Chapter 1
Routing XNS

1

This chapter describes how to configure your router to perform routing for packets using the Xerox Network System (XNS) protocols.

You will find information about the following topics and tasks:

- How to configure a routing process for XNS routing. This includes information on the principal XNS protocols, XNS addressing, how to configure your router to route XNS traffic, managing security issues, and maximizing performance.
- How to configure helper addresses and flooding options for broadcast traffic.
- How to configure options for access lists and filters.
- Configuration restrictions and requirements for encapsulation of all important lower layer protocols, including Token Ring, FDDI, Ethernet, and others.

This chapter also contains configuration information on Ungermann-Bass' and 3Com's XNS protocols.

The section "XNS Configuration Examples," later in this chapter, includes examples of actual configurations for working XNS networks, including Ungermann-Bass and 3Com.

Additionally, EXEC-level commands for monitoring and debugging the XNS network are described. These commands are found in the last sections of the chapter, along with concise summaries of the global and interface-specific configuration commands.

Cisco's Implementation of XNS

Cisco provides a subset of the XNS protocol stack to support XNS routing on its routers. The same Cisco routers that route XNS can also route another protocol stack like TCP/IP or DECnet. At the physical and data-link layers, XNS traffic can be routed over Ethernets, FDDI, Token Rings, or over point-to-point serial lines running HDLC or LAPB, or (with optional software) over X.25, Frame Relay, or SMDS networks.

Ethernet and Token Ring

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 is also used as the node address for non-LAN media, notably serial links.

XNS was originally designed to run over Ethernet. If you implement an XNS stack using Token Ring at the first two layers, you must use a different encapsulation than you would on Ethernet. Encapsulation defines the kind of envelope (layer 1 and layer 2 bits) in which three through seven of the XNS protocol and data packet are wrapped prior to being transmitted. Because there is no agreed-upon way of encapsulating XNS packets on a Token Ring network, many vendors have invented their own encapsulations. See the section "Configuring XNS over Token Ring," later in this chapter for more information.

XNS Addresses

Both data-link (MAC) and network-layer addressing are needed in any network that supports routing; XNS is no exception. An XNS network layer address is composed of three fields:

- The network number uniquely identifies a network in an internet.
- The host number uniquely identifies a host on a network.
- The socket number uniquely identifies a socket within the operating system of the host. A socket is a transport address that is the source or destination of packets.

Network and Host Numbers

The network number is expressed in decimal format in Cisco configuration files and routing tables. When configuring a Cisco router, enter the network number in decimal.

Addresses must be unique throughout an XNS internet. Because both the network number and the host address are needed to deliver traffic to a host, addresses are usually given as network numbers, followed by host addresses, separated with dots. An example address follows.

Example:

```
47.0000.0c00.23fe
```

Here, the network number is 47 (decimal), and the host address is 0000.0c00.23fe (hex). A network number of 0 is used when the network number is unknown; this is the local network number.

Socket Numbers

An XNS socket number is a 16-bit field in the IDP header. Sockets are selected by the client processes of each host before a connection is established. Certain socket numbers are considered to be well-known sockets (WKS), meaning that the service performed by the software using them is statically defined. Each system element supplying a specific well-known service does so at the same WKS. Socket numbers above the well-known range may be selected and reused at random.

Configuring XNS

Follow these steps to configure XNS routing:

Step 1: Enable XNS routing using the **xns routing** command.

Step 2: Assign a unique XNS network number to each interface using the **xns network** command.

Step 3: Optionally configure performance parameters and helper addresses (which help manage broadcast traffic).

Step 4: Optionally configure access lists and filters.

There are specific configurations for the Ungermann-Bass Net/One networks and these are described in a later section.

Enabling XNS Routing

The first step in the configuration process is to specify XNS as the protocol you are enabling. Use the **xns routing** global configuration command to enable XNS routing. The full syntax of this command follows.

xns routing [*address*]
no xns routing

The optional argument *address* is used as the router's address on interfaces that have no native MAC-level addresses (principally serial lines). The **no xns routing** command disables all XNS processing.

If the argument *address* is omitted, the router uses the address of the first IEEE-compliant (Token Ring, FDDI, or Ethernet) interface hardware address it finds in its interface list. The address 0123.4567.abcd is used as a default for non-IEEE interfaces.

Next, use the **xns network** interface subcommand to assign a decimal XNS network number to each interface and enable each interface to run XNS protocols. The full syntax of the command follows:

xns network *number*
no xns network

The argument *number* is the network number, in decimal format. Interfaces not enabled to run XNS ignore any XNS packets that they receive. Every XNS interface in a system must have a unique XNS network number.

The **no** version of the command deassigns the XNS network number for the interface and disables XNS on that interface.

Example:

This example starts XNS routing, and assigns XNS network numbers to the physical networks connected to two of the router's Ethernet interfaces.

```
xns routing
interface ethernet 0
xns network 20
interface ethernet 1
xns net 21
```

Configuring Static Routing

To add a static route from your router to a remote destination in the XNS routing table, use the **xns route** global configuration command. The full syntax follows.

xns route *network host-address*
no xns route *network host-address*

The argument *network* is the destination XNS network number in decimal. The argument *host-address* is a decimal XNS network number and a hexadecimal host number, separated by a dot. The argument *host-address* must refer to a node directly connected to one of the Cisco router's physical networks.

Static routes are not usually used in XNS environments, as nearly all XNS routers support dynamic routing with RIP. Dynamic routing is enabled by default in Cisco routers.The **no** version of the command removes the static route.

Example:

The following example sets up a host (router) address of 21.0456.acd3.1243 as the static recipient of packets destined for network 25.

```
xns route 25 21.0456.acd3.1243
```

Managing Throughput

You have several options for managing throughput while dynamically routing. You can set up multiple paths and send packets over these paths in a round-robin fashion. You can also enable or disable fast switching. You can even adjust how often your router uses RIP to send routing table updates to its neighbors provided all the neighbors are Cisco routers.

Setting Multiple Paths

A Cisco router can use multiple paths to an XNS destination in order to increase throughput in the network. By default, the router will pick one best path and send all traffic on this path, but you can configure it to remember two or more paths that have equal costs (as computed by the routing protocol; hop count for standard XNS RIP) and balance the traffic load across all the available paths.

To set the maximum number of multiple paths used for any single destination, use the **xns maximum-paths** global configuration command. The full syntax follows.

xns maximum-paths *paths*
no xns maximum-paths

The argument *paths* is the number of paths to be assigned. The default value for *paths* is 1. Packets are distributed over the multiple paths in round-robin fashion on a packet-by-packet basis. To disable multiple paths, use the command with the default value of 1. The **no xns maximum-paths** command restores the default.

The EXEC command **show xns route** displays the entire routing table, so you can examine your additional routes and the maximum path cost for each. See the section "Monitoring an XNS Network" when you are ready to use this command.

Example:

The following example asks the router to send packets for each destination over two alternate paths, if available.

```
xns maximum-paths 2
```

Configuring XNS Fast Switching

XNS fast switching achieves higher throughput by using a cache created by previous transit packets. Fast switching for XNS provides load sharing on a per-packet basis. When you enable XNS routing, fast switching is automatically enabled as well. Use the **no xns route-cache** interface command to disable fast switching. The full syntax of the command follows.

xns route-cache
no xns route-cache

Use the **no xns route-cache** command to disable fast switching and the **xns route-cache** interface subcommand to re-enable fast switching.

Adjusting Timers

XNS RIP sends routing table updates to its neighbor routers every 30 seconds, unless you specify some other time interval. You can reset the routing update timers on a per-interface basis using the **xns update-time** interface subcommand:

xns update-time *seconds*

Note: This command does not affect the Ungermann-Bass routing protocol.

XNS routing timers are affected by the value set for the *seconds* argument:

- Routing updates are sent through the affected interface.
- XNS routes are marked invalid and placed in holddown if no routing updates for those routes are heard within three times the value of the update timer.
- XNS routes are removed from the routing table if no routing updates are heard within six times the value of the update timer.

There is not a **no** version of this command. If you want to return to the default condition, use the **xns update-time** command with a value for the *seconds* argument of 30. The minimum is ten seconds.

Note: Be careful with this command. Use it only in an all-Cisco environment. Make sure that all timers are the same for all routers attached to the same network segment.

The EXEC command **show xns route** displays the value of these timers. See the section "Monitoring an XNS Network" when you are ready to use this command.

Configuring XNS over Token Ring

There is no standard way of packaging an XNS packet for transit across an 802.5 Token Ring, as XNS was designed to exist above Ethernet at the lowest two network layers.The general process of wrapping physical and link-layer header information around a packet so that it can traverse a network is called *encapsulation*. Each Token Ring vendor has its own method of encapsulating XNS packets. Cisco supports the encapsulation methods of Ungermann-Bass, 3Com, and IBM.

Use the **xns encapsulation** interface subcommand to select the encapsulation method:

xns encapsulation *keyword*

If your Token Ring is an IBM installation, you use the default *keyword* **snap**. Other options are:

- **ub** for Ungermann-Bass
- **3com** for older 3Com Corporation products

Note: Some 3COM 3+ hosts do not recognize Token Ring packets with the source-route bridging RIF field set. You can work around this discrepancy by using the **no multiring xns** interface subcommand on Token Ring interfaces that are used for 3Com XNS routing. See the chapter "Configuring Source-Route Bridging" for more information.

Example:

In the following example, XNS is enabled, an interface is defined for Token Ring, and then Ungermann-Bass is specified as the encapsulation option, as shown:

```
xns routing
xns ub-emulation
interface tokenring 0
xns network 23
xns encapsulation ub
```

The **xns ub-emulation** command is used to enable use of the Ungermann-Bass XNS routing protocol as an alternative to standard XNS RIP, and is included in this example for consistency. You will find more information on the Ungermann-Bass version of XNS in the following section.

Configuring Ungermann-Bass Net/One XNS

This section describes the protocols and configurations specific to Ungermann-Bass' Net/One networks. Net/One end nodes communicate using the XNS protocol, but there are a number of differences between Net/One's usage of the protocol and the usage common among other XNS nodes. This section discusses the Net/One implementation and how to configure routers to work in Net/One networks.

Ungermann-Bass Net/One on a Cisco Router

Ungermann-Bass's Net/One products use XNS at the network layer. However, Net/One as a whole is not equivalent to standard XNS. When you use Cisco routers in Net/One environments, you must consider a few factors. The following are the important differences between the Net/One environment and more common XNS usage:

- Net/One routers use an Ungermann-Bass proprietary routing protocol instead of standard XNS RIP. Although they generate both Ungermann-Bass update packets and standard RIP update packets, Net/One routers only listen to Ungermann-Bass updates. Cisco supports both reception and generation of Ungermann-Bass routing packets, and 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 in sending packets to destinations that are not on the local cable. Ordinary XNS end hosts use RIP for this purpose. Cisco routers can be configured to generate Ungermann-Bass Hello packets.
- Net/One equipment uses a non-XNS protocol for network software downloads. 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. 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 configured specially. Contact Ungermann-Bass to find out how to configure your Ungermann-Bass Network Management Console.
- The Ungermann-Bass 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 broadcast (network -1). These packets are sent as MAC-layer broadcasts, and are expected to be flooded throughout the XNS internet. Flooding of these packets can be enabled on Cisco routers with the **xns flood specific allnets** interface configuration subcommand, which is described in the section "XNS Broadcasts, Flooding, and Helpering."
- Net/One equipment uses a set of proprietary network management protocols. Cisco routers do not participate in these protocols.

Configuring Ungermann-Bass Net/One Routing

To enable Ungermann-Bass Net/One routing, use the **xns ub-emulation** global configuration command. The full syntax of this command follows:

xns ub-emulation
no xns ub-emulation

This command causes Hello packets and routing updates in Ungermann-Bass format to be sent out through all the interfaces on which XNS is enabled. It also causes the Cisco router to get its routing information for remote networks from Ungermann-Bass updates. Unless the **xns hear-rip** command is issued, the Cisco router will completely ignore standard RIP updates, and will behave as would an Ungermann-Bass router.

The **no** version of the command restores the router to standard XNS routing mode.

If you need your Cisco router to receive RIP updates on some interfaces, you can configure it to do so using the **xns hear-rip** interface configuration subcommand:

xns hear-rip [*access-list*]
no xns hear-rip

The **xns hear-rip** command causes the Cisco router to treat standard RIP updates incoming on the interface as if they were Ungermann-Bass updates, with delay metrics computed as though each hop mentioned by the RIP update were a 9.6 Kbps serial link. The result is ordinarily that the Cisco router will prefer an all-Ungermann-Bass path to an all-RIP path. If you only want certain routes to be learned through standard RIP, you can specify an XNS access list as an argument to the **xns hear-rip** command. The Cisco router will learn from RIP packets only routes to networks permitted by the access list. Note that the access list is applied to individual routes within the RIP packet, not to the address of the packet's sender. The **no** version of the command causes the interface not to receive RIP updates.

In an Ungermann-Bass environment, all interfaces should be configured with the **xns flood broadcast allnets** and **xns flood specific allnets** interface configuration subcommands. Interfaces should *not* be configured with **xns flood broadcast net-zero**. Token Ring interfaces whose attached rings are directly connected to Ungermann-Bass nodes should be configured with **xns encapsulation ub**.

You can set up a standard Ungermann-Bass configuration using the global configuration command **xns ub-routing**, which has been retained mostly for backward compatibility with older releases. The 9.0 Software Release changes the 8.3 behavior of this command. Issuing **xns ub-routing** is exactly equivalent to issuing **xns ub-emulation** for the system as a whole, as well as **xns hear-rip**, **xns flood broadcast allnets**, **no xns flood broadcast net-zero**, and **xns flood specific allnets** for all interfaces on which XNS is enabled. It does not modify the encapsulation used on Token Ring interfaces. The full syntax of this command follows:

xns ub-routing
no xns ub-routing

Note: The **xns ub-routing** command is never written to nonvolatile configuration memory; the equivalent individual commands are written instead.

The commands **xns flood broadcast allnets** and **xns flood specific allnets** are described in the section "XNS Broadcasts, Flooding, and Helpering."

An Ungermann-Bass configuration example is found in the "XNS Configuration Examples" section of this chapter.

Ungermann-Bass Routing Protocol and Interactions with RIP

The routing protocol used by Net/One routers is a distance-vector or Bellman-Ford protocol, similar to standard XNS RIP. The major difference between the two protocols lies in the metrics used. While standard RIP uses a hop count to determine the best route to a distant network, the Ungermann-Bass protocol uses a path-delay metric. The standard RIP protocol maintains information only about hop counts, while the Ungermann-Bass protocol maintains information both about hop counts and about 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) as Ungermann-Bass routers would.

Cisco routers may also be configured to listen to standard RIP updates when in Ungermann-Bass emulation mode, using the **xns hear-rip** interface configuration subcommand. 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: a 9.6 Kbps serial link. Information from RIP packets is used in creating outgoing Ungermann-Bass updates, and vice versa.

Older Cisco software implemented a restricted version of the Ungermann-Bass routing protocol; using that software in certain configurations could create routing instability and forwarding loops. Users planning to use Software Releases 8.3 and earlier in Ungermann-Bass environments should consult their 8.3 documentation for information on the restrictions.

Handling XNS Broadcasts, Flooding, and Helpering

This section explains how XNS handles broadcasts, and how flooding and helpering help you manage broadcast traffic.

Overview of XNS Broadcasts

Network end nodes often use broadcast packets for service discovery; a request is broadcast to many or all nodes in the internet, and one or more of the nodes that can offer the needed service reply to the broadcast. Both end nodes and routers sometimes use broadcast packets to contain data that must be received by many other nodes; an example would be a RIP routing update.

Although broadcasts 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. If the broadcast is propagated to more than one physical network, extra load is presented on all the networks involved, and on all the intervening routers.

The following describes 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 by a specific network number in the packet's destination field. If a node does not know the number of the local XNS network (common at bootup time), it may 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 by an all-ones destination network field (typically written as -1 or as FFFFFFFF).
- A *directed* broadcast is one that is intended for all nodes on a specific physical network other than the network on which the packet originates. Directed broadcasts are denoted by the use of a specific network number, other than that of the network on which the packet originates, in the destination field.

All these broadcast types use the node address FFFF.FFFF.FFFF in the packet's destination node field. The destination MAC address used in the underlying LAN frame is the broadcast address. Directed broadcasts intended for remote networks may 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.

Flooding and Helpering

Some implementations expect all broadcasts to be treated as local broadcasts. Others expect broadcasts with zero destination network fields to be treated as all-nets broadcasts. Some do not support directed broadcasts. In addition, some implementations expect packets with all-ones destination network fields, 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 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, its destination address is changed by the router to be the configured helper address, and it 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 may 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 hop-count bookkeeping fields. Flooding is useful in cases where many nodes throughout the internet need to receive a packet, or where a service that may be anywhere in the internet must be discovered. Avoid flooding in large, slow, or heavily loaded networks, as the load presented to routers, links, and end nodes by heavy flooded traffic is large.

Configure XNS helpering using the **xns helper-address** interface configuration command and the **xns forward-protocol** global configuration command.

> **xns helper-address** *host-address*
> **no xns helper-address** *host-address*

The argument *host-address* is a dotted combination of the network and host addresses as explained in the **xns route** command. Broadcasts received on this interface are considered candidates for helpering as described above. The **no** version of the command disables XNS helpering.

The **xns forward-protocol** global configuration command allows you to specify which XNS protocols will be considered for helpering when received in broadcast packets. The full syntax of the command is:

> **xns forward-protocol** *type*
> **no xns forward-protocol** *type*

The argument *type* is a decimal number corresponding to an appropriate XNS protocol. Only packets with types listed in **xns forward-protocol** commands are considered candidates for helpering. See the documentation accompanying your host XNS implementation to determine the protocol type number.

Use the **no xns forward-protocol** command and the appropriate argument to disable the forwarding of the specified protocol.

Whenever a Cisco router receives an XNS broadcast packet, it processes the packet as follows:

- If the packet is a routing update, or requests services that are offered by the Cisco router itself, the packet is processed by the Cisco 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 may be a directed broadcast address.
- If the packet is neither locally serviceable nor a candidate for helpering, it is considered for flooding. Three classes of packets may be flooded:
    - — Packets with destinations of -1.FFFF.FFFF.FFFF (enabled with **xns flood broadcast allnets**)
    - — Packets with destinations of 0.FFFF.FFFF.FFFF (enabled with **xns flood broadcast net-zero**)
    - — Packets with destinations of -1.*specific-host* (enabled with **xns flood specific allnets**)

The XNS flooding interface configuration commands are:

xns flood broadcast allnets
no xns flood broadcast allnets

xns flood broadcast net-zero
no xns flood broadcast net-zero

xns flood specific allnets
no xns flood specific allnets

All of the above are interface configuration subcommands and apply to flooding of packets *received* on the interfaces for which they are specified.

Different XNS host implementations require different flooding and helpering behavior. By default, Cisco routers do no flooding at all. It is most closely in accordance with the XNS specification to set **xns flood broadcast allnets** and **xns flood specific allnets**, but not to set **xns flood broadcast net-zero**. 3Com environments often require flooding of net-zero broadcasts.

Cisco chooses the interfaces through which flooded packets are sent according to 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 the Cisco router is configured to flood are sent out through all interfaces, except that:

■ 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, only packets received on the primary path (the first path the Cisco router learned) are flooded. If a packet is received on an interface which fails this rule, the interface that passes it will receive another copy of that packet.

■ Packets are never flooded *out* of the Cisco router through any interface that is one of the Cisco 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 Cisco 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. Users can use access lists to get control over flooding behavior.

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.

For backward compatibility, any attempt to set a helper address of -1.FFFF.FFFF.FFFF on an interface will result in that interface having no helper address set, but having **xns flood broadcast allnets** enabled.

Both helpering and flooding increment packets' hop-count fields.

Notes and Tips on Configuring XNS Helpering

You need to decide how you want the router to handle different kinds of broadcast packets. This is an excellent opportunity for you to reduce unnecessary network traffic, if you think about all your options carefully.

Many broadcasts occur when a node first becomes active on the network. A host will generate a broadcast packet when it does not know the current address of whatever host is supposed to receive its next packet—the local server, for instance. It is generally 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 internet with unnecessary traffic.

You can configure your routers to block all broadcasts. You have more than simply this all-or-nothing choice, however, by using the **xns helper-address**, **xns forward-protocol**, and **xns flood** commands.

Ignore the protocol-filtering issue for a moment and just consider some examples of how helper addresses are used. In Figure 1-1, the E0 interface has a helper address set, with the helper on network 12, available through the E2 interface. Some broadcast packets are being received on this E0 interface:

Figure 1-1 Helper Addresses

- A broadcast packet with a network address of 5 will be forwarded to the helper address on network 12.
- A broadcast packet addressed to network 0 will also be forwarded to the helper address on network 12.
- A broadcast packet addressed to network 13 is a directed broadcast, and will be sent through the E1 interface directly to network 13. It will not be sent to the helper address.

Forwarding Specific Protocols

In Figure 1-1, the E0 interface is set to a helper address and forwarding for protocol 1 only. (The protocol numbers will vary depending on your XNS host implementation.) Broadcast packets are *arriving* on the E0 interface from network 5:

- A broadcast packet destined for network 5 and protocol 1 will be sent to the helper address.
- A broadcast packet destined for network 5 and another protocol is discarded because it fails the protocol test.
- A broadcast packet destined for network 0 and protocol 1 is sent to the helper address.
- A broadcast packet destined for network 0 and a protocol other than 1 is discarded.

A broadcast packet destined to network 12 is a directed broadcast, and is sent out, still in broadcast form, on the E2 interface to network 12. This has nothing to do with the helper-address or protocol forwarding.

Example:

In this example, a helper address is specified and protocol type 2 is forwarded to that address.

```
xns forward-protocol 2
interface ethernet 0
xns helper-address  26.FFFF.FFFF.FFFF
```

Configuring XNS Access Lists and Filters

You may configure XNS access lists to filter traffic on XNS interfaces.

An access list is defined by a series of commands. Each access list entry contains only one address parameter and the list must be within the range of 400 to 499. How this address is interpreted is defined by the command that will use the list.

XNS access lists filter on the source and destination addresses only. This means that they can prevent traffic from going to specific hosts or coming from specific hosts. Extended XNS access lists are numbered from 500 to 599 and filter on XNS protocol and socket fields. The extended filters can prevent entire classes of packets (SPP, Echo, and so on) from passing a router interface and they can also filter traffic going to or coming from specific processes.

As with all other Cisco access lists, an implicit *deny everything* is defined at the end of the list. If this is not desired, an explicit *permit everything* definition must be included at the end of the list. Complex configuration examples are shown in the “XNS Configuration Examples” section.

Configuring XNS Access Lists

To configure an XNS access list, use the **access-list** global configuration command with the following syntax:

> **access-list** *number* {**deny**|**permit**} *XNS-source-network*.[*source-address*[*source-mask*]]*XNS-destination-network*.[*destination-address*[*destination-mask*]]
> **no access-list** *number*

Note: For typographic reasons, access list command syntax in this section is shown on multiple lines; an access list command must be on a single line when given as a configuration command.

The argument *number* must be a number between 400 and 499. The **no** version of the command removes the entire access list.

The only required parameter for standard XNS access lists is the XNS source network address. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered. Note that XNS uses MAC addresses as the node ID; see the examples for further clarification.

Example 1:

This example denies access from source network –1 (all XNS networks) to destination network 2. All other packets are permitted.

```
access-list 400 deny -1 2
access-list 400 permit -1
```

Example 2:

This example denies access from XNS source address 21.0000.0c00.1111. The destination network does not make any difference. All other packets are permitted.

```
access-list 400 deny 21.0000.0c00.1111
access-list 400 permit -1
```

Example 3:

This example denies access from all hosts on network 1 that have a source address beginning with 0000.0c. All other packets are permitted.

```
access-list 400 deny 1.0000.0c00.0000 0000.00ff.ffff
access-list 400 permit -1
```

Example 4:

In the following example, access is denied from source address 0011.1622.0015 on network 21 to destination address 01D3.020C.0022 on network 31. All other packets are permitted.

```
access-list 500 deny 1 21.0011.1622.0015 0000.0000.0000 31.01D3.020C.0022
0000.0000.0000
access-list 400 permit -1
```

Configuring Extended Access Lists

For extended access lists, the **access-list** command again must be typed on one line using this syntax:

access-list *number* {**deny**|**permit**} *xns-protocol source-network*.[*source-address* [*source-mask*]] *source-socket destination-network*.[*destination-address* [*destination-mask*]] *destination-socket*
no access-list *number*

The argument *number* must be a number between 500 and 599. The **no** version of the command removes the entire access list.

The source and destination addresses and masks are optional. The protocol number *xns-protocol* is the only required parameter. A network number of –1 matches all networks; a socket number of 0 matches all sockets. The **no** version of the command removes the entire access list.

Example 1:

This example denies access to packets with protocol 5 from source network 1, source socket 1234 that are trying to be routed to destination network 2, destination socket 1234:

```
access-list 500 deny 5 1 1234 2 1234
```

Example 2:

An example of applying a mask for both the source and destination networks is as follows:

```
access-list 500 permit 1 21.0011.1622.0015 0000.0000.0000 1234 31.01D3.020C.0022
0000.00ff.ffff 1234
```

Filtering Outgoing Packets

An XNS access list group number is assigned to an interface with the **xns access-group** interface subcommand. The full syntax of this command follows.

xns access-group *number*
no xns access-group *number*

The argument *number* specifies the access list number defined by the XNS global **access list** command. This command causes all outgoing XNS packets that would ordinarily have been forwarded by the router through this interface to be compared to the access list. If the packet fails the access list, it is not transmitted, but is discarded instead.

Example:

This example uses the **xns access-list** command to deny access to protocol 1 from source network 1, source socket 1234 to destination network 2, destination socket 1234, then assigns number 500 to a group to be filtered:

```
access-list 500 deny 1 1 1234 2 1234
access-list 500 permit -1
!
interface ethernet 0
xns access-group 500
```

Filtering XNS Routing Updates

This section describes the filtering commands that use access lists to control what routing information is accepted, or passed on, within XNS networks. The commands filter incoming traffic and outgoing routing information, and specific routers.

Input Filters: Adding to the Routing Table

To control which networks are added to your router's routing table, use the **xns input-network-filter** interface subcommand.

xns input-network-filter *access-list-number*
no xns input-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In this example, access list 476 controls which networks are added to the routing table when RIP packets are received. The address in the access list is the address of the network that you want to be able to receive routing updates about.

```
access-list 476 permit 16
interface ethernet 1
xns input-network-filter 476
```

This set of configuration commands ensures that network 16 is the only network whose information will be added to the routing table from Ethernet 1.

Output Filters: Controlling the List of Networks

To control the list of networks that are sent out in routing updates by your router, use this interface subcommand:

xns output-network-filter *access-list-number*
no xns output-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In the following example, access list 496 controls which networks are sent out in routing updates. The second line specifies interface serial 1, and the third line causes network 27 to be the only network advertised in routing update packets. Information about other networks will not be advertised in routing updates (for the specified interface only; other interfaces may advertise the full routing table).

```
access-list 496 permit 27
interface serial 1
xns output-network-filter 496
```

Router Filters

To control the list of routers from which data will be accepted, use the **xns router-filter** interface subcommand:

xns router-filter *access-list-number*
no xns router-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS **access-list** command.

Example:

In this example, access list 466 defines the only router that data will be accepted from. In this case, the address parameter is the address of a router.

```
access-list 466 permit 26.0000.000c0.047d
interface serial 0
xns router-filter 466
```

Information from a disallowed router is ignored.

XNS Configuration Examples

This section includes examples of common configurations, designed to help you put all the specific command information into a complex, real-world configuration file. Included are basic configuration examples, examples of flooding, and both simple and complex access lists. Some of the examples refer to 3Com or Ungermann-Bass XNS, rather than standard XNS; all examples are clearly marked.

Creating a Routing Process

The following example establishes XNS routing on a Cisco router (creating a routing process), then three interfaces are named and given their individual network numbers. The router will use the preassigned MAC-level addresses as XNS node addresses on the Ethernet interfaces. The serial interface will use the MAC address associated with the first IEEE 802.x interface found on the router.

Example:

```
xns routing
!
interface ethernet 0
xns network 20
!
interface ethernet 1
xns network 21
!
interface serial 1
xns network 24
```

Setting Timers

This example creates a routing process specifying a specific address for use on serial lines and other non-802.x interfaces. Next, specify the interfaces and assign them network numbers. The RIP routing update timers for the serial and the Ethernet interfaces have also been changed.

Example:

```
xns routing 0000.0C53.4679
!
interface ethernet 0
xns network 20
xns update-time 20
!
interface serial 0
xns network 24
xns update-time 40
!
interface ethernet 1
xns network 21
xns update-time 25
```

Configuring for Multiprotocol Routing

This example illustrates one way of enabling XNS and another protocol. Do a pencil copy of your proposed configuration commands and check them against the other protocol chapters in this manual before you proceed with configuring more than one protocol in a session.

Example:

```
xns routing
novell routing
!
interface ethernet 0
xns network 20
novell network 4e
!
interface ethernet 1
xns network 21
novell network 6bb
!
interface serial 0
xns network 24
novell network 4ad
```

Configuring for Ungermann-Bass Routing

In this example, basic Ungermann-Bass Net/One routing is enabled by:

Step 1: Enabling XNS routing

Step 2: Specifying Ungermann-Bass routing

Step 3: Defining the interfaces and networks

Interface serial 0 is connected to a non-Ungermann-Bass part of the XNS internet, so the **xns hear-rip** command is used 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 Ungermann-Bass software.

Example:

```
xns routing
xns ub-emulation
!
interface token 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

This partial example permits and denies specific services between networks 1002 and 1006 in a 3Com network. Echo and error packets can go from 1002 to 1006, as well as all SPP and PEP (normal data traffic). However, all NetBIOS requests are denied. The final three lines are blanket permissions for RIP, SPP, and PEP, because access lists assume you wish to deny anything you do not specifically permit. These blanket permissions must come at the end of the list, as shown in the example configuration.

Example:

```
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
```

Monitoring an XNS Network

Use the EXEC commands described in this section to obtain displays of activity on your XNS network.

Displaying Cache Entries

Use the **show xns cache** command to display a list of fast-switching cache entries. Enter this command at the EXEC prompt:

show xns cache

Displaying Interface Parameters

Use the **show xns interface** command to display interface-specific XNS parameters. Enter this command at the EXEC prompt:

show xns interface [*name*]

The optional argument *name* may be used to request a display a particular interface.

The following sample output shows a display of all interface statistics.

```
Ethernet 0 is up, line protocol is up
XNS address is 60.0000.0c00.1d23
xns encapsulation is ARPA
Helper address is 912.ffff.ffff.ffff
Outgoing address list is not set
Input filter list is not set
Output filter list is not set
Router filter list is not set
Update timer is not set
XNS fast-switching enabled

Ethernet 1 is administratively down, line protocol is down
XNS protocol processing disabled

Serial 1 is up, line protocol is up
XNS protocol processing disabled
```

In this display, the Ethernet 0 interface has a helper address set but all other parameters are set to the defaults. The Update Timer refers to itself as not set when it is set to default value. The Ethernet 1 interface is down and XNS processing is disabled. The serial 1 interface is up but it is not processing XNS packets.

For the Ethernet 1 and serial 1 interfaces, the **no xns network** command is in force. You can enable XNS processing by using the **xns network** configuration command.

Displaying the Routing Table

Use the EXEC command **show xns route** to display the entire XNS routing table. Enter this command at the EXEC prompt:

show xns route *network-number*

The optional network number argument specifies a particular network.

Following is sample output:

```
Codes: R - RIP derived, C - connected, S - static, 1 learned routes

Maximum allowed path(s) are/is 1
C Net 14 is directly connected, 0 uses, Ethernet0
C Net 15 is directly connected, 0 uses, Ethernet1
R Net 16 [1/0] via 14.0000.0c00.3e3b,  10 sec, 0 uses, Ethernet0
```

In this display, RIP-derived routes are indicated by the letter R. Networks that are reachable are listed in numerical order within each category—RIP or connected, in this case.

The routing table also tells you the address of the router that constitutes the first hop in the route (always the same router in this case), the round-trip delay encountered on the route, and the interface through which the route is available.

Displaying Traffic Statistics

Use the EXEC command **show xns traffic** to display packet statistics, including packets sent, received, and forwarded. Enter this command at the EXEC prompt:

show xns traffic

Sample output follows. Table 1-1 describes the fields displayed.

```
Rec: 3968 total, 0 format errors, 0 checksum errors, 0 bad hop count,
3968 local destination,  0 multicast
Bcast: 2912 received, 925 sent
Sent:  5923 generated, 500 forwarded, 0 encapsulation failed, 0 not
routable
Errors:  10 received, 20 sent
Echo:  Recd:  100 requests, 89 replies   Sent:  20 requests, 20 replies
Unknown:  5 packets
```

Table 1-1 XNS Traffic Statistics Field Descriptions

Field	Description
Rec:	The total number of XNS packets received by the router.
format errors	Number of packets received with format errors in the header; they were discarded.
checksum errors	Number of packets received and discarded because of check-sum errors.
bad hop count	Number of packets discarded because their hop count fields were equal to or greater than 16.
local destination	Number of packets received on the interface that were broadcast or were destined for internal processing by the router, including helpering or flooding.
multicast	Number of packets received with multicast addresses.
Bcast:	Number of broadcast packets received and sent.
Sent:	
generated	Number of packets created by the router itself.
forwarded	Number of packets received from other nodes and for-warded.
encapsulation failed	Number of packets discarded because encapsulation could not be accomplished.
not routable	Number of packets discarded because the router had no path to their destination networks.
Errors:	Number of Error packets sent and received.

Field	Description
Echo:	Number of Echo packets received and sent, specifying how many replies were received.
Unknown:	Packets destined for local processing, but which the router could neither handle, helper, nor flood.

Debugging an XNS Network

Use the EXEC commands described in this section to troubleshoot and monitor your XNS network. For each **debug** command listed, there is a corresponding **undebug** command that turns off the message logging.

The command **debug xns-packet** enables logging of XNS packet traffic, including the addresses for source, destination, and next hop router of each packet.

debug xns-packet

The command **debug xns-routing** displays XNS routing transactions.

debug xns-routing

XNS Global Configuration Command Summary

Following is an alphabetically arranged list of the XNS global configuration commands.

[**no**] **access-list** *number* {**deny**|**permit**} *XNS-protocol source-network*.[*source-address* [*source-mask*]] *source-socket destination-network*.[*destination-address* [*destination-mask*]] *destination-socket*
no access-list *number*

Configures an entry in an extended XNS access list. The argument *number* is an access list number in the range 500 and 599. The keyword **permit** or **deny** specifies the filtering action. The argument *XNS-protocol* is the XNS protocol number. The **no access-list** *number* command removes the specified access list.

[**no**] **access-list** *number* {**deny**|**permit**} *XNS-source-network*.[*source-address*[*source-mask*]]*XNS-destination-network*.[*destination-address*[*destination-mask*]]
no access-list *number*

Configures an entry in an XNS access list. The argument *number* is an access list number in the range 400 and 499. The keyword **permit** or **deny** specifies the filtering action. The arguments *XNS-source-network* and *XNS-destination-network* are the only required addresses. The source and destination masks are optional. The **no access-list** *number* command removes the specified access list.

[**no**] **xns forward-protocol** *type*

Determines which protocol types will be forwarded when a broadcast is received on an interface that has an XNS helper address set. The argument *type* is a decimal number corresponding to an appropriate XNS protocol.Use the **no xns forward-protocol** command and the appropriate argument to disable the forwarding of the specified protocol.

[**no**] **xns maximum-paths** *paths*

Sets the maximum number of equal-cost paths the router will use. The default value of *paths* is one. The **no** version of the command restores the default.

[**no**] **xns route** *network host-address*

Adds a static route to the XNS routing table. The argument *network* is the destination XNS network number in decimal. The argument *host-address* is a decimal XNS network number and the hexadecimal host number.The **no** version of the command removes the static route.

[**no**] **xns routing** [*address*]

Enables XNS routing. The optional argument *address* is the XNS address the router is to use. If the argument *address* is omitted, the first Token Ring, FDDI, or Ethernet interface hardware address found is used. The **no** version of the command disables all XNS packet processing.

[**no**] **xns ub-emulation**

Enables Ungermann-Bass Net/One routing. Causes Hello packets and routing updates in Ungermann-Bass format to be sent out through all the interfaces on which XNS is enabled. Also causes the Cisco router to get its routing information for remote networks from Ungermann-Bass updates. Unless the **xns hear-rip** command is issued, the Cisco router will completely ignore standard RIP updates, and will behave as would a UB router.The **no** version of the command restores the router to standard XNS mode.

[**no**] **xns ub-routing**

Exactly equivalent to issuing **xns ub-emulation** for the system as a whole, as well as **xns hear-rip**, **xns flood broadcast allnets**, and **xns flood specific allnets** for all interfaces on which XNS is enabled.

XNS Interface Subcommand Summary

Following is an alphabetically arranged list of the XNS interface subcommands. These commands follow an **interface** command.

[**no**] **xns access-group** *number*

Assigns an access list number to an interface, and causes all XNS packets that would ordinarily have been forwarded by the router through this interface to be compared to the access list. If a packet fails the access list, it is not transmitted, but is discarded instead. The argument *number* specifies the access list number.The **no** version of the command disables this feature.

xns encapsulation *keyword*

Selects encapsulation for a Token Ring interface. Choices for *keyword* are: **snap**, **ub**, and **3com**.

[**no**] **xns flood broadcast allnets**

Configures an interface on a router such that any packets received with this destination address will be flooded with packets of destinations -1.FFFF.FFFF.FFFF. The **no** version of the command disables this feature.

[**no**] **xns flood broadcast net-zero**

Configures an interface on a router such that any packets received with this destination address will be flooded with packets of destinations 0.FFFF.FFFF.FFFF. The **no** version of the command disables this feature.

[**no**] **xns flood specific allnets**

Configures an interface on a router such that any packets received with this destination address will be flooded with packets of destinations -1. *specific-host*.The **no** version of the command disables this feature.

[**no**] **xns hear-rip** [*access-list*]

Causes the Cisco router in Ungermann-Bass mode to receive RIP updates on some interfaces as if they were Ungermann-Bass updates. The argument *access-list* specifies an XNS access list when you only want certain routes to be learned through standard RIP. The Cisco router will learn from RIP packets only routes to networks permitted by the access list. The **no** version of the command disables this feature.

[**no**] **xns helper-address** *host-address*

Sets a helper address to forward broadcasts. The argument *host-address* is a dotted combination of the network and host addresses. The **no** version of the command disables XNS helpering.

[**no**] **xns input-network-filter** *access-list-number*

Controls which networks are added to your router's routing table. The argument *access-list-number* is the access list number specified in the XNS **access-list** command.The **no** version of the command disables this feature.

[**no**] **xns network** *number*

Assigns a decimal XNS network number to an interface. The **no** version of the command deassigns the XNS network number for the interface and disables XNS on that interface.

[**no**] **xns output-network-filter** *access-list-number*

Controls the list of networks that are sent out in routing updates by your router. The argument *access-list-number* is the access list number specified in the XNS **access-list** command.The **no** version of the command disables this feature.

[**no**] **xns route-cache**

Enables fast switching for XNS packets outgoing on this interface. The **no** version of the command disables fast switching.

[**no**] **xns router-filter** *access-list-number*

Controls the list of routers from which data will be accepted. The argument *access-list-number* is the access list number specified in the XNS **access-list** command.The **no** version of the command disables this feature.

xns update-time *seconds*

Sets the XNS routing update timers to the value assigned to the *seconds* argument. The default is 30 seconds.