CHAPTER 18

Configuring Novell IPX

Novell Internet Packet Exchange (IPX) is derived from the Xerox Network Systems (XNS) Internet Datagram Protocol (IDP). IPX and XNS have the following differences:

- IPX and XNS do not always use the same Ethernet encapsulation format.
- IPX uses Novell's proprietary Service Advertisement Protocol (SAP) to advertise special network services. File servers and print servers are examples of services that are typically advertised.
- IPX uses ticks while XNS uses hop count as the primary metric in determining the best path to a destination.

This chapter describes how to configure Novell IPX and provides configuration examples. For a complete description of the commands mentioned in this chapter, refer to the *Communication Server Command Reference* publication. For historical background and a technical overview of Novell IPX, see the *Internetworking Technology Overview* publication.

Cisco's Implementation of Novell IPX

Cisco's implementation of Novell's IPX protocol provides all the functionality of a Novell communication server. A Cisco communication server connects Ethernet, Token Ring, and FDDI networks, either directly or through high-speed serial lines (56 kbps to T1 speeds), X.25, or Frame Relay. At this time, the Cisco X.25 and T1 support is not compatible with Novell. This means that Cisco communication servers must be used on both ends of T1 and X.25 circuits.

IPX Addresses

An IPX network address consists of a network number and a node number expressed in the format *network.node*.

The network number identifies a physical network. It is a four-byte (32-bit) quantity that must be unique throughout the entire IPX internetwork. The network number is expressed as eight hexadecimal digits. Our communication server software does not require that you enter all eight digits: you can omit leading zeros.

The node number identifies a node on the network. It is a 48-bit quantity, represented by dotted triplets of four-digit hexadecimal numbers.

The following is an example of an IPX network address:

```
4a.0000.0c00.23fe
```

Here, the network number is 4a (more specifically, it is 0000004a), and the node number is 0000.0c00.23fe. All digits in the address are hexadecimal.

IPX Configuration Task List

To configure IPX routing, complete the following tasks. At a minimum, you must enable IPX routing. The remaining tasks are optional.

- Enable IPX Routing
- Control Access to IPX Networks
- Tune IPX Network Performance
- Configure IPX over WANs

This chapter describes how to perform these configuration tasks.

Once you have configured routing, you can monitor and maintain the network. The tasks for doing this also are described in this chapter.

Enable IPX Routing

To enable IPX routing, you must perform the following two tasks:

- Enable IPX routing on the communication server
- Assign network numbers to individual interfaces

These tasks are described in the following sections.

Enable IPX Routing on the Communication Server

The first step in enabling IPX routing is to enable it on the communication server. To do so, perform the following global configuration task:

Task	Command
Enable IPX routing on the communication server.	**ipx routing** [*node*]

For an example of how to enable IPX routing, see the section "Enabling IPX Routing Example" later in this chapter.

Assign Network Numbers to Individual Interfaces

After you have enabled IPX routing on the communication server, you assign network numbers to individual interfaces. This has the effect of enabling IPX routing on those interfaces. When you enable IPX routing on an interface, you also can specify an encapsulation (frame type) to use for packets being transmitted on that network.

A single interface can support a single network or multiple logical networks. For a single network, you can configure any encapsulation type. Of course, it should match the encapsulation type of the servers and clients using that network number.

When assigning network numbers to an interface that supports multiple networks, you must specify a different encapsulation type for each network. Because multiple networks share the physical medium, this allows the communication server to determine which packets belong to which network. For example, you can configure up to four IPX networks on a single Ethernet cable because four encapsulation types are supported for ethernet. Again, the encapsulation type should match the servers and clients using the same network number.

The following sections describe how to enable IPX routing on interfaces that support a single network and those that support multiple networks.

Assign Network Numbers to Interfaces that Support a Single Network

To assign a network number to an interface that supports a single network, perform the following interface configuration task:

Task	Command
Enable IPX routing on an interface.	**ipx network** *number* [**encapsulation** *encapsulation-type*]

For an example of how to enable IPX routing, see the section "Enabling IPX Routing Example" later in this chapter.

If you specify an encapsulation type, make sure you choose the one that matches that used by the servers and clients on that network.

Assign Network Numbers to Interfaces that Support Multiple Networks

The first logical network you configure on an interface is considered to be the primary network. Any additional networks are considered to be secondary networks. Remember that each network on an interface must use a distinct encapsulation and that it should match the clients and servers using the same network number.

To configure multiple IPX networks on an interface, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Enable IPX routing on the primary network.	**ipx network** *number* **encapsulation** *encapsulation-type*
Step 2	Enable IPX routing on a secondary network.	**ipx network** *number* **encapsulation** *encapsulation-type* **secondary**

To configure more than one secondary network, repeat step 2 as appropriate.

For an example of this type of configuration, see the section "Enabling IPX Routing on Multiple Networks Example.

Table 1-1 lists the encapsulation types you can use on IEEE interfaces and shows the correspondence between the encapsulation type and the IPX frame type.

Table 1-1 Novell IPX Encapsulation Types on IEEE Interfaces

Interface Type	Encapsulation Type	IPX Frame Type
Ethernet	novell-ether (default)	Ethernet_802.3
	arpa	Ethernet_II
	sap	Ethernet_802.2
	snap	Ethernet_Snap
Token Ring	sap (default)	Token-Ring
	snap	Token-Ring_Snap

Control Access to IPX Networks

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

There are four types of IPX access lists that you can use to filter various kinds of 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 IPX access lists have numbers from 800 to 899.
- Extended access list—Restricts traffic based on the IPX protocol type. You can further restrict traffic by specifying source and destination addresses and address masks, and source and destination sockets. Extended IPX access lists have numbers from 900 to 999.
- SAP access list—Restricts traffic based on the IPX Service Advertising Protocol (SAP) type. These lists are used for SAP filters and Get Nearest Server (GNS) response filters. Novell SAP access lists have numbers from 1000 to 1099.
- IPX NetBIOS access list —Restricts traffic based on names, not numbers.

There are 13 different types of IPX filters that you can define for IPX interfaces. They fall into five groups:

- Generic output filters. These filters control which packets are sent out an interface based on the packet's source and destination addresses and IPX protocol type.
- Routing table filters. These filters control which routing (RIP) updates are accepted and advertised by the communication server and which communication servers the local communication server accepts RIP updates from.
- Service Advertisement Protocol (SAP) filters. These filters control which SAP services the communication server accepts and advertises and which Get Nearest Server (GNS) response messages it sends out.
- IPX NetBIOS filters. These filters control incoming and outgoing IPX NetBIOS packets.
- Broadcast filters. These filters control which broadcast packets are forwarded.

Table 1-2 summarizes the types of filters and the commands you use to define them. Use the **show ipx interfaces** command to display the filters defined on an interface.

Table 1-2 IPX Filters

Filter Type	Command Used to Define Filter
Generic filters	
Filter outbound packets based on protocol, address and address mask, and socket.	**ipx access-group** *access-list-number*
Routing table filters	
Control which networks are added to the routing table.	**ipx input-network-filter** *access-list-number*
Control which networks are advertised in routing updates.	**ipx output-network-filter** *access-list-number*
Control the communication servers from which updates are accepted.	**ipx router-filter** *access-list-number*
SAP filters	
Filter incoming service advertisements.	**ipx input-sap-filter** *access-list-number*
Filter outgoing service advertisements.	**ipx output-sap-filter** *access-list-number*

Filter Type	Command Used to Define Filter
Control the communication servers from which SAP updates are accepted.	**ipx router-sap-filter** *access-list-number*
Filter list of servers in GNS response messages.	**ipx output-gns-filter** *access-list-number*
IPX NetBIOS filters	
Filter incoming packets by node name.	**ipx netbios input-access-filter host** *name*
Filter incoming packets by byte pattern.	**ipx netbios input-access-filter bytes** *name*
Filter outgoing packets by node name.	**ipx netbios output-access-filter host** *name*
Filter outgoing packets by byte pattern.	**ipx netbios output-access-filter bytes** *name*
Broadcast filters	
Control which broadcast packets are forwarded.	**ipx helper-list** *access-list-number*

You perform one or more of the following tasks to control access to IPX networks:

- Create access lists
- Create generic filters
- Create filters for updating the routing table
- Create Service Advertisement Protocol (SAP) filters
- Create Get Nearest Server (GNS) response filters
- Create IPX NetBIOS filters
- Create broadcast message filters

Keep the following in mind when configuring IPX network access control:

- Access lists entries are scanned in the order you enter them. The first matching entry is used. To improve performance, it is recommended that you place the most commonly used entries near the beginning of the access list.
- An implicit *deny everything* entry is defined at the end of an access list unless you include an explicit *permit everything* entry at the end of the list.
- All new entries to an existing list are placed at the end of the list. You cannot add an entry to the middle of a list. This means that if you have previously included an explicit *permit everything* entry, new entries will never be scanned. The solution is to delete the access list and re-enter it with the new entries.
- Take care not to set up conditions that result in packets' getting lost. One way this can happen is when a communication server or interface is configured to advertise services on a network that has access lists that deny these packets.

Create Access Lists

To create access lists, you can perform one or more of the following tasks in global configuration mode:

Task	Command
Create a standard IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-node*[*source-node-mask*]] [*destination-network*[.*destination-node* [*destination-node-mask*]]]
Create an extended IPX access list (for generic, routing, and broadcast filters).	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[.*source-node* [[*source-network-mask.*]*source-node-mask*]] *source-socket* [*destination-network* [.*destination-node* [[*destination-network-mask.*]*destination-node-mask*]] *destination-socket*]]
Create an IPX access list for SAP filters.	**access-list** *access-list-number* {**deny** | **permit**} *network*[.*node*] [*network.node-mask*] [*service-type* [*server-name*]]
Create an access list for filtering IPX NetBIOS packets by node name.	**netbios access-list host** *name* {**deny** | **permit**} *string*
Create an access list for filtering IPX NetBIOS packets by arbitrary byte pattern.	**netbios access-list bytes** *name* {**deny** | **permit**} *offset byte-pattern*

Standard access list numbers are from 800 to 899. Extended access list numbers are from 900 to 999. SAP access list numbers are from 1000 to 1099.

Once you have created an access list, apply it to a filter on the appropriate interfaces as described in the sections that follow. This activates the access list.

Create Generic Filters

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

To create generic filters, perform the following tasks:

Step 1 Create a standard or an extended access list.

Step 2 Apply a filter to an interface.

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

Task	Command
Create a standard IPX access list.	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-node*[*source-node-mask*]] [*destination-network*[.*destination-node* [*destination-node-mask*]]]
Create an extended IPX access list.	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[.*source-node* [[*source-network-mask.*]*source-node-mask*]] *source-socket* [*destination-network* [.*destination-node* [[*destination-network-mask.*]*destination-node-mask*]] *destination-socket*]]

Standard access list numbers are from 800 to 899. Extended access list numbers are from 900 to 999.

To apply a generic filter to an interface, perform the following task in interface configuration mode:

Task	Command
Apply a generic filter to an interface.	**ipx access-group** *access-list-number*

For an example of creating a generic filter, see the section "IPX Network Access Example" later in this chapter.

Create Filters for Updating the Routing Table

Routing table update filters control the entries that the communication 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:

Step 1 Create a standard or an extended access list.

Step 2 Apply one or more routing filters to an interface.

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

Task	Command
Create a standard IPX access list.	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-node*[*source-node-mask*]] [*destination-network*[.*destination-node* [*destination-node-mask*]]]
Create an extended IPX access list.	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[.*source-node* [[*source-network-mask.*]*source-node-mask*]] *source-socket* [*destination-network* [.*destination-node* [[*destination-network-mask.*]*destination-node-mask*]] *destination-socket*]]

Standard access list numbers are from 800 to 899. Extended access list numbers are from 900 to 999.

To apply routing table update filters to an interface, perform one or more of the following tasks in interface configuration mode:

Task	Command
Control which networks are added to the routing table when IPX routing updates are received.	**ipx input-network-filter** *access-list-number*
Control which networks are advertised in routing updates sent out by the communication server.	**ipx output-network-filter** *access-list-number*
Control the communication servers from which routing updates are accepted.	**ipx router-filter** *access-list-number*

You can apply one of each of the following filters to each interface.

Create Service Advertisement Protocol (SAP) Filters

A common source of traffic on Novell networks is Service Advertisement Protocol (SAP) messages, which are generated by NetWare servers and Cisco communication servers when they broadcast their available services. To control how SAP messages from network segments or specific servers are routed among IPX networks, perform the following steps: ”

To create SAP filters, perform the following tasks:

Step 1 Create a SAP access list.

Step 2 Apply one or more filters to an interface.

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

Task	Command
Create a SAP access list.	**access-list** *access-list-number* {**deny** | **permit**} *network*[**.***node*] [*service-type* [*server-name*]

SAP access lists numbers are from 1000 to 1099.

To apply SAP filters to an interface, perform one or more of the following tasks:

Task	Command
Filter incoming service advertisements.	**ipx input-sap-filter** *access-list-number*
Filter outgoing service advertisements.	**ipx output-sap-filter** *access-list-number*
Filter service advertisements received from a particular communication server.	**ipx router-sap-filter** *access-list-number*

For examples of creating and applying SAP filters, see the sections “SAP Input Filter Example” and “SAP Output Filter Example” later in this chapter.

You can apply one of each of the following filters to each interface.

Create Get Nearest Server (GNS) Response Filters

To create filters for controlling which servers are included in the Get Nearest Server (GNS) responses sent by the communication server, perform the following tasks:

Step 1 Create a SAP access list.

Step 2 Apply a GNS filter to an interface.

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

Task	Command
Create a SAP access list.	**access-list** *access-list-number* {**deny** | **permit**} *network*[**.***node*] [*service-type* [*server-name*]

SAP access lists numbers are from 1000 to 1099.

To apply a GNS filter to an interface, perform the following task in interface configuration mode:

Task	Command
Filter the list of servers in GNS response messages.	**ipx output-gns-filter** *access-list-number*

Create IPX NetBIOS Filters

Novell's IPX NetBIOS allows messages to be exchanged between nodes using alphanumeric names as well as node addresses. Therefore, the communication server lets you filter incoming and outgoing NetBIOS packets by the node name or by an arbitrary byte pattern (such as the node address) in the packet.

Note These filters apply to IPX NetBIOS packets only. They have no effect on LLC2 NetBIOS packets.

Keep the following in mind when configuring IPX NetBIOS access control:

- Host (node) names are case sensitive.
- Host and byte access lists can have the same names, because the two types of lists are independent of each other.
- When filtering by node name, the names in the access lists are compared with the destination name field for IPX NetBIOS "find name" requests.
- Access filters that filter by byte offset can have a significant impact on the packet transmission rate, because each packet must be examined. You should use these access lists only when absolutely necessary.
- If a node name is not found in an access list, the default action is to deny access.

To create filters for controlling IPX NetBIOS access, perform the following tasks:

Step 1 Create a NetBIOS access list.

Step 2 Apply the access list to an interface.

To create one or more NetBIOS access lists, perform one or both of the following tasks in global configuration mode:

Task	Command
Create an access list for filtering IPX NetBIOS packets by node name.	**netbios access-list host** *name* {**deny** | **permit**} *string*
Create an access list for filtering IPX NetBIOS packets by arbitrary byte pattern.	**netbios access-list bytes** *name* {**deny** | **permit**} *offset-byte-pattern*

To apply a NetBIOS access list to an interface, perform one or more of the following tasks in interface configuration mode:

Task	Command
Filter incoming packets by node name.	**ipx netbios input-access-filter host** *name*
Filter incoming packets by byte pattern.	**ipx netbios input-access-filter bytes** *name*
Filter outgoing packets by node name.	**ipx netbios output-access-filter host** *name*
Filter outgoing packets by byte pattern.	**ipx netbios output-access-filter bytes** *name*

You can apply one of each of these four filters to each interface.

Create Broadcast Message Filters

communication servers normally block all broadcast requests and do not forward them to other network segments. This is done to prevent the degradation of performance inherent in broadcast traffic over the entire network. The **ipx helper-list** command defines which broadcast messages get forwarded to other networks.

To create filters for controlling broadcast messages, perform the following tasks:

Step 1 Create an access list.

Step 2 Apply a broadcast message filter to an interface.

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

Task	Command
Create a standard IPX access list.	**access-list** *access-list-number* {**deny** | **permit**} *source-network*[.*source-node*[*source-node-mask*]] [*destination-network*[.*destination-node* [*destination-node-mask*]]]
Create an extended IPX access list.	**access-list** *access-list-number* {**deny** | **permit**} *protocol* [*source-network*[.*source-node* [[*source-network-mask.*]*source-node-mask*]] *source-socket* [*destination-network* [.*destination-node* [[*destination-network-mask.*]*destination-node-mask*]] *destination-socket*]]

Standard access lists have numbers from 800 to 899. Extended access lists have numbers from 900 to 999.

To apply a broadcast message filter to an interface, perform the following tasks in interface configuration mode:

Task		Command
Step 1	Specify a helper address for forwarding broadcast messages.	**ipx helper-address** *network.node*
Step 2	Apply a broadcast message filter to an interface.	**ipx helper-list** *access-list-number*

For examples of creating and applying broadcast message filters, see the section "Helper Facilities to Control Broadcasts Example" later in this chapter.

Note that a broadcast message filter has no effect unless you have issued an **ipx helper-address** command on the interface to enable and control the forwarding of broadcast messages. This command is discussed in the section "Use Helper Addresses to Forward Broadcast Messages" later in this chapter.

Tune IPX Network Performance

To tune IPX network performance, perform one or more of the following tasks:

- Control Novell IPX compliance
- Configure static routes
- Adjust routing table update timers
- Configure static SAP table entries
- Configure the queue length for SAP requests
- Adjust SAP update timers
- Set maximum paths
- Control responses to GNS requests
- Use helper addresses to forward broadcast messages
- Control the forwarding of type 20 packets
- Repair corrupted network numbers

Control Novell IPX Compliance

In general, our communication server's implementation of IPX complies with the Novell IPX communication server Specification.

To control spcific aspects of IPX compliance, you can use a combination of global configuration and interface configuration commands. You can perform one or more of the following tasks in global configuration mode:

Task	Command
Impose additional controls on the forwarding of IPX type 20 propagation packets.	**ipx type-20-input-checks**
Perform additional controls on the forwarding of IPX type 20 propagation packets.	**ipx type-20-output-checks**

You can perform one or more of the following tasks in interface configuration mode:

Task	Command
Set the tick count, which is used in the IPX Routing Information Protocol (RIP) delay field.	**ipx delay** *number*
Administratively bring down an IPX network on an interface. This removes the network from the interface.	**ipx down** *network*
Set the delay between multiple-packet routing updates.	**ipx output-rip-delay** *delay*
Set the delay between packets sent in multiple-packet SAP updates.	**ipx output-sap-delay** *delay*
Control the forwarding of IPX type 20 propagation packets.	**ipx type-20-propagation**

To achieve full compliance, issue the following interface configuration commands on each interface configured for IPX:

Task		Command
Step 1	Set the delay between multiple-packet routing updates to 55 ms.	**ipx output-rip-delay 55**
Step 2	Set the delay between packets sent in multiple-packet SAP updates.	**ipx output-sap-delay 55**
Step 3	Enable type 20 packet propagation. Do this only if you want to forward type 20 broadcast trafiic across the router.	**ipx type-20-propagation**
Step 4	Lift restrictions on the output of type 20 propagation packet broadcasts.	**no ipx type-20-output-checks**

Configure Static Routes

IPX uses the Routing Information Protocol (RIP) to determine the best path when several paths to a destination exist. RIP then dynamically updates the routing table. However, you may want to add static routes to the routing table to explicitly specify paths to certain destinations. Static routes always override any dynamically learned paths.

Be careful when assigning static routes. When links associated with static routes are lost, traffic may stop being forwarded, even though an alternative path might be available.

To add a static route to the communication server's routing table, perform the following task in global configuration mode:

Task	Command
Add a static route to the routing table.	**ipx route** *network network.node*

Adjust Routing Table Update Timers

You can set the interval between IPX RIP updates on a per-interface basis. You also can specify that a delay be inserted between the packets of a multiple-packet update.

You can set RIP update times only in a configuration in which all communication servers are our communication servers or in which the IPX communication servers allow configurable timers. The timers for all communication servers connected to the same network segment should be the same. The RIP update value you choose affects internal IPX timers as follows:

- IPX routes are marked invalid if no routing updates are heard within three times the value of the update interval (3**interval*) and are advertised with a metric of infinity.
- IPX routes are removed from the routing table if no routing updates are heard within four times the value of the update interval (4**interval*).
- If you define an update timer for more than one interface in a communication server, the granularity of the update timer is determined by the lowest value defined for one of the interfaces in the communication server. The communication server "wakes up" at this granularity interval and sends out updates as appropriate. For more information about granularity, see the *Communication Server Command Reference* publication.

You might want to set a delay between the packets in a multiple-packet update if there are some slower PCs on the network.

To adjust RIP update times on the communication server, perform one or both of the following tasks in interface configuration mode:

Task	Command
Adjust RIP update interval.	**ipx update-time** *interval*
Adjust the delay between multiple-packet routing updates.	**ipx output-rip-delay** *delay*

For an example of configuring the RIP update interval, see the section "Enabling IPX Routing on Multiple Networks Example" later in this chapter.

Configure Static SAP Table Entries

Servers use SAP to advertise their services via broadcast packets. communication servers store this information in the SAP table, also known as the Server Information Table (SIT). This table is updated dynamically. You may want to explicitly add an entry to the SIT so that clients always use the services of a particular server. Static SAP assignments always override any identical entries in the SAP table that are learned dynamically, regardless of hop count. If a dynamic route that is associated with a static SAP entry is lost or deleted, the communication server will not announce the static SAP entry until it relearns the route.

To add a static entry to the communication server's SAP table, perform the following task in global configuration mode:

Task	Command
Specify a static SAP table entry.	**ipx sap** *service-type name network.node socket hop-count*

Configure the Queue Length for SAP Requests

The communication server maintains a list of SAP requests to process, including all pending Get Nearest Server (GNS) queries from clients attempting to reach servers. When the network is restarted, the communication server can be inundated with hundreds of requests for servers. Typically, many of these are repeated requests from the same clients. The **ipx sap-queue-maximum** command allows you to configure the maximum length allowed for the pending SAP requests queue. SAP requests received when the queue is full are dropped, and the client must resend them.

To set the queue length for SAP requests, perform the following task in global configuration mode:

Task	Command
Configure the maximum SAP queue length.	**ipx sap-queue-maximum** *number*

Adjust SAP Update Timers

You can adjust the interval at which SAP updates are sent, and you can set the delay between packets sent in multipacket SAP updates.

Changing the interval at which SAP updates are sent is most useful on limited-bandwidth, point-to-point links or on X.25 interfaces. You should ensure that all Novell servers and communication servers on a given network have the same SAP interval. Otherwise, they may decide that a server is down when it is really up.

Adjusting the delay between packets sent in a multipacket SAP update is useful when the IPX network has slow IPX servers and/or communication servers. Setting a delay between packets in a multipacket SAP update forces our communication server interface to slow its output of SAP packets.

To modify the SAP timers used by the communication server, perform one or both of the following tasks in interface configuration mode:

Task	Command
Adjust the interval at which SAP updates are sent.	**ipx sap-interval** *interval*
Adjust the delay between packets sent in multiple-packet SAP updates.	**ipx output-sap-delay** *delay*

For an example of setting SAP update intervals, see the section “Enabling IPX Routing on Multiple Networks Example” later in this chapter.

Set Maximum Paths

You can set the maximum number of equal-cost, parallel paths to a destination. (Note that when paths have differing costs, the communication server chooses lower-cost routes in preference to higher-cost routes.) The communication server then distributes output on a packet-by-packet basis in round-robin fashion. That is, the first packet is sent along the first path, the second packet along the second path, and so on. When the final path is reached, the next packet is sent to the first path, the next to the second path, and so on.

The cost of a path is determined by ticks, with hop count used as a tie-breaker.

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

To set the maximum number of paths on the communication server, perform the following task in global configuration mode:

Task	Command
Set the maximum number of equal-cost paths to a destination.	**ipx maximum-paths** *paths*

Control Responses to GNS Requests

You can set the method in which the communication server responds to SAP Get Nearest Service (GNS) requests, and you can set the delay time in responding to these requests.

The default method of responding to GNS requests is to respond with the server whose availability was learned most recently.

To control responses to GNS requests, perform one or both of the following tasks in global configuration mode:

Task	Command
Respond to GNS requests using a round-robin selection method.	**ipx gns-round-robin**
Set the delay when responding to GNS requests.	**ipx gns-response-delay** [*time*]

Use Helper Addresses to Forward Broadcast Messages

communication servers normally block all broadcast requests and do not forward them to other network segments. This is done to prevent the degradation of performance over the entire network. The **ipx helper-address** command enables the forwarding of broadcast messages (except type 20 broadcasts) to other networks. It forwards all other unrecognized broadcast messages. These are non-RIP and non-SAP packets that are not addressed to the local network. Forwarding broadcast messages is sometimes useful when a network segment does not have an end-host capable of servicing a particular type of broadcast request. Using the **ipx helper-address** command, you can specify the address of a server, network, or networks that can process the broadcast messages.

Our communication servers support all-networks flooded broadcasts (sometimes referred to as *all-nets flooding*). These are broadcast messages that are forwarded to all networks. Use all-nets flooding carefully and only when necessary, because the receiving networks may be overwhelmed.

Use the **ipx helper-list** command, described earlier in this chapter, to define access lists that control which broadcast packets get forwarded.

To specify a helper address for forwarding broadcast messages, perform the following task in interface configuration mode:

Task	Command
Specify a helper address for forwarding broadcast messages.	**ipx helper-address** *network.node*

For an example of this type of configuration, see the section “Helper Facilities to Control Broadcasts Example” later in this chapter.

You can specify multiple **ipx helper-address** commands on a given interface.

Control the Forwarding of Type 20 Packets

NetBIOS over IPX uses type 20 propagation broadcast packets flooded to all networks to get information about the named nodes on the network. NetBIOS uses a broadcast mechanism to get this information, because it does not implement a network layer.

Communication servers normally block all broadcast requests. By enabling type 20 packet propagation, IPX interfaces on the communication server accept and forward type 20 propagation packets. Before flooding (forwarding) the packets, the communication server performs loop detection as described by the IPX communication server specification.

The communication server can be configured to apply extra checks to type 20 propagation packets above and beyond the loop detection described in the IPX specification. These checks are the same ones that are applied to helpered all-nets broadcast packets. They can limit the unnecessary duplication of type 20 broadcast packets. The extra helper checks are as follows:

- Accept type 20 propagation packets only on the primary network, which is the network that is the primary path back to the source network.
- Forward type 20 propagation packets only via networks that do not lead back to the source network.

While this extra checking increases the robustness of type 20 propagation packet handling by decreasing the amount of unnecessary packet replication, it has two side effects:

- If type 20 packet propagation is not configured on all interfaces or if the primary interface changes, these packets may be blocked when they are not allowed on the primary interface.
- It may be impossible to configure an arbitrary, manual spanning tree for type 20 packet propagation.

You can enable the forwarding of type 20 packets on individual interfaces, and you can restrict the acceptance and forwarding of type 20 packets. The tasks to do this are described in the following sections.

Enable the Forwarding of Type 20 Packets

By default, type 20 propagation packets are dropped by the router. If enabled, type 20 propagation broadcast packets are received by the communication server and forwarded (flooded) to other network segments, subject to loop detection.

To enable the receipt and forwarding of type 20 packets, perform the following task in interface configuration mode:

Task	Command
Enable the receipt or forwarding of IPX type 20 propagation packet broadcasts.	**ipx type-20-propagation**

Restrict the Acceptance of Incoming Type 20 Packets

For incoming type 20 propagation packets, the communication server is configured by default to accept packets on all type 20 enabled interfaces When this command is in effect, the communication server accepts packets only from the single network that is the primary route back to the source network. This means that similar packets from the same source that are received from other networks will be dropped.

Checking of incoming type 20 propagation broadcast packets is done only if the interface is configured to receive and forward type 20 packets.

To impose restrictions on the receipt of incoming type 20 propagation packets in addition to the checks defined in the IPX specification, perform the following global configuration task:

Task	Command
Impose restrictions on the acceptance of IPX type 20 propagation packet broadcasts.	**ipx type-20-input-checks**

Restrict the Forwarding of Outgoing Type 20 Packets

For outgoing type 20 propagation packets, the communication server is configured by default to send packets on all type 20 enabled interfaces, subject to loop detection. When this command is in effect, the communication server will send these packets only to networks that are not routes back to the source network. (The communication server uses the current routing table to determine routes.)

Checking of outgoing type 20 propagation broadcast packets is done only if the interface is configured to receive and forward type 20 packets.

To impose restrictions on the transmission of type 20 propagation packets and to forward these packets to all networks using only the checks defined in the IPX specification, perform the following global configuration task:

Task	Command
Impose restrictions on the transmission of IPX type 20 propagation packet broadcasts.	**ipx type-20-output-checks**

Repair Corrupted Network Numbers

To repair corrupted network numbers on an interface, perform the following tasks in interface configuration mode:

Task		Command
Step 5	Repair corrupted network numbers.	**ipx source-network-update**

Caution The **ipx source-network-update** command interferes with the proper working of OS/2 Requestors. Do not use this command in a network that has OS/2 Requestors.

Caution Do not use the **ipx source-network-update** command on interfaces on which NetWare servers are using internal network numbers.

Configure IPX over WANs

You can configure IPX over X.25, Frame Relay, SMDS, and DDR networks. To do this, you configure the appropriate address mappings. You also can route IPX packets over serial interfaces configured for dial-on-demand routing (DDR).

IPX sends periodic watchdog (keepalive) packets. Therefore, when configuring IPX over DDR, you may want to disable the generation of these packets. This is not an issue for the other WAN protocols, because they establish dedicated connections rather than establishing connections only as needed.

Novell IPX watchdog packets are keepalive packets that are sent from servers to clients after a client session has been idle for approximately 5 minutes. On a DDR link, this would mean that a call would be made every 5 minutes, regardless of whether there were data packets to send. You can prevent these calls from being made by configuring the communication server to respond to the server's watchdog packets on a remote client's behalf. This is sometimes referred to as "spoofing the server."

Monitor and Maintain the IPX Network

To monitor and maintain a Novell IPX network, perform one or more of the following tasks at the EXEC prompt:

Task	Command
Delete entries in the IPX routing table.	**clear ipx route** [*network* | *****]
Diagnose basic IPX network connectivity (user-level command).	**ping ipx** {*host* | *address*}
Diagnose basic IPX network connectivity (privileged command).	**ping**
Display the status of the IPX interfaces configured in the communication server and the parameters configured on each interface.	**show ipx interface** [*interface unit*]
List the entries in the IPX routing table.	**show ipx route** [*network*]
List the servers discovered through SAP advertisements.	**show ipx servers** [**sorted** [{**name** | **net** | **type**}]]
Display information about the number and type of IPX packets transmitted and received.	**show ipx traffic**

Configuration Examples

This section provides configuration examples for the following IPX configuration situations:

- Enabling IPX routing (page 18-19)
- Enabling IPX routing on multiple networks (page 18-20)
- IPX network access (page 18-20)
- SAP input filter (page 18-21)
- SAP output filter (page 18-22)
- Helper facilities to control broadcasts (page 18-24)

Enabling IPX Routing Example

The following configuration commands enable IPX routing, defaulting the IPX host address to that of the first IEEE-conformance interface (in this example, Ethernet 0). Routing is then enabled on Ethernet 0 and Ethernet 1 for IPX networks 2abc and 1def, respectively.

```
ipx routing
interface ethernet 0
ipx network 2abc
interface ethernet 1
ipx network 1def
```

Enabling IPX Routing on Multiple Networks Example

The following example creates four networks on Ethernet interface 0.

```
interface ethernet 0
ipx network 1
ipx encapsulation novell-ether
ipx network 2 encapsulation snap secondary
ipx network 3 encapsulation arpa secondary
ipx network 4 encapsulation sap secondary
```

Any configuration parameters that you specify on this interface are applied to all the logical networks. For example, if you set the routing update timer to 120 seconds, this value is used on all four networks.

If you administratively bring down Ethernet interface 0 using the **shutdown** interface configuration command, all four networks are shut down. You cannot bring down each network independently using the **shut** command; however, you can do this using the **ipx down** command.

To bring down network 1, use the following command:

```
ipx down 1
```

To shut down all four networks on the interface and remove all the networks on the interface, use one of the following commands:

```
no ipx network

no ipx network 1
```

To remove one of the secondary networks on the interface (in this case, network 2), use this command:

```
no ipx network 2
```

Enabling IPX over a WAN Interface Example

When you configure the communication server to transport IPX packets over a serial interface that is running a WAN protocol such as Frame Relay or PPP, you specify how the packet will be encapsulated for transport. This encapsulation is not the same as the encapsulation used on the IPX LAN interfaces.

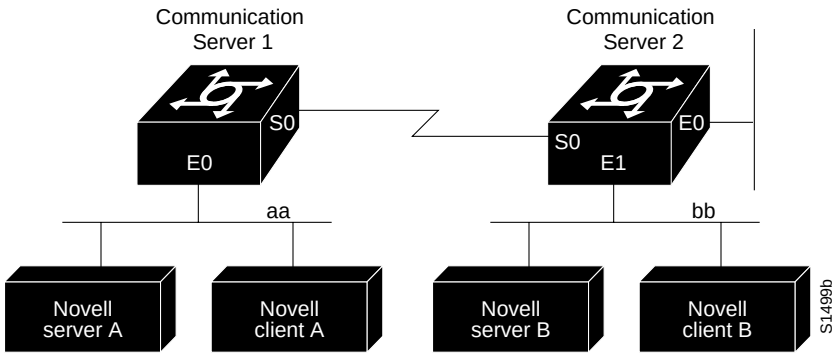
The following example configures a serial interface for Frame Relay encapsulation and for several IPX virtual networks:

```
interface serial 0
encapsulation frame-relay
interface serial 0.1
ipx network 1
interface serial 0.2
ipx network 2
```

IPX Network Access Example

Using access lists to manage traffic routing can be a powerful tool in overall network control. However, it requires a certain amount of planning and the appropriate application of several related commands. Figure 1-1 illustrates a network featuring two communication servers on two network segments.

Figure 1-1 Novell IPX Servers Requiring Access Control



Suppose you want to prevent clients and servers on network aa from using the services on network bb, but you want to allow the clients and servers on network bb to use the services on network aa. To do this, you would need an access list on the E1 interface on communication server 2 that blocks all packets coming from network aa and destined for network bb. You would not need any access list on the E0 interface on communication server 1.

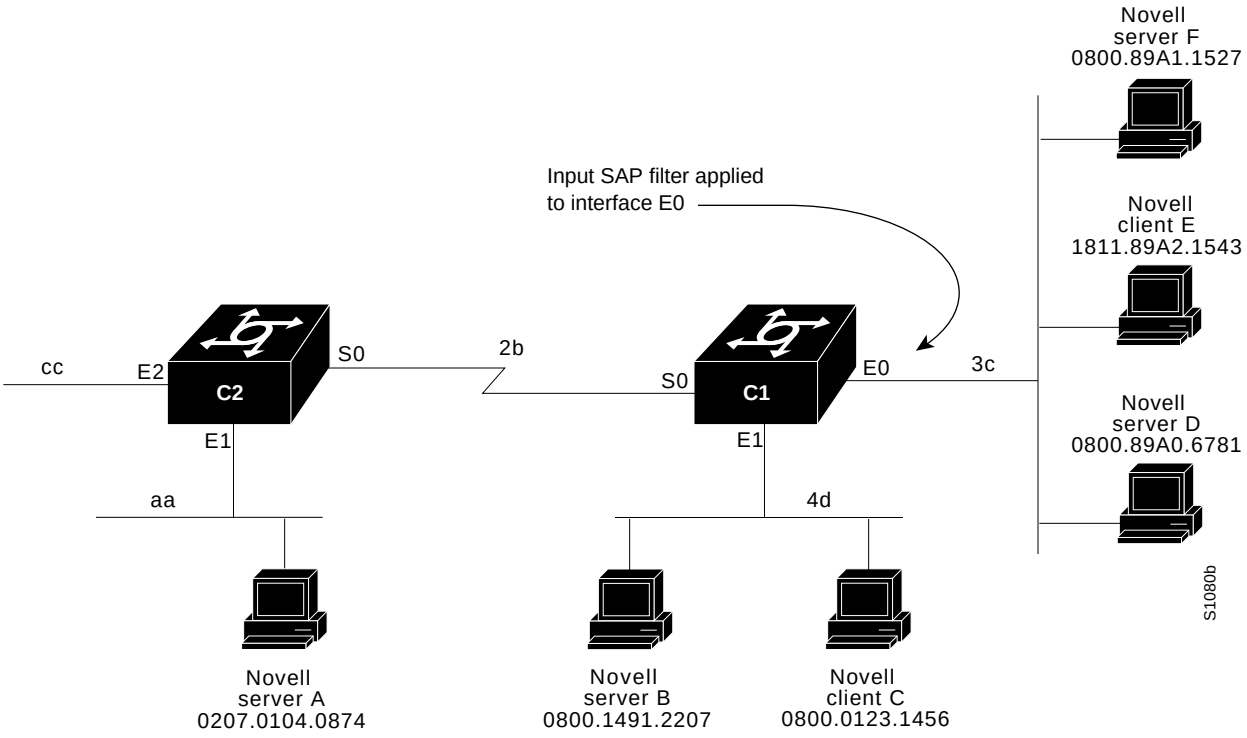
You would configure the E1 interface on communication server 2 with the following commands:

```
ipx routing
access-list 800 deny bb aa
access-list 800 permit -1 -1
interface ethernet 1
ipx network bb
ipx access-group 800
```

SAP Input Filter Example

SAP input filters allow a communication server to determine whether or not to accept information about a service. Router C1, illustrated in Figure 1-2, will not accept and, consequently not advertise, any information about Novell server F. However, Router C1 will accept information about all other servers on the network 3c. Router C2 receives information about servers D and B.

Figure 1-2 SAP Input Filter



Example

This example configures communication server C1. The first line denies server F. It accepts all other servers.

```
access-list 1000 deny 3c.0800.89a1.1527
access-list 1000 permit -1
interface ethernet 0
ipx network 3c
ipx input-sap-filter 1000
interface ethernet 1
ipx network 4d
interface serial 0
ipx network 2b
```

Note NetWare version 3.11 (and later) servers use an internal network and node number as their address for access list commands (the first configuration command in this example).

SAP Output Filter Example

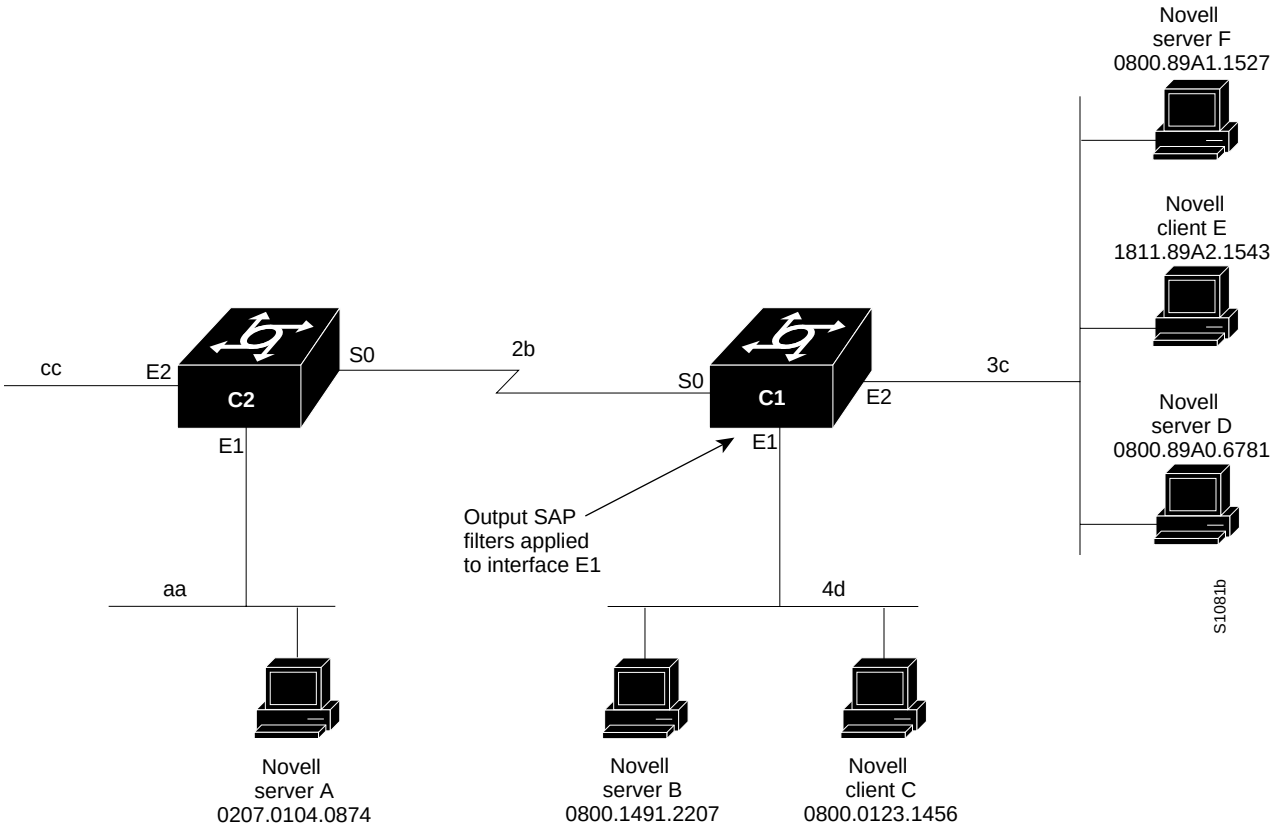
SAP output filters are applied prior to the communication server sending information out a specific interface. In the example that follows, Router C1 (illustrated in Figure 1-3) is prevented from advertising information about Novell server A out interface Ethernet 1, but can advertise server A on network 3c.

Example

The following example refers to Router C1. The first line denies server A. All other servers are permitted.

```
access-list 1000 deny aa.0207.0104.0874
access-list 1000 permit -1
interface ethernet 0
novell net 3c
interface ethernet 1
ipx network 4d
ipx output-sap-filter 1000
interface serial 0
ipx network 2b
```

Figure 1-3 SAP Output Filter



Helper Facilities to Control Broadcasts Example

The following examples illustrate how to control broadcast messages on IPX networks. Note that in the following examples, packet type 2 is used. This type has been chosen as an example,. The actual type to use will depend on the specific application.

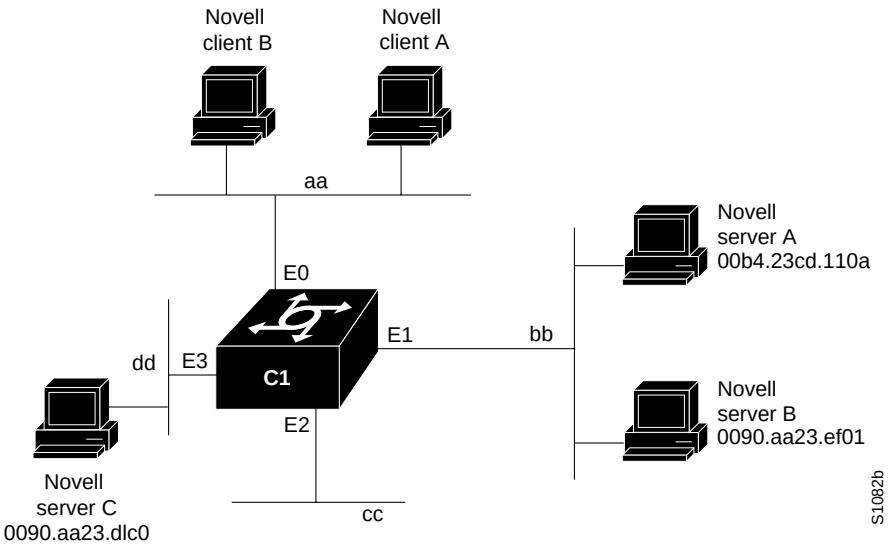
Forwarding to an Address Example

All broadcast packets are normally blocked by the communication server. However, type 20 propagation packets might not be forwarded, subject to certain loop prevention checks. Other broadcasts might be directed to a set of networks or a specific host (communication server) on a segment. The following examples illustrate these options.

Figure 1-4 shows a communication server (C1) connected to several Ethernets. In this environment, all Novell clients are attached to segment aa, while all servers are attached to segments bb and dd. In controlling broadcasts, the following conditions are to be applied:

- Only type 2 and type 20 broadcasts are to be forwarded.
- The IPX clients on network aa are allowed to broadcast via type 10 to any server on networks bb and dd.
- The IPX clients are allowed to broadcast using type 20 broadcasts to any server on network dd.

Figure 1-4 Novell Clients Requiring Server Access through a Router



Example

This example configures the communication server shown in Figure 1-4. The first line permits broadcast traffic of type 2 from network aa. The interface and network commands configure each specific interface. The **ipx helper-address** commands permit broadcast forwarding from network aa to bb and from network aa to dd. The helper list allows type 2 broadcasts to be forwarded. A specific permission to allow type 20 broadcasts to be forwarded between networks aa and dd is also required.

```
access-list 900 permit 2 aa
interface ethernet 0
ipx network aa
ipx type-20-propagation
ipx helper-address bb.ffff.ffff.ffff
ipx helper-address dd.ffff.ffff.ffff
ipx helper-list 900
interface ethernet 1
ipx network bb
interface ethernet 3
ipx network dd
ipx type-20-propagation
```

This configuration means that any network that is downstream from network aa (for example, some arbitrary network aa1) will not be able to broadcast (type 2) to network bb through Communication server C1 unless the router(s) partitioning networks aa and aa1 are configured to forward these broadcasts with a series of configuration entries analogous to the example provided for Figure 1-4. These entries must be applied to the input interface and be set to forward broadcasts between directly connected networks. In this way, such traffic can be passed along in a directed manner from network to network. A similar situation exists for type 20 packets.

Example

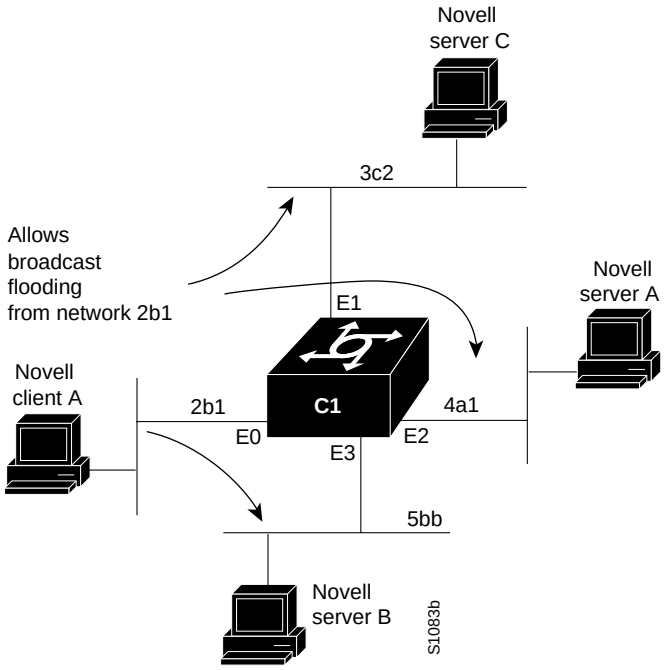
The following example rewrites the **ipx helper-address** command line to direct broadcasts to server A:

```
ipx helper-address bb.00b4.23cd.110a
! Permits node-specific broadcast forwarding to
! Server A at address 00b4.23cd.110a on network bb
```

Forwarding to All Networks Example

In some networks, it may be necessary to allow client nodes to broadcast to servers on multiple networks. If you configure your communication server to forward broadcasts to all attached networks, you are flooding the interfaces. In the environment illustrated in Figure 1-5, client nodes on network 2b1 must obtain services from IPX servers on networks 3c2, 4a1, and 5bb through Router C1. To support this requirement, use the flooding address (-1.ffff.ffff.ffff) in your **ipx helper-address** interface subcommand specifications.

Figure 1-5 Type 20 Broadcast Flooding



In this example, the first line permits traffic of type 20 to network 2b1. Then the first Ethernet interface is configured with a network number. The helper address is defined and the helper list limits forwarding to type 20 traffic. Type 20 broadcasts from network 2b1 are forwarded to all directly connected networks. All other broadcasts are blocked. To permit broadcasts, delete the **ipx helper-list** entry.

```
access-list 901 permit 20 2b1
interface ethernet 0
ipx network 2b1
ipx helper-address -1.ffff.ffff.ffff
ipx helper-list 901
interface ethernet 1
ipx network 3c2
interface ethernet 2
ipx network 4a1
interface ethernet 3
ipx network 5bb
```

All-Networks Flooded Broadcasts Example

This example configures all-nets flooding on an interface.

```
interface ethernet 0
ipx network 23
ipx helper-address -1.FFFF.FFFF.FFFF
```

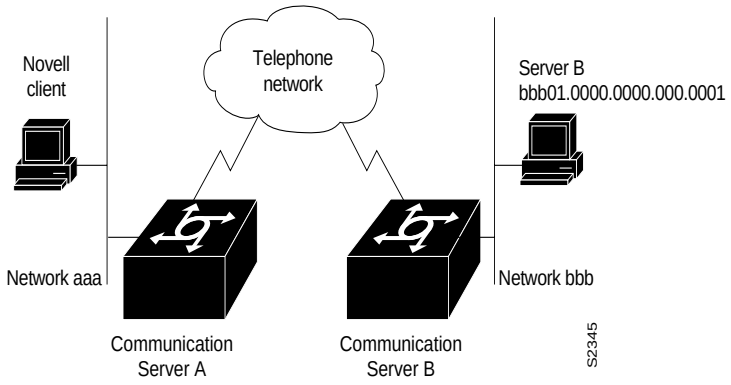
As a result of this configuration, Ethernet interface 0 will forward all broadcast messages (except type 20) to all the networks it knows how to reach. This flooding of broadcast messages may overwhelm these networks with so much broadcast traffic that no other traffic may be able to pass on them.

IPX over DDR Example

The following example configures IPX to run over a dial-on-demand (DDR) configuration illustration in Figure 1-6. In this configuration, an IPX client is separated from its server by a DDR telephone line. Once the server and client have established contact, the server will send keepalive packets regularly. The purpose of these packets is to ensure that the connection between the server and the client is still functional; they contain no other information. Server's send keepalive packets approximately every 5 minutes. If you were to allow communication server B to forward the server's watchdog packets to communication server A and the client, communication server B would have to telephone communication server A every 5 minutes just to send watchdog packets.

Instead of having communication server B telephone communication server A only to send keepalive packets, you can enable watchdog spoofing on communication server B. This way, when the server connected to this communication server sends keepalive (watchdog) packets, the communication server B will respond on behalf of the remote client (the client connected to communication server A).

Figure 1-6 IPX over DDR Configuration



The following is an example configuration for communication server A:

```
! configure the communication server to which the client is connected
ipx routing 0000.0c00.59e8
interface serial 0
no keepalive
dialer in-band
dialer string 8986
ipx network aaa
pulse-time 1
dialer-group 1
!
ipx route 42 aaa.0000.0x01.d877
!
access-list 800 permit ffffffff 42.0000.0000.0001
dialer-list 1 list 800
```

The following is an example configuration for communication server B:

```
! configure the communication server to which the server is connected
ipx routing 0000.0x01.d877
interface serial 1
no ip address
bandwidth 56
no keepalive
dialer in-band
ipx network bbb
pulse-time 1
! enable watchdog spoofing on the server’s communication server
ipx watchdog-spoof
```