
bridge-group 1
!
bridge 1 protocol dec
```

In order to enable SR/TLB, one aspect of this configuration must change immediately—a third ring must be configured. Before SR/TLB, the two Token Ring interfaces were communicating with two-port local source-route bridging; after SR/TLB, these two interfaces must be reconfigured to communicate through a virtual ring, as follows:

```
source-bridge ring-group 10
!
interface tokenring 0
source-bridge 1 1 10
!
interface tokenring 1
source-bridge 2 1 10
!
interface ethernet 0
bridge-group 1
!
interface ethernet 1
bridge-group 1
!
bridge 1 protocol dec
```

Now you are ready to determine two things:

- A ring number for the pseudo-ring that is unique throughout the source-route bridged network. For the preceding example configuration, use a 3.
- A bridge number for the path to the pseudo-ring. For the preceding example configuration, use a 1.

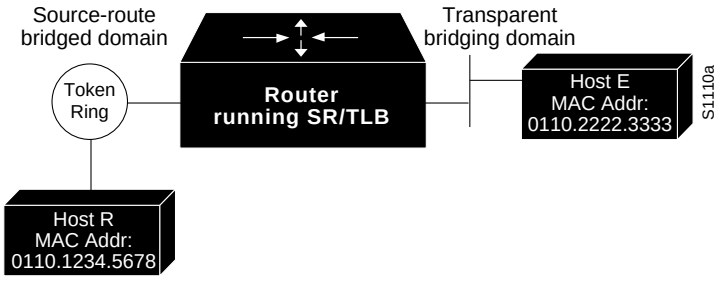
Once you have determined the ring number and the bridge number, you can add the **source-bridge transparent** command to the file, including these two values as parameters for the command. The following partial configuration includes this **source-bridge transparent** entry:

```
!
source-bridge ring-group 10
source-bridge transparent 10 3 1 1
!
interface tokenring 0
source-bridge 1 1 10
!
interface tokenring 1
source-bridge 2 1 10
!
interface ethernet 0
bridge-group 1
!
interface ethernet 1
bridge-group 1
!
bridge 1 protocol dec
```

SR/TLB with Access Filtering Example

In the example shown in Figure 23-29, you want to connect only a single machine, Host E, on an Ethernet to a single machine, Host R, on the Token Ring.

Figure 23-29 Example of a Bit-Swapped Address



You want to allow only these two machines to communicate across the router. Therefore, you might create the following configuration to restrict the access. However, this configuration will not work, as explained in the paragraph following the sample configuration file.

Note For the sake of readability, the commands to control the bridging are not shown here, just the commands to control the filtering.

```
interface tokenring 0
access-expression output smac(701)
!
interface ethernet 0
bridge-group 1 input-address-list 701
!
access-list 701 permit 0110.2222.3333
```

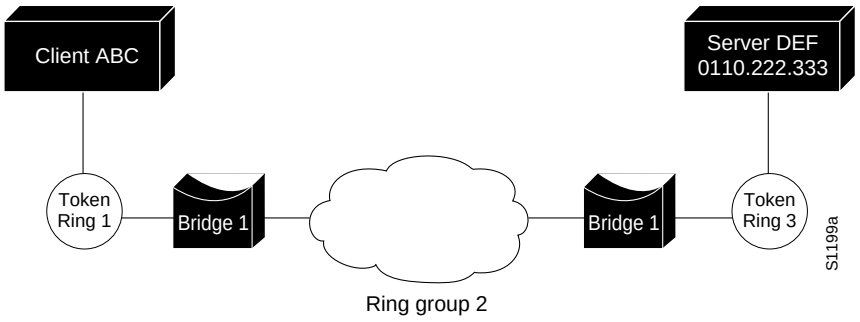
The command for the Token Ring interface specifies that the access list 701 be applied on the source address of frames going out to the Token Ring, and the command for the Ethernet interface specifies that this access list be applied on the source address frames entering the interface from Ethernet. This would work if both interfaces used the same bit ordering, but Token Rings and Ethernets use opposite (swapped) bit orderings in their addresses in relationship to each other. Therefore, the address of Host E on the Token Ring is not 0110.2222.3333, but rather 8008.4444.cccc, resulting in the following configuration. The following configuration is better. This example shows that access lists for Token Ring and Ethernet should be kept completely separate from each other.

```
interface tokenring 0
source-bridge input-address-list 702
!
interface ethernet 0
bridge-group 1 input-address-list 701
!
access-list 701 permit 0110.2222.3333
!
access-list 702 permit 0110.1234.5678
```

NetBIOS Support with a Static NetBIOS Cache Entry Example

Figure 23-30 shows a NetBIOS client on a Token Ring connected through a cloud to a NetBIOS server on another Token Ring.

Figure 23-30 Specifying a Static Entry



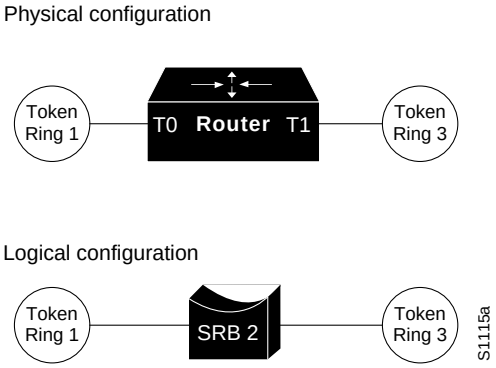
In Figure 23-30, a static entry is created in the router attached to ring 1 on the client side of the ring group. The static entry is to the server DEF, which is reached through the router attached to ring 3. If server DEF has the MAC address 0110.2222.3333, the configuration for the static entry on the client side is as follows:

```
rif 0110.2222.3333 0630.0021.0030 ring-group 2
netbios name-cache 0110.2222.3333 DEF ring-group 2
```

LNM for a Simple Network Example

Figure 23-31 shows a router with two Token Rings configured as a local source-route bridge.

Figure 23-31 Router with Two Token Rings Configured as a Local Source-Route Bridge



The associated configuration file follows:

```
interface TokenRing 0
source-bridge 1 2 3
!
interface TokenRing 1
source-bridge 3 2 1
```

The **show lnm config** command displays the logical configuration of this bridge, including the LNM configuration information that needs to be entered at the LNM Station. A sample **show lnm config** display follows:

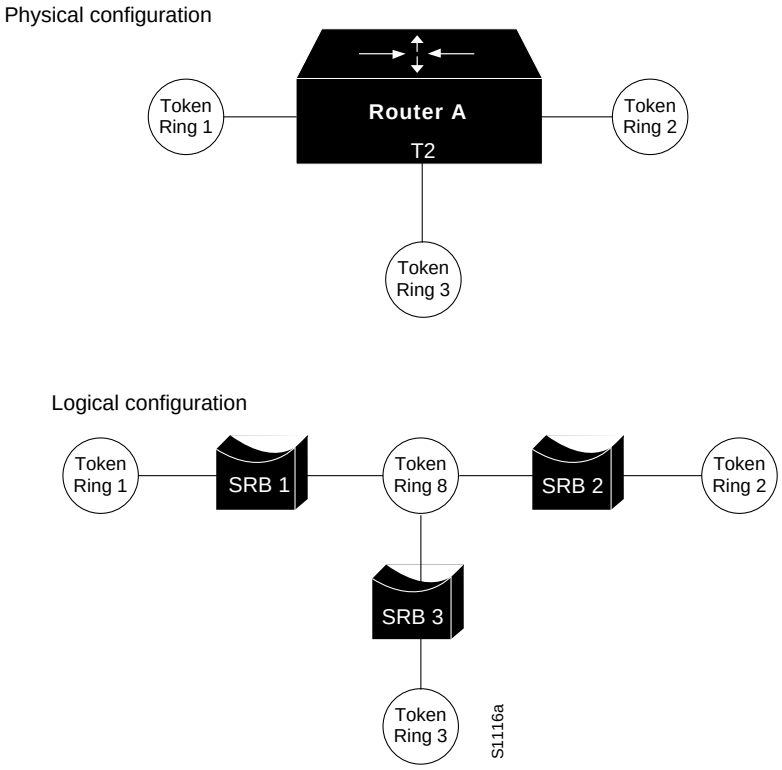
```
Wayfarer# show lnm config

Bridge(s) currently configured:
From     ring 001, address 0000.3000.abc4
Across bridge 002
To       ring 003, address 0000.3000.5735
```

In this example, the MAC addresses 0000.3000.abc4 and 000.3000.5735 must be configured as Adapter Addresses at the LNM Station.

LNM for a More Complex Network Example

Figure 23-32 shows a router with three Token Rings configured as a multiport bridge, thus employing the concept of the virtual ring.

Figure 23-32 Router with Three Token Rings Configured as a Multiport Bridge



The associated configuration file follows.

```
source-bridge ring-group 8
!
interface TokenRing 0
source-bridge 1 1 8
!
interface TokenRing 1
source-bridge 2 2 8
!
interface TokenRing 2
source-bridge 3 3 8
```

The **show lnm config** command displays the logical configuration of this bridge, including all the pertinent information for configuring this router into LNM:

```
Wayfarer# show lnm config
Bridge(s) currently configured:

           From     ring 001, address 0000.0028.abcd
           Across bridge 001
           To       ring 008, address 4000.0028.abcd

           From     ring 002, address 0000.3000.abc4
           Across bridge 002
```

```
To       ring 008, address 4000.3000.abc4

From     ring 003, address 0000.3000.5735
Across bridge 003
To       ring 008, address 4000.3000.5735
```

In this example, six station definitions must be entered at the LNM Station, one for each of the MAC addresses listed in this sample **show lnm config** display.

NetBIOS Access Filters Example

The following command permits packets that include the station name ABCD to pass through the router, but denies passage to packets that do not include the station name ABCD:

```
netbios access-list host marketing permit ABCD
```

The following command specifies a prefix where the pattern matches any name beginning with the characters DEFG. Note that the string DEFG itself is included in this condition.

```
netbios access-list host marketing deny DEFG*
```

The following command permits any station name with the letter W as the first character and the letter Y as the third character in the name. The second and fourth letters in the name can be any character. This example would allow stations named WXYZ and WAYB; however, stations named WY and WXY would not be included in this statement, because the question mark must match some specific character in the name.

```
netbios access-list host marketing permit W?Y?
```

The following command illustrates how to combine wildcard characters:

```
netbios access-list host marketing deny AC?*
```

The command specifies that the marketing list deny any name beginning with AC that is at least three characters in length (the question mark would match any third character). The string ACBD and ACB would match, but the string AC would not.

The following command removes the entire marketing NetBIOS access list.

```
no netbios access-list host marketing
```

To remove single entries from the list, use a command such as the following:

```
no netbios access-list host marketing deny AC?*
```

This example removes only the list that filters station names with the letters AC at the beginning of the name.

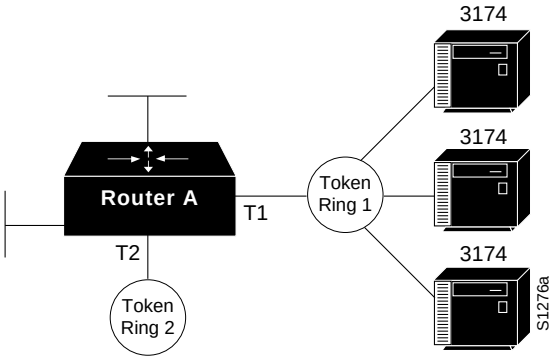
Keep in mind that the access lists are scanned in order. In the following example, the first list denies all entries beginning with the letters ABC, including one named ABCD. This voids the second command, because the entry permitting a name with ABCD comes after the entry denying it.

```
netbios access-list host marketing deny ABC*
netbios access-list host marketing permit ABCD
```

Filtering Bridged Token Ring Packets to IBM Machines Example

The example in Figure 23-33 disallows the bridging of Token Ring packets to all IBM workstations on Token Ring 1.

Figure 23-33 Router Filtering Bridged Token Ring Packets to IBM Machines



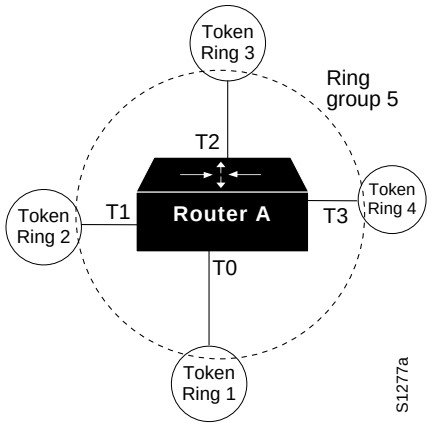
This example assumes that all hosts on Token Ring 1 have Token Ring addresses with the vendor code 1000.5A00.0000. The first line of the access list denies access to all IBM workstations, while the second line permits everything else. Then, the access list is assigned to the input side of Token Ring 1.

```
! deny access to all IBM workstations
access-list 700 deny 1000.5A00.0000   8000.00FF.FFFF
! permit all other traffic
access-list 700 permit 0000.0000.0000   FFFF.FFFF.FFFF
!
interface token ring 1
! apply access list 700 to the input side of Token Ring 1
source-bridge input-address-list 700
```

Administrative Access Filters—Filtering SNAP Frames on Output Example

Figure 23-34 shows a router connecting four Token Rings.

Figure 23-34 Router Filtering SNAP Frames on Output



The following example allows only AppleTalk Phase 2 packets to be source-route bridged between Token Rings 0 and 1, and allows Novell packets only to be source-route bridged between Token Rings 2 and 3.

```
source-bridge ring-group 5
```

```
!
interface tokenring 0
ip address 131.108.1.1 255.255.255.0
source-bridge 1000 1 5
source-bridge spanning
source-bridge input-type-list 202
!
interface tokenring 1
ip address 131.108.11.1 255.255.255.0
source-bridge 1001 1 5
source-bridge spanning
source-bridge input-type-list 202
!
interface tokenring 2
ip address 131.108.101.1 255.255.255.0
source-bridge 1002 1 5
source-bridge spanning
source-bridge input-lsap-list 203
!
interface tokenring 3
ip address 131.108.111.1 255.255.255.0
source-bridge 1003 1 5
source-bridge spanning
source-bridge input-lsap-list 203
!
! SNAP type code filtering
! permit ATp2 data (0x809B)
! permit ATp2 AARP (0x80F3)
access-list 202 permit 0x809B 0x0000
access-list 202 permit 0x80F3 0x0000
access-list 202 deny 0x0000 0xFFFF
!
! LSAP filtering
! permit IPX (0xE0E0)
access-list 203 permit 0xE0E0 0x0101
access-list 203 deny 0x0000 0xFFFF
```

Note that it is not necessary to check for an LSAP of 0xAAAA when filtering SNAP-encapsulated AppleTalk packets, because for source-route bridging, the use of type filters implies SNAP encapsulation.

Creating Access Expressions Example

In math, you have the following:

3 * 4 + 2 = 14 but 3 * (4 + 2) = 18

Similarly, the following access expressions would return TRUE if lsap(201) and dmac(701) returned TRUE or if smac(702) returned TRUE:

lsap(201) & dmac(701) | smac(702)

However, the following access expression would return TRUE only if lsap(201) returned TRUE and either of dmac(701) or smac(702) returned TRUE:

lsap(201) & (dmac(701) | smac(702))

Referring to the earlier example, “An Example Using NetBIOS Access Filters,” we had the phrase:

“Pass the frame if it is NetBIOS, or if it is an SNA frame destined to address 0110.2222.3333.”

This phrase was converted to the simpler form of:

Pass if “NetBIOS or (SNA and destined to 0110.2222.3333).”

So, for the following configuration:

```
! Access list 201 passes NetBIOS frames (command or response)
access-list 201 permit 0xF0F0 0x0001
!
access-list 202 permit 0x0404 0x0001 ! Permits SNA frames (command or response)
access-list 202 permit 0x0004 0x0001 ! Permits SNA Explorers with NULL DSAP
!
! Access list 701 will permit the FEP MAC address
! of 0110.2222.3333
access-list 701 permit 0110.2222.3333
```

The following access expression would result:

```
access-expression in lsap(201) | (lsap(202) & dmac(701))
```

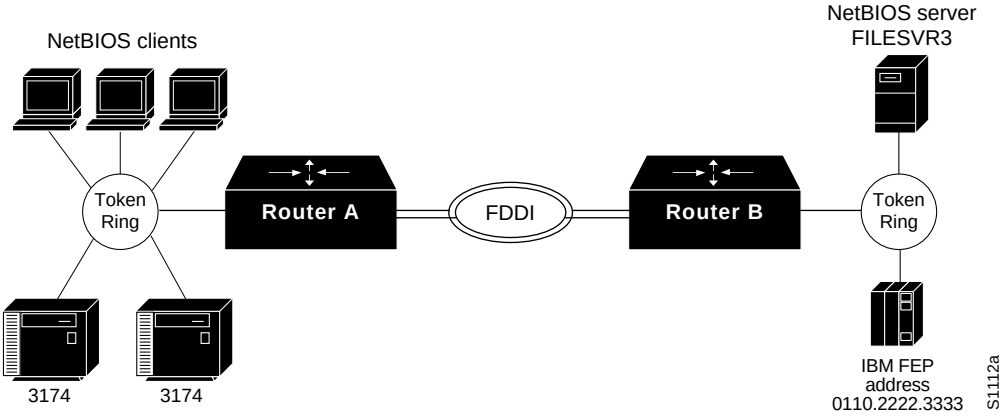
Therefore, the full configuration example is as follows:

```
interface tokenring 0
access-expression in lsap(201 | (lsap(202) & dmac(701))
!
! Access list 201 passes NetBIOS frames (command or response
access-list 201 permit 0xF0F0 0x0001
!
access-list 202 permit 0x0404 0x0001 ! Permits SNA frames (command or response)
access-list 202 permit 0x0004 0x0001 ! Permits NSA Explorers with NULL DSAP
!
! Access list 701 will permit the FEP MAC address
! of 0110.2222.3333
access-list 701 permit 0110.2222.3333
```

Access Expressions Example

Figure 23-35 shows two routers connecting two Token Rings to an FDDI backbone.

Figure 23-35 Network Configuration Using NetBIOS Access Filters



Suppose you want to permit the IBM 3174s to access the FEP at address 0110.2222.3333, and also want the NetBIOS clients to access the NetBIOS server named FILESVR3. The following set of router configuration commands would meet this need:

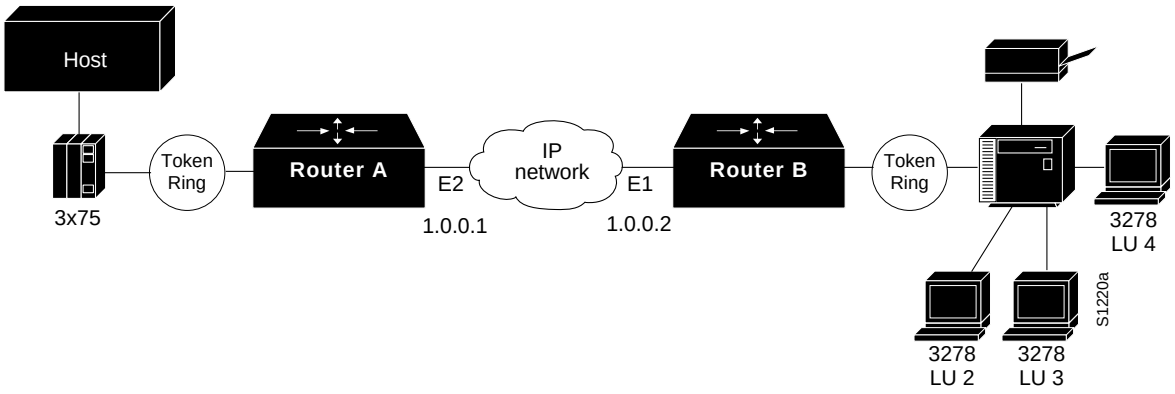
```
netbios access-list host MIS permit FILESVR3
netbios access-list host MIS deny *
!
```

```
access-list 202 permit 0x0404 0x0001 ! Permits SNA frames (command or response)
access-list 202 permit 0x0004 0x0001 ! Permits SNA Explorers with NULL DSAP
!
access-list 701 permit 0110.2222.3333
!
interface tokenring 0
access-expression in (lsap(202) & dmac(701)) | netbios-host(MIS)
```

Configuring Priority for Locally Terminated Token Ring Interfaces in RSRB Example

Figure 23-36 shows a network that uses RSRB to bridge Token Ring traffic.

Figure 23-36 RSRB Configuration Example



The configuration for the network shown in Figure 23-36 follows.

Configuration for Router/Bridge A

```
source-bridge ring-group 2624
source-bridge remote-peer 2624 tcp 1.0.0.1
source-bridge remote-peer 2624 tcp 1.0.0.2 local-ack priority
!
interface TokenRing 0
source-bridge 2576 8 2624
source-bridge spanning
multiring all
locaddr-priority 1
!
interface Ethernet 0
ip address 1.0.0.1 255.255.255.0
priority-group 1
!
locaddr-priority-list 1 02 high
locaddr-priority-list 1 03 high
locaddr-priority-list 1 04 medium
locaddr-priority-list 1 05 low
!
priority-list protocol ip high tcp 1996
priority-list protocol ip medium tcp 1987
priority-list protocol ip normal tcp 1988
priority-list protocol ip low tcp 1989
```

Configuration for Router/Bridge B

```
source-bridge ring-group 2624
source-bridge remote-peer 2624 tcp 1.0.0.2
source-bridge remote-peer 2624 tcp 1.0.0.1 local-ack priority
!
interface TokenRing 0
source-bridge 2626 8 2624
source-bridge spanning
multiring all
locaddr-priority 1
!
interface Ethernet 0
ip address 1.0.0.2 255.255.255.0
priority-group 1
!
locaddr-priority-list 1 02 high
locaddr-priority-list 1 03 high
locaddr-priority-list 1 04 medium
locaddr-priority-list 1 05 low
!
priority-list protocol ip high tcp 1996
priority-list protocol ip medium tcp 1987
priority-list protocol ip normal tcp 1988
priority-list protocol ip low tcp 1989
```

Fast Switching Example

The following example disables fast switching between two Token Ring interfaces in the same router/bridge:

```
! global command establishing the ring group for the interface configuration commands
source-bridge ring-group 2
!
! commands that follow apply to interface token 0
interface token 0
! enable srb between local ring 1, bridge 1, and target ring 2
source-bridge 1 1 2
!disable source-route fast-switching cache on interface token 0
no source-bridge route-cache
!
interface token 1
! enable srb between local ring 2, bridge 1, and target ring 1
source-bridge 2 1 1
no source-bridge route-cache
```

Frames entering Token Ring interfaces 0 or 1 will not be fast switched to the other interface.

Autonomous Switching Example

The following example enables use of autonomous switching between two ciscoBus Token Ring interfaces in the same router/bridge:

```
! global command to apply interface configuration commands to the ring group
source-bridge ring-group 2
!
! commands that follow apply to interface token 0
interface token 0
! enable srb between local ring 1, bridge 1, and target ring 2
source-bridge 1 1 2
! enable autonomous switching for interface token 0
source-bridge route-cache cbus
!
```

```
interface token 1
! enable srb between local ring 2, bridge 1, and target ring 1
source-bridge 2 1 1
source-bridge route-cache cbus
```

Frames entering interface Token Ring interfaces 0 or 1 will be autonomously switched to the other interface.

SNA Traffic Prioritization by LU Address Example

The following example enables SNA traffic prioritization by LU address:

```
locaddr-priority-list 1 01 medium
locaddr-priority-list 1 02 normal
locaddr-priority-list 1 03 low
locaddr-priority-list 1 04 high
!
priority-list 2 protocol ip low tcp 1996
priority-list 2 protocol ip high tcp 1987
priority-list 2 protocol ip medium tcp 1988
priority-list 2 protocol ip normal tcp 1989
!
interface tokenring 0
source-bridge 123
locaddr-priority 1

interface serial 0
priority-group 2
```

CHAPTER 24

Configuring STUN

Cisco's serial tunnel (STUN) implementation allows Synchronous Data Link Control (SDLC) devices and High-Level Data Link Control (HDLC) devices to connect to one another through a multiprotocol internetwork.

This chapter describes the STUN features and lists the tasks you must perform to configure a STUN network in either passthrough or local acknowledgment mode. For a complete description of the commands mentioned in this chapter, refer to the "STUN Commands" chapter in the *Router Products Command Reference* publication.

Note The use of Software Release 9.0 or earlier is discouraged. If you have Software Release 9.0 or earlier, you can enable STUN in passthrough mode only. In lieu of local acknowledgment, the proxy-polling feature is available. However, the functions provided by proxy polling have been enhanced and superseded by the STUN local acknowledgment feature and the use of proxy polling is no longer supported.

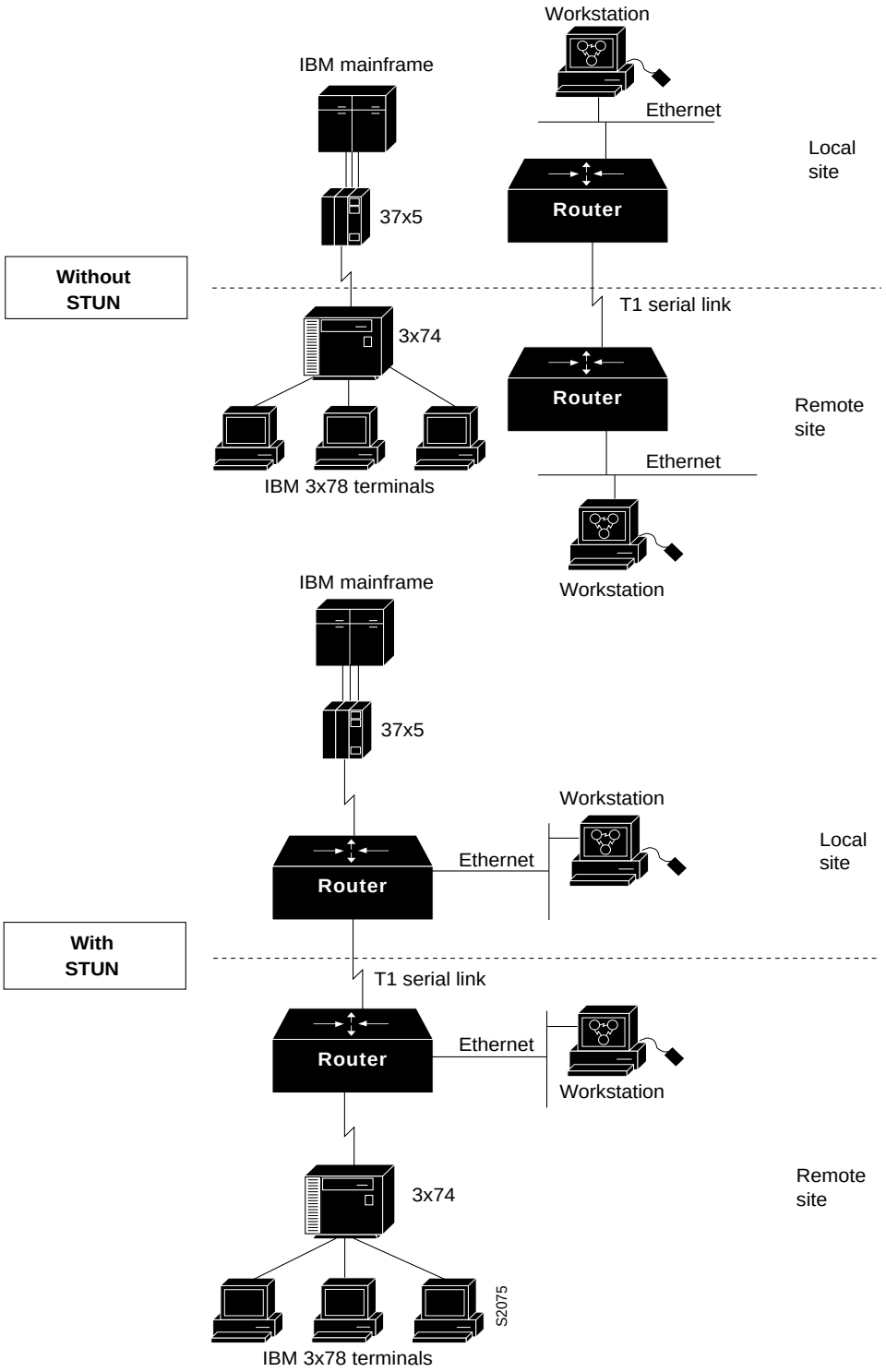
Cisco's Implementation of Serial Tunneling

Our STUN implementation provides the following features:

- Encapsulates SDLC frames in either the Transmission Control Protocol/Internet Protocol (TCP/IP) or the HDLC protocol.
- Allows two devices using SDLC- or HDLC-compliant protocols that are normally connected by a direct serial link to be connected through one or more Cisco routers, reducing leased-line costs.

  When you replace direct serial links with routers, serial frames can be propagated over arbitrary media and topologies to another router with a STUN link to an appropriate end point. The intervening network is not restricted to STUN traffic, but rather, is multiprotocol. For example, instead of running parallel backbones for DECnet and SNA/SDLC traffic, this traffic now can be integrated into an enterprise backbone network.
- Allows networks with IBM mainframes and communications controllers to share data using Cisco routers and existing network links. As an SDLC function, STUN fully supports the IBM Systems Network Architecture (SNA), and allows IBM SDLC frames to be transmitted across the network media and and/or shared serial links. Figure 24-1 illustrates a typical network configuration with and without STUN.
- Encapsulates SDLC frame traffic packets and routes them over any of the supported network media—serial, Fiber Distributed Data Interface (FDDI), Ethernet, and Token Ring, X.25, Switched Multimegabit Data Service (SMDS), and T1/T3—using TCP/IP encapsulation. Because TCP/IP encapsulation is used, you can use any of the Cisco routing protocols to route the packets.
- Copies frames to destinations based on address. STUN in passthrough mode does not modify the frames in any way or participate in SDLC windowing or retransmission; these functions are left to the communicating hosts. However, STUN in local acknowledgment mode does participate in SDLC windowing and retransmission through local termination of the SDLC session.
- Ensures reliable data transmission across serial media having minimal or predictable time delays. With the advent of STUN and wide-area network (WAN) backbones, serial links now can be separated by wide, geographic distances spanning countries and continents. As a result, these serial links have time delays that are longer than SDLC allows for bidirectional communication between hosts. The STUN local acknowledgment feature addresses the problems of unpredictable time delays, multiple retransmissions, or loss of sessions.
- Provides for configuration of redundant links to provide transport paths in the event part of the network goes down.

Figure 24-1 IBM Network Configuration with and without STUN



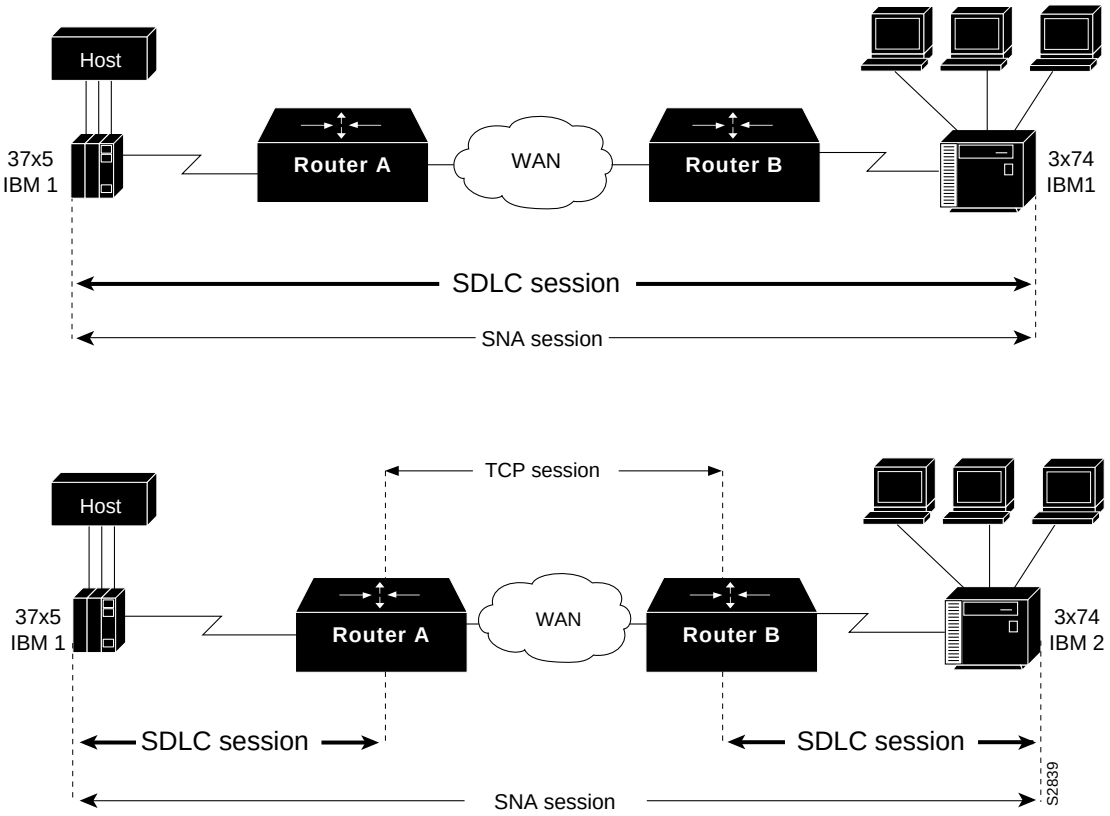
The STUN Network

STUN operates in two modes: passthrough and local acknowledgment. Figure 24-2 shows the difference between passthrough mode and local acknowledgment mode.

The upper half of Figure 24-2 shows STUN configured in passthrough mode. In passthrough mode, the routers act as a wire and the SDLC session remains between the end stations. In this mode, STUN provides a straight pass-through of all SDLC traffic, including control frames.

The lower half of Figure 24-2 shows STUN configured in local acknowledgment mode. In local acknowledgment mode, the routers terminate the SDLC sessions and send only data across the WAN. Control frames no longer travel the WAN backbone networks.

Figure 24-2 Comparison of STUN in Passthrough Mode and Local Acknowledgment Mode



Note To enable STUN local acknowledgment, routers first must be enabled for STUN and configured to appear on the network as primary or secondary SDLC nodes. TCP/IP encapsulation must be enabled. Our STUN local acknowledgment feature also provides priority queuing for TCP-encapsulated frames.

STUN Configuration Task List

To configure and monitor STUN, or STUN Local Acknowledgment, complete the tasks in the following sections:

- Enable STUN
- Configure SDLC Broadcast
- Specify a STUN Protocol Group
- Enable STUN Interfaces and Place in STUN Group
- Establish the Frame Encapsulation Method
- Configure STUN with Multilink Transmission Groups
- Set up Traffic Priorities
- Monitor STUN Network Activity

See the end of the chapter for "STUN Configuration Examples".

Enable STUN

Perform the following task in global configuration mode to enable STUN:

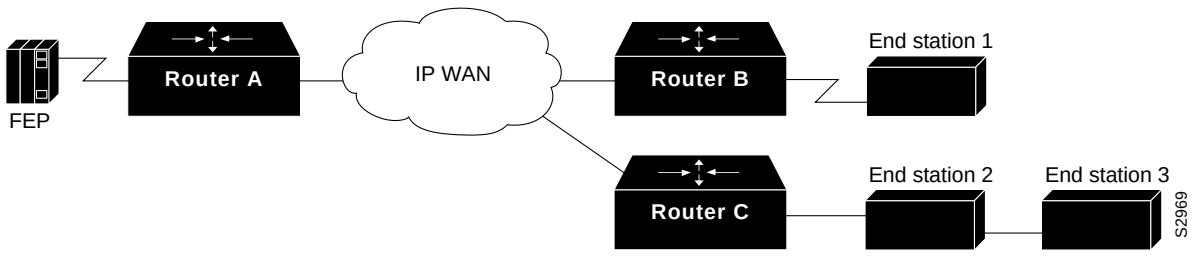
Task	Command
Enable STUN for a particular IP address.	**stun peer-name** *ip-address*

When configuring redundant links, ensure that the STUN peer names you choose on each router are the IP addresses of the most stable interfaces on each router, such as a loopback or Ethernet interface. See "STUN Configuration Examples" later in this chapter.

Configure SDLC Broadcast

The SDLC broadcast feature allows SDLC broadcast address FF to be replicated for each of the STUN peers, so each of the end stations receives the broadcast frame. For example, in Figure 24-3, the FEP views the end stations 1, 2 and 3 as if they are on an SDLC multidrop link. Any broadcast frame sent from FEP to Router A is duplicated and sent to each of the downstream routers (B and C).

Figure 24-3 SDLC Broadcast across Virtual Multidrop Lines



To enable SDLC broadcast, perform the following task in interface configuration mode:

Task	Command
Enable SDLC broadcast.	**sdlc virtual-multidrop**

Only enable SDLC broadcast on the router that is configured to be the secondary station on the SDLC link (Router A in Figure 23-3).

You must also configure SDLC address FF on Router A for each of the STUN peers. To do so, perform the following task in global configuration mode:

Task	Command
Configure SDLC address FF on Router A for each STUN peer.	**stun route address** *address-number* **tcp** *ip-address* [**local-ack**] [**priority**] [**tcp-queue-max**]

Specify a STUN Protocol Group

Each STUN interface must be placed in a group that defines the ISO 3309-compliant framed protocol running on that link. Packets will only travel between STUN interfaces that are in the same protocol group.

There are three predefined STUN protocols:

- Basic
- SDLC
- SDLC transmission group

You also can specify a custom STUN protocol.

You must specify either the SDLC protocol or the SDLC transmission group protocol if you want to use the STUN Local Acknowledgment feature.

Note Before you can specify a custom protocol, you must first define the protocol; see the section "Create and Specify a Custom STUN Protocol" later in this chapter for the procedure.

Specify a Basic STUN Group

The basic STUN protocol is unconcerned with details of serial protocol addressing and is used when addressing is unimportant. Use this when your goal with STUN is to replace one or more sets of point-to-point (not multidrop) serial links by using a protocol other than SDLC. Perform the following task in global configuration mode:

Task	Command
Specify a basic protocol group and assign a group number.	**stun protocol-group** *group-number* **basic**

Specify an SDLC Group

You can specify SDLC protocol groups to associate interfaces with the SDLC protocol. The SDLC STUN protocol is used for placing the routers in the midst of either point-to-point or multipoint (multidrop) SDLC links. To define an SDLC protocol group, perform the following task in global configuration mode:

Task	Command
Specify an SDLC protocol group and assign a group number.	**stun protocol-group** *group-number* **sdlc**

If you specify an SDLC protocol group, you cannot specify the **stun route all** command on any interface of that group.

For an example of how to configure an SDLC protocol group, see "Example of Configuring Serial Link Address Prioritization using STUN TCP/IP Encapsulation" later in this chapter.

Specify an SDLC Transmission Group

An SNA transmission group is a set of lines providing parallel links to the same pair of SNA front-end-processor (FEP) devices. This provides redundancy of paths for fault tolerance and load sharing. To define an SDLC transmission group, perform the following task in global configuration mode:

Task	Command
Specify an SDLC protocol group, assign a group number, and create an SNA transmission group.	**stun protocol-group** *group-number* **sdlc-tg**

All STUN connections in a transmission group must connect to the same IP address and use the SDLC local acknowledgment feature.

For an example of how to configure a transmission group, see "Example of Configuring Transmission Groups" later in this chapter.

Create and Specify a Custom STUN Protocol

The STUN implementation allows you to create your own STUN protocols. To define a custom protocol and tie STUN groups to the new protocol, perform the following tasks in global configuration mode:

Task		Command
Step 1	Create a custom protocol.	**stun schema** *name* **offset** *constant-offset* **length** *address-length* **format** *format-keyword*
Step 2	Specify the custom protocol group and assign a group number.	**stun protocol-group** *group-number* **schema**

Enable STUN Interfaces and Place in STUN Group

You must enable STUN on serial interfaces and place these interfaces in the protocol groups you have defined. To enable STUN on an interface and to place the interface in a STUN group, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Enable STUN function on a serial interface.	**encapsulation stun**
Step 2	Place the interface in a previously defined STUN group.	**stun group** *group-number*

Once a given serial link is configured for the STUN function, it is no longer a shared multiprotocol link. All traffic that arrives on the link will be transported to the corresponding peer as determined by the current STUN configuration.

Establish the Frame Encapsulation Method

To allow SDLC frames to travel across a multimedia, multiprotocol network, you must encapsulate them using one of the following methods:

- Configure HDLC Encapsulation without Local Acknowledgment
- Configure TCP Encapsulation without Local Acknowledgment
- Configure TCP Encapsulation with SDLC Local Acknowledgment and Priority Queuing

Configure HDLC Encapsulation without Local Acknowledgment

You can encapsulate SDLC or HDLC frames using the HDLC protocol. The outgoing serial link still can be used for other kinds of traffic. The frame is not TCP encapsulated. To configure HDLC encapsulation, perform one of the following tasks in global configuration mode:

Task	Command
Forward all HDLC or SDLC traffic of the identified interface number.	**stun route all interface serial** *interface-number*
Forward all HDLC or SDLC traffic on a direct STUN link.	**stun route all interface serial** *interface-number* **direct**
Forward HDLC or SDLC traffic of the identified address.	**stun route address** *address-number* **interface serial** *interface-number*
Forward HDLC or SDLC traffic of the identified address across a direct STUN link.	**stun route address** *address-number* **interface serial** *interface-number* **direct**

Use the **no** forms of these commands to disable HDLC encapsulation.

Note You can only forward all traffic if you are using basic STUN protocol groups.

Configure TCP Encapsulation without Local Acknowledgment

If you do not want to use SDLC local acknowledgment and only need to forward all SDLC frames encapsulated in TCP, complete the following tasks:

Task	Command
Forward all TCP traffic for this IP address.	**stun route all tcp** *ip-address*
Specify TCP encapsulation.	**stun route address** *address-number* **tcp** *ip-address* [**local-ack**] [**priority**] [**tcp-queue-max**]

Use the **no** form of these commands to disable forwarding of all TCP traffic.

This configuration is typically used when the two routers may be connected via an IP network as opposed to a point-to-point link. Otherwise, HDLC should always be used.

Configure TCP Encapsulation with SDLC Local Acknowledgment and Priority Queuing

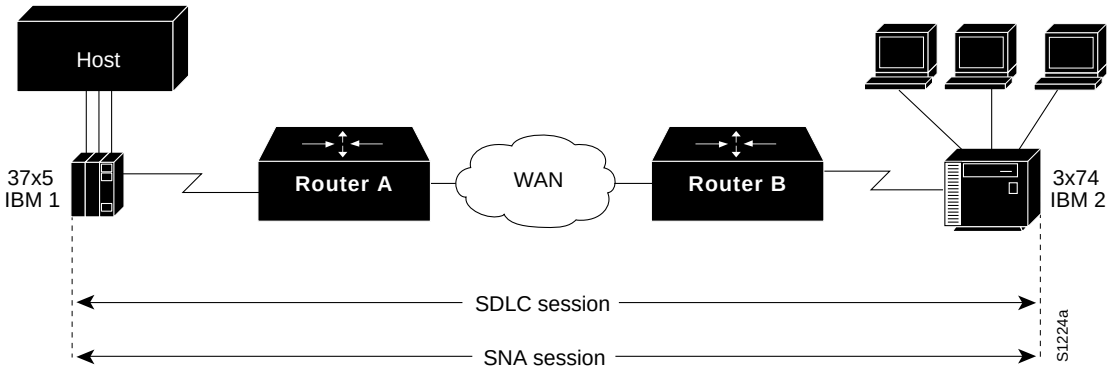
You can only configure SDLC local acknowledgment using TCP encapsulation. When you configure SDLC local acknowledgment, you also have the option of enabling support for priority queuing.

Note To enable SDLC local acknowledgment, you must have specified an SDLC or SDLC transmission group.

SDLC local acknowledgment provides local termination of the SDLC session so that control frames no longer travel the WAN backbone networks. This means that time-outs are less likely to occur.

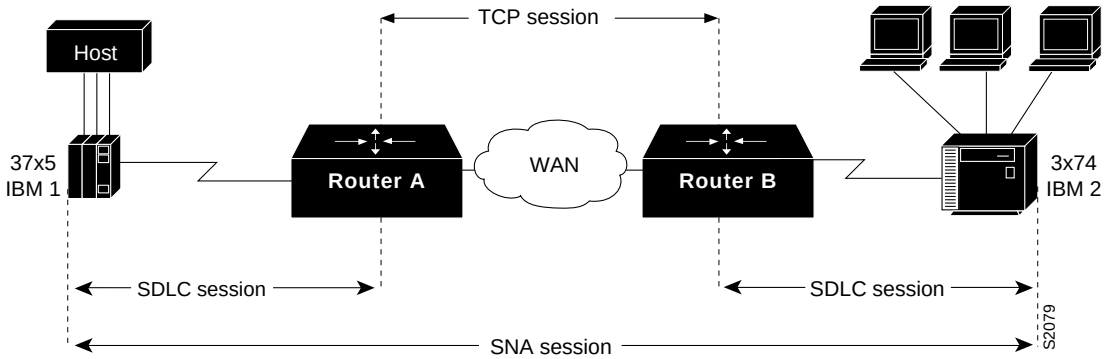
Figure 24-4 illustrates an SDLC session. IBM 1, using a serial link, can communicate with IBM 2 on a different serial link separated by a wide-area backbone network. Frames are transported between Router A and Router B using STUN. However, the SDLC session between IBM 1 and IBM 2 is still end-to-end. Every frame generated by IBM 1 traverses the backbone network to IBM 2, which, upon receipt of the frame, acknowledges it.

Figure 24-4 SDLC Session without Local Acknowledgment



With SDLC local acknowledgment, the SDLC session between the two end nodes is not end-to-end but instead terminates at the two local routers, as shown in Figure 24-5. The SDLC session with IBM 1 ends at Router A, and the SDLC session with IBM 2 ends at Router B. Both Router A and Router B execute the full SDLC protocol as part of SDLC Local Acknowledgment. Router A acknowledges frames received from IBM 1. The node IBM 1 treats the acknowledgments it receives as if they are from IBM 2. Similarly, Router B acknowledges frames received from IBM 2. The node IBM 2 treats the acknowledgments it receives as if they are from IBM 1.

Figure 24-5 SDLC Session with Local Acknowledgment



To configure TCP encapsulation with SDLC local acknowledgment and priority queuing, perform the tasks in the following sections:

- Assign the Router an SDLC Primary or Secondary Role
- Enable the SDLC Local Acknowledgment Feature
- Establish Priority Queuing Levels

Assign the Router an SDLC Primary or Secondary Role

To establish local acknowledgment, the router must play the role of an SDLC primary or secondary node. Primary nodes poll secondary nodes in a predetermined order. Secondaries then transmit if they have outgoing data.

For example, in the IBM environment, an FEP is the primary station and cluster controllers are secondary stations. If the router is connected to a cluster controller, it should appear as an FEP and must therefore be assigned the role of a primary SDLC node. If the router is connected to an FEP, it should appear as a cluster controller and must therefore be assigned the role of a secondary SDLC node. Routers connected to SDLC primary end-stations must play the role of an SDLC secondary and routers attached to SDLC secondary end stations must play the role of an SDLC primary station.

To assign the router a primary or secondary role, perform one of the following tasks in interface configuration mode:

Task	Command
Assign the STUN-enabled router an SDLC primary role.	**stun sdlc-role primary**
Assign the STUN-enabled router an SDLC secondary role.	**stun sdlc-role secondary**

Use the **no** form of these commands to remove SDLC role assignments.

Enable the SDLC Local Acknowledgment Feature

To enable SDLC local acknowledgment, complete the following task in global configuration mode:

Task	Command
Establish SDLC local acknowledgment using TCP encapsulation.	**stun route address** *address-number* **tcp** *ip-address* [**local-ack**] [**priority**] [**tcp-queue-max**]

The **stun route address 1 tcp local-ack priority tcp-queue-max** interface configuration command enables local acknowledgment and TCP encapsulation. Both these options are required to use transmission groups. You should specify the SDLC address with the echo bit turned off for transmission group interfaces. The SDLC broadcast address 0xFF is routed automatically for transmission group interfaces. The **priority** keyword creates multiple TCP sessions for this route. The **tcp-queue-max** keyword sets the maximum size of the outbound TCP queue for the SDLC. The default TCP queue size is 100. The value for **hold-queue in** should be greater than the value for **tcp-queue-max**.

You can use the **priority** keyword (to set up the four levels of priorities to be used for TCP encapsulated frames) at the same time you enable local acknowledgment. The **priority** keyword is described in the following section. Use the **no** form of this command to disable SDLC Local Acknowledgment. For an example of how to enable local acknowledgment, see "Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example" later in this chapter.

Establish Priority Queuing Levels

With SDLC local acknowledgment enabled, you can establish priority levels used in priority queuing for serial interfaces. The priority levels are as follows:

- Low
- Medium
- Normal
- High

To set the priority queuing level, perform the following task in interface configuration mode:

Task	Command
Establish the four levels of priorities to be used in priority queuing.	**stun route address** *address-number* **tcp** *ip-address* [l**ocal-ack**] **priority** [**tcp-queue-max**]

Use the **no** form of this command to disable priority settings. For an example of how to establish priority queuing levels, see "Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example" later in this chapter.

Configure STUN with Multilink Transmission Groups

You can configure multilink SDLC transmission groups across STUN connections between IBM communications controllers such as IBM 37x5s. Multilink transmission group allow you to collapse multiple WAN leased lines into one leased line.

SDLC multilink transmission groups provide the following features:

- Network control program (NCP) SDLC address allowances, including echo and broadcast addressing.
- Remote NCP load sequence. After a SIM/RIM exchange but before a SNRM/UA exchange, NCPs send numbered I-frames. During this period, I-frames are not locally acknowledged but instead are passed through. After the SNRM/UA exchange, local acknowledgment occurs.
- Rerouting of I-frames sent from the router to the NCP if a link is lost in a multilink transmission group.
- Flow control rate tuning causes a sending NCP to "feel" WAN congestion and hold frames that would otherwise be held in the router waiting to be transmitted on the WAN. This allows the NCP to perform its class-of-service algorithm more efficiently based on a greater knowledge of network congestion.

STUN connections that are part of a transmission group must have local acknowledgment enabled. Local acknowledgment keeps SDLC poll traffic off the WAN and reduces store-and-forward delays through the router. It also might minimize the number of NCP timers that expire due to network delay. Also, these STUN connections must go to the same IP address. This is because SNA transmission groups are parallel links between the same pair of IBM communications controllers.

Design Recommendations

This section provides some recommendations that are useful in configuring SDLC multilink transmission groups.

The bandwidth of the WAN should be larger than or equal to the aggregate bandwidth of all serial lines to avoid excessive flow control and ensure no degradation in response time. If other protocols also are using the WAN, ensure that the WAN bandwidth is significantly greater than the aggregate SNA serial line bandwidth to ensure that the SNA traffic does not monopolize the WAN.

When you are using a combination of routed transmission groups and directly connected NCP transmission groups, you need to plan the configuration carefully to ensure that SNA sessions do not stop unexpectedly. Assuming that hardware reliability is not an issue, from a software point of view, single-link routed transmission group are as reliable as direct NCP-to-NCP single-link transmission groups.This is true because neither the NCP nor the router can reroute I-frames when a transmission group has only one link. Additionally, multilink transmission group directed between NCPs and multilink transmission group through router are equally reliable. Both can perform rerouting.

However, you might run into problems if you have a configuration in which two NCPs are directly connected (via one or more transmission group links) and one link in the transmission group is routed. The NCPs will view this as a multilink transmission group. However, the router views the transmission group as a single-link transmission group. A problem can arise in the following situation: Assume that an I-frame is being transmitted from NCP A (connected to router A) to NCP B (connected to router B) and that all SDLC links are currently active. Router A will acknowledge the I-frame sent from NCP A and will send it over the WAN. If, before the I-frame reaches router B, the SDLC link between router B and NCP B goes down, router B will attempt to reroute the I-frame on another link in the transmission group when it receives the I-frame. However, because this is a single-link transmission group, there are no other routes, and router B drops the I-frame. NCP B will never receive this I-frame because router A acknowledged its receipt, and NCP A marked it as transmitted and deleted it. NCP B detects a gap in the transmission group sequence numbers and waits to receive the missing I-frame. It will wait forever for this I-frame, and in the meantime will not send or receive any other frames. This means that NCP B is technically inoperational and that all SNA sessions through NCP B will be lost.

One final design recommendation note concerns a configuration in which one or more lines of an NCP transmission group are router and one or more lines are directly connected between NCPs. If the network delay associated with one line of an NCP transmission group is different from the delay of another line in the same NCP transmission group, the receiving NCP will spend additional time resequencing PIUs.

Set up Traffic Priorities

You can use the methods described in the following sections to determine the order in which traffic should be handled on the network:

- Assign Queuing Priorities
- Prioritize STUN Traffic over All Other Traffic

Assign Queuing Priorities

You can assign queuing priorities by one of the following:

- Serial interface address or TCP port
- Logical unit (LU) address

Prioritize by Serial Interface Address or TCP Port

You can prioritize traffic on a per-serial-interface address or TCP port basis. You might want to do this so that traffic between one source-destination pair will always be sent before traffic between another source-destination pair.

Note You must first enable local acknowledgment and priority levels.

To prioritize traffic, perform one of the following tasks in global configuration mode:

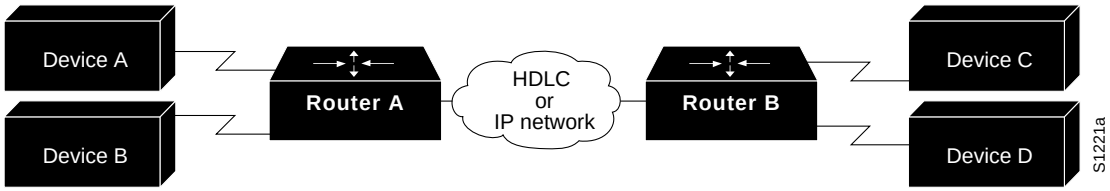
Task	Command
Assign a queuing priority to the address of the STUN serial interface.	**priority-list** *list-number* **stun** *queue-keyword* **address** *group-number address-number*
Assign a queuing priority to TCP port.	**priority-list** *list-number* **ip** *queue-keyword* **tcp** *tcp-port-number*

You must also perform the following task in interface configuration mode:

Task	Command
Assign a priority list to a priority group.	**priority-group** *list-number*

Figure 24-6 illustrates serial link address prioritization. Device A communicates with Device C, and Device B communicates with Device D. With the serial link address prioritization, you can choose to give A-C a higher priority over B-D across the serial tunnel.

Figure 24-6 Serial Link Address Prioritization



To disable priorities, use the **no** forms of these commands.

For an example of how to prioritize traffic, see "Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example" later in this chapter.

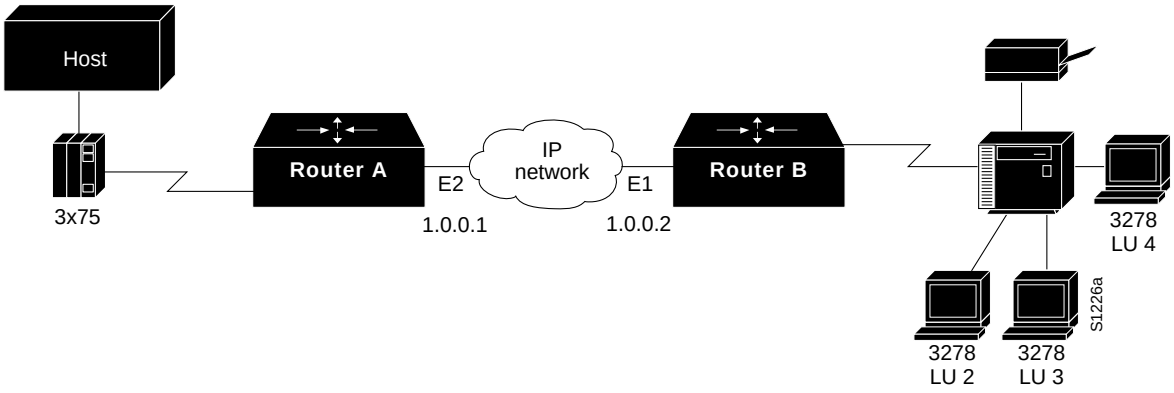
Prioritize by Logical Unit Address

SNA local logical unit (LU) address prioritization is specific to IBM SNA connectivity and is used to prioritize SNA traffic on either STUN or remote source-route bridging (RSRB). To set the queuing priority by LU address, perform the following task in interface configuration mode:

Task	Command
Assign a queuing priority based upon logical unit addresses.	**locaddr-priority-list** *list-number address-number queue-keyword*

In Figure 24-7, LU address prioritization can be set so that particular LUs receive data in preference to others or so that LUs have priority over the printer, for example.

Figure 24-7 SNA LU Address Prioritization



To disable this priority, use the **no** form of this command.

For an example of how to prioritize traffic, see "Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example" later in this chapter.

Prioritize STUN Traffic over All Other Traffic

You can prioritize STUN traffic to be routed first before all other traffic on the network. To give STUN traffic this priority, perform the following task in global configuration mode:

Task	Command
Prioritize STUN traffic in your network over that of other protocols.	**priority-list** *list-number* **stun** *queue-keyword* **address** *group-number address-number*

To disable this priority, use the **no** form of this command.

For an example of how to prioritize STUN traffic over all other traffic, see "Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example" later in this chapter.

Monitor STUN Network Activity

You can list statistics regarding STUN interfaces, protocol groups, number of packets sent and received, local acknowledgment states, and more. To get activity information, perform the following task in EXEC mode:

Task	Command
List the status display fields for STUN interfaces.	**show stun**

STUN Configuration Examples

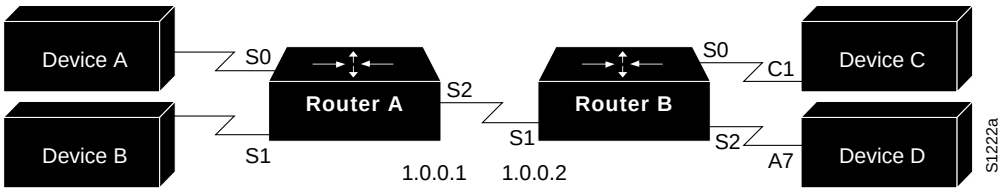
The following sections provide STUN configuration examples:

- Configuring STUN Priorities Using HDLC Encapsulation Example
- Configuring SDLC Broadcast Example
- Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example
- Configuring STUN Multipoint Implementation Using a Line-Sharing Device Example
- Configuring STUN Local Acknowledgment Example
- Configuring LOCADDR Priority Groups—Simple Example
- Configuring LOCADDR Priority Groups for STUN Example

Configuring STUN Priorities Using HDLC Encapsulation Example

Assume that the link between Router A and Router B in Figure 24-8 is a serial tunnel that uses the simple serial transport mechanism. Device A communicates with Device C (SDLC address C1) with a high priority. Device B communicates with Device D (SDLC address A7) with a normal priority.

Figure 24-8 STUN Simple Serial Transport



The following configurations set the priority of STUN hosts A, B, C, and D.

Configuration for Router A

```
stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 interface serial 2
!
interface serial 1
no ip address
encapsulation stun
stun group 2
stun route address A7 interface serial 2
!
interface serial 2
ip address 1.0.0.1 255.0.0.0
priority-group 1
!
priority-list 1 stun high address 1 C1
priority-list 1 stun low address 2 A7
```

Configuration for Router B

```
stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 interface serial 1
!
interface serial 1
ip address 1.0.0.2 255.0.0.0
priority-group 1
!
interface serial 2
no ip address
encapsulation stun
stun group 2
stun route address A7 interface serial 1
!
priority-list 1 stun high address 1 C1
priority-list 1 stun low address 2 A7
```

Configuring SDLC Broadcast Example

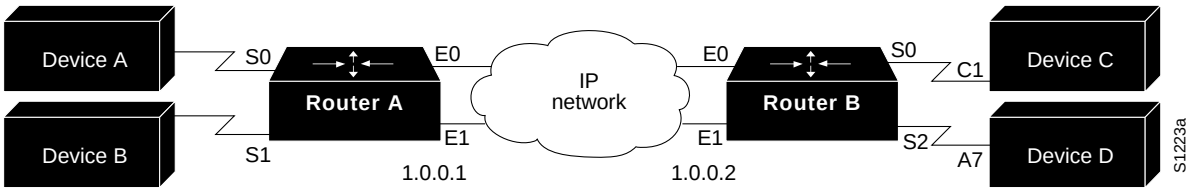
In the following example, an FEP views end stations 1, 2, and 3 as if they were on an SDLC multidrop link. Any broadcast frame sent from the FEP to Router A is duplicated and sent to each of the downstream routers (B and C):

```
stun peer-name xxx.xxx.xxx.xxx
stun protocol-group 1 sdlc
interface serial 1
encapsulation stun
stun group 1
stun sdlc-role secondary
sdlc virtual-multidrop
sdlc address 1
sdlc address 2
sdlc address 3
stun route address 1 tcp yyy.yyy.yyy.yyy local-ack
stun route address 2 tcp zzz.zzz.zzz.zzz local-ack
stun route address 3 tcp zzz.zzz.zzz.zzz local-ack
stun route address FF tcp yyy.yyy.yyy.yyy
stun route address FF tcp zzz.zzz.zzz.zzz
```

Configuring Serial Link Address Prioritization Using STUN TCP/IP Encapsulation Example

Assume that the link between Router A and Router B is a serial tunnel that uses the TCP/IP encapsulation as shown in Figure 24-9. Device A communicates with Device C (SDLC address C1) with a high priority. Device B communicates with Device D (SDLC address A7) with a normal priority.

Figure 24-9 STUN TCP/IP Encapsulation



The configuration of each router follows.

Configuration for Router A

```
stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.2 local-ack priority
priority-group 1
!
interface serial 1
no ip address
encapsulation stun
stun group 2
stun route address A7 tcp 1.0.0.2 local-ack priority
```

```
priority-group 2
!
interface ethernet 0
ip address 1.0.0.1 255.0.0.0
!
interface ethernet 1
ip address 1.0.0.3 255.0.0.0
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
priority-list 1 stun high address 1 C1
!
priority-list 2 protocol ip high tcp 1994
priority-list 2 protocol ip medium tcp 1990
priority-list 2 protocol ip normal tcp 1991
priority-list 2 protocol ip low tcp 1992
priority-list 2 stun normal address 2 A7
!
hostname routerA
router igrp
network 1.0.0.0
```

Configuration for Router B

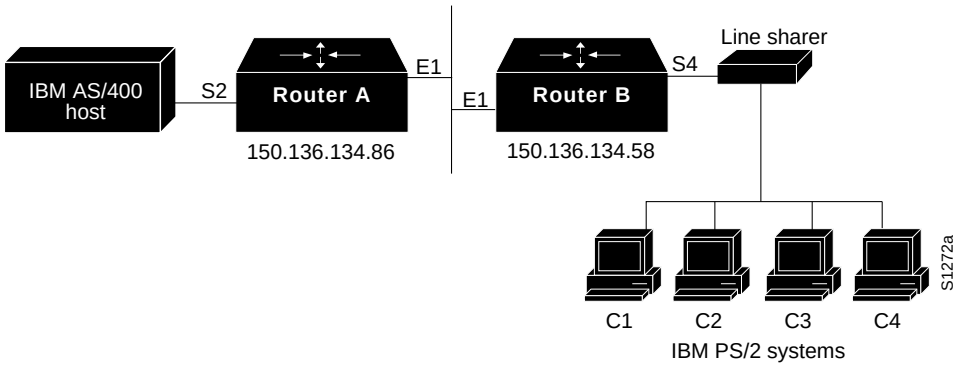
```
stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.1 local-ack priority
priority-group 1
!
interface serial 2
no ip address
encapsulation stun
stun group 2
stun route address A7 tcp 1.0.0.1 local-ack priority
priority-group 2
!
interface ethernet 0
ip address 1.0.0.2 255.0.0.0
!
interface ethernet 1
ip address 1.0.0.4 255.0.0.0
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
priority-list 1 stun high address 1 C1
!
priority-list 2 protocol ip high tcp 1994
priority-list 2 protocol ip medium tcp 1990
priority-list 2 protocol ip normal tcp 1991
priority-list 2 protocol ip low tcp 1992
priority-list 2 stun normal address 2 A7
!
hostname routerB
router igrp 109
```

```
network 1.0.0.0
```

Configuring STUN Multipoint Implementation Using a Line-Sharing Device Example

In Figure 24-10, four separate PS/2 computers are connected to a line-sharing device off of Router B. Each PS/2 computer has four sessions open on an AS/400 device attached to Router A. Router B functions as the primary station, while Router A functions as the secondary station. Both routers locally acknowledge packets from the IBM PS/2 systems.

Figure 24-10 STUN Communication Involving a Line-Sharing Device



The configuration file for the routers shown in Figure 24-10 follows.

Configuration for Router A

```
! enter the address of the stun peer
stun peer-name 150.136.134.86
! specify that group 4 uses the SDLC protocol
stun protocol-group 4 sdlc
stun remote-peer-keepalive

interface ethernet 1
! enter the IP address for the Ethernet interface
ip address 150.136.134.86 255.255.255.0
!
! description of IBM AS/400 link
interface serial 2
! description of IBM AS/400 link; disable the IP address on a serial interface
no ip address
! enable STUN encapsulation on this interface
encapsulation stun
! apply previously defined stun group 4 to serial interface 2
stun group 4
! establish this router as a secondary station
stun sdlc-role secondary
! wait up to 63000 msec for a poll from the primary before timing out
sdlc poll-wait-timeout 63000
! list addresses of secondary stations (PS/2 systems) attached to link
sdlc address C1
sdlc address C2
sdlc address C3
sdlc address C4
! use tcp encapsulation to send frames to SDLC stations C1, C2, C3, or
! C4 and locally terminate sessions with these stations
stun route address C1 tcp 150.136.134.58 local-ack
```

```
stun route address C2 tcp 150.136.134.58 local-ack
stun route address C3 tcp 150.136.134.58 local-ack
stun route address C4 tcp 150.136.134.58 local-ack
```

Configuration for Router B

```
! enter the address of the stun peer
stun peer-name 150.136.134.58
! this router is part of SDLC group 4
stun protocol-group 4 sdlc
stun remote-peer-keepalive
!
interface ethernet 1
! enter the IP address for the Ethernet interface
ip address 150.136.134.58 255.255.255.0
!
! description of PS/2 link
interface serial 4
! disable the IP address on a serial interface
no ip address
! enable STUN encapsulation on this interface
encapsulation stun
! apply previously defined stun group 4 to serial interface 2
stun group 4
! establish this router as a primary station
stun sdlc-role primary
sdlc line-speed 9600
! wait 2000 milliseconds for a reply to a frame before resending it
sdlc t1 2000
! resend a frame up to four times if not acknowledged
sdlc n2 4
! list addresses of secondary stations (PS/2 systems) attached to link
sdlc address C1
sdlc address C2
sdlc address C3
sdlc address C4
! use tcp encapsulation to send frames to SDLC stations C1, C2, C3, or
! C4 and locally terminate sessions with these stations
stun route address C3 tcp 150.136.134.86 local-ack
stun route address C1 tcp 150.136.134.86 local-ack
stun route address C4 tcp 150.136.134.86 local-ack
stun route address C2 tcp 150.136.134.86 local-ack
! set the clockrate on this interface to 9600 bits per second
clockrate 9600
```

Configuring STUN Local Acknowledgment Example

The following example shows a sample configuration for a pair of routers performing SDLC local acknowledgment.

Configuration for Router A

```
stun peer-name 150.136.64.92
stun protocol-group 1 sdlc
stun remote-peer-keepalive
!
interface Serial 0
no ip address
encapsulation stun
stun group 1
stun sdlc-role secondary
sdlc address C1
stun route address C1 tcp 150.136.64.93 local-ack
clockrate 19200
```

Configuration for Router B

```
stun peer-name 150.136.64.93
stun protocol-group 1 sdlc
stun remote-peer-keepalive
!
interface Serial 0
no ip address
encapsulation stun
stun group 1
stun sdlc-role primary
sdlc line-speed 19200
sdlc address C1
stun route address C1 tcp 150.136.64.92 local-ack
clockrate 19200
```

Configuring LOCADDR Priority Groups—Simple Example

The following example shows how to establish queuing priorities on a STUN interface based on an LU address:

```
! sample stun peer-name global command
stun peer-name 131.108.254.6
! sample protocol-group command for reference
stun protocol-group 1 sdlc
!
interface serial 0
! disable the ip address for interface serial 0
no ip address
! enable the interface for STUN
encapsulation stun
! sample stun group command
stun group 2
! sample stun route command
stun route address 10 tcp 131.108.254.8 local-ack priority
!
! assign priority group 1 to the input side of interface serial 0
locaddr-priority 1
priority-group 1
interface Ethernet 0
! give locaddr-priority-list 1 a high priority for LU 02
locaddr-priority-list 1 02 high
```

```
! give locaddr-priority-list 1 a low priority for LU 05
locaddr-priority-list 1 05 low
```

Configuring LOCADDR Priority Groups for STUN Example

The following configuration example shows how to assign a priority group to an input interface:

Configuration for Router A

```
stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.2 local-ack priority
clockrate 19200
locaddr-priority 1
priority-group 1
!
interface Ethernet 0
ip address 1.0.0.1 255.255.255.0
!
locaddr-priority-list 1 02 high
locaddr-priority-list 1 03 high
locaddr-priority-list 1 04 medium
locaddr-priority-list 1 05 low
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
```

Configuration for Router B

```
stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.1 local-ack priority
clockrate 19200
locaddr-priority 1
priority-group 1
!
interface Ethernet 0
ip address 1.0.0.2 255.255.255.0
!
locaddr-priority-list 1 02 high
locaddr-priority-list 1 03 high
locaddr-priority-list 1 04 medium
locaddr-priority-list 1 05 low
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
```

CHAPTER 25

Configuring LLC2 and SDLC Parameters

The Logical Link Control, type 2 (LLC2) and Synchronous Data Link Control (SDLC) protocols provide data link-level support for higher-level network protocols and features such as SDLLC and remote source-route bridging (RSRB) with local acknowledgment. The features that are affected by LLC2 parameter settings are listed in the next section, "LLC2." The features that require SDLC configuration and use SDLC parameters are listed in the section "SDLC" later in this chapter.

LLC2 and SDLC package data in frames. LLC2 and SDLC stations require acknowledgments from receiving stations after a set amount of frames have been sent before sending further data. The tasks described in this chapter modify default settings regarding the control field of the data frames. By modifying the control field parameters, you can determine the amount of acknowledgments sent for frames received and the level of polling used to determine available stations. In this manner, you can set the amount of resources used for frame checking and optimize the network load.

SDLC is used as the primary Systems Network Architecture (SNA) link-layer protocol for wide-area network (WAN) links. SDLC defines two types of network nodes: primary and secondary. Primary nodes poll secondary nodes in a predetermined order. Secondaries then transmit if they have outgoing data. When configured as primary and secondary nodes, our routers are established as SDLC stations.

You do not need to configure LLC2 because it is already enabled on Token Ring interfaces. You can change the default settings of LLC2 parameters as needed. To support SDLC, you need to configure the router as a primary or secondary SDLC station. You also can change default settings on any of the SDLC parameters listed. Configuration examples for both LLC2 and SDLC are given at the end of the chapter.

For a complete description of the commands mentioned in this chapter, refer to the "LLC2 and SDLC Commands" chapter in the *Router Products Command Reference* publication. For historical background and a technical overview of LLC2 and SDLC, see the *Internetworking Technology Overview* publication.

LLC2

The following features use LLC services:

- Local acknowledgment for RSRB

  This feature is used in our implementation of RSRB as described in the chapter "Configuring Source-Route Bridging."

  Because local-area networks (LANs) are now connected through RSRB and wide-area network (WAN) backbones, the delays that occur are longer than LLC2 allows for bidirectional communication between hosts. Our local acknowledgment feature addresses the problem of delays, retransmissions, and loss of user sessions.

- IBM LAN Network Manager (LNM) support

  Routers using 4- or 16-Mbps Token Ring interfaces configured for SRB support LNM and provide all IBM Bridge Program functions. With LNM, a router appears as an IBM source-route bridge, and can manage or monitor any connected Token Ring interface.

  LNM support is described in the chapter “Configuring Source-Route Bridging.”

- SDLLC media translation

  The SDLLC feature provides media translation between the serial lines running SDLC and Token Rings running LLC2. SDLLC consolidates the IBM SNA networks running SDLC into a LAN-based, multiprotocol, multimedia backbone network.

  SDLLC is described in the chapter “Configuring SDLLC.”

- ISO Connection-Mode Network Services (CMNS)

  Our CMNS implementation runs X.25 packets over LLC2 so that X.25 can be extended to Ethernet, FDDI, and Token Ring media.

LLC2 Configuration Task List

Since LLC2 is already enabled on Token Rings, you do not need to enable it on the router. However, you can complete the tasks in the following sections to enhance LLC2 performance:

- Control Transmission of I-Frames
- Establish Polling Level
- Set Up XID Transmissions

To determine which LLC2 parameters need adjustment, you can perform the following task:

- Monitor LLC2 Stations

See the end of this chapter for an LLC2 configuration example

Control Transmission of I-Frames

You can control the number of information-frames (I-frames) and acknowledgments sent on the LLC2 network by completing the tasks described in THE following sectionS.

- Set the Maximum Number of I-Frames Received before Sending an Acknowledgment
- Set the Maximum Delay for Acknowledgments
- Set the Maximum Number of I-Frames Sent before Requiring Acknowledgment
- Set the Number of Retries Allowed
- Set the Time for Resending I-Frames
- Set the Time for Resending Rejected Frames

Set the Maximum Number of I-Frames Received before Sending an Acknowledgment

You can reduce overhead on the network by increasing the maximum number of frames the router can receive at once before it must send the sender an acknowledgment. To do so, perform the following task in interface configuration mode:

Task	Command
Set maximum number of I-frames the router can receive before it sends an acknowledgment.	**llc2 ack-max** *packet-count*

Set the Maximum Delay for Acknowledgments

You can ensure timely receipt of acknowledgments so that transmission of data is not delayed. Even if the maximum amount of frames has not been reached, you can set a timer forcing the router to send an acknowledgment and reset the maximum amount counter to 0.

To set the maximum delay time, perform the following task in interface configuration mode:

Task	Command
Set the I-frame acknowledgment time.	**llc2 ack-delay-time** *milliseconds*

Set the Maximum Number of I-Frames Sent before Requiring Acknowledgment

You can set the maximum number of I-frames that the router sends to an LLC2 station before the router requires an acknowledgment from the receiving end. A higher value reduces overhead on the network. Ensure that the receiving LLC2 station can handle the number of frames set by this value.

To set this value, perform the following task in interface configuration mode:

Task	Command
Set the maximum number of I-frames the router sends before it requires an acknowledgment.	**llc2 local-window** *packet-count*

Set the Number of Retries Allowed

You can set the number of times the router will resend a frame when the receiving station does not acknowledge the frame. Once this value is reached, the session is dropped. This value also is used to determine how often the router will retry polling a busy station. This task should be done in conjunction with the task for setting the time for resending I-frames (described next). Performing them together ensures that frame transmission is monitored at a reasonable level, while limiting the number of unsuccessful repeated tries.

To set the number of retries, perform the following task in interface configuration mode:

Task	Command
Establish number of times the router will resend unacknowledged frames or try polling a busy station.	**llc2 n2** *retry-count*

Set the Time for Resending I-Frames

You can set the amount of time the router will wait before resending unacknowledged I-frames. This interval is called the T1 time. Perform this task in conjunction with the tasks of setting the number of retries and setting the transit poll-frame timer. Performing these tasks in conjunction with each other provides a balance of network monitoring and performance.

To set the T1 time, perform the following task in interface configuration mode:

Task	Command
Control how long the router waits for an acknowledgment of transmitted I-frames.	**llc2 t1-time** *milliseconds*

Note Ensure that you allow enough time to account for the round trip between the router and its LLC2-speaking stations. Under heavy network loading conditions, resending I-frames every 3000 milliseconds is appropriate.

Set the Time for Resending Rejected Frames

You can set the amount of time that the router will wait for an expected frame before sending a reject command (REJ). Typically, when an LLC2 station sends an I-frame, a sequence number is included in the frame. The LLC2 station that receives these frames will expect to receive them in order. If it does not, it can reject a frame and indicate which frame it is expecting to receive instead. If the correct frame is not sent to the router before the reject timer expires, the router sends a REJ to the remote station and disconnects the LLC2 session.

To set the reject timer, perform the following task in interface configuration mode:

Task	Command
Set the time a router waits for a resend of a rejected frame before sending a reject command to the remote station.	**llc2 trej-time** *milliseconds*

Establish Polling Level

You can control the amount of polling that occurs on the LLC2 network by completing the tasks described in the following sections:

- Set the Polling Frequency
- Set the Polling Interval
- Set the Transmit-Poll-Frame Timer

Set the Polling Frequency

You can set the optimum interval of time after which the router sends Receiver Readys, or frames that tell other LLC2 stations that the router is available. These polls occur during periods of idle time on the network.

To set polling frequency, perform the following task in interface configuration mode:

Task	Command
Control the polling frequency during idle traffic.	**llc2 idle-time** *milliseconds*

Set the Polling Interval

The amount of time the router waits until repolling a busy station can also be set. Perform this task in conjunction with setting the number of retries. Typically, you do not need to perform this task unless an LLC2 station has unusually long busy periods before clearing the busy state. In this case, you should increase the value so that the station does not time out.

To set the polling interval, perform the following task in interface configuration mode:

Task	Command
Set the amount of time the router will wait before repolling a busy station.	**llc2 tbusy-time** *milliseconds*

Set the Transmit-Poll-Frame Timer

When sending a command that must receive a response, a poll bit is sent in the frame. Once the router sends the poll bit, it cannot send any other frame with the poll bit set until the receiver replies to that poll frame with a frame containing a final bit set. When the timer expires, the router assumes that it can send another frame with a poll bit.

Set the transmit-poll-frame timer to reduce problems with receiving stations that are faulty and cannot send the frame with the final bit set by performing the following task in interface configuration mode:

Task	Command
Set the amount of time the router waits for a final response to a poll frame before the resending it.	**llc2 tpf-time** *milliseconds*

This value should be larger than the T1 time. The T1 time determines how long the router waits for receipt of an acknowledgment before sending the next set of frames. See the section “Set the Time for Resending I-Frames” earlier in this chapter for more information.

Set Up XID Transmissions

You can control the number of frames used for identification on the LLC2 network by completing the tasks described in the following sections:

- Set the Frequency of XID Transmissions
- Set the Time for XID Retries

Set the Frequency of XID Transmissions

Exchange of identification (XID) frames identify LLC2 stations at a higher level than the MAC address and contain information about the configuration of the stations.You can set how often the router sends an XID frame by performing the following task in interface configuration mode:

Task	Command
Set the frequency of XID transmissions.	**llc2 xid-neg-val-time** *milliseconds*

Caution Do not change the value unless requested by your technical support representative.

Set the Time for XID Retries

You can set the amount of time the router waits for a reply to the XID frames it sends to remote stations. The value should be larger than the T1 time, which indicates how long the router waits for an acknowledgment before dropping the session.

To set the time for XID retries, perform the following task in interface configuration mode:

Task	Command
Set how long the router waits for a reply to the XID frames it sends to remote stations.	**llc2 xid-retry-time** *milliseconds*

Monitor LLC2 Stations

You can display the configuration of LLC2 stations to determine which LLC2 parameters need adjustment. Perform the following task in EXEC mode:

Task	Command
Display the configuration of LLC2 stations.	**show llc2**

SDLC

The SDLC tasks described in this section configure the router as an SDLC station. This is in contrast to a router configured for SDLC Transport, where the router is not an SDLC station, but passes SDLC frames between two SDLC stations across a mixed-media, multiprotocol environment.

The tasks described in this section support the following features:

- Frame Relay access support

  Our Frame Relay access support feature enables a router to function as a Frame Relay Access Device (FRAD) for SDLC, Token Ring, and Ethernet-attached devices over a Frame Relay Boundary Network Node (BNN) link.

  Frame Relay access support is described in the chapter "Configuring SNA Frame Relay Access Support."

- SDLLC media translation

  Our SDLLC feature provides media translation between the serial lines running SDLC and Token Rings running LLC2. SDLLC consolidates the IBM SNA networks running SDLC into a LAN-based, multiprotocol, multimedia backbone network.

  SDLLC is described in the chapter "Configuring SDLLC."

- SDLC local acknowledgment

  SDLC local acknowledgment is used with SDLC serial tunnel (STUN). The Transmission Control Protocol/Internet Protocol (TCP/IP) must be enabled. With local acknowledgment, STUN SDLC connections can be terminated locally at the router, eliminating the need for acknowledgments to be sent across a WAN.

  SDLC local acknowledgment is described in the chapter "Configuring STUN."

SDLC Configuration Task List

Perform the tasks in the following sections to enable the router as an SDLC station. The first task is required; you accomplish it with the appropriate set of commands for your network needs. The remaining tasks are optional; you can use them as necessary to enhance SDLC performance.

- Enable the Router as a Primary or Secondary SDLC Station
- Enable SDLC Two-Way Simultaneous Mode
- Determine Use of Frame Rejects
- Set SDLC Timer and Retry Counts
- Set SDLC Frame and Window Sizes
- Control the Buffer Size
- Control Polling of Secondary Stations
- Configure an SDLC Interface for Half-Duplex Mode
- Specify the XID Value
- Set the Largest SDLC I-Frame Size

To determine which SDLC parameters need adjustment, you can perform the following task:

- Monitor SDLC Stations

See the end of this chapter for SDLC configuration examples.

Enable the Router as a Primary or Secondary SDLC Station

SDLC defines two types of network nodes: primary and secondary. Primary nodes poll secondary nodes in a predetermined order. Secondaries then transmit if they have outgoing data. When configured as primary and secondary nodes, our routers are established as SDLC stations.

Depending on your particular network needs, perform the tasks in one of the following sections to enable the router as an SDLC station:

- Establish an SDLC Station for Frame Relay Access Support
- Establish an SDLC Station for SDLLC Media Translation

Establish an SDLC Station for Frame Relay Access Support

You can establish the router to be any of the following:

- A primary SDLC station
- A secondary SDLC station
- Either primary or secondary, depending on the role of the end stations or on XID negotiations
- A primary Node Type 2.1 (NT2.1) node

To establish routers as SDLC stations when you plan to configure Frame Relay access support, complete the following tasks in interface configuration mode:

Task	Command
Set the encapsulation type of the serial interface to SDLC.	**encapsulation sdlc**
Establish the role of the interface.	**sdlc role** {**none** | **primary** | **secondary** | **prim-xid-poll**}

If the interface does not play a role, the router can be either primary or secondary, depending on the end stations. The SDLC end station must be configured as negotiable or primary NT2.1. When the end stations are configured as physical unit (PU) type 2, you can set the role of the interface to primary or secondary. When the end station is configured as secondary NT2.1, you must set the role of the interface to poll the primary XID.

Note Currently, Frame Relay access support does not support the secondary role.

Establish an SDLC Station for DLSw+ Support

To establish routers as SDLC stations when you plan to configure our DLSw+ feature, complete the following tasks in interface configuration mode:

Task	Command
Set the encapsulation type of the serial interface to SDLC.	**encapsulation sdlc**
Establish the role of the interface.	**sdlc role** {**none** | **primary** | **secondary** | **prim-xid-poll**}
Configure a MAC address for the serial interface.	**sdlc vmac** *mac-address*[1]

Task	Command
Specify the destination address with which an LLC session is established for the SDLC station.	**sdlc partner** *mac-address sdlc-address*
Attach SDLC addresses to DLSw+	**sdlc dlsw** *sdlc-address...sdlc-address*

1. The last byte of the MAC address must be 00.

For additional DLSw+ configuration tasks, refer to the chapter "Configuring DLSw+".

Establish an SDLC Station for SDLLC Media Translation

To establish routers as SDLC stations when you plan to configure our SDLLC media translation feature, complete tasks in the order listed in the following table. One serial interface can have two or more secondary stations attached to it through a modem sharing device (MSD). Each secondary station address must be assigned to the primary station. You must perform the following tasks in interface configuration mode for the serial interface:

Task		Command
Step 1	Establish a router as the primary SDLC station on the serial line.	**encapsulation sdlc-primary**
Step 2	Establish other routers as secondary SDLC stations.	**encapsulation sdlc-secondary**
Step 3	Assign secondary stations to a primary station.	s**dlc address** *hexbyte* [*echo*]

Use the **show interfaces** command to list the configuration of the SDLC serial lines. Use the **no sdlc address** command to remove a secondary address assignment. Addresses are hexadecimal (base 16).

Enable SDLC Two-Way Simultaneous Mode

SDLC two-way simultaneous mode allows a primary SDLC link station to achieve better utilization of a full-duplex serial line. With two-way simultaneous mode, the primary link station can send data to one secondary link station while there is a poll outstanding. Two-way simultaneous mode works on the SDLC primary side only. On a secondary link station, it responds to a poll from the primary station.

SDLC two-way simultaneous mode operates in either a multidrop link environment or point-to-point link environment.

In a multidrop link environment, a two-way simultaneous primary station is able to poll a secondary station and receive data from the station, and send data (I-frames) to other secondary stations.

In a point-to-point link environment, a two-way simultaneous primary station can send data (I-frames) to the secondary station although there is a poll outstanding, as long as the window limit is not reached.

To enable two-way simultaneous mode, perform either of the following tasks in interface configuration mode:

Determine Use of Frame Rejects

You can specify that a secondary station does not send Frame Rejects (FRMRs), or reject commands indicating frame errors. If you do so, the router will drop an SDLC connection if it receives an error from the secondary station. To determine handling of FRMRs, perform the following task in interface configuration mode:

To specify that the secondary station does support FRMRs, use the sdlc frmr-disable command.

Set SDLC Timer and Retry Counts

When an SDLC station sends a frame, it waits for an acknowledgment from the receiver saying that this frame has been received. You can modify the time the router allows for an acknowledgment before resending the frame. You can also determine the number of times that a router resends a frame

The SDLLC software allows a physical unit (PU) 4, PU 2.1, or PU 2 to communicate with a PU 2 SDLC device as follows:

- SDLLC with direct connection—A 37x5 FEP on a Token Ring and the 3x74 cluster controller connected to a serial line are each connected to an interface on the same router configured with SDLLC.
- SDLLC with Remote Source-Route Bridging (RSRB)—A 37x5 FEP on a Token Ring and a 3x74 cluster controller connected to a serial line are connected to different routers. Only the router to which the 3x74 is connected is configured with SDLLC. The routers communicate via RSRB using direct encapsulation, RSRB over a Fast Sequenced Transport (FST) connection, or RSRB over a TCP connection.
- SDLLC with RSRB and local acknowledgment—A 37x5 front-end processor (FEP) on a Token Ring and a 3x74 cluster controller connected to a serial line are connected to different routers. Only the router to which the 3x74 is connected is configured with SDLLC. The routers communicate via RSRB over a TCP connection that has local acknowledgment enabled.

In all of these topologies, each IBM end node (the FEP and cluster controller) has no indication that its counterpart is connected to a different medium running a different protocol. The 37x5 FEP responds as if the 3x74 cluster controller were communicating over a Token Ring, whereas the 3x74 responds as though the 37x5 FEP were communicating over a serial line. That is, the SDLLC software provides translation between the two media to be transparent to the end nodes.

Virtual Token Ring Concept Implementation

Central to Cisco's SDLLC feature is the concept of a virtual Token Ring device residing on a virtual Token Ring. Because the Token Ring device expects the node with which it is communicating also to be on a Token Ring, each SDLLC device on a serial line must be assigned an SDLLC virtual token ring address (SDLLC VTRA). Like real Token Ring addresses, SDLLC VTRAs must be unique across the network.

In addition to the SDLLC VTRA, an SDLLC virtual ring number (SDLLC VRN) also must be assigned to each SDLLC device on a serial line. (The SDLLC VRN differs from the virtual ring group numbers that are used to configure RSRB and multiport bridging.)

As part of its Virtual Telecommunications Access Method (VTAM) configuration, the IBM node on the Token Ring has knowledge of the SDLLC VTRA of the serial device with which it communicates. The SDLC VTRA and the SDLLC VRN are a part of the SDLLC configuration for the router's serial interface. When the Token Ring host sends out explorer packets with the SDLLC VTRA as the destination address in the Media Access Control (MAC) headers, the router configured with that SDLLC VTRA intercepts the frame, fills in the SDLLC VRNA and the bridge number in the routing information field (RIF), and then sends the response back to the Token Ring host. A route is then established between the Token Ring host and the router. After the router performs the appropriate frame conversion, it uses this route to forward frames to this serial device.

Resolving Differences in LLC2 and SDLC Frame Size

IBM nodes on Token Ring media normally use frame sizes greater than 1 KB, whereas the IBM nodes on serial lines normally limit frame sizes to 265 or 521 bytes. To reduce traffic on backbone networks and provide better performance, Token Ring nodes should send as large frames as possible. As part of the SDLLC configuration on the router's serial interface, the largest frame size the two media will support should be selected. The router can fragment the frames it receives from the Token Ring device before forwarding them to the SDLC device; however, it does not assemble the frames it receives from the serial device before forwarding them to the Token Ring device.

Maintaining a Dynamic RIF Cache

SDLCC maintains a dynamic RIF cache. SDLLC caches the entire RIF; that is, the RIF from the source station to destination station. The cached entry is based on the best path at the time the session began. SDLLC uses the RIF cache to maintain the LLC2 session between the router and the host FEP. SDLLC does not age these RIF entries. Instead, SDLLC places an entry in the RIF cache for a session when the session begins and flushes the cache when the session terminates. You cannot flush these RIFs because if you flushed the RIF entries randomly, the router would not be able to maintain the LLC2 session to the host FEP.

Other Implementation Considerations

- As part of Cisco's SDLC implementation, only modulus 8 Normal Response Mode (NRM) sessions are maintained for the SDLC session.
- SDLC sessions are always locally acknowledged. LLC2 sessions can be optionally configured for local acknowledgment.
- SDLLC does not apply to SNA subarea networks, such as 37x5 FEP-to 37x5 FEP communication.
- Parameters such as the maximum number of information frames (I-frames) outstanding before acknowledgment, frequency of polls, and response time to poll frames can be modified per interface. If local acknowledgment is not enabled, these parameters are modified on the SDLC interface; if local acknowledgment is enabled, these parameters are modified on the Token Ring interface.
- Local acknowledgment only applies when the remote peer is defined for RSRB using Internet Protocol (IP) encapsulation over a TCP connection. If no local acknowledgment is used, the remote peer can be defined for RSRB using direct encapsulation, RSRB using IP encapsulation over an FST connection, or RSRB using IP encapsulation over a TCP connection.

SDLLC Configuration Task List

You can perform the tasks in the following sections to configure SDLLC:

- Configure SDLLC with Direct Connection
- Configure SDLLC with RSRB
- Configure SDLLC with RSRB and Local Acknowledgment
- Configure SDLLC with Ethernet and Translational Bridging
- Customize SDLLC Media Translation
- Monitor SDLLC Media Translation

See the end of this chapter for SDLLC configuration examples.

Configure SDLLC with Direct Connection

In the SDLLC configuration with direct connection, a 37x5 FEP on a Token Ring and a 3x74 cluster controller connected to a serial line are each connected to an interface on the same router configured with SDLLC. In this configuration, the LLC2 session extends from the 37x5 FEP across the Token Ring to the router. The SDLLC session extends from the router across the serial line to the 3x74 cluster controller. The SNA session extends across the Token Ring and the serial line to provide an end-to-end connection. The router is configured with source-route bridging (SRB).

To configure SDLLC with direct connection, you must perform the tasks in the following sections:

- Enable SDLLC Media Translation
- Associate a SAP Value
- Specify the XID Value
- Initiate Connection to Token Ring Host

For an example of how to configure SDLLC with direct connection, see the "Example of SDLLC with Direct Connection" later in this chapter.

Enable SDLLC Media Translation

The interfaces you will configure for SDLLC media translation are the serial interfaces that connect to the serial lines linking the remote SDLC devices. To configure them, perform the following task in interface configuration mode:

Task	Command
Enable SDLLC media translation on a serial interface.	**sdllc traddr** *xxxx.xxxx.xx00 lr bn tr*

Associate a SAP Value

You can associate a SAP value by performing the following task in interface configuration mode:

Task	Command
Associate a SAP value.	**sdllc sap** *sdlc-address ssap dsap*

Specify the XID Value

The exchange of identification (XID) value you define on the router must match that of the IDBLK and IDNUM system generation parameters defined in VTAM of the Token Ring host to which the SDLC device will be communicating. To define XID on the router, perform the following task in interface configuration mode:

Task	Command
Specify the XID value appropriate for the SDLC station to match VTAM values.	**sdllc xid** *address xxxxxxxx*

Initiate Connection to Token Ring Host

The Token Ring host is always kept in a state ready to accept a connection from the remote serial device. The remote serial device is responsible for initiating connections. The advantage of this scheme is that the serial device can communicate with the Token Ring host whenever it chooses without requiring personnel to be on the host site.

The router actually initiates the connection on behalf of the serial device. To initiate connections, both the MAC address of the Token Ring host and the SDLC line address are required. You must configure the router to define the Token Ring host as the partner of the serial device. To do so, perform the following task in interface configuration mode:

Task	Command
Enable connections for SDLLC.	**sdllc partner** *mac-address sdlc-address*

Configure SDLLC with RSRB

A router need not directly connect the two IBM end nodes: a 37x5 FEP on a Token Ring and a 3x74 cluster controller connected to a serial line can be connected to different routers. However, the router to which the 3x74 is connected must be configured with SDLLC. The routers communicate via RSRB using direct encapsulation, RSRB over an FST connection, or RSRB over a TCP connection. RSRB transports packets between Router A and Router B, while Router B performs all conversion between the LLC2 and SDLC protocols by means of the SDLLC software.

To configure the router for SDLLC with RSRB you must perform all the tasks in the "Configure SDLLC with Direct Connection" section earlier in this chapter. In addition, you must perform one of the sets of tasks in the following sections:

- Configure RSRB Using Direct Encapsulation
- Configure RSRB over FST Connection
- Configure RSRB over TCP Connection

For more information about configuring RSRB, see the chapter "Configuring Source-Route Bridging" in this manual and "Source-Route Bridging Commands" in the *Router Products Command Reference* publication.

Note When you configure RSRB, you must configure include a **source-bridge remote peer** command on the router connected to the serial line and another **source-bridge remote peer** command on the one connected to the Token Ring. If you have more than one serial line connected to the same router, then you will have a **source-bridge remote peer** command for each interface in its configuration that will be using SDLLC with RSRB.

For an example of how to configure SDLLC with RSRB, see the section "Example of SDLLC with RSRB (Multiple 3x74s)" later in this chapter.

Configure RSRB Using Direct Encapsulation

To configure SDLLC with RSRB using direct encapsulation, perform the following tasks in global configuration mode:

Task	Command
Define a ring group.	**source-bridge ring-group** *ring-group*
Define a remote peer.	**source-bridge remote-peer** *ring-group* **interface** *interface-name* [*mac-address*]

Configure RSRB over FST Connection

To configure SDLLC with RSRB over an FST connection, perform the following tasks in global configuration mode:

Task	Command
Define a ring group.	**source-bridge ring-group** *ring-group*
For FST connection only, set up an FST peer name.	**source-bridge fst-peer***name local-interface-address*
Define a remote peer.	**source-bridge remote-peer** *ring-group* **fst** *ip-address*

Configure RSRB over TCP Connection

To configure SDLLC with RSRB over a TCP connection, perform the following tasks in global configuration mode:

Task	Command
Define a ring group.	**source-bridge ring-group** *ring-group*
Define a remote peer.	**source-bridge remote-peer** *ring-group* **tcp** *ip-address*

Configure SDLLC with RSRB and Local Acknowledgment

RSRB can be configured for only local acknowledgment with RSRB using IP encapsulation over a TCP connection. Configuring SDLLC local acknowledgment can reduce time-outs and keepalive traffic on the connection.

If LLC2 local acknowledgment is configured, it must be configured on the serial interface of the router on the 3x74 cluster controller side of the connection and on the Token Ring interface of the router on the 37x5 FEP side of the connection. Whether or not local acknowledgment is configured, the SNA session extends end-to-end and the SDLC session extends from the router configured with the serial interface to the 3x74 cluster controller. However, the LLC2 session extends from the 37x5 FEP to the router with the Token Ring interface configured. The LLC2 session is locally terminated at that router. A TCP session is then established across the wide-area network (WAN) to router on the 3x74 side of the connection.

To configure the router for SDLLC with RSRB and local acknowledgment, you must perform all the tasks in the "Configure SDLLC with Direct Connection" section earlier in this chapter. In addition, you must perform the following tasks in global configuration mode:

Task	Command
Define a ring group.	**source-bridge ring-group** *ring-group*
Define a remote peer with the local acknowledgment feature.	**source-bridge remote-peer** *ring-group* **tcp** *ip-address* **local-ack**
Enable local acknowledgment for connections involving SDLLC media translation.	**source-bridge sdllc-local-ack**

Local acknowledgment is not supported when the LLC2 device is attached to an Ethernet rather than to a Token Ring.

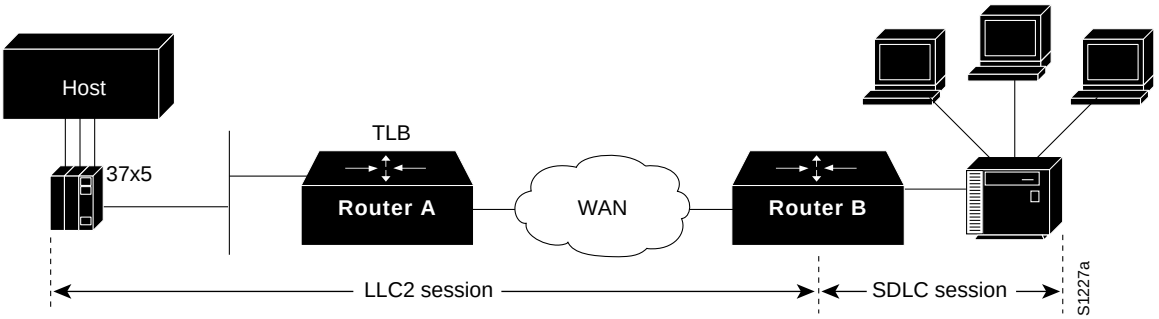
For an example of how to configure SDLCC with RSRB and local acknowledgment, see the section "Example of SDLLC with RSRB and Local Acknowledgment" later in this chapter.

For more information about configuring RSRB and local acknowledgment, see the chapter "Configuring Source-Route Bridging" in this manual and "Source-Route Bridging Commands" in the *Router Products Command Reference* publication.

Configure SDLLC with Ethernet and Translational Bridging

SDLLC support over Ethernet combines translational bridging with Ethernet support of 37x5 FEP connections. Figure 26-2 shows SDLLC with Ethernet and translational bridging. The 3x75 FEP is attached to Router A through Ethernet. The same router is configured for translational bridging, which translates Ethernet packets into Token Ring packets and passes them across the WAN to Router B connected to the 3x74 cluster controller via a serial line. The LLC2 session terminates at the Router B connected to the 3x74 cluster controller. In addition, Router B maintains an SDLC session from itself to the cluster controller.

Figure 26-2 SDLLC with Ethernet and Translational Bridging



Customize SDLLC Media Translation

To increase performance on connections involving SDLLC media translation, perform the tasks in the following sections:

- Set the Largest LLC2 I-Frame Size
- Set the Largest SDLC I-Frame Size
- Increase the SDLC Line Speed

Note the additional information in the section "Other Customizing Considerations" later in this chapter.

Set the Largest LLC2 I-Frame Size

Generally, the router and the LLC2 device with which it communicates should support the same maximum SDLC I-frame size. The larger this value, the better the line is used, thus increasing performance.

Faster screen updates to 3278-style terminals often result by configuring the Token Ring FEP to send as large an I-frame as possible and then allowing the router to segment the frame into multiple SDLC I-frames.

After the Token Ring FEP has been configured to send the largest possible I-frame, it is best to configure the router to support the same maximum I-frame size. The default is 516 bytes. The maximum value the router can support is 8144 bytes.

To set the largest LLC2 I-frame size, perform the following task in interface configuration mode:

Task	Command
Specify the largest I-frame size that can be sent or received by the designated LLC2 primary station.	**sdllc ring-largest-frame** *value*

Set the Largest SDLC I-Frame Size

Generally, the router and the SDLC device with which it communicates should support the same maximum SDLC I-frame size. The larger this value, the better the line is utilized, thus increasing performance.

After the SDLC device has been configured to send the largest possible I-frame, you must configure the router to support the same maximum I-frame size. The default is 265 bytes. The maximum value the router can support must be less than the value of the LLC2 largest frame value defined when setting the largest LLC2 I-frame size.

To set the largest SDLC I-frame size, perform the following task in interface configuration mode:

Task	Command
Set the largest I-frame size that can be sent or received by the designated SDLC station.	**sdlc sdlc-largest-frame** *address value*

Increase the SDLC Line Speed

You can increase the data transfer rate by increasing the SDLC line speed on the serial interface. If possible, increase the link speed of the 3x74 to 19.2 kbps on older units, or to 64 kbps on new units.

To increase the SDLC line speed, perform the following tasks in interface configuration mode:

Task	Command
Adjust the clock rate on the serial interface of the SCI and MCI cards to an acceptable bit rate.	**clockrate** *speed*[1]

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

Other Customizing Considerations

In addition to adjusting the SDLLC parameters described in this section, you can improve performance on the connection by adjusting the LLC2 and SDLC parameters described in the chapter "Configuring LLC2 and SDLC Parameters."

For IBM host configuration consider changing the default MAXOUT (window size) value. Widely used installation guides for IBM equipment show a MAXOUT value of 1 in the VTAM-switched major node for the 3174 PU. Changing this value to 7 improves the performance, because VTAM can send seven frames before requiring an acknowledgment.

Monitor SDLLC Media Translation

To monitor connections using SDLLC media translation, perform the following monitoring tasks in privileged EXEC mode:

Task	Command
Display information about SDLC and LLC2 connections involving interfaces on which SDLLC media translation has been enabled.	**show interfaces**
Display the current state of any connections using local acknowledgment for LLC2 and SDLLC connections.	**show sdllc local-ack**
Display information about LLC2 connections involving interfaces on which SDLLC media translation has been enabled.	**show llc2**[1]

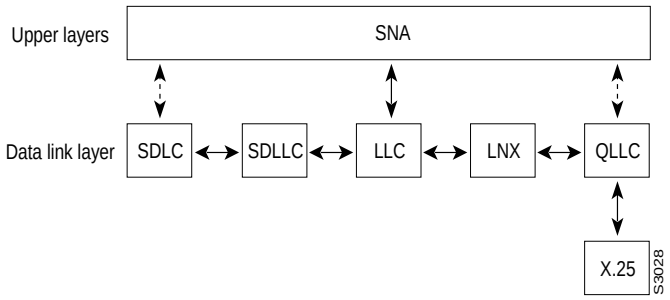
1. This command is documented in the "LLC2 and SDLC Commands" chapter of the *Router Products Command Reference* publication.

In **show llc2** output, look for the LLC2 connections that correspond to the MAC addresses you assigned to the SDLLC interfaces using the **sdllc traddr** command. For information about these commands, see the chapter "LLC2 and SDLC Commands" and "SDLLC Commands" in the *Router Products Command Reference* publication.

QLLC Conversion

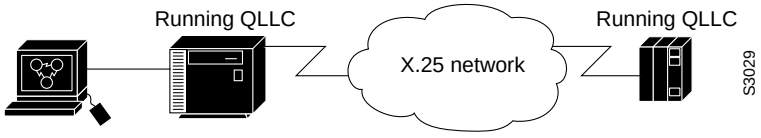
Qualified Logical Link Control (QLLC) is a data link protocol defined by IBM that allows SNA data to be transported across X.25 networks. (Although IBM has defined other protocols for transporting SNA traffic over an X.25 network, QLLC is the most widely used.) Figure 26-3 illustrates how QLLC conversion provides data link layer support for SNA communication.

Figure 26-3 SNA Data Link Layer Support.



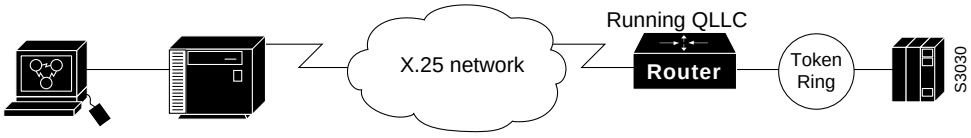
As shown in Figure 26-4, any devices in the SNA communication path that use X.25, whether end systems or intermediate systems, require a QLLC implementation.

Figure 26-4 SNA Devices Running QLLC



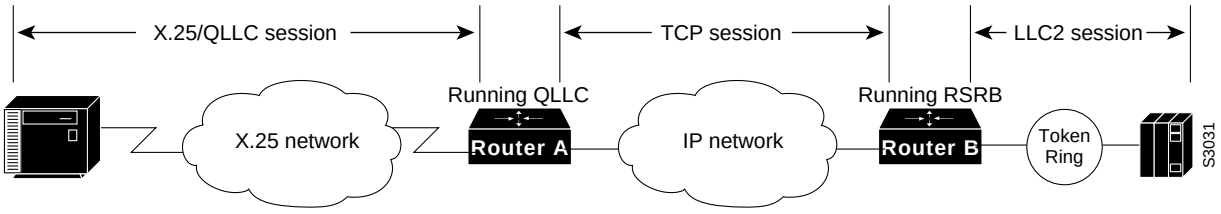
As shown in Figure 26-5, the QLLC conversion feature eliminates the need to install the X.25 software on local IBM equipment. A device that is locally attached to a Token Ring network can communicate through a router that is running the QLLC Conversion feature with a remote device that is attached to an X.25 network using QLLC. Typically, the locally attached device is a front-end processor (FEP), an AS 400, or a PS/2, and the remote device is a terminal controller or a PS/2. In this case, only the remote device needs an X.25 interface and the FEP can communicate with the terminal controller as if it were directly attached via a Token Ring network.

Figure 26-5 Router Running QLLC Conversion Feature



More elaborate configurations are possible. The router that implements QLLC conversion need not be on the same Token Ring network as the FEP. As shown in Figure 26-6, QLLC/LLC2 conversion is possible even when an intermediate IP WAN exists between the router connected to the X.25 network and the router connected to the Token Ring.

Figure 26-6 QLLC Conversion Running on Router with Intermediate IP Network

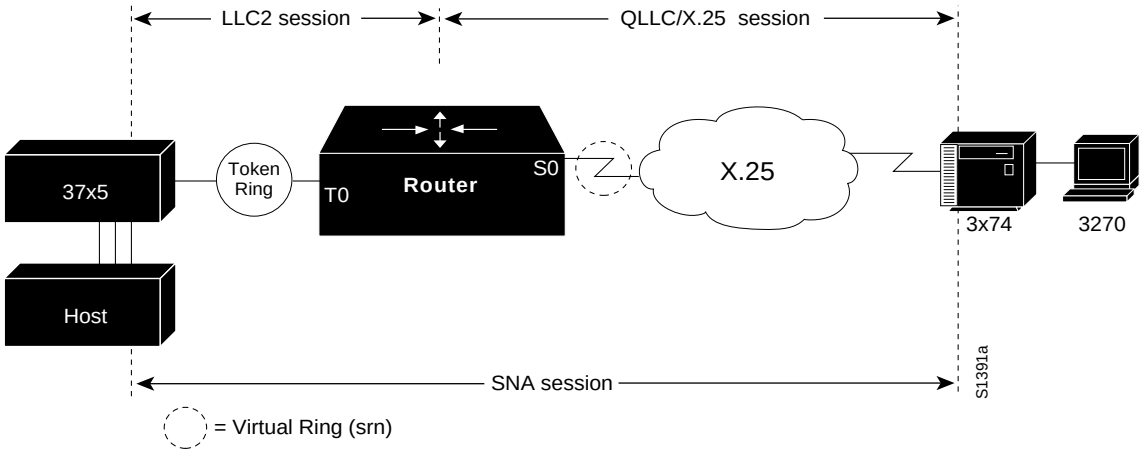


Cisco's Implementation of QLLC Conversion

SNA uses QLLC and X.25 as link-layer protocols to provide a reliable connection. QLLC itself processes QLLC control packets. In a Token Ring environment, SNA uses LLC to provide a reliable connection. The LAN-to-X.25 (LNX) software provides a QLLC conversion function to translate between LLC and QLLC.

Figure 26-7 shows the simplest QLLC conversion topology: a single Token Ring device (for example, a 37x5 FEP) communicates with a single remote X.25 device (in this case a 3x74 cluster controller). A router connects the Token Ring network to the X.25 network.

Figure 26-7 QLLC Conversion between a Single 37x5 and a Single 3x74

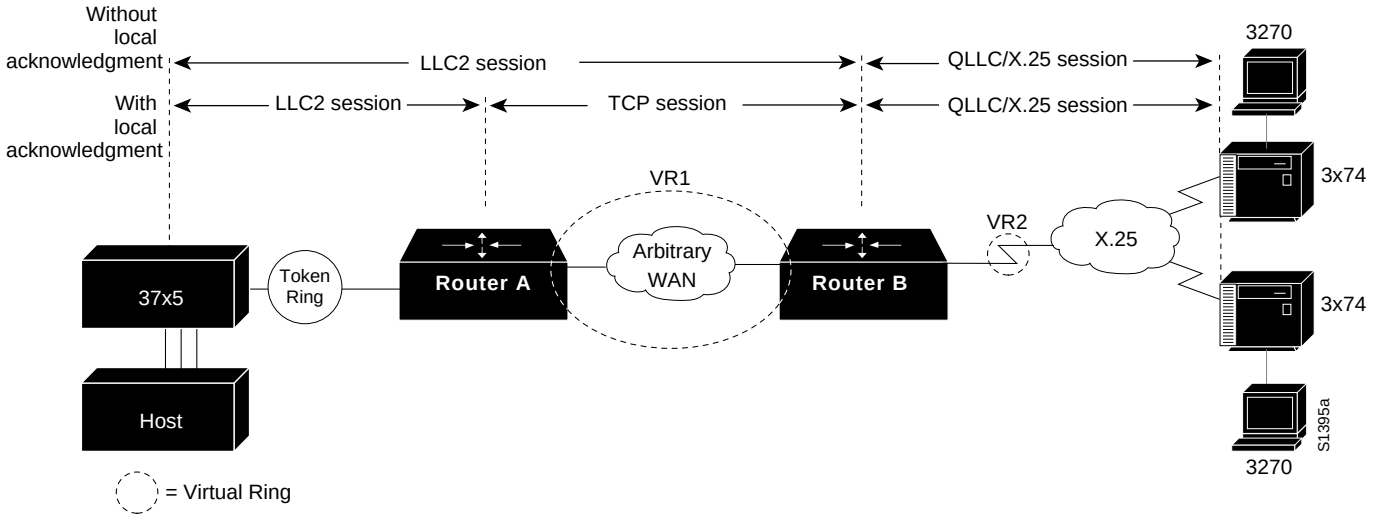


In Figure 26-7, each IBM end node has no indication that its counterpart is connected to a different medium running a different protocol. The 37x5 FEP responds as if the 3x74 cluster controller were communicating over a Token Ring, whereas the 3x74 responds as though the 37x5 FEP were communicating over an X.25 network. This is accomplished by configuring the router's X.25 interface as a virtual Token Ring, so that the X.25 virtual circuit appears to the Token Ring device (and to the router itself) as if it were a Token Ring to which the remote X.25 device is attached.

Also in this figure, the LLC2 connection extends from the 37x5 FEP across the Token Ring network to the router. The QLLC/X.25 session extends from the router across the X.25 network to the 3x74 cluster controller. Only the SNA session extends across the Token Ring and X.25 networks to provide an end-to-end connection from the 37x5 FEP to the 3x74 cluster controller.

As Figure 26-8 shows, a router need not directly connect the two IBM end nodes; instead, some type of backbone WAN can connect them. Here, RSRB transports packets between Router A and Router B, while Router B performs all conversion between the LLC2 and X.25 protocols. Only the router attached to the serial line (Router B) needs to be configured for QLLC conversion. Both Router A and Router B are configured for normal RSRB.

Figure 26-8 QLLC Conversion between a Single 37x5 and Multiple 3x74s across an Arbitrary WAN



How communication sessions are established over the communication link varies depending on whether or not LLC2 local acknowledgment has been configured on Router A's Token Ring interface. In both cases, the SNA session extends end-to-end and the QLLC/X.25 session extends from Router B to the 3x74 cluster controller. If LLC2 local acknowledgment has not been configured, the LLC2 session extends from the 37x5 FEP across the Token Ring network and the arbitrary WAN to Router B. In contrast, when LLC2 local acknowledgment has been configured, the LLC2 session extends from the 37x5 FEP Router A, where it is locally terminated. A TCP session is then used across the arbitrary WAN to Router B.

Comparing QLLC Conversion to SDLLC

Although the procedures you use to configure QLLC are very similar to those used to configure SDLLC, there are structural and philosophical differences between the point-to-point links that SDLC uses and the multiplexed virtual circuits that X.25 uses.

The most significant structural difference between QLLC conversion and SDLLC is the addressing. To allow a device to use LLC2 to transfer data, both SDLLC and QLLC provide virtual MAC addresses. In SDLLC, the actual MAC address is built by combining the defined virtual MAC (whose last byte is 0x00) with the secondary address used on the SDLC link; in this way, SDLLC supports multidrop. In QLLC conversion, multidrop is meaningless, so the virtual MAC address represents just one session and is defined as part of the X.25 configuration. Because one physical X.25 interface can support many simultaneous connections for many different remote devices, you only need one physical link to the X.25 network. The different connections on different virtual circuits all use the same physical link.

The most significant difference between QLLC conversion and SDLLC is the fact that a typical SDLC/SDLLC operation uses a leased line. In SDLC, dial-up connections are possible, but the maximum data rate is limited. In QLLC, both switched virtual circuits (SVCs) and permanent virtual circuits (PVCs) are available, but the favored use is SVC. While the router maintains a permanent connection to the X.25 network, a remote device can use each SVC for some bounded period of time and then relinquish it for use by another device. Using a PVC is very much like using a leased line.

Table 26-1 shows how the QLLC commands correspond to the SDLLC commands.

Table 26-1 QLLC and SDLLC Command Comparison

QLLC Command	Analogous SDLLC Command
qllc largest-packet	**sdllc ring-largest-frame, sdllc sdlc-largest-frame**
qllc partner	**sdllc partner**
qllc sap	**sdllc sap**
qllc srb, x25 map qllc, x25 pvc qllc	**sdllc traddr**
qllc xid	**sdllc xid**
source-bridge qllc-local-ack	**source-bridge sdllc-local-ack**

Other Implementation Considerations

Consider the following when implementing QLLC conversion:

- To use the QLLC conversion feature, a router needs to have a physical link to an X.25 public data network (PDN). It also needs to have an SRB/RSRB path to an IBM FEP. This link could be a Token Ring or Ethernet interface, or even FDDI, if RSRB is being used.
- QLLC conversion can run on any router with at least one serial interface configured for X.25 communication and at least one other interface configured for SRB or RSRB.
- QLLC conversion security depends upon access control in SRB/RSRB and X.25 and upon XID validation.

QLLC Conversion Configuration Task List

Perform the tasks in the following sections to configure QLLC conversion. The first task is required; all others are optional and depend on your specific needs.

- Enable QLLC Conversion on a Serial Interface
- Customize QLLC Conversion
- Monitor QLLC Conversion

See the end of this chapter for QLLC configuration examples.

Enable QLLC Conversion on a Serial Interface

The interfaces you configure for QLLC conversion are the serial interfaces that connect to the X.25 network linking the remote devices with which you plan to communicate. To enable QLLC conversion, you must perform the first of the following tasks. Perform the remaining tasks as appropriate.

- Enable QLLC conversion on the appropriate serial interfaces.
- Define the XID value associated with a remote X.25 device.
- Enable the router to open a connection to the local Token Ring device on behalf of the remote X.25 device.

Enable QLLC Conversion on the Appropriate Serial Interfaces

You can enable QLLC conversion on a serial interface to support either a switched virtual circuit (SVC) or a permanent virtual circuit (PVC). The tasks you perform differ somewhat depending on the type of virtual circuit you plan to support on the interface. In either case, first verify that RSRB is enabled by performing the following task in privileged EXEC mode:

Task	Command
Ensure that RSRB is enabled on the interfaces.	**show configuration**[1]

1. This command is documented in the "System Image, Microcode Image, and Configuration File Load Commands" chapter of the *Router Products Command Reference* publication.

In the sections for the appropriate serial interfaces of the **show configuration** display, look for one or more **source-bridge remote-peer** entries and a **source-bridge** *rn* entry. For more information about configuring a serial interface for RSRB, see the chapter "Configuring LLC2 and SDLC Parameters" in this manual.

To enable QLLC conversion to support an SVC, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Map a virtual Token Ring MAC address for the interface to its X.121 address.	**x25 map qllc** *virtual-mac-addr x121-addr*
Step 2	Enable the use of QLLC conversion on the interface.	**qllc srb** *virtual-mac-addr srn trn*

To enable QLLC conversion to support a PVC, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Set up a PVC for QLLC conversion.	**x25 pvc** *circuit* **qllc** *virtual-mac-addr x121-addr*
Step 2	Enable the use of QLLC conversion on the interface.	**qllc srb** *virtual-mac-addr srn trn*

Define the XID Value Associated with an X.25 Device

The exchange identification (XID) serves as a password to ensure that only those devices that should communicate with the Token Ring host have that privilege. If the XID is defined in NCP on the host, you must enable the router to reply (on behalf of the X.25 device) to the Token Ring host's requests for an XID reply. Although the XID value is used to reply to XID requests received on the LLC2 side of the connection, you apply this command on the serial interface defined for X.25. This XID value must match that of IDBLK and IDNUM defined in the NCP.

Note For most QLLC installations, you do not need to define the XID value. You only need to do so if the remote X.25 device is not configured to send its own XID. This is only possible for a device that is attached through a PVC, although most devices that are connected through X.25 send their own XIDs.

To define the XID value associated with an X.25 device, perform the following task in interface configuration mode:

Task	Command
Specify the XID value appropriate for the X.25 device associated with the Token Ring interface.	**qllc xid** *virtual-mac-addr xid*

Enable the Router to Open a Connection to the Local Token Ring Device

If you plan to use SVCs rather than PVCs, you must enable the router to open a connection to the local Token Ring device on behalf of the remote X.25 device when an incoming call is received. When QLLC conversion is used over an SVC, the remote X.25 device typically initiates the X.25/QLLC session, and the router in turn initiates the LLC2 session.

To enable the router to open a connection to the local Token Ring device, perform the following task in interface configuration mode:

Task	Command
Enable the router to open a connection to the local Token Ring device.	**qllc partner** *virtual-mac-addr mac-addr*

Customize QLLC Conversion

To customize your configuration of QLLC conversion, you can perform one or more of the following tasks:

- Enable QLLC local acknowledgment for remote source-route-bridged connections.
- Specify a SAP value other than the IBM default SAP value.
- Specify the largest packet that can be sent or received on the X.25 interface.

These tasks are described in the following sections.

Enable QLLC Local Acknowledgment for Remote Source-Route-Bridged Connections

Enable local acknowledgment when the round-trip time through the TCP/IP network is as large or larger than the LLC2 timeout period.

To enable QLLC local acknowledgment for RSRB connections, perform the following global configuration task on the router connected to the X.25 interface and configure the remote peers for local acknowledgment:

Task	Command
Enable QLLC local acknowledgment for remote source-route-bridged connections.	**source-bridge qllc-local-ack**

If, for example, Router B with X.25 interface has the IP address *ip1*, and the remote peer (Router A) has the address *ip2*, and they use a virtual ring group *vrg*, then both routers use the following configuration commands:

```
source-bridge ring-group vrg
source-bridge remote-peer vrg tcp ip1 local-ack
source-bridge remote-peer vrg ip2 local-ack
```

You only need to enable QLLC local acknowledgement on Router B, as follows:

```
source-bridge qllc-local-ack.
```

This will not affect Router A.

Specify SAP Values Other Than the Default IBM SAP Values

To use SAP values other than the default IBM SAP values, perform the following task in interface configuration mode:

Task	Command
Specify a SAP value other than the default IBM SAP value.	**qllc sap** *virtual-mac-addr ssap dsap*

Specify the Largest Packet That Can Be Sent or Received on the X.25 Interface

There are two ways for a packet to become segmented:

- The X.25 software performs the segmentation and the other X.25 station re-assembles the packet.
- The QLLC conversion performs SNA header segmentation. In this case, QLLC does not re-assemble, but passes smaller SNA segments to the IBM end station.

If the QLLC software does not perform SNA segmentation, then the X.25 software must be capable of performing X.25 segmentation of the largest packet that it can receive from the LLC2 side. This packet can be several thousand bytes long and the typical size for X.25 packets is 1024 bytes or less. (The default is 128, but that can be overridden with larger values.) The X.25 software, especially in the X.25 attached IBM end station, might not be able to reassemble a very large packet. In this situation, specifying the largest QLLC packet can be useful.

By default, the maximum SNA data unit size established for the virtual circuit is the maximum packet size that can be sent or received on the X.25 interface. If packets received on the LLC2 interface are larger than the largest value allowed on the X.25 connection, they can be segmented by the X.25 software before being sent on the X.25 interface. Moreover, there is no reassembly on receiving packets on the X.25 interface before sending them on the LLC2 interface. Thus, you might need to reconfigure the maximum packet size for the X.25 interface to match that for the LLC2 interface.

When the remote X.25 device has a limit on the maximum total length of recombined X.25 segments it will support, you must ensure the length is not exceeded. For example, a device whose maximum SNA packet size is limited to 265 bytes might not be able to handle a series of X.25 packets that it has to recombine to make a 4, 8, or 17 KB SNA packet, such as one often encounters in an LLC2 environment.

You cannot configure the X.25 interface with a larger packet size than the LLC2 interface.

To specify the largest packet that can be sent or received on the X.25 interface, perform the following task in interface configuration mode:

Task	Command
Specify the largest packet that can be sent or received on the X.25 interface.	**qllc largest-packet** *virtual-mac-address max-size*

Monitor QLLC Conversion

To monitor connections using QLLC conversion, perform the following tasks in privileged EXEC mode:

Task	Command
Display information about X.25 and LLC2 connections involving interfaces on which QLLC conversion has been enabled.	**show interfaces serial** *unit*[1]
Display the current state of any connections using QLLC local acknowledgment.	**show qllc**
Display information about LLC2 connections involving interfaces on which QLLC conversion has been enabled.	**show llc2**[2]

1. This command is documented in the " Interface Commands" chapter of the *Router Products Command Reference* publication.
2. This command is documented in the "LLC2 and SDLC Commands" chapter of the *Router Products Command Reference* publication.

SDLLC Configuration Examples

The following sections provide SDLLC configuration examples:

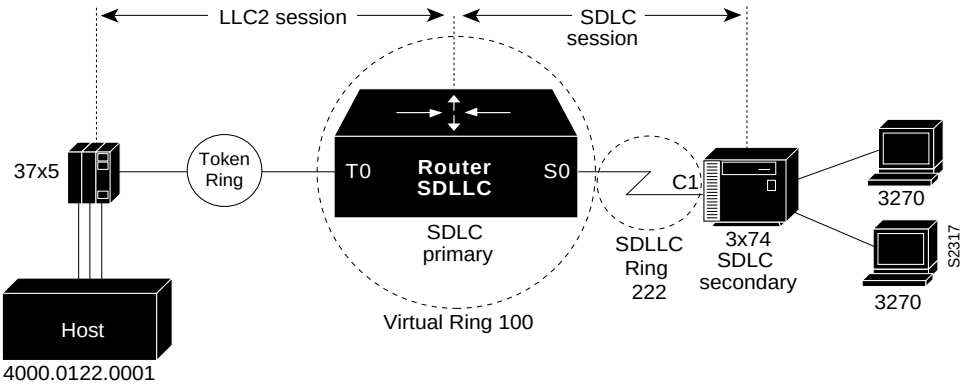
- Example of SDLLC with Direct Connection
- Example of SDLLC with Single Router Using RSRB
- Example of SDLLC with RSRB (Single 3x74)
- Example of SDLLC with RSRB (Multiple 3x74s)
- Example of SDLLC with RSRB and Local Acknowledgment
- NCP and VTAM Sysgen Parameters

Following the examples are sample NCP definitions that the 37x5 FEP in these topologies could use and VTAM definitions that the IBM host in these topologies could use to reflect the routers in the communication path.

Example of SDLLC with Direct Connection

Figure 26-9 shows a router configuration when the router directly connects the Token Ring and the serial line. The router is configured with SRB.

Figure 26-9 SDLLC Communication between a 37x5 and a 3x74 Connected to the Same Router (Direct Connection)



A configuration file that enables direct connection follows:

```
source-bridge ring-group 100
!
interface tokenring 0
source-bridge 111 1 100
!
interface serial 0
encapsulation sdlc-primary
sdlc address c1
sdllc traddr 0110.2222.3300 222 2 100
sdllc partner 4000.0122.0001 c1
sdllc xid c1 1720001
```

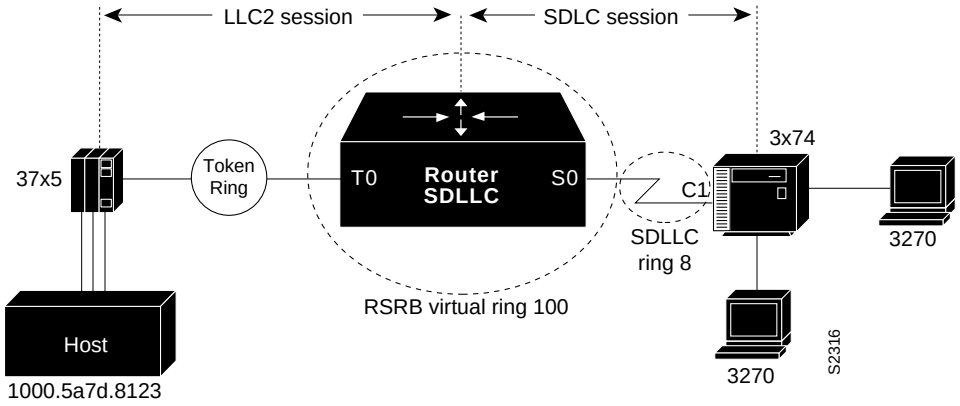
Example of SDLLC with Single Router Using RSRB

Figure 26-10 shows a router configuration in which the router directly connects the Token Ring and the serial line but uses RSRB to create a virtual ring 100. This configuration has the following characteristics:

- The FEP sees c1 3174 at MAC address 0110.2222.33c1
- The RIF from the FEP to the devices would appear as:

  ring 1—bridge 1—ring 100—bridge 1—ring 8

Figure 26-10 SDLLC with Single Router with RSRB



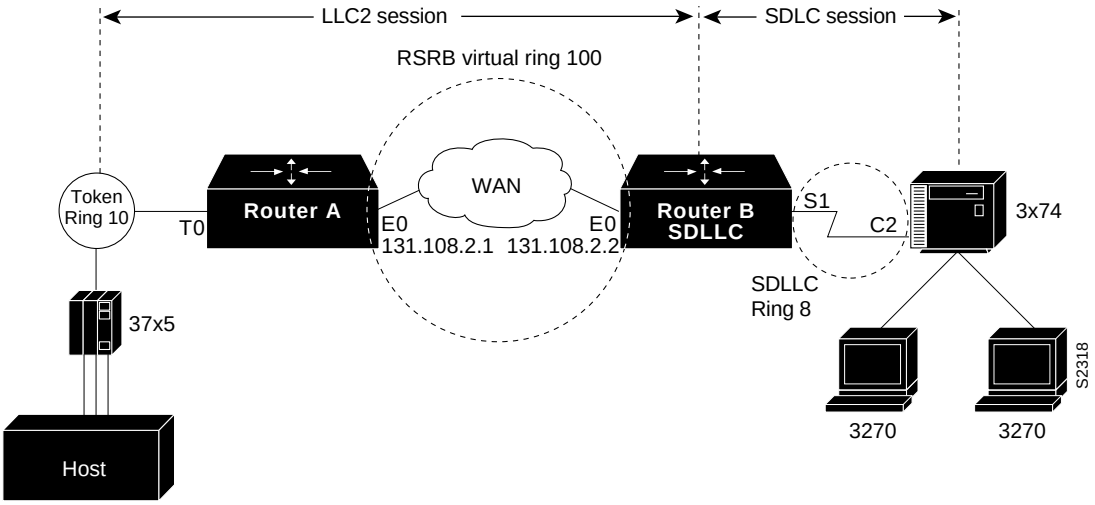
The following sample configuration file is for SDLLC with a single router using RSRB:

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.1.1
source-bridge remote-peer 100 tcp 131.108.2.2

interface tokenring 0
ip address 131.108.2.2 255.255.255.0
source-bridge 111 1 100

interface serial 0
encapsulation sdlc-primary
sdlc address c1
sdllc traddr 0110.2222.3300 8 1 100
sdllc partner 1000.5a7d.8123 c1
sdllc xid c1 17200c1
```

Example of SDLLC with RSRB (Single 3x74)

In Figure 26-11, SDLLC with RSRB connects an FEP and a single 3x74 cluster controller. The host wants to communicate with a single 3174 that its FEP sees on a Token Ring. However, the 3x74 seen by the FEP is in fact SDLC device C1 connected by means of a serial link through a remote router.

Figure 26-11 SDLLC with RSRB with Single 3x74



The configuration files for the network shown in Figure 26-11 follow.

Configuration for Router A

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.2.2
!
interface tokenring 0
ip address 131.108.1.1 255.255.255.0
source-bridge 10 1 100
!
interface ethernet 0
ip address 131.108.2.1 255.255.255.0
```

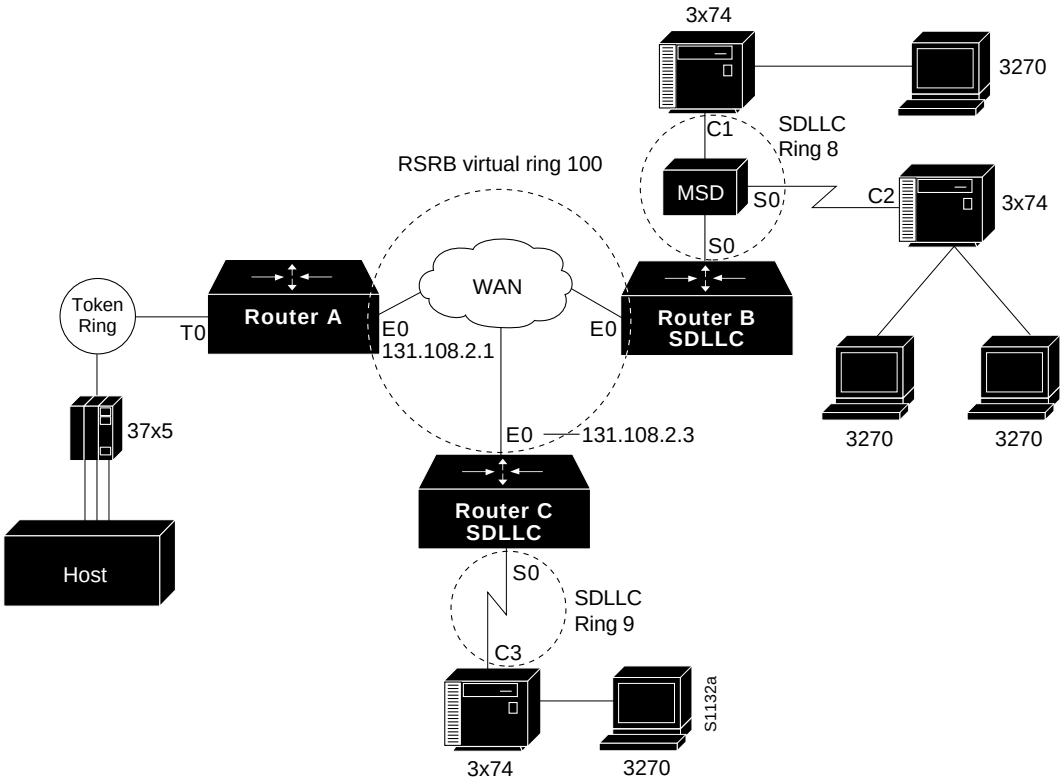
Configuration for Router B

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.1.1
source-bridge remote-peer 100 tcp 131.108.2.2
!
interface tokenring 0
ip address 131.108.2.2 255.255.255.0
source-bridge 1 1 100
!
interface serial 0
encapsulation sdlc-primary
sdlc address c1
sdllc traddr 0110.2222.3300 8 1 100
sdllc partner 1000.5a7d.8123 c1
sdllc xid c1 17200c1
```

Example of SDLLC with RSRB (Multiple 3x74s)

In the setup shown in Figure 26-12, Router A needs no SDLLC configuration, Router B has the SDLLC configuration and supports multipoint on the SDLC link with a modem-sharing device and Router C is also configured with SDLLC. For information about the NCP and VTAM system generation (sysgen) parameters that are used in this configuration see the "NCP and VTAM Sysgen Parameters" section later in this chapter.

Figure 26-12 SDLLC with RSRB (Multiple 3x74s)



The following configuration files describe the network shown in Figure 26-12. The note references to the right of the configuration files refer to the "Notes" section at the end of this chapter.

Configuration for Router A

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.2.1
source-bridge remote-peer 100 tcp 131.108.2.2
source-bridge remote-peer 100 tcp 131.108.2.3
!
interface tokenring 0
ip address 131.108.1.1 255.255.255.0
source-bridge 10 1 100
!
interface ethernet 0
ip address 131.108.2.1 255.255.255.0
```

Configuration for Router B

```
source-bridge ring-group 100
```

```
source-bridge remote-peer 100 tcp 131.108.2.1
source-bridge remote-peer 100 tcp 131.108.2.2
source-bridge remote-peer 100 tcp 131.108.2.3
!
interface ethernet 0
ip address 131.108.2.2 255.255.255.0
!
interface serial 0
encapsulation sdlc-primary
sdlc address c1
sdllc traddr 0110.2222.3300 7 1 100
sdllc partner 1000.5a7d.8123 c1
sdllc xid c1 17200c1
!
interface serial 1
encapsulation sdlc-primary
sdlc address c2
sdllc traddr 0220.3333.4400 7 1 100
sdllc partner 1000.5a7d.8123 c2 MUST MATCH TIC LOCADD, NOTE 2
sdllc xid c2 17200c2MUST MATCH VTAM IDBLK/IDNUM, NOTE 4
```

Configuration for Router C

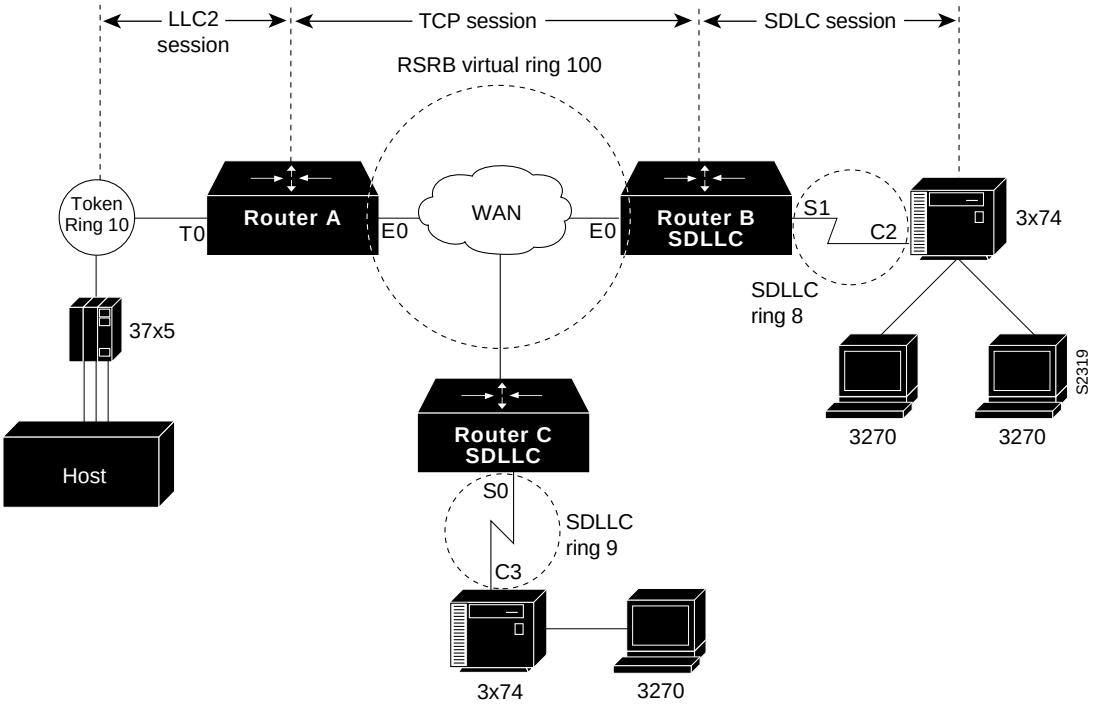
```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.2.1
source-bridge remote-peer 100 tcp 131.108.2.2
source-bridge remote-peer 100 tcp 131.108.2.3
!
interface ethernet 0
ip address 131.108.2.3 255.255.255.0
!
interface serial 0
encapsulation sdlc-primary
sdlc address c3
sdllc traddr 0110.2222.3300 9 1 100
sdllc partner 1000.5a7d.8123 c3 MUST MATCH TIC LOCADD, NOTE 2
sdllc xid c3 17200c3MUST MATCH VTAM IDBLK/IDNUM, NOTE 4
```

Example of SDLLC with RSRB and Local Acknowledgment

The configuration shown in Figure 26-13 enables local acknowledgment for Router B, which means that the LLC session terminates at Router A. However, the LLC2 session between Router A and Router C is not locally acknowledged and terminates at Router C.

For information about the NCP and VTAM system generation (sysgen) parameters that are used in this configuration see the "NCP and VTAM Sysgen Parameters" section later in this chapter.

Figure 26-13 SDLLC with RSRB and Local Acknowledgment



The following sample configuration files describe the network shown in Figure 26-13. (The notes in the sample configuration files refer to the "Notes" section at the end of this chapter.)

Configuration for Router A

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.2.1
source-bridge remote-peer 100 tcp 131.108.2.2 local-ack
source-bridge remote-peer 100 tcp 131.108.2.3
!
interface tokenring 0
ip address 131.108.1.1 255.255.255.0
source-bridge 1 1 100
!
interface ethernet 0
ip address 131.108.2.1 255.255.255.0
```

Configuration for Router B

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.2.1 local-ack
source-bridge remote-peer 100 tcp 131.108.2.2
source-bridge remote-peer 100 tcp 131.108.2.3

source-bridge sdllc local-ack
!
interface ethernet 0
ip address 131.108.2.2 255.255.255.0
!
interface serial 0
encapsulation sdlc-primary
sdlc address c1
sdllc traddr 4000.3174.0b0d 7 1 100
sdllc partner 1000.5a7d.8123 c1
!Must match TIC LOCADD [See NOTE 2]
sdllc xid c1 017200c1
!Must match VTAM IDBLK/IDNUM [See NOTE 4]
interface serial 1
encapsulation sdlc-primary
sdlc address c2
sdllc traddr 0110.2222.3200 8 1 100
sdllc partner 1000.5a7d.8123 c2
!Must match TIC LOCADD [See NOTE 2]
sdllc xid c2 017200c2
!Must match VTAM IDBLK/IDNUM [See NOTE 4]
```

Configuration for Router C

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.2.1
source-bridge remote-peer 100 tcp 131.108.2.2
source-bridge remote-peer 100 tcp 131.108.2.3
!
interface ethernet 0
ip address 131.108.2.3 255.255.255.0
!
interface serial 0
encapsulation sdlc-primary
sdlc address c3
sdllc traddr 4000.3174.0c00 9 1 100
sdllc partner 1000.5a7d.8123 c3
Must match TIC LOCADD [See NOTE 2]
sdllc xid c3 017200c3
!Must match VTAM IDBLK/IDNUM [See NOTE 4]
```

QLLC Conversion Configuration Examples

The following sections provide QLLC conversion configuration examples:

- QLLC Conversion between a Single 37x5 and a Single 3x74 Example
- QLLC Conversion between a Single 37x5 and Multiple 3x74s Example
- QLLC Conversion between Multiple 37x5s and Multiple 3x74s Example
- QLLC Conversion between a Single 37x5 and Multiple 3x74s across an Arbitrary WAN Example

- NCP and VTAM Sysgen Parameters

The examples describe four increasingly complex QLLC conversion topologies and possible router configurations for each. Following the examples are sample NCP definitions that the 37x5 FEP in these topologies could use and VTAM definitions that the IBM host in these topologies could use to reflect the routers in the communication path.

QLLC Conversion between a Single 37x5 and a Single 3x74 Example

Figure 26-7, shown previously, illustrates the simplest QLLC conversion topology—a single 37x5 FEP on a Token Ring communicating with a single 3x74 cluster controller across an X.25 network. A router connects the Token Ring to the X.25 network. In Figure 26-7, notice that the router's X.25 interface is treated as a virtual ring for configuration purposes.

The following configuration file configures the router to support the network topology shown in Figure 26-7:

```
source-bridge ring-group 100
!
interface serial 0
encapsulation x25
x25 address 31102120100
x25 map qllc 0100.0000.0001 31104150101
qllc srb 0100.0000.0001 201 100
!
! Allow the 3174 to initiate the connection.
!
qllc partner 0100.0000.0001 4000.0101.0132

interface tokenring 0
source-bridge 100 1 201
```

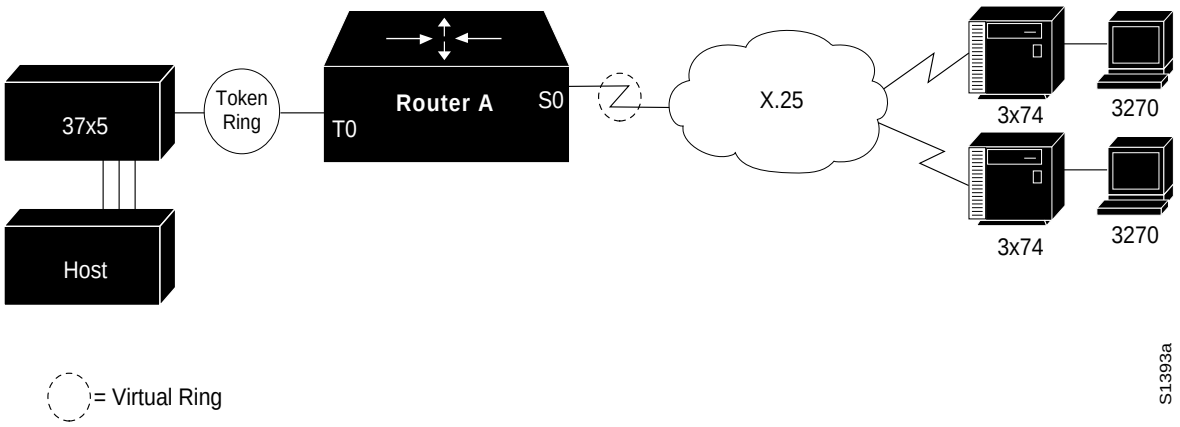
In this configuration file, the **source-bridge ring-group** command defines a virtual ring number 100. The serial 0 interface that connects to the X.25 network is then configured for X.25 DTE operation using the **encapsulation x25** command and assigned the X.121 address of 31102120100 using the **x25 address** command. The **x25 map qllc** command associates the X.121 address of the remote X.25 device (31104150101) with a virtual Token Ring MAC address (0100.0000.0001) the Token Ring device will use to communicate with this remote X.25 device. The **qllc srb** command indicates that the virtual MAC address of the X.25 device will be used to communicate with the real MAC address of the Token Ring device.

The **qllc partner** command enables the router to open a connection to the local Token Ring device at MAC address 4000.0101.0132 on behalf of the remote X.25 device at virtual Token Ring MAC address 0100.0000.0001. The **source-bridge** command configures the router's Token Ring 0 interface for local source-route bridging by associating the router's virtual ring number 100 with the ring number (1) of the local Token Ring and the bridge number (1) that uniquely identifies this bridge interface.

QLLC Conversion between a Single 37x5 and Multiple 3x74s Example

Figure 26-14 shows a slightly more complex QLLC conversion topology—the same 37x5 FEP on a Token Ring connects through a router to an X.25 network—but communicates with multiple 3x74 cluster controllers through X.25.

Figure 26-14 QLLC Conversion between a Single 37x5 and Multiple 3x74s



The following configuration file configures the router to support the network topology shown in Figure 26-14:

```
source-bridge ring-group 100
!
interface serial 0
encapsulation x25
x25 address 3137005469
!
! configure the first 3174
!
x25 map qllc 0000.0cff.0001 31370054111
!
! 1001 - virtual ring used by all qllc devices
! 100 - the virtual ring group
!
qllc srb 0000.0cff.0001 1001 100
qllc partner 0000.0cff.0001 4000.1160.0000
qllc xid 0000.0cff.0001 01710017
!
! configure the second 3174
!
x25 map qllc 0000.0cff.0002 313700543247
!
! 1001 - virtual ring used by all qllc devices
! 100 - the virtual ring group
!
qllc srb 0000.0cff.0002 1001 100
qllc partner 0000.0cff.0002 4000.1160.0000
qllc xid 0000.0cff.0002 01710017
!
interface Tokenring 0
!
! Since this is a real bridge, we have to define the way it
! bridges to the Qllc virtual ring.
!
source-bridge 1 1 100
source-bridge spanning
```

QLLC Conversion between Multiple 37x5s and Multiple 3x74s Example

In the following example, two 3x74s on a Token Ring each attach to a different 37x5 on the other side of an X.25 network. Only one Token Ring interface is used. Do not create a bridge from the QLLC virtual ring (1001) to the physical Token Ring (1). Instead, define a virtual ring group (for example, 100).

```
interface serial 0
encapsulation x25
x25 address 3137005469
!
! configure the router for the first 3x74
!
x25 map qllc 0000.0cff.0001 31370054111
!
! 1001 - virtual ring used by all qllc devices
! 1 - the local Token Ring number
!
qllc srb 0000.0cff.0001 1001 1
qllc partner 0000.0cff.0001 4000.1160.0000
!
! configure the router for the second 3x74
!
x25 map qllc 0000.0cff.0002 31370053247
!
! 1001 - virtual ring used by all qllc devices
! 1 - the local Token Ring number
!
! Note that the partner's MAC address and XID are different from
! those in the first 3x74.
!
qllc srb 0000.0cff.0001 1002 1
qllc partner 0000.0cff.0002 4000.1161.1234
!
interface Tokenring 0
!
! Since this is a real bridge, we have to define the way it bridges
! to the QLLC virtual ring.
!
source-bridge 1 1 1001
source-bridge spanning
```

QLLC Conversion between a Single 37x5 and Multiple 3x74s across an Arbitrary WAN Example

Figure 26-8, shown previously, includes an added arbitrary WAN in the communication path between the 37x5 FEP and the multiple 3x74 cluster controllers. The arbitrary WAN can be a multihop network, whereas QLLC conversion treats the X.25 network as a single-hop network.

In Figure 26-8, notice that the arbitrary WAN and the routers on either side of it form a single virtual ring, as configured using the **source-bridge ring group** global command.

In this configuration file, Router A uses an IP address of 131.108.2.2 and its Token Ring interface is attached to Token Ring 1. Because Router A connects to the Token Ring, it does not need to be configured for QLLC conversion. Router B, configured for QLLC conversion because it connects directly to the X.25 network through its serial interface, uses an X.121 address of 31102120100 and an IP address of 131.108.1.1. The 37x5 device uses a MAC address of 4000.0101.0132. The virtual MAC address of 0100.0000.0001 has been assigned to the 3x74 device.

Sample Configuration for Router A

The following configuration file configures the Router A in Figure 26-8:

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.1.1 local-ack
source-bridge remote-peer 100 tcp 131.108.2.2 local-ack
!
interface ethernet 0
ip address 131.108.3.3 255.255.255.0
!
source-bridge 1 1 100
source-bridge spanning
!
interface tokenring 0
ip address 131.108.2.2 255.255.255.0
```

Sample Configuration for Router B

The following configuration file configures the Router B in Figure 26-8:

```
source-bridge ring-group 100
source-bridge remote-peer 100 tcp 131.108.1.1 local-ack
source-bridge remote-peer 100 tcp 131.108.2.2 local-ack
source-bridge qllc-local-ack
!
interface serial 0
encapsulation x25
x25 address 31102120100
x25 map qllc 0100.0000.0001 31104150101
x25 map qllc 0100.0000.0002 31104150102
qllc srb 0100.0000.0001 201 100
qllc srb 0100.0000.0002 201 100
!
! Allow the 3174 to initiate the connection.
!
qllc partner 0100.0000.0001 4000.0101.0132
qllc partner 0100.0000.0002 4000.0101.0132
!
interface ethernet 0
ip address 131.108.1.1 255.255.255.0
```

NCP and VTAM Sysgen Parameters

The sample system generation (sysgen) parameters in this section show typical NCP and VTAM values that correspond with the Router A, Router B, and Router C configurations shown in Figure 26-12 and Figure 26-13 for SDLLC media translation and in Figure 26-8 for QLLC conversion.

IBM's ACF/NCP uses a function called NTRI (NCP/Token Ring Interconnection) to support Token Ring-attached SNA devices. NTRI also provides translation from Token Ring-attached SNA devices (Physical Units) to switched (dial-up) devices. VTAM provides the resolution for these devices in a Switched Major Node. VTAM treats these devices on NTRI logical lines as switched devices. (For more information consult IBM documentation *NCP/SSP/EP Resource Definition Reference,* SC30-3448-04.)

Using SDLLC, the Cisco router translates SDLC leased line protocol into Token Ring LLC2 protocol, then the NTRI function in ACF/NCP translates Token Ring LLC2 protocol into an SNA switched protocol.

NCP Generation Definitions

```
*****************************************************************
***              SAMPLES BASED ON ACF/NCP V5 R4.
***            NOT ALL NCP PARAMETERS ARE SHOWN
*****************************************************************
*
*****************************************************************
*     OPTIONS DEFINITION STATEMENT
*****************************************************************
NCPOPTOPTIONS      NEWDEFN=YES NTRI GENERATION, MUST BE FIRST STMT
*
*****************************************************************
*     BUILD MACRO
*****************************************************************
NCPBU BUILD        LOCALTO=1.5,NTRI ACK TIMER FOR LOCAL TOKEN RINGS
                   REMOTTO=2.5,NTRI ACK TIMER FOR REMOTE TOKEN RINGS
                               USED IN SDLLC CONFIGURATIONS, NOTE 1
*
*****************************************************************
*     DYNAMIC RECONFIGURATION POOL SPACE
*****************************************************************
DRPOOLLUDRPOOL     NUMTYP2=50  RESERVE 50 LUS ON PU. T2 PUS
*
*****************************************************************
*     PHYSICAL GROUP FOR NTRI TIC #1, DEFINITIONS FOR THE TOKEN RING
*     ADAPTER TO ESTABLISH PHYSICAL CONNECTIVITY
*****************************************************************
EPHYG GROUP        ECLTYPE=PHYSICAL
*
EPHYL LINE         ADAPTER=TIC2,          TYPE OF ADAPTER
                   ADDRESS=(16,FULL),     INTERNAL FEP TIC ADDRESS
                   PORTADD=0,
                   LOCADD=10005a7d8123,TIC ADDRESS, NOTE 2
                   RCVBUFC=1440,
                   MAXTSL=2012,
                   TRSPEED=16             TOKEN RING SPEED
*
EPHYPUPU
*
EPHYLULU           ISTATUS=INACTIVE
*
*****************************************************************
*     NTRI PERIPHERAL LOGICAL LINE GROUP, LINE AND PU PAIRS ARE
*     GENERATED BY THE AUTOGEN PARAMETER.
*****************************************************************
ELOGG GROUP        ECLTYPE=LOGICAL,
                   PHYPORT=0,
                   CALL=INOUT,
                   AUTOGEN=3              ONE PER SDLLC CONTROLLER,
                                          NOTE 3
*****************************************************************
```

VTAM Definitions

```
*************************************************************
*         VTAM SWITCHED MAJOR NODE, BASED ON ACF/VTAM V3 R4.
*         THE CODING BELOW SUPPORTS DIAL IN OPERATION ONLY. TYPICALLY,
*         NTRI IMPLEMENTATIONS USE ONLY DIAL IN. IF DIAL OUT FROM AN
*         APPLICATION IS REQUIRED, PATH MACROS MUST BE USED. CONSULT
*         THE APPROPRIATE VTAM INSTALLATION REFERENCE MANUAL.
*************************************************************
VSWITCH   VBUILD         TYPE=SWNET
*
VPU1      PU             ADDR=13,       COULD BE ANYTHING (NOT USED)
                         IDBLK=017,     XID PARM, NOTE 4
                         IDNUM=200c1,   XID PARM, NOTE 4
                         MAXOUT=7,
                         MAXDATA=265,
                         MODETAB=AMODETAB,
                         DLOGMOD=US327X,
                         PUTYPE=2,
                         USSTAB=USS327X
*
VLU1A     LU             LOCADDR=2,
VLU1B     LU             LOCADDR=3
*
VPU2      PU             ADDR=13,       COULD BE ANYTHING (NOT USED)
                         IDBLK=017,     XID PARM, NOTE 4
                         IDNUM=200c2,   XID PARM, NOTE 4
                         MAXOUT=7,
                         MAXDATA=265,
                         MODETAB=AMODETAB,
                         DLOGMOD=US327X,
                         PUTYPE=2,
                         USSTAB=USS327X
*
VLU2A     LU             LOCADDR=2,
VLU2B     LU             LOCADDR=3
*
VPU3      PU             ADDR=13,       COULD BE ANYTHING (NOT USED)
                         IDBLK=017,     XID PARM, NOTE 4
                         IDNUM=200c3,   XID PARM, NOTE 4
                         MAXOUT=7,
                         MAXDATA=265,
                         MODETAB=AMODETAB,
                         DLOGMOD=US327X,
                         PUTYPE=2,
                         USSTAB=USS327X
*
VLU3A     LU             LOCADDR=2,
VLU3B     LU             LOCADDR=3
*
```

Notes

In these sample definitions:

1. REMOTTO is the NCP's T1 timer for remote Token Rings. All connections use RIF information and therefore look like remote Token Ring devices. The default is 2.5 seconds, which is adequate for most situations; however, when slow-speed links are used, this parameter should be reviewed to ensure enough time for link-level acknowledgments.
2. The LOCADD parameter defines the locally administered address of the TIC in the NCP. The Cisco routers, configured for SDLLC, will insert this address as the 802.5 destination address field in TEST and XID frames to establish connectivity and then in data frames during the session. The **sdllc partner** and **qllc partner** commands define this connection in the Cisco routers. Each SDLC control unit is defined with an **sdllc partner** or **qllc partner** command.
3. The AUTOGEN parameter specifies the number of LINE and PU pairs that are automatically generated by NDF (Network Definition Facility). Each controller requires a LINE and PU definition in the ELCTYPE LOGICAL group. These represent control block space in the NCP simulating switched line as described earlier.
4. The IDBLK and IDNUM parameters in VTAM are used to identify incoming connection requests. IDBLK is typically unique for each type of IBM device. IDNUM is any five hex digit combination. The Cisco routers configured for SDLLC or QLLC conversion must associate an IDBLK/IDNUM combination with a controller by using the **sdllc xid** or **qllc xid** command. If not using the **qllc xid** command, then IDBLK/IDNUM must agree with the values of the X.25 attached devices. During activation, an XID will be sent to the NCP containing the specific IDBLK/IDNUM. NCP will send these values to VTAM in an SNA command called REQCONT. VTAM will search its switched major nodes to find a match. If found, VTAM will establish sessions with the device by sending activation commands (ACTPU, ACTLUs).

CHAPTER 27

Configuring DSPU

This chapter describes our support for Systems Network Architecture (SNA) downstream physical unit (DSPU) devices. For a complete description of the commands mentioned in this chapter, refer to the "DSPU Configuration Commands" chapter of the *Router Products Command Reference* publication.

Cisco's Implementation of DSPU

DSPU is a software feature that enables the router to function as a physical unit (PU) concentrator for SNA PU type 2 nodes. PU concentration at the router simplifies the task of PU definition at the upstream host while providing additional flexibility and mobility for downstream PU devices.

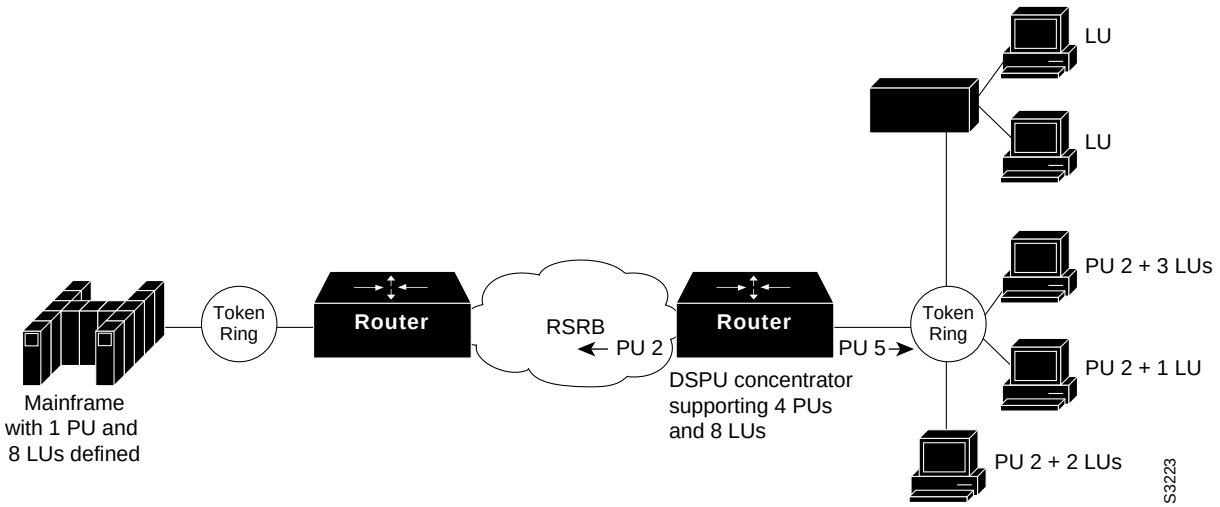
The DSPU feature allows you to define downstream PU type 2 devices in the router. DSPU reduces the complexity of host configuration by letting you replace multiple PU definitions that represent each downstream device with one PU definition that represents the router.

Because you define the downstream PUs at the router rather than the host, you isolate the host from changes in the downstream network topology. Therefore you can insert and remove downstream PUs from the network without making any changes on the host.

The concentration of downstream PUs at the router also reduces network traffic on the wide-area Network (WAN) by limiting the number of sessions that must be established and maintained with the host. The termination of downstream sessions at the router ensures that idle session traffic does not appear on the WAN.

Figure 27-1 shows a router functioning as a DSPU concentrator.

Figure 27-1 Router Acting as a DSPU Concentrator

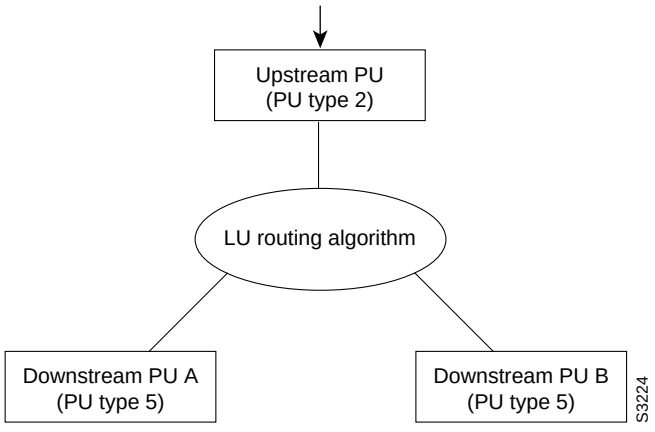


Typically, the router establishes one or more upstream connections with one or more hosts and many downstream connections with PU type 2 devices. From an SNA perspective, the router appears as a PU type 2 device to the upstream host and assumes the role of a System Services Control Point (SSCP) appearing as a PU type 5 device to its downstream PUs.

The SSCP sessions established between the router and its upstream host are completely independent of the SSCP sessions established between the router and its downstream PUs. SNA traffic is routed at a logical unit (LU) level using a routing algorithm that maps downstream LUs onto upstream LUs.

Figure 27-2 illustrates the SNA perspective of DSPU.

Figure 27-2 SNA Perspective of DSPU



DSPU Configuration Task List

To configure DSPU, perform the tasks in the following sections. The last two tasks are optional.

- Define DSPU Upstream Hosts
- Define Downstream PUs
- Define DSPU LUs
- Configure DSPU to Use a Data Link Control
- Define the Number of Outstanding, Unacknowledged Activation RUs
- Monitor DSPU Feature Status

See the end of this chapter for "DSPU Configuration Examples."

Define DSPU Upstream Hosts

The upstream host provides LUs that the router will assign for use by its downstream PUs. Since one upstream host can only provide a maximum of 255 LUs, the DSPU feature supports multiple hosts. Multiple upstream host support allows the DSPU router to provide more than 255 LUs for use by its downstream PUs.

To define a DSPU host, perform the following task in global configuration mode:

Task	Command
Define a DSPU host.	**dspu host** *host-name* **xid-snd** *xid* **rmac** *remote-mac* [**rsap** *remote-sap*] [**lsap** *local-sap*] [**window** *window-size*] [**maxiframe** *max-iframe*] [**retries** *retry-count*] [**retry-timeout** *retry-timeout*] [**focalpoint**]

Define Downstream PUs

To define the downstream PUs, perform either of the tasks in the following sections, depending on your circumstances:

- Explicitly Define a Downstream PU
- Enable the Default PU Option

Explicitly Define a Downstream PU

Explicitly define a downstream PU if you require the router to perform verification checking on incoming downstream connections or to initiate an outgoing downstream connection.

To explicitly define a downstream PU, perform the following task in global configuration mode:

Task	Command
Explicitly define a downstream PU.	**dspu pu** *pu-name* [**rmac** *remote-mac*] [**rsap** *remote-sap*] [**lsap** *local-sap*] [**xid-rcv** *xid*] [**window** *window-size*] [**maxiframe** *max-iframe*] [**retries** *retry-count*] [**retry-timeout** *retry-timeout*]

If the router will perform verification checking on incoming downstream connections, there are several combinations of parameters that you can configure for verification matching:

- **xid-rcv**—Match on *xid* value only.
- **rmac**/**rsap**—Match on *remote-mac* and *remote-sap* only.
- **rmac**/**rsap** and **xid-rcv**—Match on values r*emote-mac*, *remote-sap*, and *xid*.

The router will reject any incoming downstream connections that do not match the parameters of a defined downstream PU unless the default PU option is also enabled.

Enable the Default PU Option

Configure the DSPU default PU option if you do not require the router to verify incoming downstream connections. The default PU option allows the router to accept incoming downstream connections without an explicit definition for the downstream PU.

To enable the default PU option, perform the following task in global configuration mode:

Task	Command
Enable the default PU option.	**dspu default-pu** [**window** *window-size*] [**maxiframe** *max-iframe*]

Define DSPU LUs

You must specify the LU routing algorithm used to map the upstream LUs to the downstream LUs and to define all LUs for each upstream and downstream PU.

The DSPU feature assigns upstream LUs onto downstream LUs based on the selected LU routing algorithm and performs the mapping necessary for SNA data transfer.

The DSPU feature supports two alternative mapping algorithms that are described in the following sections:

- Dedicated LU Routing
- Pooled LU Routing

An upstream host PU or downstream PU can support up to 255 LU sessions. The DSPU feature allows each LU to be individually configured for either dedicated LU routing or pooled LU routing.

Dedicated LU Routing

You can configure an upstream LU such that it is reserved, or dedicated, for use by a specific downstream LU.

To define a dedicated LU or a range of dedicated LUs for an upstream host and downstream PU, perform the following task in global configuration mode:

Task	Command
Define a dedicated LU or a range of dedicated LUs for a downstream PU.	**dspu lu** *lu-start* [*lu-end*] **host** *host-name host-lu-start* [**pu** *pu-name*]

See the “Dedicated LU Routing Example” section later in this chapter for an example of dedicated LU routing.

Pooled LU Routing

You can configure an upstream host LU such that it is a member of a pool of LUs. When a downstream connection is established and the downstream LU is configured as a pooled LU, the router selects an upstream LU from the pool for assignment to the downstream LU.

Pooled LU routing allows a limited number of upstream host LUs to be shared (at different times) among many downstream PUs.

To define a host LU or a range of host LUs in an LU pool, perform the following task in global configuration mode:

Task	Command
Define a host LU or a range of host LUs in an LU pool.	**dspu pool** *pool-name* **host** *host-name* **lu** *lu-start* [*lu-end*] [**inactivity-timeout** *inactivity-minutes*]

You can configure a downstream LU as a pooled LU. When a downstream connection is established and the downstream LU is configured as a pooled LU, the router selects an upstream LU from the specified pool for assignment to the downstream LU.

To define a pooled LU or a range of pooled LUs for a downstream PU, perform the following task in global configuration mode:

Task	Command
Define a pooled LU or a range of pooled LUs for a downstream PU.	**dspu lu** *lu-start* [*lu-end*] **pool** *pool-name* [**pu** *pu-name*]

See the "Pooled LU Routing Example" section later in this chapter for an example of pooled LU routing.

Configure DSPU to Use a Data Link Control

The final step in configuring DSPU is to define the data link controls that will be used for upstream host and downstream PU connections.

The DSPU feature supports the data link controls described in the following sections:

- Configure DSPU to Use Token Ring
- Configure DSPU to Use RSRB
- Configure DSPU to Use RSRB with Local Acknowledgment

You configure DSPU for a data link control by enabling a local SAP for either upstream or downstream connections. Each data link control owns 255 SAPs that can be enabled for DSPU connections. A local SAP can be enabled for either upstream or downstream connections; a local SAP cannot be enabled for both upstream and downstream connections on the same data link control.

Configure DSPU to Use Token Ring

To enable a local SAP on the Token Ring interface for use by upstream hosts, perform the following task in interface configuration mode:

Task	Command
Enable local SAP for upstream hosts.	**dspu enable-host** [**lsap** *local-sap*]

To enable a local SAP on the Token Ring interface for use by downstream PUs, perform the following task in interface configuration mode:

Task	Command
Enable local SAP for downstream PUs.	**dspu enable-pu** [**lsap** *local-sap*]

To initiate a connection with a remote upstream host or a downstream PU on the Token Ring interface, perform the following task in interface configuration mode:

Task	Command
Initiate connection with upstream host or downstream PU via Token Ring.	**dspu start** {*host-name* | *pu-name*}

Configure DSPU to Use RSRB

To configure DSPU to use RSRB you must create a DSPU/RSRB data link control.

Similar to our implementation of SDLLC, the DSPU/RSRB data link control uses the concept of a virtual Token Ring device residing on a virtual Token Ring to represent the router to upstream hosts and downstream PUs across an RSRB network.

Since the upstream host and/or downstream PU expects its peer to also be on a Token Ring, you must assign a virtual Token Ring address (the DSPU virtual MAC address) to the DSPU/RSRB data link control. Like real Token Ring addresses, the DSPU virtual MAC address must be unique across the network.

In addition to assigning the DSPU virtual Mac address, you must also assign a DSPU virtual ring number to the DSPU/RSRB data link control. The DSPU virtual ring number must be unique across the network.

Note The DSPU virtual ring number is a different number than the virtual ring group numbers that you use to configure RSRB and multiport bridging.

The combination of the DSPU virtual MAC address and the DSPU virtual ring number identifies the DSPU/RSRB data link control interface to the rest of an RSRB network.

When an end station (either an upstream host or a downstream PU) attempts to connect with the DSPU router, the following occurs:

1. The end station sends explorer packets with the DSPU virtual MAC address as the destination address in the MAC headers.
2. The router configured with that DSPU virtual MAC address intercepts the frame, fills in the DSPU virtual ring number and the DSPU bridge number in the Routing Information Field (RIF), and sends the response back to the end station.
3. The end station establishes a session with the DSPU router.

To define the DSPU/RSRB data link control interface, perform the following tasks in global configuration mode:

Task	Command
Define an RSRB ring group.	**source-bridge ring-group** *ring-group*

Task	Command
Define the DSPU/RSRB interface.	**dspu rsrb** *local-virtual-ring bridge-number target-virtual-ring virtual-macaddr*

After you define the DSPU RSRB data link control, you configure DSPU to use the RSRB data link control by enabling a local SAP for either upstream or downstream connections.

To enable a local SAP on RSRB for use by upstream hosts, perform the following task in global configuration mode:

Task	Command
Enable local SAP for upstream hosts.	**dspu rsrb enable-host** [**lsap** *local-sap*]

To enable a local SAP on RSRB for use by downstream PUs, perform the following task in global configuration mode:

Task	Command
Enable local SAP for downstream PUs.	**dspu rsrb enable-pu** [**lsap** *local-sap*]

To initiate a connection with a remote upstream host or downstream PU over RSRB, perform the following task in global configuration mode:

Task	Command
Initiate connection with upstream host or downstream PU via RSRB.	**dspu rsrb start** {*host-name* | *pu-name*}

Configure DSPU to Use RSRB with Local Acknowledgment

Configuring DSPU to use RSRB with local acknowledgment is identical to configuring RSRB with local acknowledgment. If you add the **local-ack** keyword to the **source-bridge remote-peer** configuration command DSPU will use local-acknowledgment for any end stations that connect to DSPU from that peer.

To configure DSPU to use RSRB with local acknowledgment, perform the following tasks in global configuration mode:

Task	Command
Define an RSRB ring group.	**source-bridge ring-group** *ring-group*
Define a remote peer with the local acknowledgment feature.	**source-bridge remote-peername ring-group** *ring-group* **tcp** *ip-address* **local-ack**
Define the DSPU/RSRB interface.	**dspu rsrb** *local-virtual-ring bridge-number target-virtual-ring virtual-macaddr*

Define the Number of Outstanding, Unacknowledged Activation RUs

The DSPU feature allows you to define the number of activation request units (RUs) such as ACTLUs or DDDLU NMVTs that can be sent by the router before waiting for responses from the remote PU.

The DSPU activation window provides pacing to avoid depleting the router buffer pool during PU activation. Increasing the window size allows more LUs to become active in a shorter amount of time (assuming the required buffers for activation RUs are available). Decreasing the window size limits the amount of buffers the DSPU may use during PU activation. Typically, you do not need to change the default window size.

To define the number of unacknowledged activation RUs that can be outstanding, perform the following task in global configuration mode:

Task	Command
Define number of unacknowledged activation RUs.	**dspu activation-window** *window-size*

Monitor DSPU Feature Status

You can monitor the status of the DSPU feature. To display information about the state of the DSPU feature, perform the following tasks in EXEC mode:

Task	Command
Show the status of DSPU hosts or downstream PUs.	**show dspu pu** {*host-name* | *pu-name*} [**all**]
Show the status of the DSPU pool.	**show dspu pool** *pool-name* [**all**]

DSPU Configuration Examples

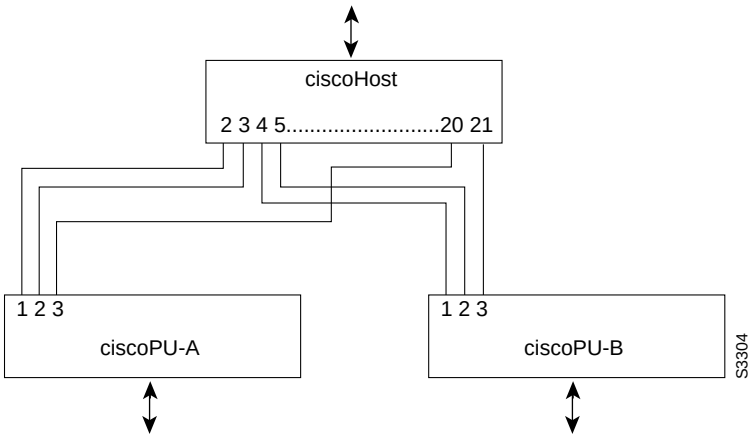
The following sections provide DSPU configuration examples:

- Dedicated LU Routing Example
- Pooled LU Routing Example
- DSPU Configuration Example

Dedicated LU Routing Example

Figure 27-3 illustrates the use of dedicated LU routing. Each upstream host LU is dedicated for use by a specific downstream LU.

Figure 27-3 Dedicated LU Routing



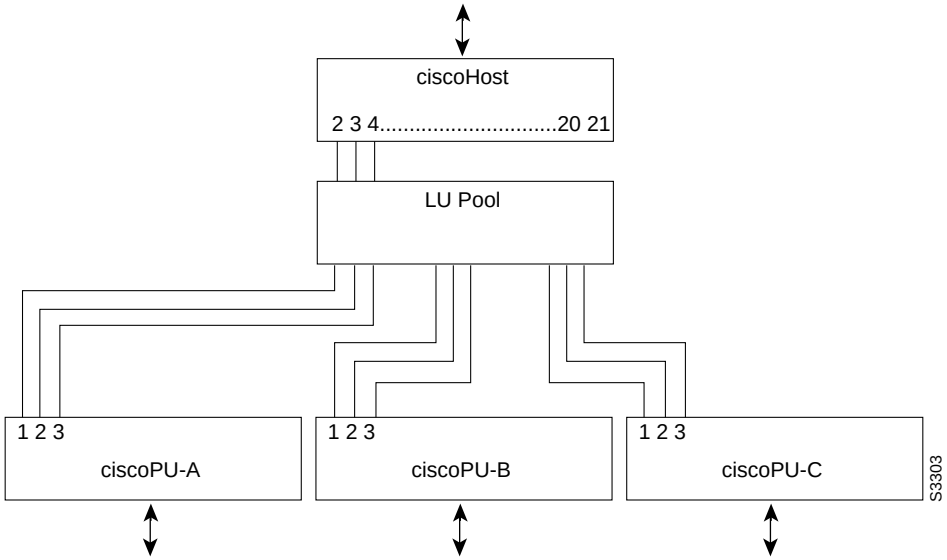
The following is a configuration file for the dedicated LU routing shown in Figure 27-3:

```
dspu host ciscohost xid-snd 06500001 rmac 4000.3745.0001
dspu pu ciscopu-a xid-rcv 05D00001 rmac 1000.5AED.0001
dspu lu 1 2 host ciscohost 2
dspu lu 3 3 host ciscohost 20
dspu pu ciscopu-b xid-rcv 05D00002 rmac 1000.5AED.0002
dspu lu 1 2 host ciscohost 4
dspu lu 3 3 host ciscohost 21
```

Pooled LU Routing Example

Figure 27-4 illustrates the use of pooled LU routing. Each upstream LU is configured in the LU pool and each downstream LU is configured as a pooled LU.

Figure 27-4 Pooled LU Routing



The following is the configuration example for the pooled LU routing shown in Figure 27-4:

```
dspu host ciscohost xid-snd 06500001 rmac 4000.3745.0001
dspu pool lupool host ciscohost lu 2 21
dspu pu ciscopu-a xid-rcv 05D00001 rmac 1000.5AED.0001
dspu lu 1 3 pool lupool
dspu pu ciscopu-b xid-rcv 05D00002 rmac 1000.5AED.0002
dspu lu 1 3 pool lupool
dspu pu ciscopu-c xid-rcv 05D00003 rmac 1000.5AED.0003
dspu lu 1 3 pool lupool
```

DSPU Configuration Example

The following configuration example represents one possible definition for the network topology shown earlier in Figure 27-1:

```
source-bridge ring-group 99
source-bridge remote-peer 99 tcp 150.10.13.1
source-bridge remote-peer 99 tcp 150.10.13.2 local-ack

dspu rsrb 88 1 99 4000.ffff.0001
dspu rsrb enable-host lsap 4

dspu host ciscohost xid-snd 06500001 rmac 4000.1111.0001 lsap 04 focalpoint
dspu pool ciscopool host ciscohost lu 2 8
dspu rsrb start ciscohost

dspu pu ciscopu1 xid-rcv 05d00001 lsap 04
dspu lu 2 3 pool ciscopool

dspu pu ciscopu2 xid-rcv 05d00002 lsap 04
dspu lu 2 4 pool ciscopool

dspu pu ciscopu3 xid-rcv 05d00003 lsap 12
dspu lu 2 2 pool ciscopool

dspu pu ciscopu4 xid-rcv 05d00004 lsap 20
dspu lu 2 2 pool ciscopool
dspu lu 3 3 host ciscohost 9
```

```
interface TokenRing 0
ring-speed 16
dspu enable-pu lsap 4
dspu enable-pu lsap 12
dspu enable-pu lsap 20
```

CHAPTER 28

Configuring SNA Frame Relay Access Support

This chapter describes our Frame Relay access support for System Network Architecture (SNA) devices. For a complete description of the commands mentioned in this chapter, refer to the "SNA Frame Relay Access Support Commands" chapter of the *Router Products Command Reference* publication.

Cisco's Implementation of SNA Frame Relay Access

Because Frame Relay offers cost-effective means of transporting multiple protocols on a wide-area network (WAN), IBM now supports Frame Relay multiprotocol encapsulation functions on a wide range of IBM devices.

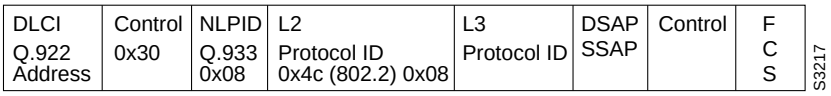
The multiprotocol encapsulation specification is described in RFC 1490 and FRF.3 Agreement from the Frame Relay Forum (FRF).

RFC 1490 Multiprotocol Encapsulation for SNA Data

RFC 1490 specifies a standard method of encapsulating multiprotocol traffic with data link (Level 2 of the OSI model) framing.The encapsulation for SNA data is specified in the FRF.3 Agreement.

The Frame Relay encapsulation method is based on the RFC 1490 frame format for "user-defined" protocols using Q.933 NLPID, as illustrated in Figure 28-1.

Figure 28-1 Frame Relay Encapsulation Based on RFC 1490



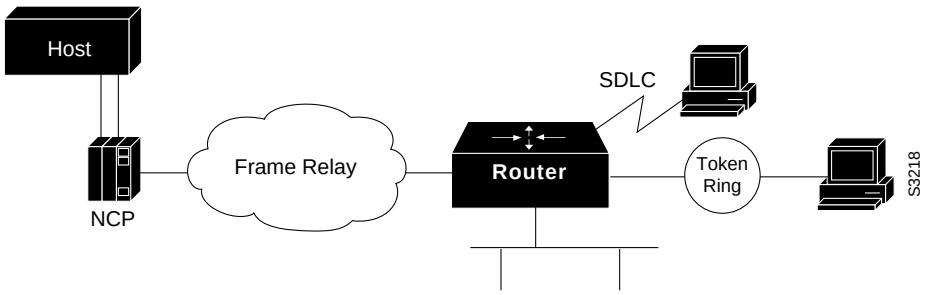
Note The protocol ID for SNA subarea FID4 is 0x81. The protocol ID for SNA subarea FID2 is 0x82. The protocol ID for APPN FID2 is 0x83.

Frame Relay Access Support

Our Frame Relay Access support consists of a router acting as a Frame Relay Access Device (FRAD) for Synchronous Data Link Control (SDLC), Token Ring, and Ethernet attached devices over a Frame Relay Boundary Network Node (BNN) link. Frame Relay access support allows the router

acting as a FRAD to take advantage of the SNA BNN support for Frame Relay provided by ACF/NCP 7.1 and OS/400 V2R3. Downstream PU 2.0 and PU 2.1 devices can be attached to the router through SDLC, Token Ring or Ethernet links. The router acting as a FRAD is connected to the NCP or AS/400 through a public or private Frame Relay network, as illustrated in Figure 28-2.

Figure 28-2 SNA BNN Support for Frame Relay



The frame format used to communicate across the Frame Relay BNN link is defined in RFC 1490 for routed SNA traffic. From the perspective of the SNA host (for example an NCP or AS/400), the Frame Relay connection is defined as a switched resource similar to a Token Ring BNN link.

The router is responsible for terminating the local data link control (DLC) frames (such as SDLC and Token Ring frames) and for modifying the DLCs to 802.2 compliant LLC frames. The LLC logical link is used to provide a reliable connection-oriented link layer transport required by SNA. (For example, 802.2 LLC is used to provide link layer acknowledgment, sequencing and flow control.)

The router encapsulates these 802.2 LLC frames according to the RFC 1490 format for SNA traffic. The frames are then forwarded to the SNA host on a Frame Relay permanent virtual circuit (PVC). In the reverse direction, the router is responsible for de-encapsulating the data from the Frame Relay PVC, and for generating and transmitting the appropriate local DLC frames to the downstream devices.

SNA Frame Relay Access Support Configuration Task List

To configure Frame Relay access support, perform the tasks described in the following sections:

- Configure Frame Relay Access Support
- Configure Frame Relay Access Support Congestion Management
- Monitor and Maintain Frame Relay Access Support

See the end of this chapter for "Frame Relay Access Support Configuration Examples."

Configure Frame Relay Access Support

To configure Frame Relay access support, perform one of the following tasks in interface configuration mode:

Task	Command
Associate an LLC connection with a Frame Relay DLCI.	**fras map llc** *mac-address lan-lsap lan-rsap* **serial** *port* **frame-relay** *dlci fr-lsap fr-rsap* [**PFID2** | **AFID2**| **FID4**]

Task	Command
Associate an SDLC link with a Frame Relay DLCI.	**fras map sdlc** *sdlc-address* **serial** *port* **frame-relay** *dlci fr-lsap fr-rsap* [**PFID2** | **AFID2**| **FID4**]

Since Frame Relay itself does not provide a reliable transport as required by SNA, the RFC 1490 support of SNA uses LLC2 as part of the encapsulation to provide link-level sequencing, acknowledgment and flow control. The serial interface configured for Internet Engineering Task Force (IETF) encapsulation (in other words, RFC 1490) can take all LLC2 interface configuration commands.

Configure Frame Relay Access Support Congestion Management

Frame Relay access support provides a congestion control mechanism based on the interaction between congestion notification bits in the Frame Relay packet and the dynamic adjustment of the LLC2 send window. This window is the number of frames the router can send before waiting for an acknowledgment. The window size decreases with the occurrence of backward explicit congestion notification (BECN), and increases when no BECN frames are received.

To configure congestion management, perform the following tasks in interface configuration mode:

Task	Command
Specify the maximum window size for each logical connection.	**llc2 local-window** *size*[1]
Enable the dynamic window flow-control mechanism.	**llc2 dynwind** [**nw** *nw-number*] [**dwc** *dwc-number*]

1. This command is documented in the "LLC2 and SDLC Commands" chapter of the *Router Products Command Reference* publication.

You can enable the dynamic window mechanism only if you are using Frame Relay IETF encapsulation.

Monitor and Maintain Frame Relay Access Support

To display information about the state of Frame Relay access support, perform the following command in privileged EXEC mode:

Task	Command
Display the mapping and connection state of the Frame Relay access support.	**show fras map**

Frame Relay Access Support Configuration Examples

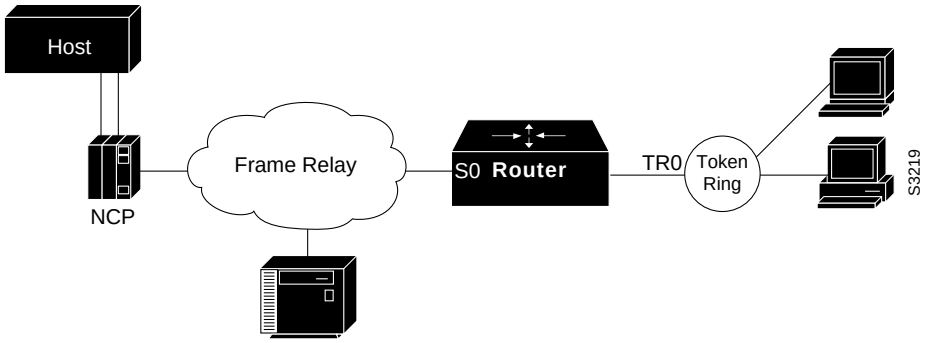
The following sections provide Frame Relay access support configuration examples:

- LAN-Attached SNA Devices Example
- SDLC-Attached SNA Devices Example

LAN-Attached SNA Devices Example

Figure 28-3 illustrates the configuration of SNA devices attached to a local-area network (LAN).

Figure 28-3 LAN-Attached SNA Devices



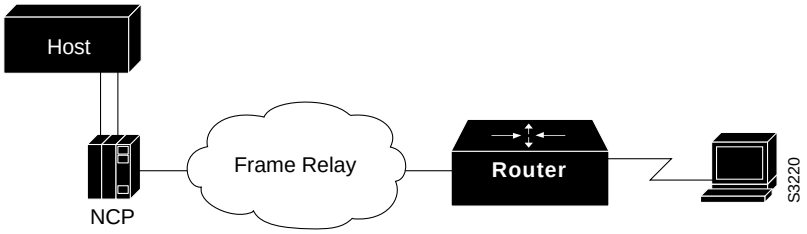
The following is the configuration for the network shown in Figure 28-3:

```
interface TokenRing 0
no ip address
no keepalive
ring-speed 16
fras map llc 0800.5a8f.8802 4 4 serial 0 frame-relay 200 4 4
!
interface serial 0
mtu 2500
no ip address
encapsulation frame-relay IETF
keepalive 12
frame-relay lmi-type ansi
frame-relay map llc2 200
```

SDLC-Attached SNA Devices Example

Figure 28-4 illustrates the configuration of SDLC-attached SNA devices.

Figure 28-4 SDLC-Attached SNA Devices



The following is the configuration file for the network shown in Figure 28-4:

```
interface serial 1
no ip address
encapsulation sdlc
no keepalive
clockrate 56000
sdlc address C1
sdlc xid C1 05D01501
sdlc role primary
fras map sdlc C1 serial 0 frame-relay 200 4 4
!
```

```
interface serial 0
mtu 2500
no ip address
encapsulation frame-relay IETF
keepalive 12
frame-relay lmi-type ansi
frame-relay map llc2 200
```

CHAPTER 29

Configuring IBM Channel Attach

This chapter describes how to configure the Cisco 7000 series mainframe Channel Interface Processor (CIP), which supports the IBM channel attach feature.

For hardware technical descriptions and information about installing the router interfaces, refer to the hardware installation and maintenance publication for your product. For command descriptions and usage information, refer to the chapter entitled "IBM Channel Attach Commands" of the *Router Products Command Reference* publication.

Cisco's Implementation of IBM Channel Attach

IBM (and compatible) mainframe hosts are connected to each other and to communication controllers through high-performance communication subsystems called mainframe channels. Cisco supports IBM channel attachment technologies, including both the fiber-optic Enterprise Systems Connection (ESCON) channel introduced on the ES/9000 mainframe and the parallel bus-and-tag channel supported on System 370 and later mainframes.

Cisco has implemented Common Link Access for Workstations (CLAW) support in the CIP, which is a link-level protocol used by channel-attached RISC System / 6000 series systems and by IBM 3172 devices running Transmission Control Protocol/Internet Protocol (TCP/IP) offload. The CLAW protocol improves efficiency of channel use and allows the CIP to provide the functionality of a 3172 in TCP/IP environments and support direct channel attachment. The output from TCP/IP mainframe processing is a series of IP datagrams that the router can switch without modifications.

The Cisco 7000 series configured with the CIP (and other interface processors) is an ideal connectivity hub for large corporate networks, providing routing services between mainframes and LANs:

- A Cisco 7000 series with a CIP can replace an IBM 3172 interconnect controller, enabling mainframe and peripheral access from LAN-based workstations.
- Because the number of network devices is reduced, a Cisco 7000 series with a CIP simplifies the network, especially in situations where a router can replace a 3172.
- To ensure 100 percent IBM compatibility, the Cisco ESCON Channel Adapter (ECA) adapter card for the CIP uses the IBM ESCON chipset.

IBM Channel Attach Hardware Requirements

Support for IBM channel attach requires the following hardware:

- A Cisco 7000 series with one available card slot

- A Channel Interface Processor (CIP) with one or two adapter cards (ECA, PCA, or a combination)
- All necessary cables for interconnecting the adapter cards to the mainframe or ESCON Director Switch

IBM Channel Attach Host Software Requirements

Your mainframe host software must meet the following minimum requirements:

- IBM TCP/IP for VM Version 2 Release 2, with PTF enhancements for RISC System / 6000 series ESCON support
- IBM TCP/IP for MVS Version 2 Release 2.1, with PTF enhancements for RISC System / 6000 series ESCON support

Interface Configuration Task List

You can perform the tasks in the following sections to configure and maintain IBM channel attach interfaces. In addition, several examples show how host configuration settings correlate to values used in the router configuration commands. The first section provides some background on the IBM channel attach feature and the Cisco 7000 series interfaces that support it.

- Understand the IBM Channel Attach Interface
- Select the Interface
- Configure IBM Channel Attach
- Select Host System Parameters
- Monitor and Maintain the Interface

See the end of this chapter for “IBM Channel Attach Interface Configuration Examples.”

Understand the IBM Channel Attach Interface

Support for IBM channel attach is provided on the Cisco 7000 series routers by the Channel Interface Processor (CIP) and an appropriate interface adapter card. With a CIP and the ESCON Channel Adapter (ECA) or bus and tag Parallel Channel Adapter (PCA), a Cisco 7000 series router can be directly connected to a mainframe, replacing the function of an IBM 3172 interconnect controller. This connectivity enables mainframe applications and peripheral access from LAN-based workstations.

Transmission Control Protocol/Internet Protocol (TCP/IP) mainframe protocol environments for IBM operating systems Multiple Virtual Storage (MVS) and Virtual Machine (VM) are supported in the initial release. This support includes TCP/IP-based applications such as terminal emulation (Telnet), the File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Network File System (NFS), a distributed file access system.

A single CIP can support up to two channel adapter cards in any combination. Because of this flexibility, upgrading from parallel bus and tag to ESCON is simplified. The CIP can be configured for ESCON support by replacing a parallel channel adapter with an ESCON adapter. Note that this upgrade procedure must be done by authorized service personnel.

A CIP configured with 8 megabytes of memory can support up to 128 common link access to workstation (CLAW) connections, or approximately 240 devices. Each CLAW connection requires two devices. Subsequent releases will support up to 256 CLAW connections per CIP, or a maximum of 128 CLAW connections per interface adapter card.

Use the **show extended channel** EXEC command to display current CIP status. This command provides a report for each physical interface configured to support IBM channel attach.

Select the Interface

Before you configure your channel attach interface, you must select the interface. Perform the following task in global configuration mode:

Task	Command
Select the channel attach interface and enter interface configuration mode.	**interface channel** *slot/port*

You need not add a space between the interface type (**channel**) and the slot and port number. For example, you can specify **interface channel 3/0** or **interface channel3/0**.

The following section describes how to configure your channel attach interface.

See the section "IBM Channel Attach Interface Configuration Examples" at the end of this chapter for example configuration commands.

Configure IBM Channel Attach

The following sections describe how to configure the IBM channel attach interface.

- Define the Routing Process
- Assign an IP Address
- Configure the IBM Channel Attach Interface
- Select a Data Rate for the Parallel Channel Adapter (PCA)
- Configure Other Interface Support

See the section "Select Host System Parameters" for guidelines on matching interface configuration values with host system values.

Define the Routing Process

You must configure the routing process that will be used by the router. We recommend using the enhanced IGRP routing process to perform IP routing on the IBM channel attach interface. Perform the following steps beginning in global configuration mode:

Task		Command
Step 1	Enter router configuration mode by selecting the routing process, preferably EIGRP, and the autonomous system the router belongs to.	**router eigrp** *autonomous-system*
Step 2	Define the directly connected networks that are part of the autonomous system.	**network** *address*

Assign an IP Address

You must assign an IP address to the ECA or PCA interface so that it can communicate with other devices (or tasks) on the network. The IP address you assign to the interface must be in the same subnetwork as the hosts with which you wish to communicate. Perform the following task in interface configuration mode:

Task	Command
Assign an IP address and network mask to the selected interface.	**ip address** *address mask*

Configure the IBM Channel Attach Interface

You must define the devices, or tasks supported on the interface. Some information you need to perform this task is derived from the following host system configuration files: MVSIOCP, IOCP, and the TCPIP configuration. Refer to the section "Select Host System Parameters" for guidelines and pointers.

Perform the following task in interface configuration mode:

Task	Command
Define the CLAW parameters for this device.	**claw** *path device-address ip-address host-name device-name host-app device-app*

See the section "IBM Channel Attach Interface Configuration Examples" for samples of **claw** commands for different configurations.

Select a Data Rate for the Parallel Channel Adapter (PCA)

When you configure a channel attach interface that supports a PCA adapter card, you must define a data rate of either 3 megabytes per second or 4.5 megabytes per second. Perform the following task in interface configuration mode:

Task	Command
Define the PCA data transfer rate.	**channel-protocol** [**s** | **s4**]

Configure Other Interface Support

You can further define how the interface and the router interoperate. You can perform any of the following tasks in interface configuration mode to enhance the usefulness of IBM channel attach support. Perform the following tasks in interface configuration mode:

Task	Command
Disable fast switching (IP route cache switching). Fast switching is on by default, but access lists can inhibit fast switching.	[**no**] **ip route-cache**[1]
Use access lists to filter connections.	**access-list** *list* {**permit** | **deny**} *source source-mask*[1]
Enable autonomous switching through either the Silicon switching engine (SSE) or the CxBus controller.	**ip route-cache** [**cbus** | **sse**][1]

Task	Command
Include autonomous switching support for multiple IP datagram applications running on the same CIP, as required.	**ip route-cache same-interface**

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

Select Host System Parameters

This section describes how to correlate values found in the VM and MVS system IOCP files with the fields in the **claw** command. In addition, you will need configuration information from the host TCP/IP application configuration file. Refer to the following IBM operating system manuals for specific IOCP configuration statement details:

- *Transmission Control Protocol/Internet Protocol TCP/IP Version 2 Release 2.1 for MVS: Planning and Customization*, SC31-6085 (or later version)
- *Transmission Control Protocol/Internet Protocol TCP/IP Version 2 Release 2 for VM: Planning and Customization*, SC31-6082 (or later version)

Values from the Host IOCP File

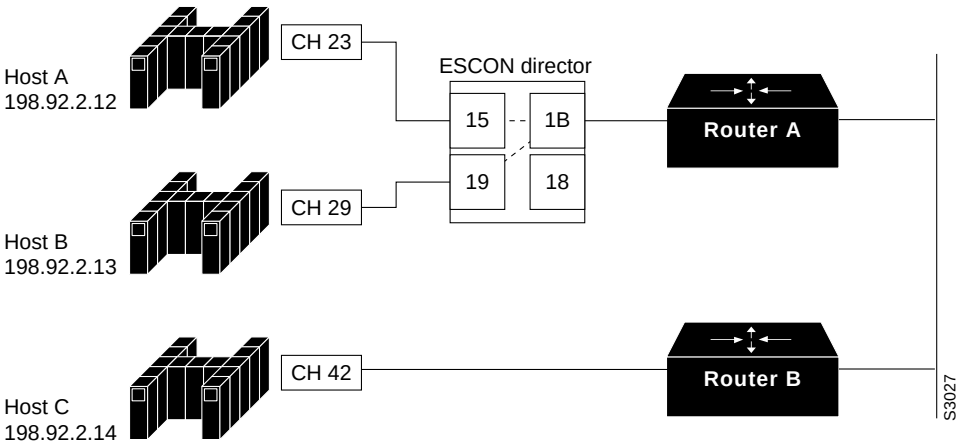
When you define CLAW parameters, you must supply path information and device address information to support routing on an IBM channel. The path information can be simple, in the case of a channel directly attached to a router, or more challenging when the path includes an ESCON director switch or multiple image facility support.

First we will examine the **claw** command *path* argument. It is a concatenation of three hexadecimal numbers that represent the following values:

CLAW Path Argument Breakdown	Values	Description
Path	01–FF	For a directly attached ESCON channel or any parallel channel, this value is 01 *unless* the system administrator has configured another value. For a channel attached through an ESCON director switch, this value will be the path that, from the router point of view, exits the switch and attaches to the host.
Channel logical address	0–F	For a directly attached ESCON channel or any parallel channel, this value is 0. If the host is running in Logical Partition (LPAR) mode, this is the channel logical address associated with the channel and defined in the IOCP. The default for this part of the path argument is 0. Otherwise, the channel logical address associated with the channel is defined in the IOCP.
Control unit logical address	0–F	For a directly attached ESCON channel or any parallel channel, this value defaults to 0. If this value is specified in the IOCP, match that value here. Otherwise, the control unit logical address is specified in the IOCP CNTLUNIT statement for the host channel in the CUADD parameter.

In Figure 29-1, two host systems connect to the ESCON director switch, on paths 23 and 29. The channels both exit the switch on path 1B and attach to Router A.

Figure 29-1 System with an ESCON Director Switch and a Directly Attached Channel



Note that the path between Host A and Host B is dynamically switched within the ESCON director. A third host is attached directly to Router B through path 42. The IOCP control unit statements would look something like the following examples:

- Host A

```
CNTLUNIT CUNUMBER=0001, PATH=(23), LINK=1B, UNITADD=((00,64)), UNIT=SCTC, CUADD=F
```

- Host B

```
CNTLUNIT CUNUMBER=0002, PATH=(29), LINK=1B, UNITADD=((00,64)), UNIT=SCTC, CUADD=A
```

- Host C

```
CNTLUNIT CUNUMBER=000A, PATH=(42), UNIT=SCTC, UNITADD=((00,64))
```

Note If you use the Hardware Configuration Definition (HCD) program to generate an IOCP and your release of HCD does not support the value RS6K, you might need to set the control unit and device value to SCTC for your ESCON channels. A device mismatch error message will be displayed, but the device will come on line and operate correctly.

The system administrator would provide you with the values, for example 15 and 19, for the return channel attachment from the switch to each host. Given these values, the **claw** command *path* argument for the two channel attachments to Router A becomes:

```
claw 150F
claw 190A
```

The **claw** command *path* argument for the directly attached channel to Router B is easy to determine:

```
claw 0100
```

Next, determine the **claw** command *device-address* argument value, which is shown as 00 in the UNITADD parameter for all three devices. This value can be any even value between 00 and 3E, as long as it matches an allowed UNITADD value in IOCP. The **claw** commands now become:

- Router A

```
claw 150F 00
claw 190A 00
```

- Router B

```
claw 0100 02
```

Values from the Host TCPIP File

The remainder of the **claw** command arguments are derived from the DEVICE, LINK, and HOME statements in the host TCPIP configuration files. The statements will be similar to the following:

- Host A

```
DEVICE EVAL CLAW 500 VMSYSTEM C7000 NONE  20  20 4096 4096
LINK EVAL1 IP 0 EVAL
HOME 198.92.2.12   EVAL1
```

- Host B

```
DEVICE EVAL CLAW 600 STSYSTEM C7000 NONE  20  20 4096 4096
LINK EVAL1 IP 0 EVAL
HOME 198.92.2.13   EVAL1
```

- Host C

```
DEVICE EVAL CLAW 700 RDUSYSTM C7000 NONE  20  20 4096 4096
LINK EVAL1 IP 0 EVAL
HOME 198.92.2.14   EVAL1
```

The DEVICE statement lists the *host-name* and *device-name* values to use, which follows the CLAW 500 entry in the DEVICE statement.

The LINK statement links the device name, EVAL, to EVAL1. The IP address for EVAL1 appears in the HOME statement.

Based on this example, you can supply the remainder of the arguments for the sample **claw** commands:

- Router A

```
claw 150F 00 198.92.2.12 VMSYSTEM C7000 TCPIP TCPIP
claw 190A 00 198.92.2.13 STSYSTEM C7000 TCPIP TCPIP
```

- Router B

```
claw 0100 02 198.92.2.14 RDUSYSTM C7000 TCPIP TCPIP
```

Example of a Derived Value

When you have a directly attached channel, the system administrator might provide you with a system IODEVICE ADDRESS that you can use. In this case, you must work backwards through the IOCP file to locate the proper *device-address* argument value for the **claw** command.

In this first example, the IODEVICE ADDRESS value is 800. Using this number, you locate the IODEVICE ADDRESS statement in the IOCP file, which points you to the CNTLUNIT statement that contains the *device-address* argument value for the **claw** command:

```
IODEVICE ADDRESS=(0800,256),CUNUMBR=(0012),UNIT=SCTC
**** Address 800 points to CUNUMBR 0012 in the following statement

CNTLUNIT CUNUMBR=0012,PATH=(28),UNIT=SCTC,UNITADD=((00,256))
**** The device-address is the UNITADD value of 00
```

From this example, the **claw** command would be similar to the following:

```
claw 0100 00 197.91.2.12 CISCOVM EVAL TCPIP TCPIP
```

In the next example, the system administrator has given you an IODEVICE ADDRESS of 350, which does not correspond exactly to a value in the IOCP file. In this instance you must calculate an offset *device-address* argument value for the **claw** command:

```
IODEVICE ADDRESS=(0340,64),CUNUMBR=(0008),UNIT=SCTC
IODEVICE ADDRESS=(0380,64),CUNUMBR=(0009),UNIT=SCTC
**** Address 350 (340 + 10) is in the range covered by CUNUMBER 0008

CNTLUNIT CUNUMBR=0008,PATH=(24),UNIT=SCTC,UNITADD=((40,64)),SHARED=N, X
**** The device-address is the UNITADD value of 40, offset by 10
**** The device-address to use is 50
```

From this example, the **claw** command would be similar to the following:

```
claw 0100 50 197.91.2.12 CISCOVM EVAL TCPIP TCPIP
```

Note In the IOCP examples for the IODEVICE and CNTLUNIT statements, UNIT=SCTC is the usual value for ESCON channels. Parallel channels will have UNIT=3088 in the CNTLUNIT statement and UNIT=CTC in the IODEVICE statement.

Caution When you are running MVS, you must disable the missing interrupt handler (MIH) to avoid introducing errors into the CLAW algorithm. Refer to the IBM publication *Transmission Control Protocol/Internet Protocol TCP/IP Version 2 Release 2.1 for MVS: Planning and Customization* (publication SC31-6085 or later) for information on disabling the MIH.

Monitor and Maintain the Interface

You can perform the tasks in the following sections to monitor and maintain the interfaces:

- Monitor Interface Status
- Clear and Reset the Interface
- Shut Down and Restart an Interface
- Run CIP Interface Loopback Diagnostics

Monitor Interface Status

The software allows you to display information about the interface, including the version of the software and the hardware, the controller status, and statistics about the interfaces. The following table lists some of the interface monitoring tasks. (The full list of **show** commands can be displayed by entering the **show ?** command at the EXEC prompt.)

Perform the following commands in EXEC mode:

Task	Command
Display information about the channel interface processor (CIP) interfaces on the Cisco 7000 series. These commands display information that is specific to the interface hardware.	**show extended channel** *slot/port* **statistics** [*path* [*device-address*]] **show extended channel** *slot/port* **subchannel**
Display current internal status information for the interface controller cards in the Cisco 7000.	**show controllers** {**cxbus** | **fddi** | **serial** | **t1** | **token**}
Display the number of packets of each protocol type that have been sent through the interface for the Cisco 7000.	**show interfaces channel** [*slot/port*]
Display the hardware configuration, software version, the names and sources of configuration files, and the boot images.	**show version**[1]

1. This command is documented in the "System Image, Microcode Image, and Configuration File Load Commands" chapter of the *Router Products Command Reference* publication.

Clear and Reset the Interface

To clear the interface counters shown with the **show interfaces** command, enter the following command at the EXEC prompt:

Task	Command
Clear interface counters for the Cisco 7000.	**clear counters** [*slot/port*]

Note This command will not clear counters retrieved using SNMP, but only those seen with the EXEC **show interfaces** command.

Complete the following tasks in EXEC mode to clear and reset interfaces. Under normal circumstances, you do not need to clear the hardware logic on interfaces.

Task	Command
Reset the hardware logic on an interface.	**clear interface** *type number*

Shut Down and Restart an Interface

You can disable an interface. Doing so disables all functions on the specified interface and marks the interface as unavailable on all monitoring command displays. This information is communicated to other network servers through all dynamic routing protocols. The interface will not be mentioned in any routing updates. On the CIP with an ECA interface adapter, a command is sent to the host to inform it of the impending shutdown. On the CIP with a PCA interface adapter, the **shutdown** command disables the adapter card's transceivers and the PCA stops responding to all commands. A a select-out bypass relay must be manually set at the cable connecting to the PCA.

One reason to shut down an interface is if you want to change the interface type of a Cisco 7000 port online. To ensure that the system recognizes the new interface type, shut down the interface, then reenable it after changing the interface. Refer to your hardware documentation for more details.

To shut down an interface and then restart it, perform the following tasks in interface configuration mode:

Task	Command
Shut down an interface.	**shutdown**
Reenable an interface.	**no shutdown**

To check whether an interface is disabled, use the EXEC command **show interfaces**. An interface that has been shut down is shown as administratively down in the **show interfaces** command display.

Run CIP Interface Loopback Diagnostics

The CIP does not provide software loopback support. You can use special "wrap" plugs to perform hardware loopback with the ECA and PCA interface cards. Hardware loopback information is included in the hardware installation notes for the CIP.

IBM Channel Attach Interface Configuration Examples

The following sections include examples to help you understand some aspects of interface configuration:

- Routing Process Configuration Example
- IP Address and Network Mask Configuration Example
- CLAW Configuration Example
- Interface Shutdown and Startup Example

Routing Process Configuration Example

The following example configures an Enhanced IGRP routing process in autonomous system 127 and defines two networks to be advertised as originating within that autonomous system:

```
router eigrp 127
network 197.91.2.0
network 197.91.0.0
```

IP Address and Network Mask Configuration Example

The following example assigns an IP address and network mask to the IBM channel attach interface on the router:

```
ip address 197.91.2.5 255.255.255.0
```

CLAW Configuration Example

The following example configures the IBM channel attach interface to support a directly connected device:

```
claw 0100 00 197.91.0.21 VMSYSTEM C7000 TCPIP TCPIP
```

Interface Shutdown and Startup Example

The following example turns off the CIP interface in slot 2 at port 0:

```
interface channel 2/0
shutdown
```

The following example enables the CIP interface in slot 3 at port 0 that had been previously shut down:

```
interface channel 3/0
no shutdown
```

CHAPTER 30

Configuring DLSw+

This chapter describes how to configure DLSw+, our implementation of the data-link switching (DLSw) standard for Systems Network Architecture (SNA) devices. For a complete description of the commands in this chapter, refer to the "DLSw+ Configuration Commands" chapter of the *Router Products Command Reference* publication.

Cisco's Implementation of DLSw: DLSw+

DLSw+ is our implementation of DLSw, an SNA-over-IP routing standard that helps to integrate SNA and local-area network (LAN) internetworks by encapsulating nonroutable SNA and NetBIOS protocols within routable IP protocols. It is a means of transporting SNA and NetBIOS traffic over an Internet Protocol (IP) network. DLSw is an alternative to source-route bridging (SRB) and addresses the following limitations of SRB:

- SRB hop-count limits (SRB limit is 7)
- Broadcast traffic (from SRB explorer frames or NetBIOS name queries)
- Unnecessary traffic (acknowledgments)
- Data link control timeouts
- Lack of flow control and prioritization

Because these limitations occur when SRB is extended across a wide-area network (WAN), DLSw is typically used to transport SNA and NetBIOS across a WAN.

DLSw Standard

The DLSw standard, documented in RFC 1795, defines the switch-to-switch protocol used between DLSw routers and defines a mechanism to locally terminate data link control connections and multiplex the traffic from the data link control connections onto a Transmission Control Protocol (TCP) connection. The standard always calls for the transport protocol to be TCP and always requires that data link control connections be locally terminated (the equivalent of our local acknowledgment option). The standard also requires that the SRB route information field (RIF) be terminated at the DLSw router. The standard describes a means for prioritization and flow control and defines error recovery procedures that assure data link control connections are appropriately disabled if any part of their associated circuits breaks.

The DLSw standard does not specify when to establish TCP connections. The capabilities exchange allows compliance to the standard but at different levels of support. The standard does not specify how to cache learned information about MAC addresses, RIFs, or NetBIOS names. It also does not describe how to keep track of both capable or preferred DLSw partners for either backup or

load-balancing purposes. It does not provide the specifics of media conversion, but leaves the details up to the implementation. It does not define how to map switch congestion to data link control flow control. Finally, the management information base (MIB) is documented under a separate RFC.

DLSw+ Features and Enhancements

DLSw+ includes the following features:

- Full compliance with the DLSw standard RFC 1795.
- A choice of transport options, including TCP, Fast-Sequenced Transport (FST), and direct encapsulation in High-Level Data Link Control (HDLC)
- Media conversion between local or remote LANs and Synchronous Data Link Control (SDLC) or Ethernet
- Scalability enhancements through peer groups
- Scalability enhancements through explorer firewalls and location learning
- Configuration reduction through on-demand peers

DLSw+ includes enhancements in the following areas:

- Modes of operation—The ability to determine the capabilities of the participating router and to operate according to those capabilities
- Scalability—The ability to construct IBM internetworks in a way that reduces the amount of broadcast traffic and therefore enhances their scalability
- Performance—The ability to offer higher-performance transport options when the line speeds and traffic conditions do not require local acknowledgment
- Availability—The ability to rapidly recover from failures by caching multiple peers that can be used to reach a given destination

Modes of Operation

DLSw+ operates in three modes:

- Backward compatibility mode—remote source-route bridging (RSRB) can be used in parallel with DLSw+ to communicate with older releases of Cisco Internetwork Operating System (Cisco IOS) running RSRB and SDLC-to-LLC2 conversion (SDLLC).
- Standards compliance mode—DLSw+ can automatically detect (through the DLSw+ capabilities exchange) that the participating router is manufactured by another vendor. DLSw+ then operates in DLSw standard mode.
- Enhanced mode—DLSw+ can automatically detect that the participating router is another DLSw+ router. DLSw+ then operates in enhanced mode, providing the additional features of DLSw+ to the SNA and NetBIOS end systems.

Note DLSw+ does not interoperate with pre-standard implementations such as RFC 1434

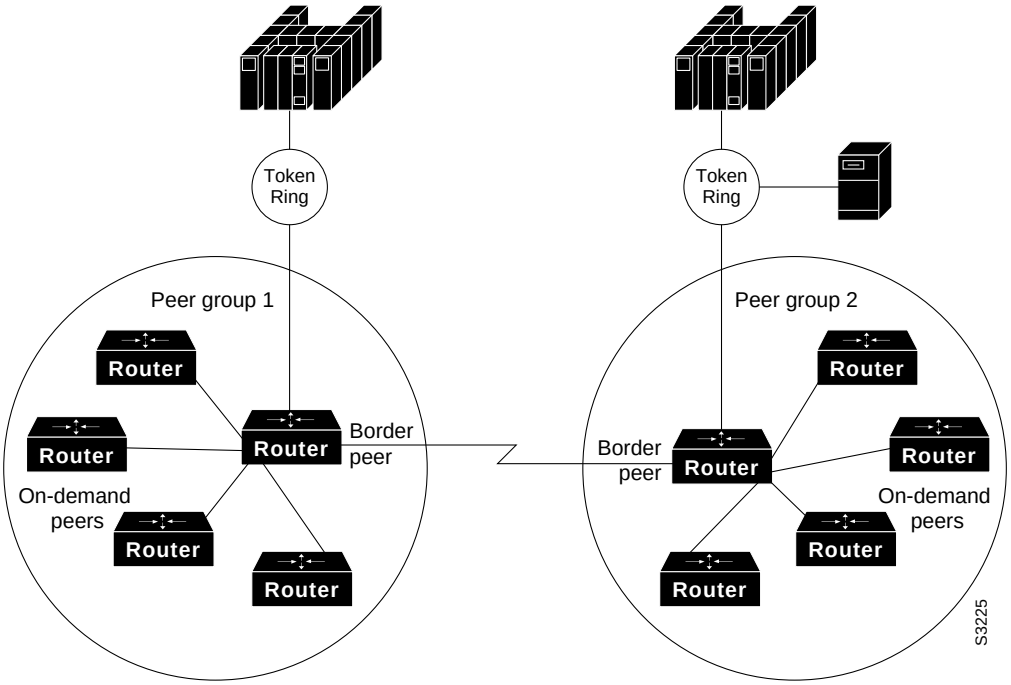
Some enhanced DLSw+ features are also available when a Cisco router is operating in standards compliance mode with another vendor's router. In particular, enhancements that are locally controlled options on a router can be accessed even though the remote router does not have DLSw+. These include location learning (the ability to determine if a destination is on a local LAN before sending "canureach" frames across a WAN), explorer firewalls, and media conversion.

Improved Scalability

One significant factor that limits the size of Token Ring internetworks is the amount of explorer traffic that traverses the WAN. DLSw+ includes the following features to reduce the number of explorers:

- Peer groups—The large Token Ring internetworks that Cisco has helped to build over the last several years have all followed a similar structure. That structure is a hierarchical grouping of routers based upon the usual flow of broadcasts through the network. A cluster of routers in a region or a division of a company are combined into a peer group.
- Border peers—Within a peer group, one or more routers are designated as border peers. When a DLSw+ router receives a test frame or NetBIOS name query, it sends a single explorer frame to its border peer. The border peer takes complete responsibility for forwarding the explorer on behalf of the peer group member. This arrangement eliminates duplicate explorers on the access links and minimizes the processing required in access routers.
- On-demand peers—On-demand peers greatly reduce the number of peers that must be configured. As Figure 30-1 shows, you can use on-demand peers to establish an end-to-end circuit even though the DLSw+ routers servicing the end systems have no specific configuration information about the peers. This configuration permits casual, any-to-any connection without the burden of configuring the connection in advance. It also allows any-to-any routing in large internetworks where persistent TCP connections would not otherwise be possible.
- Explorer firewalls—An explorer firewall permits only a single explorer for a particular destination MAC address to be sent across the WAN. While an explorer is outstanding and awaiting a response from the destination, subsequent explorers for that MAC address are merely stored. Once the explorer response is received at the originating DLSw+, all explorers receive an immediate local response. This eliminates the start-of-day explorer storm that many networks experience.

Figure 30-1 Scalability with DLSw+



Performance

The transport connection between DLSw+ routers can vary according to the needs of the network and is not necessarily tied to TCP/IP as the DLSw standard is. We support three different transport protocols between DLSw+ routers:

- TCP/IP for transport of SNA and NetBIOS traffic across WANs where bandwidth is limited and termination of data link control sessions is required. This transport option is required when DLSw+ is operating in DLSw+ standards compliance mode.
- FST/IP for transport across WANs with an arbitrary topology with sufficient bandwidth to accommodate SNA and NetBIOS.
- Direct encapsulation for transport across a point-to-point connection where the benefits of an arbitrary topology are not important.

Enhanced Availability

DLSw+ offers enhanced availability by maintaining a peer table of multiple paths to a given MAC address or NetBIOS name (where a path is either a remote peer or a local port). Each of our routers maintains a preferred path and one or more capable paths to each destination. The preferred peer is either the peer that responds first to an explorer frame or the peer with the least cost. The preferred port is always the port over which the first positive response to an explorer was received. If the preferred peer to a given destination is unavailable, the next available capable peer is promoted to the new preferred peer. No additional broadcasts are required, and recovery through an alternate peer is immediate.

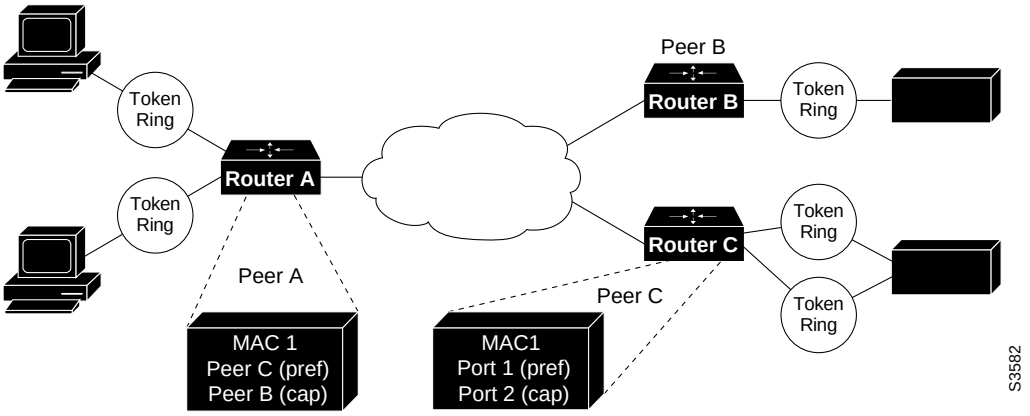
Maintaining multiple paths per destination is especially attractive in SNA networks. A common technique used in the hierarchical SNA environment is assigning the same MAC address to different Token Ring interface couplers (TICs) on the IBM front-end processors (FEPs). DLSw+ ensures that duplicate TIC addresses are found, and if multiple DLSw+ peers can be used to reach the FEPs, they are all cached.

The way that multiple capable peers are handled with DLSw+ can be biased to meet either of the following network needs:

- Fault tolerance—To rapidly reconnect if a data-link connection is lost. Whenever a new circuit is established between a pair of end systems and the end-to-end path for the circuit is already known (that is, it is cached), the originating DLSw+ router sends a circuit-establishment message directly to the preferred partner. If the preferred partner is no longer available, a circuit-establishment message is sent to the next capable router in the cache. Maintaining multiple cache entries facilitates a timely reconnection after session outages.
- Load balancing—To distribute the network traffic over multiple DLSw+ peers in the network. The routers can be configured to perform load balancing, in which case circuits are established in round-robin fashion using the list of capable routers. When used for load balancing, this technique improves overall SNA performance.

Figure 30-2 shows a peer table of preferred (Pref) and capable (Cap) routes.

Figure 30-2 Enhanced Availability and Performance



DLSw+ Configuration Task List

DLSw+ supports media conversion between local or remote LANs and SDLC or Ethernet. For clarity, the configuration task list below describes configuration in a Token Ring environment. The only differences for SDLC and Ethernet are the specific commands needed to configure those media, plus a media-specific command to associate the interface with DLSw+.

To configure DLSw+, perform tasks in the following sections:

- Define a Source-Bridge Ring Group for DLSw+
- Define a DLSw+ Local Peer for the Router
- Define a DLSw+ Ring List or Port List
- Define a DLSw+ Bridge Group List

- Define DLSw+ Remote Peers
- Configure Peer-on-Demand Defaults
- Configure Static Resources Exchanged in Capabilities Exchange
- Configure Static Paths
- Configure Duplicate Path Handling
- Enable DLSw+ on the Appropriate Token Ring Interface
- Enable DLSw+ on the Appropriate Ethernet Interface
- Enable DLSw+ on the Appropriate SDLC Interface
- Tune the DLSw+ Configuration
- Monitor and Maintain the DLSw+ Network

See the end of this chapter for "DLSw+ Configuration Examples." Media-specific configuration examples for Ethernet and SDLC are also provided. For details of SDLC commands in the sample SDLC configuration, refer to the "LLC2 and SDLC Commands" chapter of the *Router Products Command Reference* publication.

Define a Source-Bridge Ring Group for DLSw+

The source-bridge ring can be shared between DLSw+ and SRB/RSRB. In DLSw+, the source-bridge ring group specifies the virtual ring that will appear to be the last ring in the RIF. Because RIFs are terminated at the router, there is no correlation between the ring-group number specified in DLSw+ peers. They can be the same for management simplicity, but they do not have to be. To define a source-bridge ring group for DLSw+, perform the following task in global configuration mode:

Task	Command
Define a ring group.	**source-bridge ring-group** *ring-group*[1]

1. This command is documented in the "Source-Route Bridging Commands" chapter of the *Router Products Command Reference*.

Refer to the DLSw+ Using TCP Encapsulation and LLC2 Local Acknowledgment—Basic Configuration Example for an sample configuration file.

Define a DLSw+ Local Peer for the Router

Defining a DLSw+ local peer for a router enables DLSw+. You specify all local DLSw+ parameters as part of the local peer definition. To define a local peer, perform the following task in global configuration mode:

Task	Command
Define the DLSw+ local peer.	**dlsw local-peer** [**peer-id** *ip-address*] [**group** *group*] [**border**] [**cost** *cost*] [**lf** *size*] [**keepalive** *seconds*] [**passive**] [**promiscuous**]

Refer to theDLSw+ Using TCP Encapsulation and LLC2 Local Acknowledgment—Basic Configuration Example for a sample configuration.

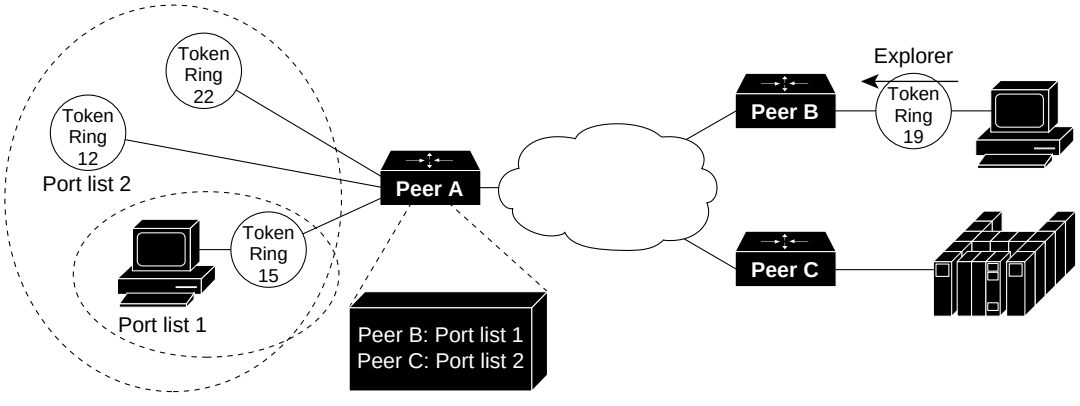
Define a DLSw+ Ring List or Port List

DLSw+ ring lists are used to map traffic on a local interface to remote peers. You can create a ring list of local ring numbers and apply the list to remote peer definitions. Traffic received from a remote peer is only forwarded to the rings specified in the ring list. Traffic received from a local interface is only forwarded to peers if the input ring number appears in the ring list applied to the remote peer definition. The definition of a ring list is optional. If you want all peers and all rings to receive all traffic, you do not have to define a ring list. Simply specify 0 for the list number in the remote peer statement.

To define a ring list, perform the following task in global configuration mode:

Task	Command
Define a ring list.	**dlsw ring-list** *list-number* **rings** *ring-number*

Figure 30-3 DLSw+ Port List Implementation



DLSw+ port lists are used to map traffic on a local interface (either Token Ring or serial) to remote peers. You can create a port list of local ports and apply the list to remote peer definitions. Traffic received from a remote peer is only forwarded to peers if the input port number appears in the port list applied to the remote peer definition. The port list command provides a single command to specify both serial and Token Ring interfaces. The definition of a port list is optional. If you want all peers and all interfaces to receive all traffic, you do not have to define a port list. Simply specify 0 for the list number in the remote peer statement.

To define a port list, perform the following task in global configuration mode:

Task	Command
Define a port list.	**dlsw port-list** *list-number* [**serial** | **tokenring**] *number* [**serial** | **tokenring**] *number ...*

Note Either the ring list or the port list command can be used to associate rings with a given ring-list. The ring list command is easier to type in if you have a large number of rings to define.

Define a DLSw+ Bridge Group List

DLSw+ bridge group lists are used to map traffic on the local Ethernet bridge group interface to remote peers. You can create a bridge group list and apply the list to remote peer definitions. Traffic received from a remote peer is only forwarded to the bridge group specified in the bridge group list. Traffic received from a local interface is only forwarded to peers if the input bridge group number appears in the bridge group list applied to the remote peer definition. The definition of a bridge group list is optional. Since each remote peer has a single list number associated with it, if you want traffic to go to a bridge group and to either a ring list or port list, you should specify the same list number in each definition.

Task	Command
Define a ring list.	**dlsw bgroup-list** *list-number* **bgroups** *number*

Define DLSw+ Remote Peers

You can define direct, FST, or TCP encapsulation remote peers by performing one of the following tasks in global configuration mode:

Task	Command
Define a direct encapsulation in HDLC for remote peer.	**dlsw remote-peer** *list-number* **interface** *interface-name* [*mac-address*] [**cost** *cost*] [**lf** *size*] [**keepalive** *seconds*] [**lsap-output-list** *list*] [**host-netbios-out** *host-list-name*] [**bytes-netbios-out** *bytes-list-name*]
Define a direct encapsulation in Frame Relay for the remote peer.	**dlsw remote-peer** *list-number* **frame-relay interface serial** *number dlci-number* [**pass-thru**] [**cost** *cost*] [**lf** *size*] [**keepalive** *seconds*] [**lsap-output-list** *list*] [**host-netbios-out** *host-list-name*] [**bytes-netbios-out** *bytes-list-name*]
Define an FST encapsulation remote peer.	**dlsw remote-peer** *list-number* **fst** *ip-address* [**cost** *cost*] [**lf** *size*] [**keepalive** *seconds*] [**lsap-output-list** *list*] [**host-netbios-out** *host-list-name*] [**bytes-netbios-out** *bytes-list-name*] [**backup-peer** *ip-address*]
Define a TCP encapsulation remote peer.	**dlsw remote-peer** *list-number* **tcp** *ip-addr*ess [**priority**] [**cost** *cost*] [**lf** *size*] [**keepalive** *seconds*] [**tcp-queue-max** *size*] [**lsap-output-list** *list*] [**host-netbios-out** *host-list-name*] [**bytes-netbios-out** *bytes-list-name*] [**backup-peer** *ip-address*]

Refer to the DLSw+ Using TCP Encapsulation and LLC2 Local Acknowledgment—Basic Configuration Example for a sample configuration.

Configure Peer-on-Demand Defaults

To configure peer-on-demand defaults, perform the following task in global configuration mode:

Task	Command
Configure peer-on-demand defaults.	**dlsw peer-on-demand-defaults** {**fst** | **tcp**} [**bytes-netbios-out** *bytes-list-name*] [**cost** *cost*] [**host-netbios-out** *host-list-name*] [**keepalive** *seconds*] [**local-ack**] [**lsap-output-lis**t *list*] [**priority**][1]

1. The **local-ack** and **priority** keywords apply to TCP only.

Configure Static Resources Exchanged in Capabilities Exchange

To reduce explorer traffic destined for this peer, the peer can send other peers a list of resources for which it has information (icanreach) or does not have information (icannotreach). This information is exchanged as part of a capabilities exchange.To configure static resources that will be exchanged as part of a capabilities exchange, perform one of the following tasks in global configuration mode:

Task	Command
Configure a resource not locally reachable by the router.	**dlsw icannotreach saps** *sap* [*sap...*]
Configure a resource locally reachable by the router.	**dlsw icanreach** {**mac-exclusive** | **netbios-exclusive** | **mac-address** *mac-addr* [**mask** *mask*] | **netbios-name** *name*}

Configure Static Paths

To configure static paths to minimize explorer traffic originating in this peer, perform one of the following tasks in global configuration mode:

Task	Command
Configure the location or path of a static MAC address.	**dlsw mac-addr** *mac-addr* {**ring-group** *ring* | **remote-peer** {**interface serial** *number* | **ip-address** *ip-address*} | **group** *group*}
Configure a static NetBIOS name.	**dlsw netbios-name** *netbios-name* {**ring-group** *ring* | **remote-peer** {**interface serial** *number* | **ip-address** *ip-address*}| **group** *group*}

Configure Duplicate Path Handling

To configure duplicate path handling, perform the following task in global configuration mode:

Task	Command
Configure duplicate path handling.	**dlsw duplicate-path-bias** [**load-balance**]

Enable DLSw+ on the Appropriate Token Ring Interface

To enable DLSw+ on a Token Ring interface, perform the following task in interface configuration mode:

Task	Command
Enable DLSw+ on a Token Ring interface.	**source-bridge** *local-ring bridge-number ring-group*[1]

1. This command is documented in the "Source-Route Bridging Command" chapter of the *Router Products Command Referenc*e.

Enable DLSw+ on the Appropriate Ethernet Interface

To enable DLSw+ on certain Ethernet interfaces, perform the following task in interface configuration mode:

Task	Command
Enable DLSw+ on an Ethernet interface.	**dlsw bridge-group** *bgroup-num*

Enable DLSw+ on the Appropriate SDLC Interface

To enable DLSw+ on an SDLC interface, perform the following task in interface configuration mode:

Task	Command
Enable DLSw+ on an SDLC interface.	**sdlc dlsw** *station-address*, *station address ...*

Tune the DLSw+ Configuration

To modify an existing configuration parameter, perform one or more of the following tasks in global configuration mode:

Task	Command
Disable and re-enable DLSw+ without altering the configuration.	**dlsw disable**
Configure the depth of DLSw+ explorer queue router.	**dlsw explorerq-depth** *queue-max*
Configure DLSw+ timers.	**dlsw timer** {**icannotreach-block-time** | **netbios-cache-timeout** | **netbios-explorer-timeout** | **netbios-retry-interval** | **netbios-verify-interval** | **sna-cache-timeout** | **sna-explorer-timeout** | **sna-retry-interval** | **sna-verify-interval**} *time*

Monitor and Maintain the DLSw+ Network

To monitor and maintain activity on the DLSw+ network, perform one or more of the following tasks in privileged EXEC mode:

Task	Command
Display capabilities of a direct-encapsulated remote peer.	**show dlsw capabilities interface** *type* | *number*
Display capabilities of a TCP/FST remote peer.	**show dlsw capabilities ip-address** *ip-address*
Display capabilities of the local peer.	**show dlsw capabilities local**
Display DLSw+ circuit information.	**show dlsw circuits**
Display DLSw+ peer information.	**show dlsw peers**
Display DLSw+ reachability information.	**show dlsw reachability**

DLSw+ Configuration Examples

The following sections provide DLSw+ configuration examples:

- DLSw+ Using TCP Encapsulation and LLC2 Local Acknowledgment—Basic Configuration Example
- DLSw+ Using TCP Encapsulation with Local Acknowledgment—Peer Group Configuration Example 1
- DLSw+ Using TCP Encapsulation with Local Acknowledgment—Peer Group Configuration Example 2
- DLSw+ Translation between SDLC and Token Ring Media Example

DLSw+ Using TCP Encapsulation and LLC2 Local Acknowledgment—Basic Configuration Example

This sample configuration requires the following tasks, which are described in earlier sections of this document:

- Define a Source-Bridge Ring Group for DLSw+
- Define a DLSw+ Local Peer for the Router
- Define DLSw+ Remote Peers
- Enable DLSw+ on the Appropriate LAN Interface

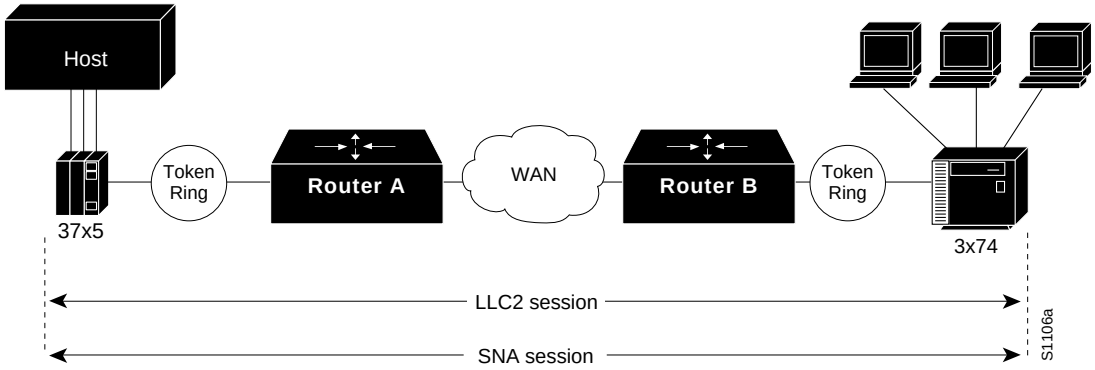
Encapsulate the source-route bridged traffic inside IP datagrams passed over a TCP connection between two routers/bridges with local acknowledgment enabled when you have LANs separated by wide geographic distances and you want to avoid multiple retransmissions or loss of user sessions that can occur with time delays.

Logical Link Control–Type 2 (LLC2) is an ISO standard data link level protocol used in Token Ring networks. LLC2 was designed to ensure reliable transmission of data across LAN media with minimal or predictable time delays. With the advent of DLSw+ and WAN backbones, LANs are now separated by wide, geographic distances spanning countries and continents. As a result, these LANs have time delays that are longer than LLC2 allows for bidirectional communication between hosts. The local acknowledgment capability in routers and bridges supporting remote source-route bridging addresses the problem of unpredictable time delays, multiple retransmissions, and loss of user sessions.

In a typical LLC2 session, when one host sends a frame to another host, the sending host expects the receiving host to respond positively or negatively in a predefined period of time commonly called the *T1 time*. If the sending host does not receive an acknowledgment of the frame it sent within the T1 time, it retries a few times (normally 8 to 10). If there is still no response, the sending host drops the session.

Figure 30-4 illustrates an LLC2 session. A 37x5 on a LAN segment can communicate with a 3x74 on a different LAN segment separated via a wide-area backbone network. Frames are transported between Router A and Router B by means of DLSw+. However, the LLC2 session between the 37x5 and the 3x74 is still end-to-end; that is, every frame generated by the 37x5 traverses the backbone network to the 3x74, and the 3x74, on receipt of the frame, acknowledges it.

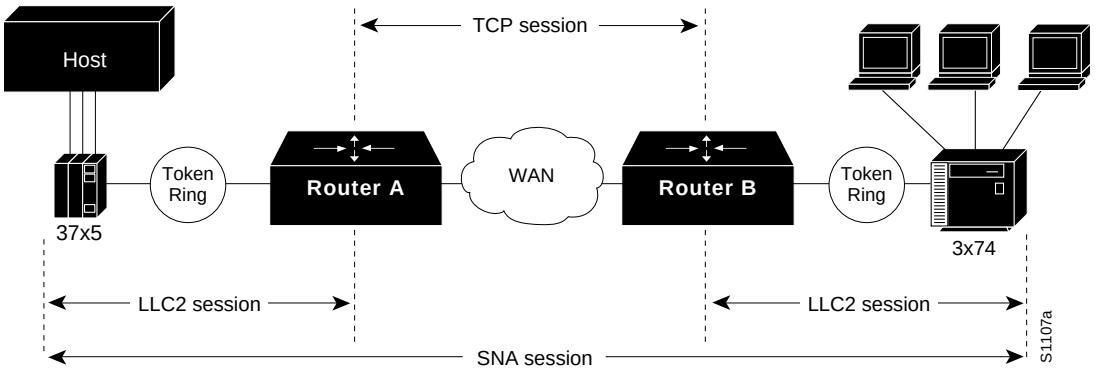
Figure 30-4 LLC2 Session without Local Acknowledgment



On backbone networks consisting of slow serial links, the T1 timer on end hosts could expire before the frames have a chance to reach the remote hosts, causing the end host to retransmit. This results in duplicate frames reaching the remote host at the same time as the first frame also reached the remote host, albeit slowly. These frame duplications break the LLC2 protocol, resulting in the loss of sessions between the two IBM machines.

One way to solve this time delay problem is to increase the timeout value on the end nodes to account for the maximum transit time between the two end machines. However, in networks consisting of hundreds or even thousands of nodes, every machine would need to be reconfigured with new values. With local acknowledgment for LLC2 turned on, the LLC2 session between the two end nodes would not be end-to-end, but instead, terminate at two local routers. Figure 30-5 shows the LLC2 session with the 37x5 ending at Router A and the LLC2 session with the 3x74 ending at Router B. Both Router A and Router B execute the full LLC2 protocol as part of local acknowledgment for LLC2.

Figure 30-5 LLC2 Session with Local Acknowledgment



With local acknowledgment for LLC2 enabled in both routers, Router A acknowledges frames received from the 37x5. The 37x5 still recognizes the acknowledgments it receives as belonging to the 3x74. Router A looks like the 3x74 to the 37x5. Similarly, Router B acknowledges frames received from the 3x74. The 3x74 recognizes the acknowledgments it receives as coming from the 37x5. Router B looks like the 3x74 to 37x5. Because the frames no longer have to travel the WAN backbone networks to be acknowledged, but instead are locally acknowledged by routers, the end machines do not time out, resulting in no loss of sessions.

The advantages of enabling local acknowledgment for LLC2 include the following:

- Local acknowledgment for LLC2 solves the T1 timer problem without having to change any configuration on the end nodes. The end nodes are unaware that the sessions are being locally acknowledged. In networks consisting of hundreds or even thousands of machines, this is a definite advantage. All the frames acknowledged by the router appear to the end hosts to be coming from the remote IBM machine. In fact, by looking at a trace from a protocol analyzer, you cannot determine whether a frame was acknowledged by the local router or by a remote IBM machine. The MAC addresses generated by the routers are identical to those generated by the remote IBM machine.
- All the supervisory (RR, RNR, REJ) frames that are locally acknowledged go no farther than the router. Without local acknowledgment for LLC2, *every* frame traverses the backbone. With local acknowledgment, only data (I-frames) traverses the backbone, resulting in less traffic on the backbone network. For installations in which customers pay for the amount of traffic passing through the backbone, this could be a definite cost-saving measure. A simple protocol exists between the two *peers* to bring up or down a TCP session.

Notes on Using LLC2 Local Acknowledgment

LLC2 local acknowledgment is enabled only with TCP remote peers (as opposed to LAN or direct serial interface remote peers).

If the LLC2 session between the local host and the router terminates in either router, the other will be informed to terminate its connection to its local host.

If the TCP queue length of the connection between the two routers reaches the highwater mark, the routers sends Receiver-Not-Ready (RNR) messages to the local hosts until the queue limit is reduced to below this limit.

The configuration of the LLC2 parameters for the local Token Ring interfaces can affect overall performance. Refer to the chapter “Configuring LLC2 and SDLC Parameters” in this manual for more details about fine-tuning your network through the LLC2 parameters.

The routers at each end of the LLC2 session execute the full LLC2 protocol, which could result in some overhead. The decision to use local acknowledgment for LLC2 should be based on the speed of the backbone network in relation to the Token Ring speed. For LAN segments separated by slow-speed serial links (for example, 56 kbps), the T1 timer problem could occur more frequently. In such cases, it might be wise to turn on local acknowledgment for LLC2. For LAN segments separated by a T1, backbone delays will be minimal; in such cases, FST or direct should be used. Speed mismatch between the LAN segments and the backbone network is one criterion to help you decide to use local acknowledgment for LLC2.

There are some situations (such as the receiving host failing between the time the sending host sends data and the time the receiving host receives it), in which the sending host would determine, *at the LLC2 layer*, that data was received when it actually was not. This error occurs because the router acknowledges that it received data from the sending host before it determines that the receiving host can actually receive the data. But because both NetBIOS and SNA have error recovery in situations where an end device goes down, these higher-level protocols will resend any missing or lost data. Because these transaction request/confirmation protocols exist above LLC2, they are not affected by tight timers, as is LLC2. They also are transparent to local acknowledgment.

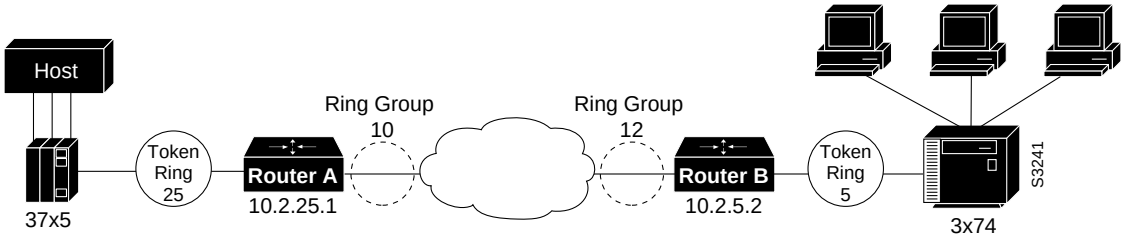
If you are using NetBIOS applications, note that there are two NetBIOS timers—one at the link level and one at the next higher level. Local acknowledgment for LLC2 is designed to solve link timeouts only. If you are experiencing NetBIOS session timeouts, you have two options:

- Experiment with increasing your NetBIOS timers and decreasing your maximum NetBIOS frame size.

- Avoid using NetBIOS applications on slow serial lines.

Figure 30-6 illustrates a DLSw+ configuration with local acknowledgment.

Figure 30-6 DLSw+ with Local Acknowledgment—Simple Configuration



Configuration for Router A

```
source-bridge ring-group 10
!
dlsw local-peer peer-id 10.2.25.1
dlsw remote-peer 0 tcp 10.2.5.2
int loopback 0
ip address 10.2.25.1 255.255.255.0
.
.
.
interface TokenRing 0
no ip address
ring-speed 16
source-bridge active 25 1 10
```

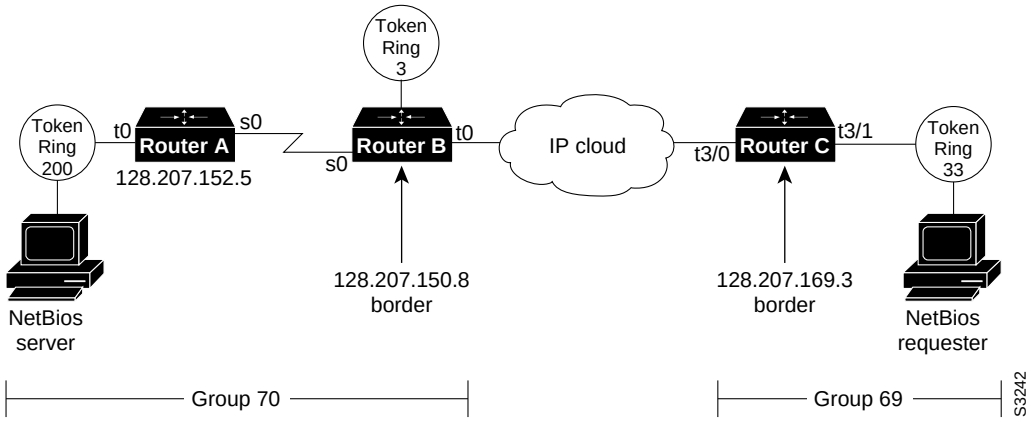
Configuration for Router B

```
source-bridge ring-group 12
dlsw local-peer peer-id 10.2.5.2
dlsw remote-peer 0 tcp 10.2.25.1
int loopback 0
ip address 10.2.5.2 255.255.255.0
.
.
.
interface TokenRing 0
no ip address
ring-speed 16
source-bridge active 5 1 12
```

DLSw+ Using TCP Encapsulation with Local Acknowledgment—Peer Group Configuration Example 1

Figure 30-7 illustrates DLSw+ configured using border peers, showing circuits to each other. Router A is configured to operate in promiscuous mode, and border peers Routers B and C forward broadcasts. This configuration reduces processing requirements in router A (the access router) and still support any-to-any networks.

Figure 30-7 DLSw with Local Acknowledgment—Peer Group Configuration 1



The following are configuration files for the routers in Figure 30-7.

Configuration for Router A

```
hostname RouterA
!
source-bridge ring group 31
dlsw local-peer peer-id 128.207.152.5 group 70 promiscuous
dlsw remote peer 0 tcp 128.207.150.8
!
interface Serial0
ip unnumbered TokenRing
clockrate 56000
!
interface TokenRing 0
ip address 128.207.152.5 255.255.255.0
ring-speed 16
source-bridge 200 13 31
source-bridge spanning
!
.
.
.
router igrp 777
network 128.207.0.0
```

Configuration for Router B

```
hostname RouterB
!
.
.
.
source-bridge ring-group 31
dlsw local-peer peer-id 128.207.150.8 group 70 border promiscuous
dlsw remote-peer 0 tcp 128.207.169.3
dlsw remote-peer 0 tcp 128.207.152.5
!
.
.
.
interface serial0
```

```
ip unnumbered tokenring 0
bandwidth 56
!
.
.
.
interface tokenring 0
ip address 128.207.150.8 255.255.255.0
ring-speed 16
source-bridge 3 14 31
source-bridge spanning
!
router igrp 777
network 128.207.0.0
```

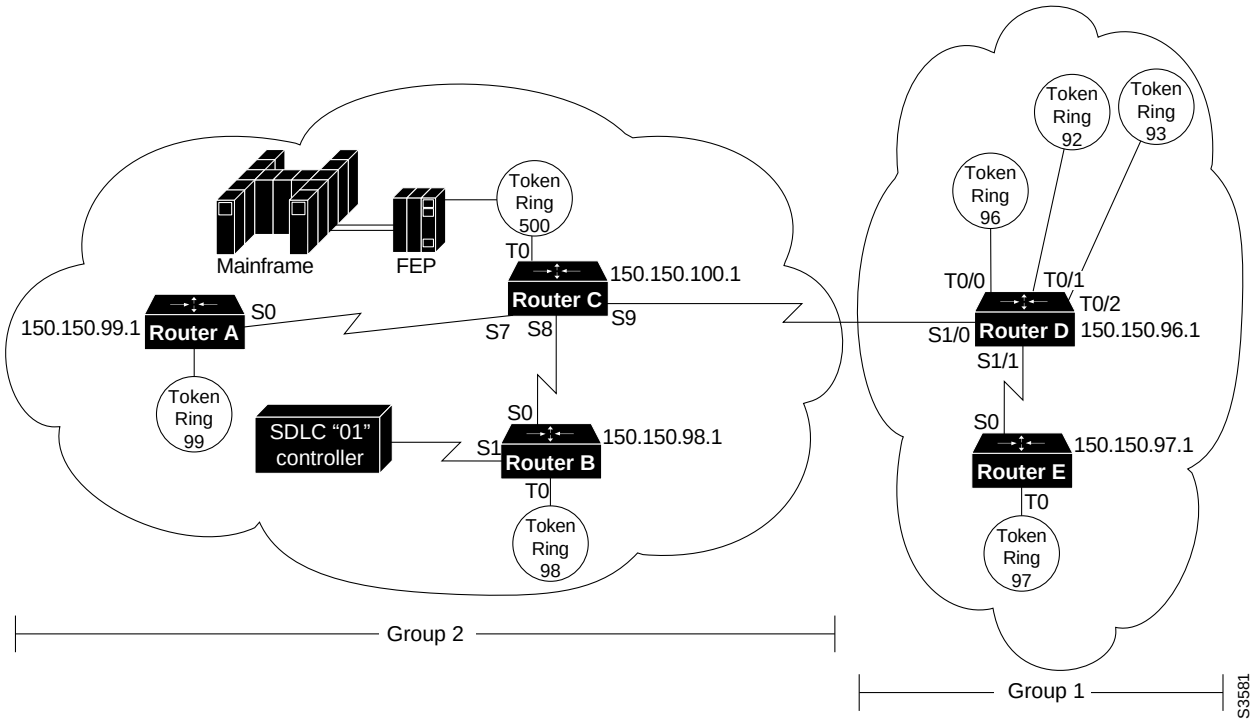
Configuration for Router C

```
hostname RouterC
!
.
.
.
source-bridge ring-group 69
dlsw local-peer peer-id 128.207.169.3 group 69 border promiscuous
dlsw remote-peer 0 tcp 128.207.150.8
!
.
.
.
interface tokenring 3/0
description fixed to flashnet
ip address 128.207.2.152 255.255.255.0
ring-speed 16
multiring all
!
interface tokenring 3/1
ip address 128.207.169.3 255.255.255.0
ring-speed 16
source-bridge 33 2 69
source-bridge spanning
!
.
.
.
router igrp 777
network 128.207.0.0
```

DLSw+ Using TCP Encapsulation with Local Acknowledgment—Peer Group Configuration Example 2

Figure 30-8 illustrates a peer group configuration that allows any-to any connection except for Router B.

Figure 30-8 DLSw+ with Local Acknowledgment—Peer Group Configuration 2



The following are configuration files for the routers in Figure 30-8:

Configuration for Router A

```
hostname Router A
!
.
.
.
source-bridge ring-group 2000
dlsw local-peer peer-id 150.150.99.1
dlsw remote-peer 0 tcp 150.150.100.1
!
interface loopback 0
 ip address 150.150.99.1 255.255.255.192
!
interface serial 0
 ip address 150.150.9.1 255.255.255.192
!
.
.
.
interface tokenring 0
 no ip address
 ring-speed 16
 source-bridge 99 1 2000
 source-bridge spanning
!
.
.
.
```

```
router eigrp 202
 network 150.150.0.0
```

Configuration for Router B

```
hostname RouterB
!
.
.
.
source-bridge ring-group 2000
dlsw local-peer peer-id 150.150.98.1 group 2
dlsw remote-peer 0 tcp 150.150.100.1
!
interface loopback 0
 ip address 150.150.98.1 255.255.255.192
!
.
.
.
interface serial 1
 no ip address
 encapsulation sdlc
 no keepalive
 priority-group 1
 clockrate 9600
 sdlc role primary
 sdlc vmac 4000.8888.0100
 sdlc address 01
 sdlc xid 01 05d20006
 sdlc partner 4000.1020.1000 01
 sdlc dlsw 1
!
interface tokenring 0
 no ip address
 ring-speed 16
 multiring all
 source-bridge 98 1 2000
 source-bridge spanning
!
.
.
.
router eigrp 202
 network 150.150.0.0
```

Configuration for Router C

```
hostname RouterC
!
.
.
.
source-bridge ring-group 2000
dlsw local-peer peer-id 150.150.100.1 group 2 border promiscuous
dlsw remote-peer 0 tcp 150.150.96.1
dlsw remote-peer 0 tcp 150.150.98.1
dlsw remote-peer 0 tcp 150.150.99.1
!
interface loopback 0
 ip address 150.150.100.1 255.255.255.192
!
.
```

```
.
.
interface serial 7
 ip address 150.150.9.2 255.255.255.192
 clockrate 56000
!
interface serial 8
 ip address 150.150.10.2 255.255.255.192
!
interface serial 9
 ip address 150.150.8.2 255.255.255.192
!
interface tokenring 0
 no ip address
 ring-speed 16
 multiring all
 source-bridge 500 1 2000
!
router eigrp 202
 network 150.150.0.0
```

Configuration for Router D

```
hostname RouterD
!
.
.
.
source-bridge ring-group 2000
dlsw local-peer peer-id 150.150.96.1 group 1 border
dlsw remote-peer 0 tcp 150.150.97.1
dlsw remote-peer 0 tcp 150.150.100.1
!
interface loopback 0
 ip address 150.150.96.1 255.255.255.192
!
.
.
.
interface serial 1/0
 ip address 150.150.8.1 255.255.255.192
 clockrate 56000
!
interface serial 1/1
 ip address 150.150.16.1 255.255.255.192
!
.
.
.
interface tokenrin g0/0
 ip address 150.150.2.1 255.255.255.192
 ring-speed 16
 source-bridge 96 1 2000
 source-bridge spanning
!
interface tokenring 0/1
 ip address 150.150.13.1 255.255.255.192
 no ip address
 ring-speed 16
 source-bridge 92 1 2000
 source-bridge spanning
!
.
.
```

```
.
router eigrp 202
 network 150.150.0.0
```

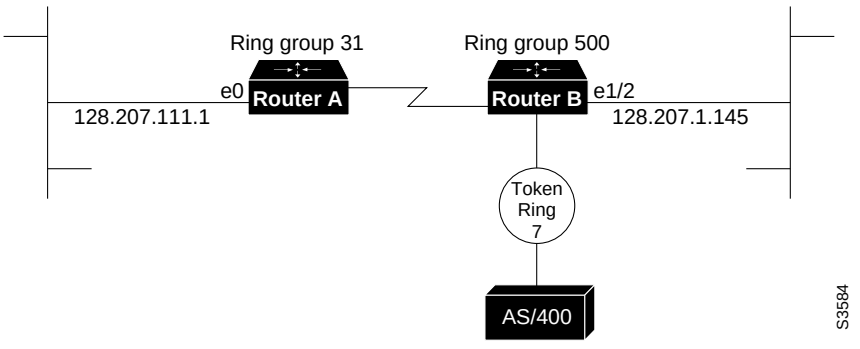
Configuration for Router E

```
hostname RouterE
!
.
.
.
source-bridge ring-group 2000
dlsw local-peer peer-id 150.150.97.1 group 1 promiscuous
dlsw remote-peer 0 tcp 150.150.96.1
!
interface loopback 0
 ip address 150.150.97.1 255.255.255.192
!
interface serial 0
 mtu 1400
 ip address 150.150.16.2 255.255.255.192
!
.
.
.
interface tokenring 0
 no ip address
 no ip route-cache
 ring-speed 16
 source-bridge 97 1 2000
 source-bridge spanning
!
.
.
.
router eigrp 202
 network 150.150.0.0
```

DLSw+ Translation between Ethernet and Token Ring Configuration Example

DLSw+ also supports Ethernet media. Except for configuring for a specific media, in this case Ethernet, the configuration is similar to other DLSw+ configuration.

Figure 30-9 DLSw+ Translation between Ethernet and Token Ring Media



The following are the configuration files for the routers in figure 26-9.

Configuration for Router A

```
hostname RouterA
!
.
.
.
source-bridge ring-group 31
dlsw local-peer peer-id 128.207.2.155
dlsw remote-peer 0 tcp 128.207.1.145 lf 1500
dlsw bridge-group 5
!
interface Ethernet 0
 ip address 128.207.111.1 255.255.255.0
 bridge-group 5
!
.
.
bridge 5 protocol ieee
!
.
.
```

Configuration for Router B

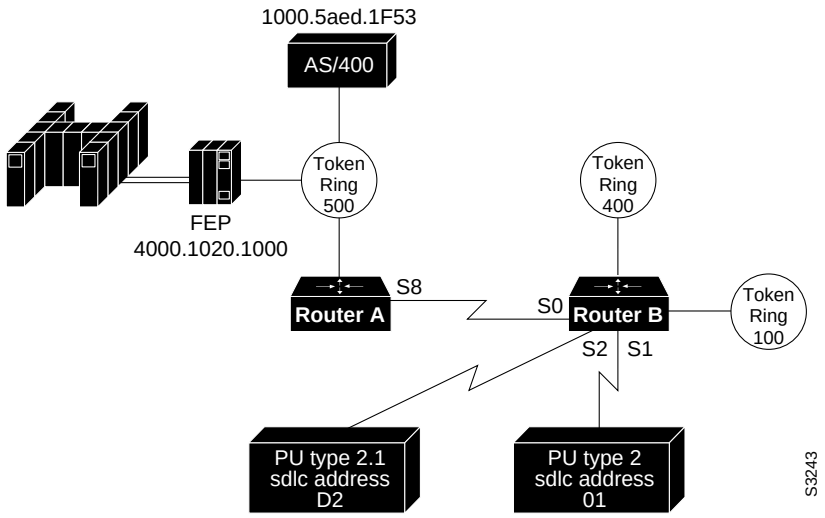
```
hostname RouterB
!
.
.
.
source-bridge ring-group 500
dlsw local-peer peer-id 128.207.1.145
dlsw remote-peer 0 tcp 128.207.2.155 lf 1500
 dlsw bridge-group 5
.
.
.
interface ethernet 1/2
 ip address 128.207.1.145 255.255.255.0
 bridge-group 5

.
.
.
interface tokenring 2/0
 no ip address
 ring-speed 16
 multiring all
 source-bridge 7 1 500
 source-bridge spanning
!
.
.
.
router igrp 777
 network 128.207.0.0
```

DLSw+ Translation between SDLC and Token Ring Media Example

DLSw+ provides media conversion between local or remote LANs and SDLC. For additional information about configuring SDLC parameters, refer to the chapter "Configuring LLC2 and SDLC Parameters."

Figure 30-10 illustrates DLSw+ with SDLC encapsulation. For this example, 4000.1020.1000 is the MAC address of the FEP host (PU4). 1000.5aed.1f53 is the MAC address of the AS/400 host, which is defined as Node Type 2.1. Router B serves as the primary station for the remote secondary stations 01. Router B can serve as either primary station or secondary station to remote station D2.

Figure 30-10 DLSw+ Translation between SDLC and Token Ring Media



The following is the configuration file for Router A:

Configuration for Router A

```
hostname RouterA
!
.
.
.
source-bridge ring-group 2000
dlsw local-peer peer-ed 150.150.10.2
dlsw remote-peer 0 tcp 150.150.10.1
!
.
.
.
interface serial 8
 ip address 150.150.10.2 255.255.255.192
 clockrate 56000
!
.
.
.
interface tokenring 0
 no ip address
 ring-speed 16
 source-bridge 500 1 2000
```

```
source-bridge spanning
!
.
.
.
router eigrp 202
 network 150.150.0.0
```

Configuration for Router B

```
hostname RouterB
!
.
.
.
source-bridge ring-group 2000
dlsw local-peer peer-id 150.150.10.1
dlsw remote-peer 0 tcp 150.150.10.2
!
.
.
.
interface serial 0
 ip address 150.150.10.1 255.255.255.192
!
interface serial 1
 description PU2 with SDLC station role set to secondary
 no ip address
 encapsulation sdlc
 no keepalive
 clockrate 9600
 sdlc role primary
 sdlc vmac 4000.9999.0100
 sdlc address 01
 sdlc xid 01 05d20006
 sdlc partner 4000.1020.1000 01
sdlc dlsw 1
!
interface serial 2
 description Node Type 2.1 with SDLC station role set to negotiable or primary
 encapsulation sdlc
 sdlc role none
 sdlc vmac 1234.3174.0000
 sdlc address d2
 sdlc partner 1000.5aed.1f53 d2
 sdlc dlsw d2
!
interface tokenring 0
 no ip address
 early-token-release
 ring-speed 16
 multiring all
 source-bridge 100 1 2000
 source-bridge spanning
!
interface tokenring 1
 no ip address
 ring-speed 16
 multiring all
 source-bridge 400 1 2000
 source-bridge spanning
!
router eigrp 202
 network 150.150.0.0\
```

INDEX

Symbols

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z