



## Configuring XNS

---

This chapter describes how to configure standard Xerox Network System (XNS) routing and Ungermann-Bass Net/One XNS routing. It also provides configuration examples. For a complete description of XNS commands, refer to the “XNS Commands” chapter in the *Cisco IOS Apollo Domain, Banyan VINES, DECnet, ISO CLNS, and XNS Command Reference* publication. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the [“Identifying Supported Platforms”](#) section in the “Using Cisco IOS Software” chapter.

**Note**

---

The XNS networking protocol will no longer be offered after Cisco IOS Release 12.2. Information about this protocol will not appear in future releases of the Cisco IOS software documentation set.

---

**Note**

---

Not all Cisco access servers support XNS. For more information, refer to the release notes for the current Cisco IOS release.

---

## 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 Cisco routers in Net/One environments, consider the following differences between Net/One and standard XNS environments:

- Net/One routers use a proprietary routing protocol instead of the standard XNS Routing Information Protocol (RIP). Although they generate both Ungermann-Bass and standard RIP update packets, Net/One routers listen only to Net/One updates. Cisco supports both the reception and the generation of Net/One routing packets. Also, Cisco 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. Cisco 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 the packets of this protocol. Ungermann-Bass routers pass the booting protocol from network to network and modify the

embedded network numbers. The Cisco equipment does not understand the Net/One booting protocol and does not modify the embedded network numbers. For network booting to work through Cisco routers, Net/One Network Management Consoles must be specially configured. Contact Ungermann-Bass for information on how to perform this type of configuration.

- 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 MAC-layer broadcasts and are expected to be flooded throughout the XNS internetwork. On Cisco routers, you enable the flooding of these packets as described in the “[Controlling Broadcast Messages](#)” section later in this chapter.
- Net/One equipment uses proprietary network management protocols. Cisco 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, Cisco routers participate in this protocol and behave (insofar as routing protocols are concerned) like Ungermann-Bass routers.

Cisco routers also can be configured to listen to standard RIP updates when in Ungermann-Bass Net/One emulation mode. When a Cisco 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 Cisco IOS 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 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 0 identifies the local network. The XNS network number is expressed in decimal format in the Cisco configuration files and routing tables. However, the router internally converts the network number into hexadecimal format. In this case, for instance, a network analyzer will display the network number in hexadecimal format.

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:

```
47.0000.0x00.23Ee
```

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.

## XNS Configuration Task List

To configure XNS routing, perform the tasks in the following sections. At a minimum, you must enable routing.

- [Enabling XNS Routing](#) (Required)
- [Controlling Access to the XNS Network](#) (Optional)
- [Tuning XNS Network Performance](#) (Optional)
- [Configuring XNS over WANs](#) (Optional)
- [Routing XNS Between Virtual LANs](#) (Optional)
- [Monitoring the XNS Network](#) (Optional)

See the “XNS Configuration Examples” section at the end of this chapter for configuration examples.

## Enabling 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, their default is to receive only Ungermann-Bass updates. To receive RIP updates, use the **xns hear-rip** global configuration command.

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. Cisco IOS software supports both the reception and the generation of Ungermann-Bass routing packets.

## Enabling Standard XNS Routing

To enable standard XNS routing, use the following commands beginning in global configuration mode:

|               | Command   | Purpose                              |
|---------------|---|--------------------------------------|
| <b>Step 1</b> | Router(config)# <b>xns routing</b> [ <i>address</i> ] | Enables XNS routing on the router.   |
| <b>Step 2</b> | Router(config)# <b>interface</b> <i>type number</i>   | Enters interface configuration mode. |
| <b>Step 3</b> | Router(config-if)# <b>xns network</b> <i>number</i>   | Enables XNS routing on an interface. |

For an example of how to enable XNS routing, see the “[XNS Routing Configuration Example](#)” section at the end of this chapter.

If you omit the address from the **xns routing** global configuration command, Cisco IOS software uses the address of the first IEEE-compliant (Token Ring, FDDI, or Ethernet) interface MAC address it finds in its interface list. The software 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 specify this encapsulation type, use one of the following commands in interface configuration mode:

| Command  | Purpose   |
|--|---|
| Router(config-if)# <b>xns encapsulation snap</b> | Encapsulates XNS packets being forwarded across an IBM Token Ring network.            |
| Router(config-if)# <b>xns encapsulation ub</b>   | Encapsulates XNS packets being forwarded across an Ungermann-Bass Token Ring network. |
| Router(config-if)# <b>xns encapsulation 3com</b> | Encapsulates XNS packets being forwarded across an older 3Com Token Ring network.     |

## Enabling Concurrent Routing and Bridging

You can route XNS on some interfaces and transparently bridge it on other interfaces simultaneously. To enable this type of routing, you must enable concurrent routing and bridging. To enable concurrent routing and bridging for the router, use the following command in global configuration mode:

| Command                           | Purpose   |
|-----------------------------------|---|
| Router(config)# <b>bridge crb</b> | Enables concurrent routing and bridging for the router. |

## Enabling Ungermann-Bass Net/One Routing

To enable Ungermann-Bass Net/One routing, use the following commands beginning in global configuration mode:

|               | Command   | Purpose   |
|---------------|---|---|
| <b>Step 1</b> | Router(config)# <b>xns ub-emulation</b>                                 | Enables Ungermann-Bass Net/One routing on the router. |
| <b>Step 2</b> | Router(config)# <b>interface</b> <i>type number</i>                     | Enters interface configuration mode.                  |
| <b>Step 3</b> | Router(config-if)# <b>xns hear-rip</b><br>[ <i>access-list-number</i> ] | Enables the receipt of RIP updates on the interface.  |

For an example of how to enable Ungermann-Bass Net/One routing, see the “[Net/One Routing Configuration Example](#)” section at the end of this chapter.

## Controlling Access to the XNS Network

To control access to XNS networks, you create access lists and then apply them with filters to individual interfaces.

Following are the 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.

Of the following two different types of filters you can define for XNS interfaces, 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 Cisco IOS software and from which routers the local router accepts RIP updates.

[Table 6](#) 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 6** XNS Filters

| Filter Type   | Command Used to Define Filter                             |
|---|---|
| <b>Generic filters</b>  |   |
| Filter based on protocol, address and address mask, port and port mask. | <code>xns access-group access-list-number</code>          |
| <b>Routing table filters</b>  |   |
| Filter which networks are added to the routing table.                   | <code>xns input-network-filter access-list-number</code>  |
| Filter which networks are advertised in routing updates.                | <code>xns output-network-filter access-list-number</code> |
| Control the routers from which updates are accepted.                    | <code>xns router-filter access-list-number</code>         |

You perform one or more of the tasks in the following sections to control access to XNS networks:

- [Creating Access Lists](#)
- [Creating Generic Filters](#)
- [Creating Filters for Updating the Routing Table](#)

Consider the following when you configure 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. Consequently, 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.

## Creating Access Lists

To create access lists, use either of the following commands in global configuration mode:

| Command  | Purpose                              |
|--|--------------------------------------|
| Router(config)# <b>access-list</b> <i>access-list-number</i> { <b>deny</b>   <b>permit</b> }<br><i>source-network</i> [. <i>source-address</i> [ <i>source-address-mask</i> ]]<br>[ <i>destination-network</i> [. <i>destination-address</i> [ <i>destination-address-mask</i> ]]]   | Creates a standard XNS access list.  |
| Router(config)# <b>access-list</b> <i>access-list-number</i> { <b>deny</b>   <b>permit</b> } <i>protocol</i><br>[ <i>source-network</i> [. <i>source-host</i> [ <i>source-network-mask</i> . <i>source-host-mask</i> ]]]<br><i>source-socket</i> [ <i>destination-network</i> [. <i>destination-host</i><br>[ <i>destination-network-mask</i> . <i>destination-host-mask</i> ]<br>[ <i>destination-socket</i> [/pep]]] | Creates an extended XNS access list. |

Once you have created an access list, you activate the access list by applying it to a filter on the appropriate interfaces as described in the sections that follow.

For an example of how to create an access list to control services between networks in a 3Com network, see the “[3Com Access List Example](#)” section at the end of this chapter.

## Creating Generic Filters

Generic filters determine which packets to send out on an interface, based on the source of the packet and destination addresses, XNS protocol type, and source and destination socket numbers.

To create generic filters, perform the following tasks:

- Create a standard or extended access list.
- Apply a filter to an interface.

To create an access list, use one of the following commands in global configuration mode:

| Command   | Purpose                              |
|---|--------------------------------------|
| Router(config)# <b>access-list</b> <i>access-list-number</i> { <b>deny</b>   <b>permit</b> } <i>source-network</i> [. <i>source-address</i> [ <i>source-address-mask</i> ]] [ <i>destination-network</i> [. <i>destination-address</i> [ <i>destination-address-mask</i> ]]]  | Creates a standard XNS access list.  |
| Router(config)# <b>access-list</b> <i>access-list-number</i> { <b>deny</b>   <b>permit</b> } <i>protocol</i> [ <i>source-network</i> [. <i>source-host</i> [ <i>source-network-mask</i> . <i>source-host-mask</i> ]]] <i>source-socket</i> [ <i>destination-network</i> [. <i>destination-host</i> [ <i>destination-network-mask</i> . <i>destination-host-mask</i> ]<br>[ <i>destination-socket</i> [/pep]]] | Creates an extended XNS access list. |

To apply a generic filter to an interface, use the following command in interface configuration mode:

| Command  | Purpose                                   |
|--|---|
| Router(config-if)# <b>xns access-group</b> <i>access-list-number</i> | Applies a generic filter to an interface. |

For an example of creating a generic access list, see the “[3Com Access List Example](#)” section at the end of this chapter.

## Creating Filters for Updating the Routing Table

Routing table update filters control the entries that the router or access server 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:

- Create a standard or an extended access list.
- Apply one or more filters to an interface.

To create an access list, use one of the following commands in global configuration mode:

| Command  | Purpose                              |
|--|--------------------------------------|
| <pre>Router(config)# <b>access-list</b> <i>access-list-number</i> {deny   permit} <i>source-network</i> [<i>.source-address</i> [<i>source-address-mask</i>]] [<i>destination-network</i> [<i>.destination-address</i> [<i>destination-address-mask</i>]]]</pre>   | Creates a standard XNS access list.  |
| <pre>Router(config)# <b>access-list</b> <i>access-list-number</i> {deny   permit} <i>protocol</i> [<i>source-network</i> [<i>.source-host</i> [<i>source-network-mask</i>.<i>source-host-mask</i>]]] <i>source-socket</i> [<i>destination-network</i> [<i>.destination-host</i> [<i>destination-network-mask</i>.<i>destination-host-mask</i>] [<i>destination-socket</i> [/pep]]]</pre> | Creates an extended XNS access list. |

Standard access list numbers can be from 400 to 499. Extended access list numbers can be from 500 to 599.

For an example of how to create an extended access list entry, see the “[Extended Access List with Network Mask Option Example](#)” section at the end of this chapter.

To assign routing table update filters to an interface, use one of the following commands in interface configuration mode. You can apply one of each of the following filters to each interface.

| Command  | Purpose   |
|--|---|
| <pre>Router(config-if)# <b>xns input-network-filter</b> <i>access-list-number</i></pre>  | Controls which networks are added to the routing table when XNS routing updates are received. |
| <pre>Router(config-if)# <b>xns output-network-filter</b> <i>access-list-number</i></pre> | Controls which networks are advertised in routing updates sent out by the router.             |
| <pre>Router(config-if)# <b>xns router-filter</b> <i>access-list-number</i></pre>         | Controls the routers from which routing updates are accepted.                                 |

## Tuning XNS Network Performance

To tune XNS network performance, perform the tasks in one or more of the following sections:

- [Configuring Static Routes](#) (Optional)
- [Setting Routing Table Update Timers](#) (Optional)
- [Setting Maximum Paths](#) (Optional)

- [Controlling Broadcast Messages](#) (Optional)
- [Disabling XNS Fast Switching](#) (Optional)

## Configuring Static Routes

XNS uses 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 routing table of the router, use the following command in global configuration mode:

| Command  | Purpose                                   |
|--|---|
| Router(config)# <b>xns route</b> <i>network network.host</i> | Adds a static route to the routing table. |

## Setting 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 Cisco 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, use the following command in interface configuration mode:

| Command   | Purpose                    |
|---|----------------------------|
| Router(config-if)# <b>xns update-time</b> <i>interval</i> | Sets the RIP update timer. |

For an example of setting the RIP update timer, see the “[Routing Update Timers Example](#)” section at the end of this chapter.

## Setting Maximum Paths

You can set the maximum number of equal-cost, parallel paths to a destination. (Note that when paths have differing costs, Cisco IOS software chooses lower-cost routes in preference to higher-cost routes.) The software 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 with 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 among many paths.

To set the maximum number of paths on the router, use the following command in global configuration mode:

| Command   | Purpose   |
|---|---|
| Router(config)# <code>xns maximum-paths number</code> | Sets the maximum number of equal-cost paths to a destination. |

## Controlling 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 an 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 destination field of the packet. If a node does not know the number of the local XNS network (common when booting), it can use a network number of 0 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 1's (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 destination host field of the packet. 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 0 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 1's, 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. Cisco supports 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, Cisco IOS software 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 sender of the packet. A helper address can be a directed broadcast address, in which case the helpered packet is forwarded to a remote network and is 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 generates 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 must ensure that users can broadcast between networks without cluttering the internetwork with unnecessary traffic.

Whenever Cisco IOS software 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 Cisco IOS software and is not forwarded any further.
- If a helper address is set on the interface on which the packet arrived, and the protocol type of the packet appears in the **xns forward-protocol** list, the packet is forwarded to the helper address. The helper address can be a directed broadcast address.

## Forwarding Broadcast Messages to Specified Hosts

To configure helpering, which forwards broadcast messages to the specified host or hosts, use the following command in interface configuration mode:

| Command  | Purpose  |
|--|--|
| Router(config-if)# <b>xns helper-address</b> <i>network.host</i> | Forwards broadcast messages to the specified host. |

You can specify multiple **xns helper-address** configuration mode commands on a given interface.

For an example of forwarding broadcast messages, see the “[Helpering Example](#)” section at the end of this chapter.

## Specifying 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 forward packets, use the following command in global configuration mode:

| Command   | Purpose  |
|---|--|
| Router(config)# <b>xns forward-protocol</b> <i>protocol</i> | Forwards packets of a specific XNS protocol to a helper address. |

## Configuring Flooding

Different XNS implementations require different flooding behavior. By default, Cisco routers do no flooding. You can, however, configure interfaces on the router to apply flooding to the packets *received* on an interface.

Whenever Cisco IOS software 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 1's (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.

Cisco IOS software 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 software 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 *source* network of the packet. If there are multiple paths to the source network, packets received only on the primary path (the first path the software 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 paths of the router back toward the source of the packet. 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 source network of the packet, the packet is not flooded. This feature prevents 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 results in that interface having no helper address set, but having **xns flood broadcast allnets** enabled.

To define the flooding behavior of an interface, use one of the following commands in interface configuration mode:

| Command  | Purpose  |
|--|--|
| Router(config-if)# <b>xns flood broadcast allnets</b>  | Floods packets whose destination address is -1.FFFF.FFFF.FFFF. |
| Router(config-if)# <b>xns flood broadcast net-zero</b> | Floods packets whose destination address is 0.FFFF.FFFF.FFFF.  |
| Router(config-if)# <b>xns flood specific allnets</b>   | Floods packets with destinations of -1. <i>specific-host</i> . |

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 broadcasts whose network address is all 0's.

## Disabling 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, use the following command in interface configuration mode:

| Command                                      | Purpose                      |
|--|------------------------------|
| Router(config-if)# <b>no xns route-cache</b> | Disables XNS fast switching. |

## Configuring XNS over WANs

You can configure XNS over X.25, Frame Relay, and Switched Multimegabit Data Service (SMDS) networks by configuring the appropriate address mappings as described in the “Configuring X.25 and LAPB,” “Configuring Frame Relay,” and “Configuring SMDS” chapters in the *Cisco IOS Wide-Area Networking Configuration Guide*.

## Routing XNS Between Virtual LANs

XNS can be routed over virtual LAN (VLAN) subinterfaces using the Inter-Switched Link (ISL) VLAN encapsulation protocol. Full-feature Cisco IOS software is supported on a per-VLAN basis, allowing standard XNS capabilities to be configured on VLANs. For detailed information about configuring XNS routing between VLANs, refer to the *Cisco IOS Switching Services Configuration Guide*.

## Monitoring the XNS Network

To monitor an XNS network, use one or more of the following commands in user EXEC or privileged EXEC mode:

| Command   | Purpose   |
|---|---|
| Router> <b>show xns cache</b>                   | Lists the entries in the XNS fast-switching cache.  |
| Router> <b>show xns interface</b> [type number] | Displays the status of the XNS interfaces configured in the router and the parameters configured on each interface. |
| Router> <b>show xns route</b> [network]         | Lists the entries in the XNS routing table.   |
| Router> <b>show xns traffic</b>                 | Displays information about the number and type of XNS packets sent and received.                                    |
| Router> <b>ping xns address</b>                 | Diagnoses basic XNS network connectivity (user-level command).  |
| Router# <b>ping</b>                             | Diagnoses basic XNS network connectivity (privileged command).  |

# XNS Configuration Examples

To help in configuring XNS routing on your router, use the following configuration examples:

- [XNS Routing Configuration Example](#)
- [Net/One Routing Configuration Example](#)
- [3Com Access List Example](#)
- [Extended Access List with Network Mask Option Example](#)
- [Routing Update Timers Example](#)
- [Helper Example](#)

## XNS Routing Configuration Example

The following example shows how to enable XNS routing on a 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
```

## Net/One Routing Configuration Example

The following example shows how to enable 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
```

## 3Com Access List Example

The following partial example shows how to control specific services between networks 1002 and 1006 in a 3Com network. Echo and error packets, along with 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 show how to allow 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
```

## Routing Update Timers Example

The following example shows how to create 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
!
```

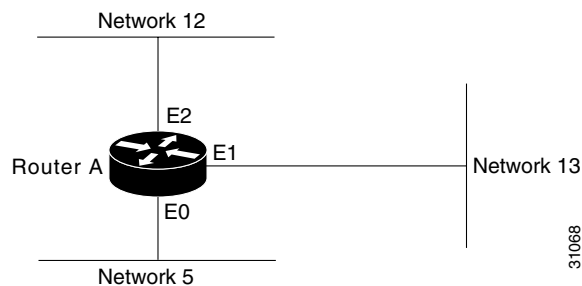
```
interface ethernet 1
  xns network 21
  xns update-time 20
```

## Helper Example

The following example shows how to set up helping for the configuration shown in [Figure 30](#). Cisco IOS software 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 30** 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 broadcast has nothing to do with helping or protocol forwarding.

