CONFIDENTIAL

Novell®

NetWare® Link Services Protocol (NLSP) Specification

Revision 0.9

Disclaimer

Novell, Inc. makes no representations or warranties with respect to any software or to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability, title, or fitness for any particular purpose. Further, Novell, Inc. reserves the right to modify or replace any and all parts of the software, this manual, or the contents of this manual at any time, without any obligation to notify any person or entity of such changes.

Further, Novell, Inc. makes no representations or warranties with respect to any NetWare software and specifically disclaims any express or implied warranties of merchantability, title, or fitness for any particular purpose. Further, Novell, Inc. reserves the right to modify or replace any and all parts of NetWare software, at any time, without any obligation to notify any person or entity of such change.

Trademarks

Novell, NetWare, and the N design are registered trademarks of Novell, Inc.

ARCnet is a registered trademark of Datapoint Corporation.
AppleTalk and LocalTalk are registered trademarks of Apple Computer, Inc.
Datapoint is a registered trademark of Datapoint Corporation.
G/Net is a registered trademark of Gateway Communications, Inc.
IBM is a registered trademark of International Business Machines Corporation.
ProNET and Proteon are registered trademarks of Proteon, Inc.
Telebit is a trademark of Telebit.
Xerox is a registered trademark and XNS is a trademark of Xerox Corporation.

NetWare Link Services Protocol
1st Edition (March 1993)
Novell Part Number 100-001708-001

Novell, Inc.
2180 Fortune Drive
San Jose, California 95131
U.S.A.

Contents

1. Introduction

This specification defines the NetWare Link Services Protocol (NLSP), which provides Link State routing for IPX networks. Routing is the function in network computing that accomplishes two objectives:

- End-to-end delivery of data traffic over an internetwork; an internetwork contains disjoint media segments, so traffic must be relayed from segment to segment to reach its destination.
- Accommodation to the characteristics of the diverse underlying datalink transmission media; the links used in modern networks vary dramatically, and routing smoothes over the differences for the benefit of more general-purpose software components.

Routing is a software function. In this document, a *router* refers to a specialized network node, but a multipurpose system can offer both routing functionality and other kinds of service simultaneously. For example, file service, electronic mail, gateway to mainframe systems, hub, and network management. For brevity, the term *router* indicates a network node that runs routing software, even if it is a multipurpose system.

A router attaches to two or more disjoint segments and forwards data traffic as needed from one segment to another. It exchanges information with other routers to acquire sufficient information to make the best forwarding choices.

NLSP is (essentially) a protocol for information exchange among routers geared to the needs of large IPX internetworks. IPX is the Network-Layer protocol used by Novell's NetWare network operating systems, and by the compatible products of other system providers.

NLSP incorporates the *Link State* approach to network-layer routing that is being adopted broadly in the industry. Link-State routing can be hierarchical; networked systems are grouped into routing areas and areas are grouped into routing domains. Hierarchy promotes scalability. This specification covers the first level, Level 1 of the hierarchy—the protocol for operation within a routing area. Subsequent specifications will expand the scope to cover hierarchical operation.

1.1. Document Organization

This document is for network software developers who build implementations compatible with Novell's NLSP products.

Section 2 provides an overview of the NLSP design. Reading it provides necessary background for the details covered later.

The subsequent sections specify the protocol and procedures in sufficient detail for implementation. Each contains a *Database* subsection (where applicable), describing constants, configurable parameters, and dynamic values pertinent to that part of the design. The specification in each chapter uses the database items defined not only in

that chapter but in previous chapters—the databases are cumulative. Where relevant, each concludes with a *Packet Structures* subsection specifying the formats of messages exchanged among routers.

Section 3 documents the IPX WAN version 2 (IW2) protocol, which specifies initial connection setup methods for various WAN media. It has relevance not only to NLSP, but also to other IPX routing protocols.

Section 4 covers the local database each router maintains to describe the router's immediate neighborhood, and covers the protocol exchanges that take place with neighboring routers to keep the information current.

Section 5 describes methods for the "immediate neighborhood" information to be disseminated throughout a routing area. The routers cooperate to ensure that they keep a consistent, up-to-date representation of the state of all routers and links in the area. Hence the name "Link State." From the representation, they can make informed decisions about how to forward data traffic.

Section 6 describes the Decision Process, which uses the Link State representation to decide the best paths for forwarding data traffic to all destinations accessible in a routing area.

Section 7 covers compatibility with the previously existing RIP and SAP protocols. These two protocols have been used pervasively in IPX internetworks for routing and resource location. With NLSP, they continue to be the protocol used for communication between end nodes and routers. Moreover, for communication among routers, there is a compatibility specification enabling NLSP routers and RIP routers to coexist constructively.

Section 8 itemizes the network management aspects of NLSP operation. The specification there identifies Management Information Base (MIB) variables accessible to management stations from NLSP routers. The MIB supports configuration, monitoring, and troubleshooting.

Section 9 lists design differences between NLSP and the ISO 10589 IS-IS Standard. The core design of NLSP is based on IS-IS, which is familiar in the industry.

Section 10 is a bibliography. References cited in the body of the specification are listed there.

1.2. Typographic Conventions

In this specification, new and important terms and concepts appear in *italic* type when introduced for the first time.

Protocol constants and management parameters appear in sansSerif type with multiple words run together. The first word is lowercase. The first character of each subsequent word is UPPERCASE.

Protocol field names appear in Sans Serif type with embedded spaces and with the first character of each word UPPERCASE.

Descriptive values of constants, parameters, and protocol fields appear enclosed in "double quotation marks." For example, with electric switches:

0 = "Off"
1 = "On"

Patches of pseudocode are in `monospace type`. Unless otherwise specified, arithmetic is performed using non-negative integers, with remainders truncated on division.

Hexadecimal values are preceded by "0x". Numbers appearing without such qualification are decimal. For example, 17 = 0x11.

In diagrams, this document uses a diamond-shaped symbol to represent a router, a horizontal or vertical line to represent a LAN, an oblique line to represent a WAN, and a cloud to represent a collection of networks whose internal organization is not applicable to the immediate discussion. See Figure 1-1.

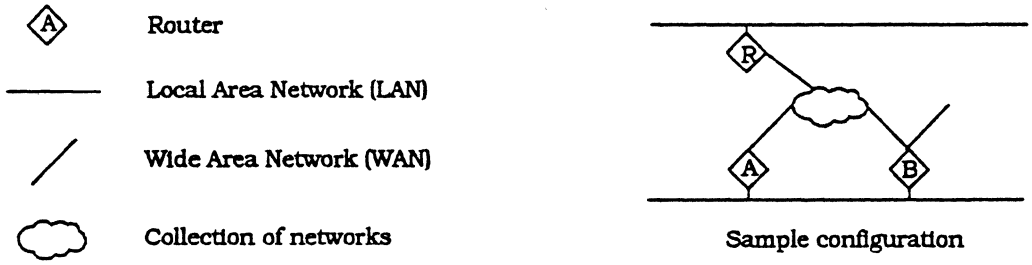


Figure 1-1: Diagram conventions

2. Basic Concepts

The goal of routing is to convey data traffic from the source end node to the intended destination. Routing methods are implemented in special network nodes called "routers," which are connected to two or more network segments of like or unlike kind. The router forwards traffic from segment to segment. A given packet might have to be forwarded several times. To make good decisions, routers exchange information with each other about the topology of the internetwork. Several approaches to the information exchange have been used in the industry. NLSP uses the *Link State* approach. With this approach, every router keeps a road map representing the states of the links and routers in a routing area.

2.1. Link State Databases and Algorithms

The method is called Link State because routers keep track of the state of every communication link in a routing area. The router keeps a record of all the routers in the area, the links connecting them, the operational status of the routers and links, and related parameters. NLSP operates within a "routing area" of links and routers. Figure 2-1 illustrates a simple example. Keep the example in mind as the discussion turns to databases, protocols, and algorithms.

Figure 2-2 illustrates the data exchanges among routers, the information databases each router maintains, and the processing steps to accomplish routing functions.

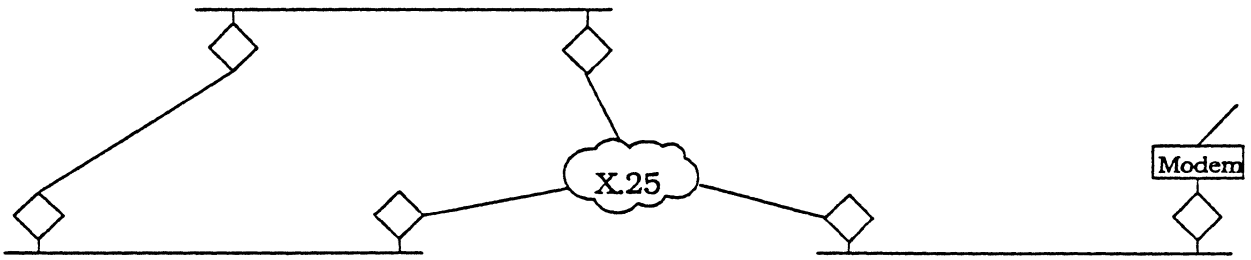


Figure 2-1: Example Internetwork

In the figure, a horizontal line represents a LAN, an oblique line represents a point-to-point link, and the cloud in the center represents a packet data network using X.25. The diamond-shaped symbols represent routers. End nodes, such as workstations, personal computers, and nonrouting servers, are not shown in the diagram.

Each router maintains a database describing the entire area. For each point-to-point link, the database records the endpoint routers and the state of the link. For each LAN, the database records the routers connected to the LAN. The X.25 network is modeled as a set of point-to-point links that go up and down as calls are established and cleared. Likewise for the dial-up modem link.

2.1.1. Maintaining Adjacencies between Routers and their Neighbors

The first database, "Adjacencies," keeps track of the router's immediate neighbors and the operational status of the directly attached links. Regular exchanges of "Hello" messages maintain this information with each neighbor. When a circuit is enabled, the router begins a periodic transmission of Hello messages and listens for the messages from its neighbor. When the router detects (through these exchanges) that the link is reliably operational, it enters a record of the neighbor in the Adjacencies database. If the exchange is interrupted, the router updates the Adjacency database to reflect this fact. This way, it keeps track of its immediate neighborhood.

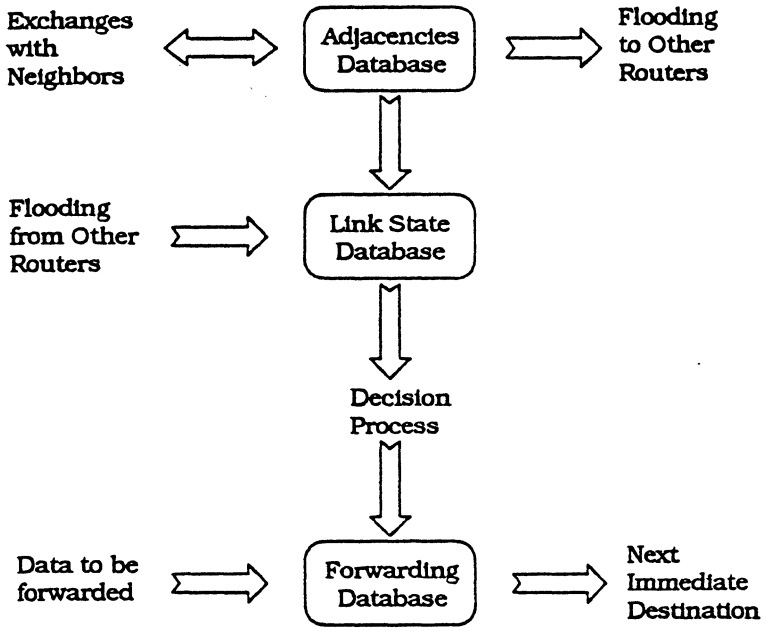


Figure 2-2: NLSP Overview

2.1.2. Synchronizing Link State Information Among Routers

Imagine aggregating the "immediate neighborhood" information from all routers into one database; it would represent the connectivity of an entire routing area. This is the Link State database. It is helpful to think of the Adjacency database as a subset of the Link State database—the portion describing the directly attached links. (The way things actually work, portions of the Adjacency database are combined with local information to become the part of Link State database describing a router's immediate neighborhood.) The NLSP design aims for every router in a routing area to have a copy of the Link State database locally. Moreover, they synchronize their views of the database among themselves. To a great extent, the work of NLSP is synchronization of a replicated database—keeping the copies of the Link State database consistent among the routers. When a topology change occurs, there is a short transitional period when the copies differ. Operation of NLSP makes them converge again.

The Link State database represents routers and links. It does not represent end nodes; that is, it does not represent client workstations, personal computers, or nonrouting servers.

When all the copies of the Link State database are identical and conform with the actual connectivity of the internetwork, forwarding decisions made by successive routers in a packet's path are consistent with each other. The packet progresses from its source to the intended destination successfully and efficiently. If the copies become different from each other, or if they fail to reflect the connectivity of the internetwork accurately, things can go wrong. A packet can find itself in an endless forwarding loop among routers and eventually be discarded. A packet can be routed into a "black hole," a router that has no way to progress the packet toward the intended destination. This is why consistency of the Link State database is so important.

When a link comes up or goes down, it takes time for knowledge of that event to percolate throughout the internetwork. Meanwhile, there is a temporary inconsistency. This might wreak havoc for a Network-Layer protocol that guarantees delivery. IPX does not—it operates on a "best effort" basis, meaning that occasional data loss is tolerable. Still, any disruption must be minimized by making the Link State databases converge to a consistent view as fast as possible. Experience in the industry indicates that Link State algorithms achieve synchronization faster than other designs for large-scale internetworks.

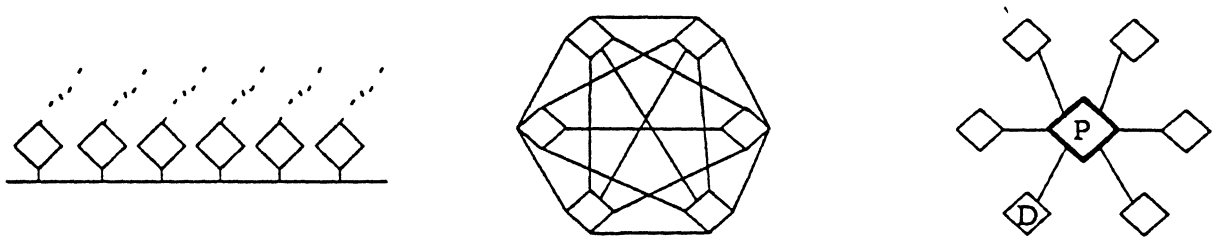
Flooding is the means used to achieve synchronization. Each router sends information from its Adjacency database to each of its neighbors. The information takes the form of *Link State Packets*, or *LSPs*. A numbering scheme detects when the same LSP arrives more than once. When a new LSP arrives—one not seen before—two things happen. First, it is retransmitted on all links except those on which that particular LSP was received. Second, it is merged into the receiving router's Link State database. The database is the collection of LSP packets of two kinds: LSP packets derived from the router's own Adjacency database and LSP packets received from other routers by flooding.

If an adjacency is lost, each router detecting this event floods an LSP indicating that the link is down. This forces routers receiving the LSP to tag the link as down. Record of the link is not removed from the Link State database at this point.

Each LSP includes a Remaining Lifetime field, initialized to the value maxAge seconds by the originating router. Each router holding a copy of the LSP counts down the Remaining Lifetime, reinitializing it if a new copy of the LSP arrives. If the LSP times out, it is purged from the Link State database. At the same time, the router doing the purging floods the expired LSP. The purging mechanism cleans up stale information that has little prospect of becoming useful.

2.1.3. Determining a Designated Router and Constructing Pseudonodes

To keep the size of the Link State database reasonable, LANs need special measures. Consider Figure 2-3, which shows many routers connected to the same LAN. Part (a) shows each router able to relay between the LAN and a particular WAN. In terms of physical connectivity, each router can reach all the others in one hop. There is full mesh adjacency, as shown in part (b) (the WAN links are not shown there). If there are n routers, this means $n \times (n-1) \div 2$ links.



(a) Physical Topology *(b) Actual Adjacencies* *(c) Adjacency Model*

Figure 2-3: Designated Router and Pseudonode

It would be unduly burdensome to convey so much information to all the other routers in the routing domain, and to have them all process the information to make routing decisions. Instead, the LAN adjacencies among the routers are modeled conceptually, as shown in part (c). A fictitious *pseudonode* ("P" in the diagram) represents the LAN as a whole in the Link State database. Each of the routers represents itself as being directly connected to P. There are *n* "links." End nodes are thought of as being directly connected to P, although they are not represented in LSPs. It's unnecessary to represent them—they are located by having the same network number as the pseudonode.

Because P is fictitious, some actual router is chosen to represent P for Link State protocol exchanges. This is the *Designated Router* ("D" in the diagram). A regular protocol exchange occurs among routers of a LAN by which one (and only one) Designated Router is chosen and a replacement takes over in case the original one goes out of service (or becomes ineligible for some other reason). Periodically, every router multicasts a Hello packet on the LAN. The one with the highest priority (a configurable parameter) wins. In case of a tie, the IEEE MAC addresses are compared numerically, and the higher value wins. If a router is attached to several LANs, it might be Designated for some but not for others—the Designated Router election is independent on different LANs, and is decided by priority and address numbers.

The Designated Router originates LSPs on behalf of the pseudonode. If it resigns because a new router has a higher MAC address, it floods the area with a null LSP indicating that the old pseudonode is no longer valid.

2.1.4. Incorporating Switched Circuits into the Design

Certain wide-area media use switched, connection-oriented datalink media. To reach a remote destination, you initiate a call. If the call is successfully established, you can then transmit data over the connection. Finally, when you are finished, the call can be cleared. Because IPX is a connectionless Network-Layer protocol, there is no necessary association between an application session and a datalink connection.

One type of switched network is a packet-switched network using the X.25 standard. These networks support multiple concurrent connections (called *virtual circuits*) using a single network attachment. In NLSP, each connection is treated as a distinct adjacency, and hence as a distinct link in the Link State database. If a router is connected to an X.25 network and has five connected IPX virtual circuits at the current time, it's as if there were five point-to-point links. X.25 is not modeled as a fully connected cloud, but as a collection of point-to-

point links that come up and down as virtual circuits are established and cleared. Operation of NLSP does not by itself cause virtual circuits to be established or cleared.

Another type of switched network is an asynchronous dial-up circuit using a modem. Each modem supports one connection at a time, at most. A modem link is treated as a point-to-point link that can be up (if connected) or down (if not connected). Operation of NLSP does not by itself cause calls to be established or cleared.

2.1.5. The Circuit Concept in Link State Design

NLSP has the concept of a *circuit* both for LANs and WANs. With LANs, a circuit is a point of attachment to a network segment, through which the router can reach many other systems. If the router has two points of attachment to the same network segment, it is treated as two circuits. With a two-party WAN (either a dedicated or a dialed link), each point of attachment is treated as a circuit. The situation is different for multipoint WANs like X.25 and Frame Relay. For these WAN media, a single point of attachment provides access to many other systems using a connection-oriented Datalink layer. NLSP treats each X.25 or Frame Relay *virtual circuit* (switched or permanent) as a separate circuit for operation of the routing protocol, even though virtual circuits can share the same network point of attachment.

2.1.6. Example of Link State Database

Figure 2-4 illustrates the same example as Figure 2-1, but with the parts labeled. Each router is labeled with a letter toward the end of the alphabet. Each LAN is labeled with a letter toward the beginning of the alphabet. A pair of letters denotes a link. For example, RT denotes the point-to-point link connecting the router R with the router T. Likewise, RA denotes the "link" between R and the pseudonode that represents the LAN A. The designated routers (R, U, and W) are shown with heavy borders.

For the sake of discussion, suppose there are two active X.25 virtual circuits: SV and UV. Also, suppose the modem attached to W is not currently connected to a remote party.

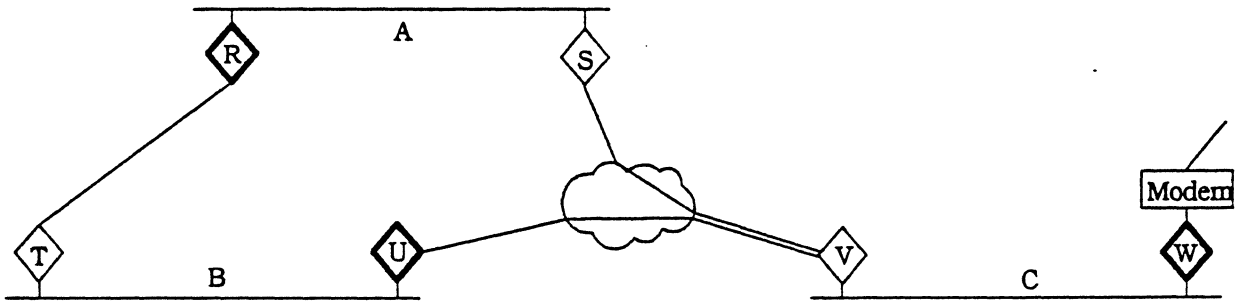


Figure 2-4: Labeled Example

Figure 2-5 shows a simplified view of the Link State database describing the example. There is one table for nodes and a second one for links. Notice that each link is represented twice: once as reported by the router at one end of the link and again as reported by the router at the other end of the link. Cost is discussed in more detail later; for now, simply consider a link to have a non-negative measurement assigned to it. Notice also that a pseudonode reports zero cost to reach the real nodes of the LAN.

Node ID	*Type*
A	Pseudonode
B	Pseudonode
C	Pseudonode
R	Router
S	Router
T	Router
U	Router
V	Router
W	Router

Link	*Cost*	*Link*	*Cost*
AR	0	RA	10
AS	0	SA	10
RT	19	TR	19
BT	0	TB	11
BU	0	UB	11
SV	40	VS	40
UV	40	VU	40
CV	0	VC	10
CW	0	WC	10

Figure 2-5: Link State Database

2.1.7. Decision Process and Forwarding

The typical routing decision is, "given that you have a packet to a particular final destination, what should the next hop be from here to get it to the destination with least cost"? (It is also possible to split the load among several next-hop links.)

The cost table in Figure 2-5 contains the information from which you can deduce routing decisions, but its form is not conducive to deciding the next hop. The *Decision Process* is a computation that puts the cost-table into a new form, called the *Forwarding database*. Unlike the Link State database, each router has a unique version of the Forwarding database, expressed from its own vantage point. The Forwarding database can be thought of as a table that maps a network number into the next hop. Figure 2-6 illustrates a Forwarding database from the vantage point of router V.

Network #	*Next Hop*
0xCCC47689	S
0xCCC47666	C
0xCCC47600	U
0x0082392C	U
0x845FAC11	S
0x05551212	W
0xC3141592	V

Figure 2-6: Forwarding Database Example

The first column in Figure 2-6 contains all the network numbers in the area. For each network number, the "next hop" in that row indicates which outgoing link—and destination on that link—to use for traffic destined to that network. When the router transmits a data frame, the table determines how to forward it. To take one new concept, 0xCCC47666 is the network number for LAN C; end nodes on C are reached by recourse to that table row. As another example, an NLSP router has an *internal network number*, and 0xC3141592 is the one for V. Packets addressed to that network are not forwarded by V. Of course, the internal representation of the Forwarding database is not typically a linear array, but a structure designed for fast lookup.

The Decision Process rebuilds the Forwarding database whenever the router detects a change in the Link State database. There is a hold-down timer, however. This is a minimum interval between reruns of the Decision Process. Topology changes can occur clustered in time. The hold-down timer allows the cluster of changes to be absorbed into the Link State database before running the Decision Process. Otherwise, it would run for each change of the cluster, wasting processing resources.

The Decision Process is called Dijkstra's algorithm, after Edsger W. Dijkstra, who devised the underlying method. It treats each router or pseudonode as a node in a graph, and each link as an arc connecting two nodes. Each link has a cost—actually, two costs, one in each direction, as reported by the router (or pseudonode) at each end. For example, the graph for Figure 2-4 and 2-5 looks like Figure 2-7.

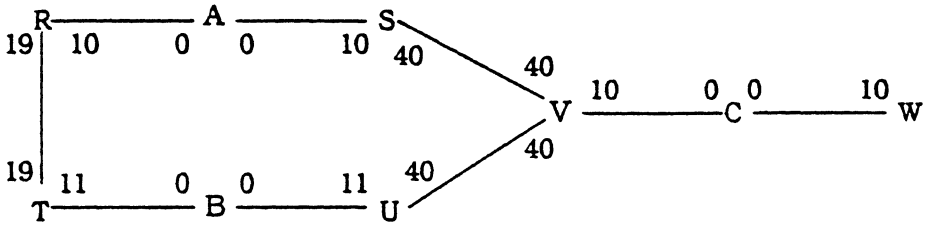


Figure 2-7: Graph with Costs

The object is to find the shortest distance from the local node (that is, from the router where the computation is running) to every other node of the graph. Here are the essential elements of Dijkstra's algorithm:

- Maintain a set of nodes whose shortest distance is already known. Call this the *Known Set*. Initially, it contains only the local node, with zero cost.
- The algorithm operates iteratively. At each iteration, add one node to the Known Set.
- How do you decide which to add? Look at the links from nodes in the Known Set to those outside it. This gives you a subset of links to consider for this step of the iteration. (The subset is equivalent to the "tentative list" used by some texts in describing Dijkstra's algorithm.) Each link in the subset has a "near node," in the Known Set, and a "far node," outside the Known Set. From the subset, choose the one whose far node has the lowest total cost from the local node. The node at the other end of that link is the one chosen. Add it to the Known Set.
- After n iterations, all the reachable nodes are in the Known Set (n is the number of reachable nodes in the graph).

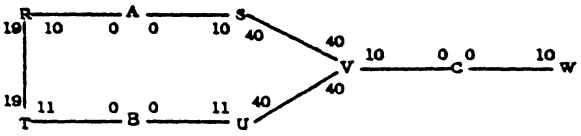
How does this method build a Forwarding database? Each iteration identifies three things: a chosen "far" node, a link connecting it to a previous member of the Known Set, and the previous "near" node itself. For the Forwarding database, the next hop to the chosen node must be determined. If the "near" node is the local node itself, the "next hop" is the link chosen in this iteration. Otherwise, the "next hop" is the same as the near node's next hop.

Figure 2-8 illustrates the iterations of Dijkstra's algorithm for the graph shown in Figure 2-7. It shows the computation performed by node "U."

Step 6 is interesting because it involves choosing between two routes to the same node. The algorithm selects the better path, rejecting the sub-optimal one. In either case, the same node (S in the example) is added to the Known Set, but it is important that the better route to it be the one recorded as the result of the step.

The computational burden to run Dijkstra's algorithm is proportional to $n \times k \times \log n$, where n is the number of routers, and k is the number of neighbors per router. Several optimizations are possible. Memory requirements are proportional to $n \times k$. For a detailed discussion, see Reference [Per92], pages 2-24 through 2-28.

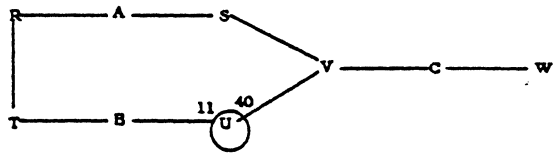
Original graph with costs



Step 1

Known Set

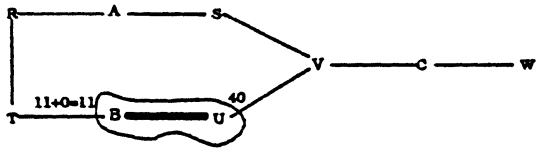
Node	Cost	Next Hop
U	0	–



Step 2

Known Set

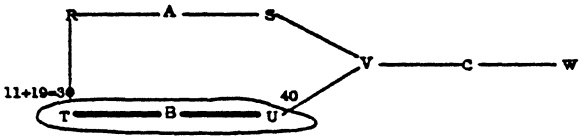
Node	Cost	Next Hop
U	0	–
B	11	B



Step 3

Known Set

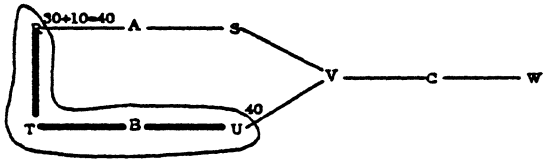
Node	Cost	Next Hop
U	0	–
B	11	B
T	11	B



Step 4

Known Set

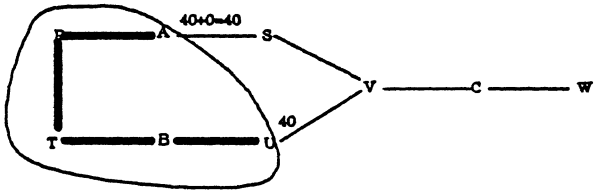
Node	Cost	Next Hop
U	0	–
B	11	B
T	11	B
R	30	B



Step 5

Known Set

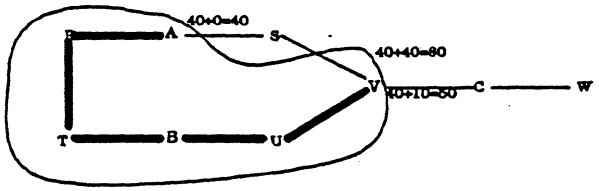
Node	Cost	Next Hop
U	0	–
B	11	B
T	11	B
R	30	B
A	40	B



Step 6

Known Set

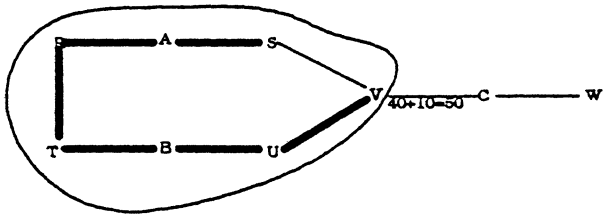
Node	Cost	Next Hop
U	0	–
B	11	B
T	11	B
R	30	B
A	40	B
V	40	V



Step 7

Known Set

Node	Cost	Next Hop
U	0	–
B	11	B
T	11	B
R	30	B
A	40	B
V	40	V
S	40	B



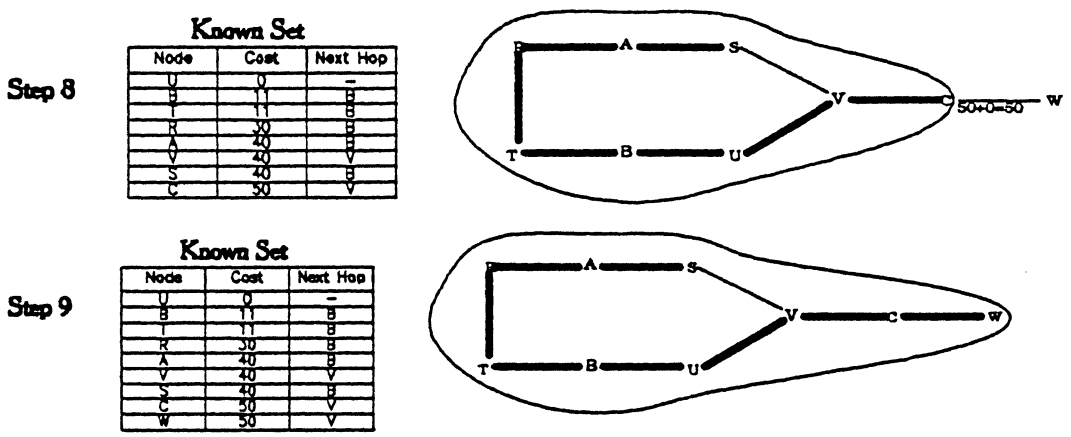


Figure 2-8: Decision Process Steps

Each iteration of Dijkstra's algorithm identifies a node and a link. The chosen links, taken together, form a subset of the graph called a *single source shortest path spanning tree*. The tree is shown in the diagram with a heavy line. Section 6 presents an alternate description of Dijkstra's algorithm. It is more akin to pseudocode.

2.1.8. Load Splitting

NLSP supports load splitting. That is, if there are equal cost paths, the traffic can be divided among them to make fuller use of the internetwork.

There is a *maximal splitting degree* defined. This is a number, MSD, assigned by the network administrator. It can be different for different routers in a routing area. If there are equal cost paths, MSD is an upper limit on the number used for routing. If MSD=1, the router does not do load splitting. Load splitting focuses on the step in Dijkstra's algorithm where you choose which node to add to the known set.

In case of a tie to the same far node, the Forwarding database has more than one entry added to it, instead of just one. How many? If the tie is among m links, the number of entries added is the smaller of m and MSD. If m is greater than MSD, the choice of which to add is based on a prescribed ordered list of criteria.

2.2. Reliability Features of NLSP

A number of NLSP features specifically ensure reliable operation.

2.2.1. Disseminating Updates Reliably

LAN and WAN links often do not provide guaranteed delivery. They operate on an unreliable "best effort" basis. The LSP exchange protocol includes reliability features to ensure that the Link State database replicas in the routers become synchronized.

As mentioned earlier, each router sends LSP packets to its neighbors. Some LSPs describe the contents of the sender's Adjacency database; others are forwarded from other neighbors. To ensure that the Link State databases throughout the routing area are synchronized, a reliability feature is applied to these transmissions. There are two methods: one for point-to-point links and the other for broadcast links such as LANs.

For point-to-point links, a router locally sets a *Send Routing Message* (SRM) flag on an LSP before transferring it to a neighbor. The flag remains set until an acknowledgment is received. Periodically, LSPs with the SRM flag are sent.

A router replies each time it receives an LSP on a point-to-point link. Suppose A sends an LSP to B. In the normal case, the LSP is equal to the one B already has, or newer. In this case, B replies with a Partial Sequence Number Packet (PSNP), with the LSP source's identity, sequence number of the LSP, remaining lifetime of the entry, and other information. This is the form of the acknowledgment. A router can acknowledge more than one LSP in a PSNP.

On the other hand, if the received LSP is older than the one already in the Link State database, B sends the newer LSP. This is actually an anomalous situation. Because the exchange is reliable, A should not have sent an old LSP, unless B had queued the new one for output and set the SRM flag. It could be a race condition: perhaps the new LSP is already in B's output queue, waiting for its turn on the wire. The design accounts for this possibility. The queued output can be considered to be the "B sends the newer LSP" packet. So if B has the SRM flag set for the LSP, it need do nothing. To an eavesdropper on the wire, it appears that B responded with the newer LSP. If the flag is not set, there is a protocol error situation; B should still send the newer LSP, but report the error.

Reliability works differently for broadcast networks. The Designated Router periodically multicasts a (CSNP). This doesn't contain the entire Link State database, but just enough information about each LSP so that each of the other routers can determine whether it is synchronized with the Designated Router. If the receiver of a CSNP determines that the Designated Router is out of date, it multicasts the newer information. On the other hand, if the CSNP receiver itself is out of date, it multicasts a PSNP identifying the LSPs for which it requests an update. The Designated Router responds with the missing information.

2.2.2. Operating with Database Overload

A router might find that it has insufficient resources to process an incoming LSP. The problem might be insufficient memory, misconfiguration, or certain transitory conditions. When this happens, the router cannot make correct decisions, so other routers must be warned. A bit in the LSP designates database overload, and the router experiencing overload floods an LSP with the bit set. Traffic is routed around the overloaded router until the situation returns to normal. A router must always preallocate space for the "database overload" LSP, so memory shortage does not prevent sending it.

2.2.3. Calculating and Testing Checksums

A checksum is used on each LSP sent, to detect corruption of data in transit. The computation is the same as the one used in ISO 8473 (Reference [ISO88]). If the receiver detects a checksum error, the packet is dropped and an error is logged.

2.2.4. Coping with System Bugs

In rare situations, a hardware or software bug can cause a memory location in a computer to be overwritten erroneously. If the computer is acting as a router, and if the corrupted data is part of the Link State database, the erroneous information causes incorrect routing decisions.

NLSP includes a provision to recover from situations like this automatically. Periodically, every router performs a checksum computation on the Link State database. The checksum result is compared with the checksum values originally received with each LSP. Detecting a mismatch is cause for drastic action—the entire LSP database is discarded and reacquired from scratch.

In the normal case, where everything checks out correctly, the Decision Process is run immediately after the checksum calculation. This protects against undetected corruption of the Forwarding database.

2.2.5. Implementing Fail-Stop Operation

Routers must be implemented with high reliability of hardware and software in mind. In particular, there should be a very low probability of corrupting data. Although NLSP does contain fault-tolerance features to recover from corrupted data and resource overload, such misadventures do take their toll on smooth operation of the network.

Routers should be implemented in a fail-stop manner. If the system detects that its routing operation might be compromised, the routing function should be deactivated, and then reactivated with an empty routing database. The routing database is a transitory one, without an archival requirement. If one router's database is lost, the information can be reacquired readily (a) from other routers, and (b) by probing the network environment anew. Newly acquired routing information is more useful than old information. Plunging ahead with a possibly corrupted database risks spreading the failure's impact widely to other routers. Consequently, it is better to restart and gather the information anew.

2.3. The Criterion Used in Determining Routes

When making routing decisions, a router tries to determine the best path for packets to follow from source to destination. But "best" according to what criterion? To address this question, a *cost* is assigned to each link. The cost is a non-negative integer. The Decision Process operates to minimize the total cost for packets to travel from source to destination.

Each type of media has a predefined default cost defined later in this specification. This cost is an indication of the throughput capacity of a given link. For example, here are the default costs for certain common media:

X.25	40
ISDN	30
AppleTalk	25
Point-to-point sync	19
ARCnet	15
IEEE 802.5	11
IEEE 802.3/Ethernet	10
FDDI	7

The end-user can override the cost assigned to a given link. This allows the user to customize routing behavior. Each side of a link has a separate view of the cost. It is recommended (but not required) that the views indicate the same cost. If they don't, routing can be asymmetric. That is, the route from A to B can traverse different links than the route from B to A.

Asymmetric routing still works, but problems can be more difficult to diagnose than with symmetric routes.

2.4. Information Useful to Higher Layers

NLSP conveys information about the characteristics of the links throughout the routing area: the connectivity, the costs, the media types, and so forth. This information is used within the IPX network layer, to do a good job of routing. Because NLSP includes all the mechanisms to convey such information reliably throughout the area, those mechanisms are exploited to convey information not used for routing, but which is useful to other software. There are three items currently defined, with room to add more in the future. The three are identified as follows:

MTU size. This is the maximum packet size (in bytes of IPX user data) that can be transmitted over a link in a given direction. Knowing this value for each link allows a router to determine the MTU size end-to-end from an originator to a final recipient. This allows the originator to send data packets of just the right size, making best use of the internetwork's capacity.

Delay. This is a measurement of the time (in microseconds) to send a byte of data (excluding protocol headers) across the link. Aggregating this measurement over the links in an end-to-end path, a router can calculate end-to-end delay with fine granularity. This information allows a transport protocol to fine-tune the timeout interval to use waiting for an acknowledgment. The better the fine-tuning, the better the performance achieved.

Throughput. This is the amount of data (in bits per second) that can flow through the interface. This information helps the Transport layer decide the maximum useful unacknowledged data to have in its output queue. It allows fine-tuning of the offered window size of a transport connection.

These values can be accessed using network management. In future protocol revisions there might be additional, more specialized protocols to access the values.

2.5. Routing Information Protocol

The Routing Information Protocol (RIP) has been used with IPX from the early days, and its use continues. RIP is a distance-vector routing protocol modeled after the Xerox XNS protocol of the same name (Reference [Xer81]). Routers exchange routing information with their neighbors through periodic broadcasts. They consolidate the information and pass the summarized data along to other RIP routers and to nonrouting end nodes.

NLSP is a successor to RIP for communicating among routers. For router/end node communication, however, RIP continues to be the one method used.

Reference [Nov92] contains a complete specification of RIP.

RIP uses a value called *RIP Link Delay* as its measurement criterion. It is measured in *ticks* (18.21 ticks per second). RIP Link Delay is the time to deliver a 576-byte packet to a destination across the link (it must be at least one). Conceptually, this value contains components of latency and throughput. NLSP deals separately with latency (called *NLSP Delay* in the protocol) and throughput (*NLSP Throughput*), but combines them for the benefit of RIP compatibility.

2.5.1. RIP for Communicating with End Nodes

This part of RIP support must be operational unconditionally. It provides the way for end nodes to learn (a) the network number of their directly connected network, and (b) which router provides the best next hop to an intended destination. It does not include periodic broadcast.

RIP is a request/reponse protocol. The request includes space to ask about routes to specific network numbers, or the requester can send a general request for routes to all networks. Every router on the directly connected segment reponds with a measurement of the distance through it to the networks. Based on the reponse, an end node decides which router provides the best path to a given network. This is purpose (b).

When an end node begins operation, it does not know the network numbers of the segments to which it is attached. It broadcasts a RIP request on its directly connected LAN segments, using zero as the source and destination network number in the IPX header. Every router on the segment responds. From a response's IPX header, the end node learns the segment network number. This is purpose (a). If there are no routers operating, the end node can still communicate with peers on the same network using network number zero.

When a router replies to a RIP request, it supplies two measurements for each destination network:

- Number of hops (the number of routers that must be traversed to reach the destination network)
- Number of ticks

NLSP routers determine the number of hops from the tree constructed by the Decision Process. The number of ticks is calculated from the delay and throughput measurements described in Section 2.4, "Information Useful to Higher Layers."

2.5.2. RIP for Backward Compatibility

Internetworks can contain both RIP and NLSP routers. NLSP routing includes the method to connect the two worlds into a unified whole. When activated, this compatibility feature involves (a) generating RIP broadcasts so that RIP routers learn how to send data traffic to NLSP routers, and (b) heeding RIP broadcasts, so that NLSP routers learn how to send data traffic to RIP routers.

This part of RIP support is operational conditionally, on a per-link basis.

RIP routes are absorbed into the Link State database as *external routes*, like those to other routing areas. This means, in particular, that NLSP routes are preferred over RIP routes if both are available to the same destination. The RIP route for a network is attached to the node that detects a RIP broadcast on a directly connected network. For a LAN, it is attached to the pseudonode. The two RIP measurements (hop count and tick count) are included. When LSPs are flooded, the RIP information is carried along.

When composing a RIP broadcast, an NLSP router combines the information from (a) the external routes, and (b) the tree calculated by the Decision Process. First, consider an NLSP

route. The router determines the number of hops from the tree constructed by the Decision Process. The number of ticks is calculated from the delay and throughput measurements described in Section 2.4, "Information Useful to Higher Layers." Next, consider a network reachable by a concatenation of an NLSP route and a RIP route. The router determines the number of hops by adding the hop count for the NLSP portion of the path to the RIP hop count originally absorbed from the RIP router. Likewise, the tick count is the sum of the value calculated for the NLSP portion of the path and the value originally absorbed from the RIP router.

RIP filtering is implemented when sending and receiving RIP broadcasts on a per-link basis. The user can configure network numbers (or number patterns) to be included or excluded from consideration. However, the RIP routes carried along in LSPs flooded among NLSP routers are not filtered.

Routers allow the RIP broadcast frequency to be configurable on each link individually. The default is once per minute.

RIP routes are purged if they are not refreshed by receipt of a new RIP broadcast. This timeout interval is configurable on each link individually. The default is four minutes.

2.6. The Service Advertising Protocol

SAP has been used with IPX from the early days, and its use continues. SAP uses a broadcast method similar to RIP for disseminating information, but instead of routing information it conveys information about network services and their addresses. Servers advertise their services and addresses, and routers gather the information to share with other routers and with end nodes.

With the newer (NDS), the use of SAP can be considerably reduced. Workstations locate services of interest by consulting an NDS server. The NDS-resident information about services is disseminated by direct, unicast-based protocols. It is not disseminated by broadcast-based SAP. NDS is part of NetWare v4.x. However, even in a network of all v4.x nodes, some uses of SAP remain. For example, SAP locates the nearest NDS server on startup. SAP usage can be considerably reduced with NDS, but is not eliminated at this point. Moreover, other IPX-based systems have a customer presence and are still being deployed: NetWare v3.x, NetWare v2.x, and products from many parties that add value to those environments. For all these reasons, NLSP routers must support SAP.

Incoming SAP broadcasts are processed in a way similar to incoming RIP broadcasts. The received SAP information is included in the receiving router's Link State database, and from there is flooded to other routers. For a LAN, the SAP information is attached to the pseudonode. Information can come from a RIP router, a nonrouting end node, or a software module running a network service on the router system itself. It is all treated similarly.

An NLSP router responds to SAP resource-location requests using the SAP information gleaned from SAP broadcasts and from other NLSP routers. It also generates periodic SAP broadcasts based on the same information. SAP support is configurable on a per-link basis.

SAP filtering is implemented on a per-link basis when sending and receiving SAP broadcasts. The user can configure names (or name patterns) to be included or excluded from

consideration. However, the SAP carried along in LSPs flooded among NLSP routers is not filtered.

Routers allow the SAP broadcast frequency to be configurable individually on each link. The default is once per minute.

SAP information is purged if it is not refreshed by receipt of a new SAP broadcast. This timeout interval is configurable individually on each link. The default is four minutes.

2.7. NetBIOS and Packet Type 20

For certain protocol implementations to function correctly, NLSP routers propagate certain broadcast packets. The protocol most affected is IPX-based NetBIOS. The IPX packet type 20 is used specifically for this purpose. The special handling applies only to broadcast packets—those with destination node number all-ones in the IPX header. Details are in Section 5 of Reference [Nov92].

End-users require the ability to set boundaries on the propagation of these packets. Routers implement a user-configurable, per-link filter governing whether packets of type 20 are sent and heeded on that link.

2.8. Enhancing the NLSP to Hierarchical Routing

This document specifies routing within a routing area—Level 1 routing. As NLSP develops in the future, there will be facilities for linking areas together into routing domains (Level 2), and routing domains into a global internetwork (Level 3). Figure 2-9 illustrates the hierarchy of domains and areas. Figure 2-9 shows the internals of one of the areas for clarity, and one of the domains. Imagine each area as containing LANs and WANs such as in Figure 2-1. Areas are connected to each other by Level 2 routers; domains are connected to each other by Level 3 routers.

Hierarchical routing intends that each independent organization is a separate routing domain. This could be a company, a university, or an agency. It could be a public carrier connecting organizations with each other. Within a routing domain, each local campus, or other suborganization having its own administrative staff constitutes a routing area.

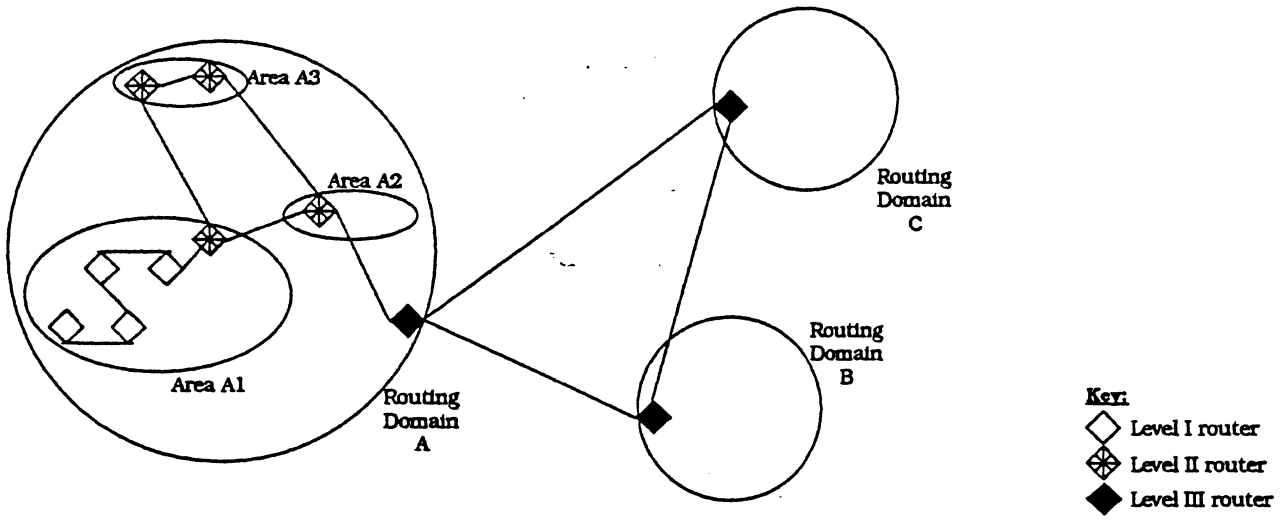


Figure 2-9: Hierarchical Routing

Level 2 routing aims to enhance scalability by reducing the amount of information that every router must store and process to route throughout a domain. Instead of keeping the Link States throughout the routing domain, it keeps this level of detail only for its own area. To get to other areas, it need only maintain information on a per-area basis, and be able to find the Level 2 router for each area other than its own. This information is a much coarser level of detail. Hierarchical routing wins information compression and economy of computation.

Level 3 routing aims first to provide another level of scalability and information compression, and then to imagine different routing domains belonging to different organizations entirely—companies or agencies with an arms-length relationship. A public network service provider that connects different organizations together might have a domain of its own. So Level 3 routing can include policy criteria in forwarding decisions. For example, imagine that each of the three domains in Figure 2-9 is a separate company's internal internetwork. Domain B might want to forbid "transit" traffic from Domain A bound for Domain C.

A Level 2 router also acts as a Level 1 router within its own area; likewise, a Level 3 router also acts as a Level 2 router within its own domain.

2.9. IPX Addressing and its Relationship to Routing

The IPX Network-Layer address contains three parts:

- A four-byte Network number
- A six-byte Node number
- A two-byte Socket number

A Network number is assigned to each LAN segment. Also, NetWare v3.x and v4.x servers have *internal network numbers*. Network numbers receive the most attention in this document, because they guide routing decisions.

The Node number identifies a specific system attached to the network. It is taken from the MAC address space administered by IEEE. This makes it globally unique. (In some situations, it must really be unique only within a network. This fact is sometimes exploited, bypassing global uniqueness.)

The Socket number identifies a higher-layer entity within a system.

Two 32-bit quantities identify each routing area: a network address and a mask. The mask contains *n* leading "one" bits and *32–n* trailing "zero" bits. The pair of numbers is called an *area address*. The term *Category n* area refers to an area where the mask has *n* leading "one" bits. The larger *n* is, the smaller the area. An example is

```
01234500
FFFFFF00
```

The first number identifies the area: address `01234500`. The second number, the mask, indicates how much of the area number identifies the area itself, and how much identifies an individual network within the area. In the example, the first 24 bits of the address (`012345`) are the routing area. (Twenty-four is six hex F's times four bits per hex digit.) When writing the area address as a number/mask pair as above, write zeros after `012345` to pad it to 32 bits. Every network number within the area starts with `012345`. The remaining eight bits are used to identify individual network numbers within the routing area; for example, `012345AB` and `01234500` are network numbers in the area. The example is a Category 24 area.

A routing area can have as many as three area addresses. NLSP treats the three as synonymous identifiers of the same routing area. The three can be of different categories. In fact, no numerical relationship is prescribed between the addresses. Having more than one address allows the routing areas to be reorganized without interrupting operation. For example, to split a routing area in half, you introduce the new area address into half the routers, one by one. Then, remove the old address from those routers, one by one. As long as one of the routers has both addresses, there is one routing area. As soon as the routers become two disjoint groups, there are two areas.

Figure 2-10 provides a simple example, with four routers (A, B, C, D) in a linear sequence. All four start in one area, and conclude with two areas of two routers each.

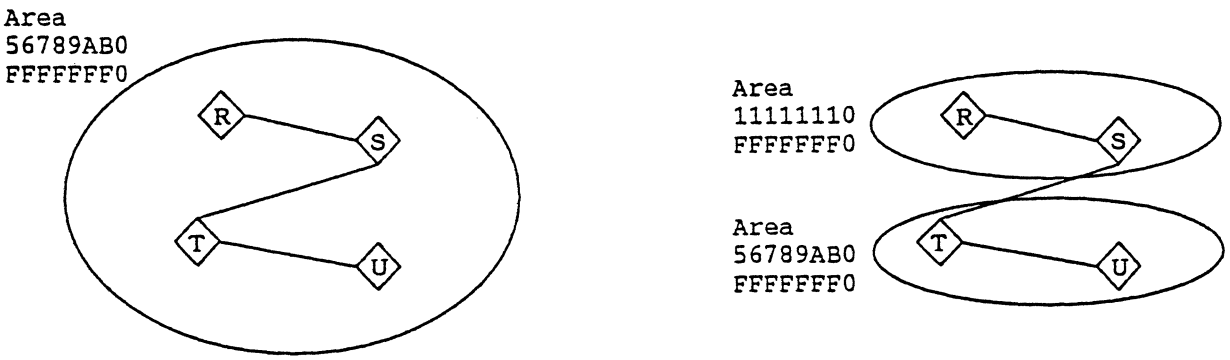


Figure 2-10: Splitting a Routing Area

To accomplish the split, the following steps can be followed:

Step 1	This is the starting point, one area address.			
Router	R	S	T	U
Level	1	1	1	1
Area Addresses	56789AB0 FFFFFFF0	56789AB0 FFFFFFF0	56789AB0 FFFFFFF0	56789AB0 FFFFFFF0

Step 2	Add the second area address to R.			
Router	R	S	T	U
Level	1	1	1	1
Area Addresses	56789AB0 11111110 FFFFFFF0 FFFFFFF0	56789AB0 FFFFFFF0	56789AB0 FFFFFFF0	56789AB0 FFFFFFF0

Step 3	Add it to S, and remove the old area address from R. It's still all one area, though.			
Router	R	S	T	U
Level	1	1	1	1
Area Addresses	11111110 FFFFFFF0	56789AB0 11111110 FFFFFFF0 FFFFFFF0	56789AB0 FFFFFFF0	56789AB0 FFFFFFF0

Step 4	Promote S and T from Level 1 to Level 2 Routers. It's still one area.			
Router	R	S	T	U
Level	1	2	2	1
Area Addresses	11111110 FFFFFFF0	56789AB0 11111110 FFFFFFF0 FFFFFFF0	56789AB0 FFFFFFF0	56789AB0 FFFFFFF0

Step 5	Remove S's original address. The routers discover that there are two nonoverlapping areas.			
Router	R	S	T	U
Level	1	2	2	1
Area Addresses	11111110 FFFFFFF0	11111110 FFFFFFF0	56789AB0 FFFFFFF0	56789AB0 FFFFFFF0

Now let's go from routing areas to the next level of aggregation: routing domains. There is no numerical relationship required among area addresses within a domain, except that no two areas can contain the same area address. When Level 3 routers convey traffic between domains, they share a description of which addresses belong in which domain. This is a list of all the area addresses contained in each domain.

Area addresses in a Routing Domain do not have to share a common prefix for NLSP to work correctly. Any area addresses can be combined in a Domain. But if common prefixes are, in fact, used, routing can operate more efficiently. First, the data shared among Domains is more compact, because the data can be expressed using the common prefix instead of by listing all the area addresses. Second, as a consequence, routers making routing decisions have less data from other domains to process, so they have less work to do. This saves compute cycles. Finally, confidentiality can play a role: a company or carrier may not want to reveal its internal network topology. Consequently, those planning address assignments should consider assigning them hierarchically, even though such an assignment is not required for correct operation. For example, if addresses in the Figure 2-9 internetwork were assigned hierarchically, the Level 3 router in Domain A might have the following address list:

Domain A		Domain B		Domain C	
Area A1	CCCC1500 FFFFFF00	All areas	CCCC2000 FFFFF000	All areas	CCCC3000 FFFFF000
Area A2	CCCC1520 FFFFFFF0				
Area A3	CCCC1600 FFFFFFF0				

Note that the information about domains B and C have collapsed to a single entry each. Likewise, information about Domain A collapses to a single entry from the point of view of B or C. This works because they all differ in the fifth hexadecimal digit: the digit is "1" for domain A; "2" for B, and "3" for C. Within Domain A, only one entry need be circulated to the Level 1 and Level 2 routers to describe Domains B and C. If address assignment is not hierarchical, there are many entries. This consumes network bandwidth to convey to all routers, memory to record the entries, and computational capacity to process them.

2.10. Network Management

NLSP routers are managed using the SNMP (Simple Network Management Protocol) standard. SNMP runs over IP and IPX.

Specifying NLSP network management amounts to defining an SNMP MIB. The MIB is a set of values that can be read and changed remotely by action of the SNMP protocol, and a set of traps that can be generated by the router to raise alarms about exceptional conditions. The NLSP MIB is divided into groups, as follows:

System Group	Global information about the IPX protocol entity
Circuit Group	Information about each interface on the system
Forwarding Group	Forwarding database used to route data traffic
Services Group	Known Services that advertise themselves using SAP
Neighbors Group	Information about neighboring NLSP routers and the adjacencies with them
LSP Group	Tables representing the LSP database
Translation Group	Various mappings (for example, System ID to Server Name)
Graph Group	Representation of the network topology

There are actually three MIBs, as follows:

- IPX MIB includes basic information about IPX Network-Layer operation. It is applicable to all IPX-speaking systems, both routers and end nodes. It includes information in the following groups: System, Circuit, Forwarding, Services.
- NLSP MIB augments IPX MIB with information specific to the NLSP routing protocol. It extends the System, Circuit, and Forwarding groups, and adds the Neighbors, Translation, Graph, and LSP groups.
- RIP/SAP MIB augments IPX MIB with information specific to the RIP and SAP protocols. It extends the System and Circuit groups.

2.11. Capabilities Assumed in the System Environment

2.11.1. Configuration of Parameters by End-Users

Certain operational values and characteristics are configurable by the end-user. To accommodate this requirement, a router needs a method for user interaction. The method is an implementation choice. For example, it could be one of the following:

a) Terminal attached directly to the router

b) Remote terminal-emulation facility

c) File edited by the user and read by the router

d) An implementation-dependent protocol

e) The NLSP MIB defined in this specification

f) Different MIB for a different network management protocol

g) Combination of methods

2.11.2. Event Handling

There are various places in the specification where it is convenient to describe behavior of the router in terms of events. The term *event* is meant in a generic sense. Implementation of event handling is system-specific.

Event handling can include notifying a system manager of abnormal conditions, making an entry in an error log, and/or incrementing a counter.

2.11.3. Characteristics of Links to Support NLSP

The links connecting routers with each other must support frames of 576 bytes. This number is the data portion of the frame, including the Network-Layer (IPX) header, but not datalink headers. If larger frames are supported, performance and scalability can improve.

Links can operate on a "best effort" datagram basis, providing a high probability of delivering individual frames. There is no requirement for the links to provide confirmed or guaranteed delivery. There is no requirement for the links to guarantee that the order of delivery matches the order of transmission. (However, if frames are reordered often, performance can be affected.) NLSP includes reliability features that allow it to operate over unreliable links. NLSP operates over all sorts of links: broadcast, nonbroadcast, local area, metropolitan area, wide area, connectionless, connection-oriented, point-to-point, multipoint, switched, dedicated, and high and low data rates.

The router needs a way to detect when a link becomes active or inactive, and when failures or degraded conditions prevail.

Certain kinds of multipoint networks have the ability to address multiple stations with the same datalink frame. Addressing all stations attached to a network segment is called *broadcast*; when a subset is addressed, the function is called *multicast*. NLSP uses multicast addressing (on those links where it is available) to address all the NLSP routers on the network segment, without burdening other systems. Where broadcast is available but not multicast, the same frames (which would have been multicast) are sent using the broadcast mechanism.

2.11.4. Imposing Jitter on Timed Operations

When packets are transmitted as a result of timer expiration, there is a danger that timers of individual systems may become synchronized. This would result in a traffic distribution containing peaks of intense activity. Where there are large numbers of synchronized systems, the peaks can overload both the transmission medium and the receiving systems. To prevent

this from occurring, periodic timers that cause packet transmissions have *jitter* introduced. When jitter is applied, the actual time interval varies randomly between 75% and 100% of the specified value.

The procedure `DefineJitteredTimer` below indicates how jitter operates in NLSP. In the description, `BaseTimeValueInSeconds` is the nominal timeout value for a specific action, and `ExpirationAction` is a procedure that can be called to perform that action. For example, `DefineJitteredTimer (10, SendHelloPacket)` causes the action *SendHelloPacket* to be performed at random intervals of between 7.5 and 10 seconds.

```
DefineJitteredTimer (BaseTimeValueInSeconds, ExpirationAction)

  CONSTANTS:

      Jitter = 25           -- Percentage jitter defined in this specification.

      Resolution = 100      -- Timer resolution in milliseconds for a
                            -- particular router.

  VARIABLES FOR THE PROCEDURE AS A WHOLE:

      BaseTimeValue         -- BaseTimeValueInSeconds, converted to timer
                            -- intervals.

      MaximumTimeModifier   -- Largest possible jitter.

      Running               -- Starts as TRUE; set to FALSE to halt the
                            -- activity.

  VARIABLES REEVALUATED IN THE INNER LOOP:

      WaitTime              -- Time interval for this loop's timeout.

      NextExpiration        -- next clock time at which the ExpirationAction occurs.

  METHOD:

      BaseTimeValue ← BaseTimeValueInSeconds x 1000 / Resolution

      MaximumTimeModifier ← BaseTimeValue x Jitter / 100

      WHILE Running

            BEGIN           -- Running loop

            WaitTime ← BaseTimeValue - Random (MaximumTimeModifier)

            NextExpiration  ← CurrentTime () + WaitTime

            ExpirationAction ()

            WaitUntil (NextExpiration)

            END          -- Running loop
```

The description assumes availability of the following functions in the environment:

```
Random (max)       -- Returns a uniformly distributed random integer with
                   -- 0 < Random(max) < max.

CurrentTime ()     -- Returns the current time in milliseconds.

WaitUntil (time)   -- This procedure waits until the current time is "time,"
                   -- then returns.
```

The essential point of the algorithm is that the expiration time is randomized within the inner loop. Note that the new expiration time is set immediately on expiration of the last interval, rather than when `ExpirationAction` has completed.

2.12. General Processing of Incoming IPX Packets

IPX (Internetwork Packet Exchange) is a connectionless datagram Network-Layer protocol, adapted from the Xerox Network Systems design, Reference [Xer81]. The IPX packet structure is defined on page 2-28.

When a router receives a valid IPX packet, it proceeds in the following order:

a) If the final destination of the packet is on the same system as the router, it is processed by the software entity identified by the Destination Socket field. This includes packets addressed directly to this system, as well as packets sent by a datalink multicast or broadcast on a directly connected network segment. Exit after completing step a) for such a packet—the remaining steps do not apply. If there is no software entity on the system prepared to receive the packet, the packet is discarded.

   In particular, the routing software entity on the local system processes the packets listed in Figure 2-11.

Protocol	Socket	Packet Type	Where Specified
RIP	0x0452	0	Section 7
SAP	0x0453	1	Section 7
IW2	0x9001	4	Section 3
Propagated Packet	Varies	20	Chapter 5 of Reference [Nov 92]
NLSP	0x9004	0	Sections 4 and 5

Figure 2-11: Packet Types Pertinent to this Document

The following general acceptance tests apply to incoming packets destined for the routing software entity on this system:

1. Size consistency tests apply to NLSP packets. Suppose

   D is the datalink user data size reported by the Datalink Layer,

   I is the Packet Length reported in the IPX header,

   H is the IPX header size (30 bytes), and

   N is the Length Indicator in an NLSP packet.

   Every valid packet has

   $D \geq I$

   $I = N + H$

   If a received packet fails these size consistency tests, a packetRxSmall event is generated and the packet is discarded.

   The "$D \geq I$" test also applies to all arriving IPX packets.

2. If the IPX Packet Type field is incompatible with the Destination Socket field, the packet is dropped. Valid combinations are in Figure 2-11.

3. If the Version field of an incoming NLSP packet is not one, a mismatchedVersion event is generated and the packet is discarded.

4. If an NLSP packet arrives on a circuit for which the NLSP protocol is not active, the packet is discarded.

b) If Packet Type = 20, process it as specified in Chapter 5 of Reference [Nov92].

c) If the Transport Control field is maxHops or greater, the packet is discarded. Traditionally, the value of maxHops is 16, and this is the default value. It is configurable. When a packet is discarded, an ipxInTooManyHops event is logged.

d) All other packets are data packets to be relayed by operation of the router's traffic forwarding role. If the next hop is impossible because the packet is too large for the medium's supported datalink frame size, the packet is discarded. When the router forwards a packet, it increments the Transport Control field by one.

Note: If either the Source or Destination Network Number field is zero, it should be filled in with the actual Network Number before being passed to the appropriate software entity, or before being forwarded. When forwarding a packet, the only two modifications a router can make to the IPX header or data are (a) incrementing the Transport Control field and (b) filling in the actual Network Number.

It is valid to forward a packet even if the Destination Node is all-ones, indicating a broadcast. If the router is directly attached to such a packet's Destination Network, the packet is forwarded as a datalink broadcast on that circuit.

2.13. Packet Structures

All the packets defined in this specification are carried as the data portion of IPX Network-Layer packets. In the packet, they immediately follow the IPX header specified later.

The IPX Network-Layer packets, in turn, are carried as the data portion of datalink frames. They immediately follow the datalink header, and immediately precede the datalink trailer (if any). Datalink headers are media-dependent, and are not defined in this specification. This is illustrated in Figure 2-12.

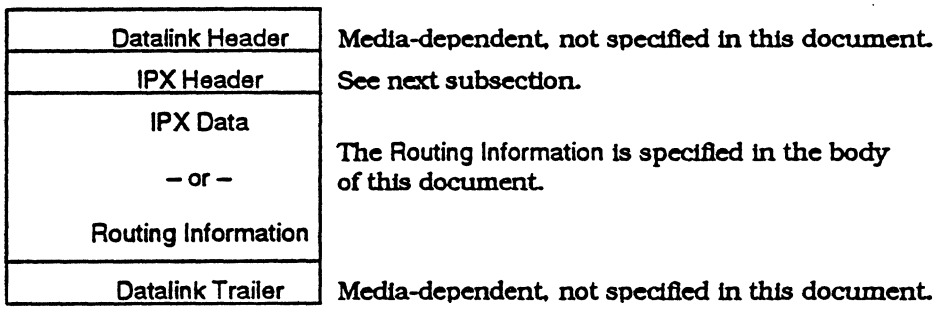


Figure 2-12: Datalink Packet

All multibyte fields are transmitted with the most significant byte first.

When NLSP packets are sent by multicast to all NLSP routers on the directly connected segment, the following destination MAC addresses are used in the datalink header:

- IEEE 802.3 — `0x09001BFFFFFF`
- IEEE 802.5 — `0xC00070000200`
- FDDI — `0x09001BFFFFFF`

2.13.1. IPX Header

IPX Header	Number of Bytes
Checksum	2
Packet Length	2
Transport Control	1
Packet Type	1
Destination Address	12
Source Address	12

Checksum: 0xFFFF. Historically, IPX has not used checksums. However, there are plans to phase in the use of checksums in the IPX header. At present, routers should set this field to 0xFFFF when generating an IPX packet, but should make no assumptions regarding the value this field has for received packets.

Packet Length: The number of bytes in the IPX packet, including the IPX header and the subsequent IPX data.

Transport Control: The number of routers a packet has traversed on its way to its destination. Sending nodes always set this field to zero when originating an IPX packet.

Packet Type: The type of service offered or required by the packet. Values pertinent to this specification are in Figure 2-11.

> **Note:** For some versions of existing products, the Packet Type field is not a reliable indicator of the type of packet encapsulated. The source and destination Socket fields should be used to determine the packet type when this determination is required. Propagated packets are an exception; the Packet Type should be checked for the value 20 to determine whether the packet is to be propagated.

Destination Address: The IPX address identifying the final destination of the packet. Each IPX address has three subfields, as follows:

IPX Address	Number of Bytes
Network Number	4
Node Number	6
Socket	2

- Network number: Identifies the physical network segment to which a system is attached. If a system is attached to more than one segment, it has a corresponding address for each. Certain systems (for example, NLSP routers and NetWare v3.x servers) also have an *internal network number* that does not correspond to an actual network segment, but is thought of as a virtual network residing inside the system. There is no broadcast network number. Network Numbers must be unique throughout an internetwork.

  Network number 0 (zero) has a special meaning: "The directly attached network segment onto which this packet is being transmitted." Systems that are starting up, and have not yet discovered the network number of a segment, use this value. Routers do not forward packets with Destination Network Number 0.

- Node number: Identifies a particular system on the network segment. The value is media-dependent. When the Network Number refers to an IEEE 802.3, 802.4, or 802.5 LAN, the value is the six-byte MAC address. If a medium has a smaller address (for example, Omninet), it is right-justified and zero-padded. The Node Number of a router or server on its internal network is one, by current convention. The broadcast value 0xFFFFFFFFFFFF means "all systems on the directly attached segment." Node numbers must be unique within a Network Number.

- Socket: Identifies a software entity within a networked system. Values pertinent to this specification are in Figure 2-11.

Source Address: The IPX address identifying the originator of the packet. It has the same three-part IPX Address structure as the destination address. Broadcast addresses are not allowed in the Source Node Number.

3. IPX WAN Version 2

This section describes how Novell IPX and routing protocols for IPX operate over various WAN media.

As its name implies, the IPX WAN version 2 (IW2) protocol specified in this section is the successor to the IPX WAN protocol specified in Reference [All92]. This document supersedes that specification, and is backward-compatible with it.

Reference [All92] specifies how to operate IPX, RIP, and SAP over the relevant media. IW2 adds what is needed to support two additional routing protocols: NLSP and Unnumbered RIP. Part of IW2 is to negotiate which routing protocol is used over a particular link.

Reference [All92] specifies how to operate over PPP and X.25. This specification adds two media types: Frame Relay and IP Relay.

3.1. Support of Several Routing Protocols

The routing protocols covered by this specification are as follows:

- RIP, as specified in Reference [Nov92].
- Unnumbered RIP. This operates the same way as RIP, but the link over which IW2 is being run is not assigned an IPX network number.
- NLSP, as specified in sections 4 through 8 of this document.

Unnumbered RIP is motivated by administrative convenience. In an internetwork in which there are many WAN circuits, it is inconvenient to set aside IPX network numbers for them all, and to make sure that there is no duplication.

Of the three routing protocols, only RIP involves assigning a network number to the link running IW2. RIP is called a *numbered* routing protocol type.

Neither NLSP nor Unnumbered RIP uses a network number for the WAN link. They are called *unnumbered* routing protocol types.

Additional routing protocols of numbered and unnumbered types may be specified in the future. The categorization of routing protocols into numbered or unnumbered is an important distinction, which must be considered when planning new designs.

3.2. Implementing Media-Dependent Functions

IW2 strives to treat all media the same way. Different treatment of different media is limited to the minimum needed for effective communication.

Link establishment and termination are procedures that vary from one medium to the next. Other operational details of IPX (for example, encapsulation) are also media-dependent. Recall that the NLSP and RIP packets ride within IPX packets, as do the IPX WAN and IW2 protocols, so no additional information about their encapsulation need be specified, beyond that of IPX.

Individual media types are described one at a time.

3.2.1. Operation over PPP

IPX and its routing protocols use PPP (Reference [Sim92]) when operating over point-to-point synchronous and asynchronous networks.

With PPP, link establishment means the IPXCP reaches the "Open" state, as described in Reference [Sim92a]. The IW2 protocol must not begin until the IPX NCP reaches the "Open" state. If IPXCP negotiates options conflicting with those negotiated later with IW2, the IW2 negotiation takes precedence.

PPP allows either side of a connection to stop forwarding IPX if one end sends an IPXCP or an LCP Terminate-Request. When a router detects this, it immediately reflects the lost connectivity in its routing information database instead of naturally aging it out.

3.2.2. Operation over X.25 Switched Virtual Circuits

With X.25, link establishment means successfully opening an X.25 switched virtual circuit (SVC). As specified in Reference [Mal92], the protocol identifier 0x800000008137 is used in the X.25 Call User Data field of the Call Request frame, and indicates that the SVC is devoted to IPX.

Each IPX packet is encapsulated directly in X.25 data frame sequences without additional framing.

Either side of the SVC can close it, thereby tearing down the IPX link. When a router detects this, it immediately reflects the lost connectivity in its routing information database instead of naturally aging it out.

3.2.3. Operation over X.25 Permanent Virtual Circuits

The nature of Permanent Virtual Circuits (PVCs) is that no call request is made. When the router is informed that X.25 Layer 2 is up, the router assumes that link establishment is complete.

Each IPX packet is encapsulated in an X.25 data frame sequence without additional framing. Novell IPX assumes a particular X.25 PVC is devoted to the use of IPX.

If a router receives a Layer 2 error condition (X.25 Restart, for example), it should reflect lost connectivity for the PVCs in its routing information database and perform the necessary steps to obtain a full IPX connection again.

3.2.4. Operation over Frame Relay

To determine when a PVC has become active or inactive, the router interacts periodically with either a private Frame Relay switch or a public Frame Relay network. The method used depends on the switch or service provider. Some support Reference [DEC90], Section 6; others support Reference [ANS91], Annex D. NLSP supports both these specifications.

When a router is restarted, IW2 exchanges over active Frame Relay PVCs (that is, PVCs that have remained active before and after restart) can begin immediately.

When a router detects that a Frame Relay PVC has switched from an inactive state to an active state, link establishment is considered complete and IW2 exchange over this newly activated PVC begins.

When an active PVC becomes inactive, the router reflects the lost connectivity in its routing information database.

Framing of IPX packets is specified in Reference [Bra92], Section 6. The NLPID (Network-Layer Protocol ID) used for IPX is 0x800000008137.

3.2.5. Operation over IP Relay

IP Relay is a streamlined method to convey IPX traffic through an IP internetwork. IP Relay operates like a collection of two-endpoint PVCs, although the traffic travels as UDP/IP datagrams (References [Pos80] and [Pos81]). Each endpoint of each PVC is an IP address. Each router implementing the IP Relay feature is configured to know the opposite end of each PVC that terminates at its IP address.

When an IPX router is first activated and notices that IP is active on its system (or if IP becomes active while NLSP is already active), all the configured PVCs are considered to be established datalink connections. IW2 exchanges can begin immediately.

The router can receive an ICMP packet (Reference [Pos81a]) indicating nondelivery of an IP Relay datagram. The following two ICMP types are relevant: Destination Unreachable and Time Exceeded. Upon receipt of any of these packets, the router considers the corresponding PVC as failed, and restarts the IW2 protocol.

If the IPX router detects that the IP protocol has become inactive, it considers that the IP Relay PVCs have all been cleared and it reflects the lost connectivity in its routing information database.

Each IPX packet comprises the data portion of a UDP packet. The UDP port used for IP Relay is awaiting assignment.

If a packet arrives at IP Relay's UDP port from an IP address not in the receiving router's PVC list, that packet is ignored.

3.2.6. Operation over other WAN media

Additional WAN media will be added here as specifications are developed.

3.3. Outline of the Stages of IW2 Operation

After the underlying datalink connection is established as described in the preceding media-dependent description, the IW2 protocol is activated to exchange identities and determine certain operational characteristics of the link.

There are three steps of IW2 operation:

- Negotiating the master/slave role and choice of routing protocol. The master/slave roles persist for the IW2 interactions only. The master keeps the rest of IW2 proceeding in an orderly way by initiating the request/reponse exchanges.
- Exchange of packets to determine certain link characteristics empirically. The details of this step depend on the routing protocol negotiated.
- Final exchange of the routers' configuration information

After these steps are concluded, transmission of IPX routing packets begins—using the routing protocol negotiated—as well as transmission of IPX data traffic.

3.4. IW2 Packets and their Usage

IW2 involves exchanges of IPX WAN packets, whose structure is defined on page 3-14. There are seven types of IW2 packets:

- Timer Request—a 576-byte packet sent from both ends of the WAN link to measure packet turnaround time
- Timer Response—response to a Timer Request; this packet is also 576 bytes long
- Throughput Request—a pair of n-byte packets sent back-to-back from the Master to measure the media throughput
- Throughput Response—acknowledgment of the Throughput Request packet pair
- Delay Request—sent by the Master to measure media delay
- Delay Response—response to the Delay Request packet; it has the same size as the request
- Information Request—sent by the Master at the conclusion of the IW2 negotiation to exchange delay and throughput measurements, a network number, and the router names
- Information Response—response packet to the Information Request
- NAK—negative acknowledgment; sent back to a party that sent an illegal packet

Certain packets apply to certain routing types but not others. Details follow.

If the router's software environment includes a priority system, IW2 packets should be processed with higher priority than all other IPX packets

3.5. Steps of the Initial Negotiation, Independent of Routing Type

The first exchange of packets decides the master/slave roles and the routing protocol to be used on the link. It also gauges the link delay, by timing the turnaround time for a moderately sized packet exchange. The initial negotiation is the same for all routing types. It is illustrated in Figure 3-1.

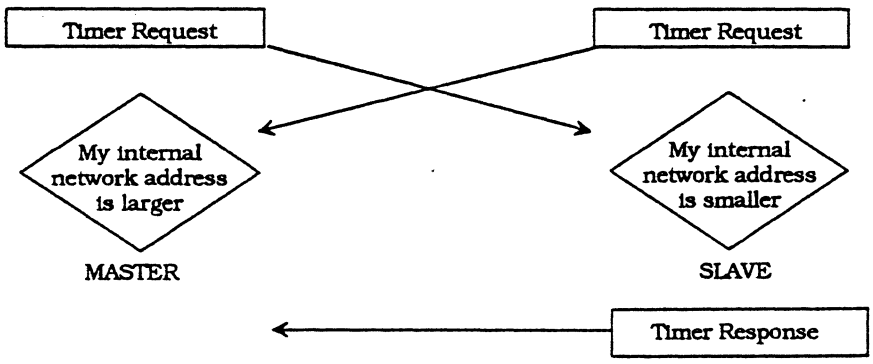


Figure 3-1: Initial Negotiation

After link establishment, both sides of the link send Timer Request packets and start a timer waiting for a Timer Response. See page 3-14 for a specification of packet contents. These

Timer Request packets are sent every 20 seconds until a response is received or retry counter is exhausted. Normally, the retry counter is set to 16.

In composing the Timer Request packet, the router takes into consideration

- Which types of routing protocols it supports
- Whether it is prepared to assign an IPX network number to the link

For each routing protocol supported, place the corresponding option in the Timer Request packet. They appear in the originator's order of preference, with the most preferred first.

For each compression method supported, place the corresponding option in the Timer Request packet.

If the router is prepared to assign an IPX network number to the link, it sends its internalNetworkNumber in the WNode ID field, and omits the Extended Node ID option. On the other hand, if the router is **not** prepared to assign an IPX network number to the link, it sends zero in the WNode ID field. In the latter case, it also includes its internalNetworkNumber in the Extended Node ID option.

Anticipated future routing types allow a system to include zero as the WNode ID field and omit the Extended Node ID option. This system is a nonrouter establishing a connection with a router.

On receiving a Timer Request packet, a router determines its role—Master or Slave—for the remainder of the IW2 exchanges. The Master role does not denote special privilege. It merely means that the router is the requester in the ensuing request/response exchanges. The decision is made as follows:

a) If the WNode ID field is zero in both the sent and the received Timer Requests

   i) If both Timer Requests include an Extended Node ID, the router with the higher numeric value of this field is the Master. If the two Extended Node ID fields are equal, a configuration error has occurred. After reporting the error, the router issues a disconnect on the underlying datalink connection. Manual intervention is needed to correct the error condition.

   ii) If only one Timer Request includes the Extended Node ID, the router sending it is the Master.

   iii) If neither Timer Request includes the Extended Node ID, a connection cannot be established. The datalink circuit is cleared by the system that initiated it.

b) If either the sent or received Timer Request (or both) contains a nonzero WNode ID field, the router with the higher WNode ID is the Master.

c) If the two WNode ID fields are equal and nonzero, a configuration error has occurred. After reporting the error, the router issues a disconnect on the underlying datalink connection. Manual intervention is needed to correct the error condition.

The numeric comparisons are done by considering each byte as an unsigned integer, and the first byte as most significant. Note that in Figure 3-1 the decisions in the diamond-shaped symbols are simplified; the actual decision takes into account the two fields: WNode ID and Extended Node ID.

The Slave responds to the Timer Request with a Timer Response. To do so, it determines the routing protocol to be used on the link according to the following rule:

```
For each routing Routing Type in the received Timer Request,
  in order of appearance
   BEGIN       -- LOOP
   If the Routing Type is one supported by this router
         BEGIN       -- TEST
         If the Routing Type is an unnumbered type,
               choose it and exit.
         If the received WNode ID is nonzero,
               choose this Routing Type and exit.
         END         -- TEST
   END         -- LOOP
```

Note: It is permitted for a router to support a numbered routing type, but not be able to assign the network number. In this case, that routing type can be selected only if the other router supports it and can assign the network number. The rules for determining the Master/Slave role and routing protocol ensure that

- Routers implementing IW2 are interoperable with those implementing Reference [All92].
- If only one router is prepared to assign a network number, it is the Master (the role which does the assignment), so that if a numbered routing type is chosen, the Master has a network number to assign.

The remainder of IW2 depends on the routing type selected. A separate subsection follows for each alternative.

3.6. Remaining Steps for the (Numbered) RIP Routing Type

When the Timer Request/Response exchange concludes with RIP as the chosen routing type for the link, Figure 3-2 illustrates that the Information Request/Reply is the next (and only) remaining step in IW2. This reiterates the specification in [All92]. Once the negotiation is completed, the link can be used for RIP, SAP, and IPX data traffic, but not for NLSP packets.

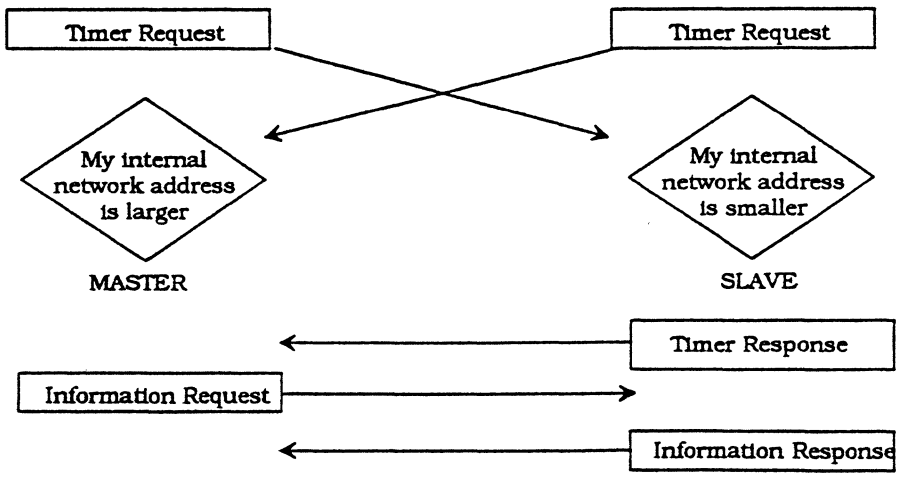


Figure 3-2: IW2 Exchanges for the RIP Routing Type

The following option subfield in the Information Request/Reponse exchange contain operational information used in RIP.

- RIP/SAP Info Exchange option—WAN Link Delay subfield. This link delay is used in the RIP information propagated to the router's other links. The following algorithm determines the IPX WAN link delay, based on the Timer Request/Response exchange. Here, `start_time` is the time that the Timer Request is sent, and `end_time` is the time that the Timer Response is received.

```
link_delay ← end_time - start_time                    -- 1/18th second

link_delay ← MAX ( link_delay, 1 )     -- Ensure link delay is at least 1

link_delay ← 6 × link_delay
```

  Note: Workstations use the Link Delay to timeout sessions. The factor of six is in anticipation of queuing delays for the WAN link that can readily impact round-trip time when data traffic starts flowing. It is a biasing value to prevent timeout of multiple workstation sessions sharing the link.

```
link_delay ← 55 × link_delay  -- Convert link delay to milliseconds
```

The Link Delay is used as the network transport time when advertised in the IPX RIP packet. For a consistent network, a common link delay is required at both ends of the link and is calculated by the link Master.

Note: IPX WAN in its initial version does not retry the Information Request exchange. If no Information Response is received after the first Information Request, the datalink connection is cleared and starts again.

Once the IW2 exchanges are complete, RIP operates over the link as specified in Reference [Nov92].

3.7. Remaining Steps for the Unnumbered RIP Routing Type

With RIP and IPX WAN, an IPX network number is assigned to each (virtual) circuit by operation of the protocol. With Unnumbered RIP, there are no network numbers used for circuits. This is an advantage, because it is easier to administer.

The Unnumbered RIP routing type exploits this administrative ease, while still using RIP and SAP as the routing protocols for the circuit. In Unnumbered RIP, no network number is assigned to the link.

The IW2 packets exchanged for Unnumbered RIP are the same as for RIP. If the Timer Response selects Unnumbered RIP, that mode will be in effect on the link. The next exchange is the Information Request/Response. The Common Network Number in the RIP/SAP Info Exchange field is zero.

Once the IW2 exchanges are complete, RIP operates over the link as specified in Reference [Nov92], with the following exceptions. The Network Numbers in the source/destination addresses of RIP/SAP packets traversing this link are set to zero. These RIP/SAP packets are originated by the routers at either end. Forwarded packets, by contrast, continue to have the originator's and the final destination's network numbers. Because the two ends of the link are routers, the fact that this link has no network number is inconsequential. If one end were an end node, there would have to be a way for systems far away in the internetwork to address it, requiring a network number. If services are accessible on the same system as the router, or if

there are software elements that act as clients, they are attached to the system's internal network number.

3.8. Remaining Steps for the NLSP Routing Type

3.8.1. Normal IW2 Exchanges for NLSP

Figure 3-3 shows the complete IW2 exchange in preparation for using NLSP on the link for the normal case of no error conditions.

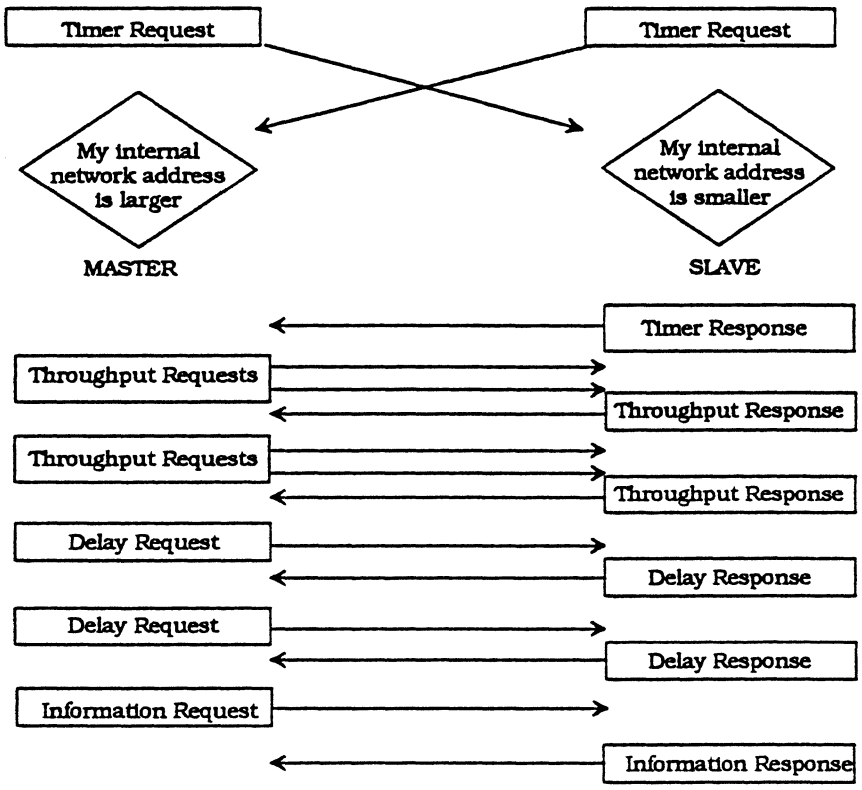


Figure 3-3: IPX WAN Version 2 Protocol for NLSP

The Master initiates Throughput Request and Delay Request packets to calculate those two circuit characteristics. See page 3-14 for a specification of packet contents. Throughput Request and Delay Request packets are re-sent every 20 seconds until a response is received or the retry count is exhausted. If no response is received after the 16 retries, the router issues a disconnect to the link. The Sequence Number field matches a response with the corresponding request. For the Throughput Request, the matched pair contains the same sequence number (not an even/odd pair).

The "Throughput" and "Delay" packet exchanges provide empirical measurements of those two characteristics of the link. The measurements are used later during NLSP operation. "Throughput" is the amount of data (in bits per second) that can flow across the link. "Delay"

is how long it takes to send a zero-byte packet to the destination. (Actually, it is an indication of the wire propagation delay to reach from one point to the other.)

Finally, the IW2 Master sends an Information Request packet to provide the Slave with operational information about the link. The Slave responds to acknowledge receipt. The exchange also informs each router of the other's textual name. If the Master does not receive the reply, NLSP retries every 20 seconds until a response is received or the retry count (16) is exhausted.

The following option subfields in the Information Request/Response exchange contain operational information used in NLSP. The following discussion indicates how to calculate the values that appear in the subfields, based on prior exchanges.

- NLSP Information option—Throughput subfield. This value is calculated based on the Throughput Request/Response exchange: two equal-sized Throughput Request packets are sent back to back. Each is *R* bytes. Nominally, R = 512, but the Master may use a different size based on local configuration, network management action, or the results of earlier exchanges on the link. The receiver (Slave) of these back-to-back packets measures the time between two packets and sends a Throughput Response packet with delta time and the Throughput Request packet length. The Master calculates the actual throughput capacity of the circuit by dividing packet size by delta time. The method assumes that packet size is large enough to offset the driver's holding time and that the lower level driver is sending the packets one after another. The method also expects that an X.25 network delivers packets one after another to the receiving side. As illustrated in Figure 3-3, the request is repeated twice; the results are averaged, as follows:

R-byte packets

--------->

--------->

---------> *t1*

---------> *t2*

throughput1 = R / (t2 – t1)

R-byte packets

--------->

--------->

---------> *t3*

---------> *t4*

throughput2 = R / (t4 – t3)

Measured throughput = (throughput1 + throughput2) / 2

- NLSP Information option—Delay subfield. This Delay measurement appears in the Link Information option of the Link State packet. The quantity conceptually represents the

one-way electrical propagation delay between the two points. It is calculated based on the Delay Request/Response exchange. The idea behind the calculation is that the measured time interval between sending a full packet and receiving its echo from the other router consists of two parts: (a) the time to clock the bits onto the wire, and (b) the propagation delay. Now, the time to clock bits onto the wire is the reciprocal of the Throughput. Combining these facts,

Echo_time / 2 = Delay + (1 / Throughput).

The division by 2 converts from round-trip to one-way. Now, Delay is the number being sought, Throughput was determined by the previous exchange, and Echo_time is measured by the Delay exchange. (Actually, it is measured twice and the measurements are averaged.) The calculation is made by solving the above equation for Delay:

Delay = Echo_time / 2 – (1 / Throughput)

with the proper conversion of units. This is depicted as follows:

576-byte packet

t1 ----------> ---------->+

t2 <---------- <----------+

echo1 = (t2 – t1) / 2 *(microseconds), the division by 2 is included here*

576-byte packet

t3 ----------> ---------->+

t4 <---------- <----------+

echo2 = (t4 – t3) / 2 *(microseconds), the division by 2 is included here*

Measured delay = (echo1 + echo2) / 2 –
[1,000,000 × 8 × 576 / Throughput]

- RIP/SAP Info Exchange option—WAN Link Delay subfield. This value is calculated as described on page 3-7. It must be present and calculated correctly even when NLSP has been chosen.

On receiving the Information Request, the Slave records the pertinent information and reponds with an Information Response packet, in which its own router name and node identifier are substituted.

After the Slave has received the Information Request and the Master has received the Information Response, NLSP packets and IPX data packets can start being sent over the link.

3.8.2. NLSP Configured Values

A router may permit the two values described in IW2 packets (NLSP Delay and NLSP Throughput) to be configured manually. This can be accomplished either by local action at the router or remotely by network management action.

Whether or not a router is configured manually, it proceeds through the IW2 exchange as specified above, reporting the measured values. After the IW2 exchange, when Link State packets begin flowing, the values in the Link State packets are the manually configured ones. The values appearing in the Information Request/Response are disregarded by routers that have overriding manually configured values.

If either (a) the router at only one end of a link is configured, or (b) the routers at the two ends of a link are configured with different values, the two routers can report different values in their respective Link State packets. This may result in asymmetric routes, but NLSP operates with asymmetric routes. (An asymmetric route is one in which traffic from *A* to *B* traverses different links than traffic from *B* to *A*.) However, asymmetric routes are considered harmful because of troubleshooting difficulties.

In no event does a router attempt to assert its configured values by manipulating the numbers sent in IW2 exchanges.

3.9. Checking and Recovery Features of IW2

Packets received on socket number 0x9004 not having the WIdentifier field set to 0x5741534D are discarded.

If an unknown WPacket Type is received, the receiver should respond with a NAK packet.

If an invalid or unsupported WOption Number is received in a request, the receiver should respond to the packet with a WAccept Option set to "No" for that option.

If a router determines that it cannot support any of the Routing Types included in the Timer Response or Timer Request packet, it issues a disconnect on the underlying datalink connection.

If a Timer Request arrives after the router has sent or received a Timer Response, the exchange reverts to the start. That is, the router reverts to sending its Timer Request packet.

When a router assumes the Slave role, it starts a 60-second timer. If the IW2 exchange has not concluded by the time the timer expires, the router issues a disconnect on the underlying datalink connection.

If a protocol error is detected, the router issues a disconnect on the underlying datalink connection. Protocol errors include (a) packets received out of order, and (b) a Slave receiving a Timer Response. The same happens if no response is detected after the request retries are exhausted. If this happens before the Master role is decided, either router issues the disconnect.

If a router receives a packet with an illegal WPacket Type, or if the options are malformed (for example, they extend beyond the end of the packet), the router sends a NAK packet, and otherwise ignores the offending packet. (There is an exception. If the routing type is RIP, or if it is not yet decided, a NAK is not sent. This is because Reference [All92] does not include the NAK.) When a router receives a NAK packet, it can try something different or it can cause a disconnect.

After a disconnect, the router that initiated the original connection can (at its option) try to establish a new datalink connection. With X.25 switched circuits, this means establishing a new virtual circuit before restarting IW2. With dial-up PPP, this means establishing a new datalink connection before restarting PPP and then IW2. With a dedicated PPP link, this means that both peers restart the PPP protocol from the beginning, then IW2.

For an X.25 PVC, a Frame Relay PVC, or IP Relay, "establishing a new datalink connection" means restarting IW2 from the beginning. In these cases, the Timer Request can be retried indefinitely.

Under certain circumstances—particularly on X.25 or Frame Relay PVCs—it is only possible to detect that the remote router went away when it comes back up again. In this case, one side of the link receives a Timer Request packet when IPX is in a fully connected state. The side receiving the Timer Request must realize that a problem occurred and start tearing down the link to reestablish the connection.

Note: Resolving call collisions is outside the scope of this specification. Call collisions occur when two routers are configured to maintain a circuit to each other on the same physical connection, and the two call requests are issued at the same time. Resolving call collisions is the responsibility of the underlying Datalink layer.

Note: It may happen that two circuits become established connecting the same two endpoints. In fact, this may be desirable to increase throughput in some situations. IW2 does not include any protocol provision to discriminate between wanted and unwanted duplicates. Every circuit is kept, unless clearing it is forced by local manual action or by operation of network management.

3.10. Recalibrating Throughput and Delay

The specification does not currently include a provision for the Throughput or Delay to be recalibrated after completion of the IW2 exchange, while the circuit is active conveying routing traffic and data traffic.

To allow this capability to be added in the future, a router that receives a Throughput Request or Delay Request after completion of the IW2 exchange ignores those packets, without generating an alarm and without disconnecting the circuit.

3.11. IW2 Database

3.11.1. Constant Values

minMTU

The packet size that every router must be prepared to handle: 576 bytes, including IPX header and data but not datalink headers or trailers.

3.11.2. Configured Values

internalNetworkNumber

Every NLSP router and every NetWare v3.x or v4.x server has an IPX network number internal to the system itself. The network number is within the relevant routing area, but is distinct from the network numbers of the physical network segments in a routing area. Any application services operating on the system are attached to the internalNetworkNumber. Network management packets to a router should be sent to the internalNetworkNumber. (In previous specifications, such as Reference [All92], there is the concept of a *primary network number* for systems not necessarily having an internal network number. A *primary network number* can be either an internal one or the network number of a distinguished permanently attached network segment. The internalNetworkNumber idea supersedes that of a primary network number.)

routerName

Every router has a readable textual, symbolic name. It is especially useful for network management purposes to identify systems by a user-defined mnemonic name. The name is 1 to 47 characters, containing uppercase English letters, underscore (_), hyphen (-), and

"at" sign (@). When a router coexists on the same system as a NetWare server, the routerName coincides with the server name.

nlspDelayOverride
A router can provide to the user a mechanism to (optionally) override the NLSP Delay, normally determined empirically by the IW2 protocol exchange.

nlspThroughputOverride
A router can provide to the user a mechanism to (optionally) override the NLSP Throughput, normally determined empirically by the IW2 protocol exchange.

3.11.3. Dynamic Values

ipxWanNetworkNumbers
As a result of IW2 exchanges, each WAN circuit is either (a) determined to run an unnumbered routing protocol, or (b) has an IPX network number assigned. The router keeps track of the network number (or lack thereof) for each circuit. The router can optionally have a pool of network numbers to assign dynamically during IW2 operation to circuits being brought into service.

3.11.4. Dynamic Values per Circuit

Delay
The electrical propagation delay for a signal to flow from one endpoint to the other in milliseconds.

Throughput
The amount of data, in bits per second, that can flow through the circuit if there is no other traffic.

RIP Link Delay (Ticks)
The amount of time it takes a 576-byte packet to reach a location. It is measured in ticks, with 18.21 ticks per second (minimum of one).

3.12. Packet Structures

IW2 employs one kind of packet structure: IPX WAN (page 3-14)

These packets ride in the data portion of IPX packets. The IPX header fields are encoded:

Destination Network	zero
Destination Node	0xFFFFFFFFFFFF
Destination Socket	0x9004
Source Network	zero
Source Node	zero
Source Socket	0x9004
Packet Type	4

IPX WAN packets never exceed minMTU bytes in size, including the IPX header but excluding the datalink headers. All multibyte fields are transmitted with the most significant byte first.

3.12.1. IPX WAN 2

IPX WAN 2

	Number of Bytes
WIdentifier	4
WPacket Type	1
WNode ID	4
WSequence Number	1
WNum Options	1
Variable Length Fields	Variable

WIdentifier: 0x5741534D (ASCII for "WASM") indicates packets of the IPX WAN family of protocols.

WPacket Type: Identifies a specific type of IW2 packet:

0 = "Timer Request"

1 = "Timer Response"

2 = "Information Request"

3 = "Information Response"

4 = "Throughput Request"

5 = "Throughput Response"

6 = "Delay Request"

7 = "Delay Response"

0xFF = "NAK;" the packet content is an exact copy of the received, rejected packet, except that (a) the Packet Type indicates NAK, and (b) the WNode ID is that of the sending router. Only packets with the proper Identifier are replied to with NAK.

Packet Types 0 through 3 are defined in Reference [All92]. Packet types 4 through 7 and 0xFF apply only to IW2.

WNode ID: The internalNetworkNumber of the sending router. There is one exception: if the router cannot assign a network number to the link and if this is a Timer Request packet, this field contains zero and the internalNetworkNumber is conveyed in the Extended Node ID option.

WSequence Number: This field starts with zero for each request packet type and is incremented by one for each retry of that packet type. Each response echoes the sequence number of the request. This allows each response to be matched with the request that provoked it.

WNum Options: Indicates how many options are present in the variable length fields. Options in a packet can occur in any order.

Option

	Number of Bytes
WOption Number	1
WAccept Option	1
WOption Data Len	2
WOption Data	WOption Data Len

Variable Length fields: A series of optional fields, each of which has the following four-part Option form.

WOption Number: Identifies a particular option. Currently defined codes are in the following bullet list.

WAccept Option: This field is used for negotiating options between the two parties:

0 = "No"

1 = "Yes"

3 = "Not Applicable"

WOption Data Len: The length in bytes of the WOptionData field.

WOption Data: Details specific to a particular code.

Currently defined codes, and the corresponding values, are as follows:

- Routing Type

  Code = 0x00 Length = 1

  Value = Type of IPX routing protocol the sender can support (or cannot support if this is a Timer Response and WAccept Option is "No"):

  0 = "RIP"
  1 = "NLSP"
  2 = "Unnumbered RIP"

- RIP/SAP Info Exchange

  Code = 0x01 Length = 54

  Value = The following three subfields:

  RIP/SAP Info Exchange

	Number of Bytes
WAN Link Delay	2
Common Network Number	4
Router Name	48

  - WAN Link Delay: The metric IPX RIP uses for routing; the RIP Delay in Ticks.
  - Common Network Number: For numbered Routing Types, this field contains the IPX network number assigned to the link; for unnumbered Routing Types, the value zero occupies this field.
  - Router Name: A unique textual identifier for the router, flush-left, null-filled.

- NLSP Information

  Code = 0x02 Length = 8

  Value = two subfields, (Delay and Throughput) containing values estimated by IW2 exchanges.

  NLSP Information

	Number of Bytes
Delay	4
Throughput	4

- NLSP Raw Throughput Data

  Code = 0x03 Length = 8

  Value = The following two subfields:

  NLSP Raw Throughput Data

	Number of Bytes
Request Size	4
Delta Time	4

  - Request Size: Size of the Throughput Request packet.
  - Delta time: The time interval (in microseconds) between receipt of equal size Throughput Request Packets.

- Extended Node ID

  Code = 0x04 Length = 4

  Value = Sending router's internalNetworkNumber. This option occurs only in Timer Requests and only when WNode ID is zero.

- Compression

  Code = 0x80 Length = variable

  Value = The first byte indicates a style of compression; the remaining bytes specify parameters pertaining to that style.

  Currently, one compression type is specified:

  Telebit compression (option length = 3 bytes), see Reference [Mat92].
  Byte 0: Contains the value "0" to indicate Telebit compression.

Byte 1: Compression options.
Byte 2: Number of compression slots.

- Pad

  Code = 0xFF Length = variable

  Value = A sequence of bytes to fill the packet to the required size. The sequence starts with zero; each byte increments by one; after 0xFF the sequence wraps to zero and repeats.

  Note: The purpose of this "ramp" is to prevent data-compressing modems from defeating accurate estimations of the link's operational characteristics.

Figure 3-4 summarizes which options are permitted in which request/response packets.

	Timer		Information		Throughput		Delay	
	Request	Response	Request	Response	Request	Response	Request	Response
Routing Type	Ry	=4c	-	-	-	-	-	-
RIP/SAP Information Exchange	-	-	My	=y	-	-	-	-
NLSP Information	-	-	Ny	=y	-	-	-	-
NLSP Raw Throughput Data	-	-	-	-	-	MPy	-	-
Extended Node ID	Ai	-	-	-	-	-	-	-
Compression	Oy	E3c	-	-	-	-	-	-
Pad	Mi	Mi	-	-	MPi	-	MPi	MPi

Figure 3-4: IW2 Option Usage

Legend for Figure 3-4

M Mandatory field.

N Mandatory for NLSP; optional otherwise.

O Optional field; can appear more than once.

= Response fields contain the same WOption Data as the corresponding request fields.

E Response fields correspond one-for-one with request fields; the response can zero out zero or more option bits of the Telebit options field; it must not set bits that were zero in the request.

P Throughput and Delay exchanges occur only with NLSP.

R Routing types appear in order of preference (most preferred first). At least one must be present.

A Appears if and only if the WNode ID field is zero.

3 Each WAccept Option field set to indicate acceptance or rejection; the responder selects **at most** one of the choices offered in the Request.

4 Each WAccept Option field set to indicate acceptance or rejection; the responder selects **exactly** one of the routing types offered in the Request.

y WAccept Option must be "Yes."

c WAccept Option must be "Yes" or "No."

i Value of WAccept Option is immaterial.

– Not permitted.

The following series of packet diagrams is an alternate representation of the option combinations occurring in the various request/reply packets.

Timer Request

IPX Header	See Section 2.
IW2 Header (fixed part)	See page 3-14
Routing Type	Routing types appear in order of preference (most preferred first). At least one must be present. WAccept Option must be "Yes".
Extended Node ID	Appears if and only if the WNode ID field is zero. Value of WAccept Option is immaterial.
Compression	Optional field; can appear more than once. WAccept Option must be "Yes".
Pad	Mandatory field. Value of WAccept Option is immaterial.

Timer Response

IPX Header	See Section 2.
IW2 Header (fixed part)	See page 3-14.
Routing Type	Response fields contain the same WOption Data as the corresponding request fields. Each WAccept Option field set to indicate acceptance or rejection; the responder selects exactly one of the routing types offered in the Request. WAccept Option must be "Yes" or "No".
Compression	Response fields correspond one-for-one with request fields; the response can zero out zero or more option bits of the Telebit options field; it must not set bits which were zero in the request. Each WAccept Option field set to indicate acceptance or rejection; the responder selects at most one of the choices offered in the Request. WAccept Option must be "Yes" or "No".
Pad	Mandatory field. Value of WAccept Option is immaterial.

Information Request

IPX Header	See Section 2.
IW2 Header (fixed part)	See page 3-14.
RIP/SAP InformationExchange	Mandatory field. WAccept Option must be "Yes".
NLSP Information	Mandatory for NLSP; optional otherwise. WAccept Option must be "Yes".

Information Response

IPX Header	See Section 2.
IW2 Header (fixed part)	See page 3-14.
RIP/SAP InformationExchange	Response fields contain the same WOption Data as the corresponding request fields. WAccept Option must be "Yes".
NLSP Information	Response fields contain the same WOption Data as the corresponding request fields. WAccept Option must be "Yes".

Throughput Request

IPX Header	See Section 2.
IW2 Header (fixed part)	See page 3-14.
Pad	Mandatory field. Throughput and Delay exchanges occur only with NLSP. Value of WAccept Option is immaterial.

Throughput Response

IPX Header	See Section 2.
IW2 Header (fixed part)	See page 3-14.
NLSP Raw Throughput Data	Mandatory field. Throughput and Delay exchanges occur only with NLSP. WAccept Option must be "Yes".

Delay Request

IPX Header	See Section 2.
IW2 Header (fixed part)	See page 3-14.
Pad	Mandatory field. Throughput and Delay exchanges occur only with NLSP. Value of WAccept Option is immaterial.

Delay Response

IPX Header	See Section 2.
IW2 Header (fixed part)	See page 3-14.
Pad	Mandatory field. Throughput and Delay exchanges occur only with NLSP. Value of WAccept Option is immaterial.

4. Adjacencies

Before routers can spread topology and status information throughout a routing area, each router determines that information for the links to which it is directly connected and for the routers to which it can communicate directly (without forwarding) over those links.

This section deals with the "local neighborhood discovery" part of the protocol.

NLSP treats two categories of networks differently. In this specification, the terms *LAN* (Local Area Network) and *WAN* (Wide Area Network) are used for these categories. For each medium of interest, it is specified whether NLSP treats it as a LAN or a WAN. In most cases, the categorization is obvious, but not always. In general terms, a LAN is capable of broadcast addressing at the Datalink Layer. IEEE 802.3 and IEEE 802.5 are LANs. Because IEEE 802.6 and FDDI support broadcast, they are treated as LANs even though they are not always "local" networks. In general terms, a WAN either is connection-oriented (requiring a call setup at the Datalink layer) or is a point-to-point network. Examples are PPP (dedicated or dialed links) and X.25.

Section 4.1 deals with WANs, while Section 4.2 deals with LANs.

By exchanging Hello packets on circuits, a router determines the reachability of its neighbors. An *adjacency* is the record that a router keeps about the state of its connectivity with a neighbor, and about the attributes of the neighboring router.

4.1. Maintaining Adjacencies over WAN Networks

Adjacency establishment goes through several stages on a WAN:

- The underlying datalink connection is established. The details depend on the type of medium.
- If a connection is successfully established, the two neighbors use the *IPX WAN version 2* (IW2) protocol to exchange identities and determine certain operational characteristics of the link. Section 3 covers IW2.
- If IW2 concludes successfully, the two neighbors begin to exchange Hello packets and update their Adjacency Databases. Details follow.
- Once the Hello packets establish an adjacency, the routers start exchanging Link State packets. These are defined in Section 5.
- Finally, the routers start forwarding IPX data packets over the link.

4.1.1. Maintaining WAN Links

For each adjacency, the router maintains a state variable that assumes one of three values: "Up," "Initializing," or "Down."

Whenever a circuit is created and the first Hello packet is received, the state initializes to "Down." Whenever there is a Layer 2 datalink reset of the circuit, an adjacencyStateChange (Down) event is generated and the state is set to "Down." The same actions are taken if the link (or the entire router) is reset by local management action or by operation of NLSP (for example, detection of a checksum error in the locally stored Link State database).

See Figure 4-1 for the complete state machine.

If a neighbor is not heard from in the time indicated by the holdingTimer recorded when accepting a WAN Hello packet, the router generates an adjacencyStateChange (Down) event and deletes the adjacency.

4.1.2. Sending WAN Hello Packets

Sending WAN Hello packets allows routers on a circuit to discover each other's identity, decide whether they are in the same routing area, and to determine whether the other router and the link between them remains operational.

A router sends a WAN Hello packet on a WAN link when

a) The circuit is first enabled and the IW2 protocol exchange is complete

b) Whenever the interval nonBcastHelloInt expires

c) The contents of the next Hello to be transmitted would be different from the contents of the previous Hello transmitted by this system and one or more seconds have elapsed since the previous Hello

Hello packets are sent regardless of the value of the local state, as long as the circuit exists.

The WAN Hello packet is constructed as follows:

d) The Local WAN Circuit ID field is set to the localCircuitID value assigned by this router when the circuit is created. This value must be unique among all WAN circuits attached to this router. (It is recorded in the circuitID entry of the Adjacency database.)

e) The WAN State field is set to this router's current state for this link.

f) The MTU Size field must be included. It indicates the largest packet (in bytes, including IPX header but not datalink header) that the router is capable of receiving on the circuit; that is, the localMaxPacketSize.

4.1.3. Receiving WAN Hello Packets

Upon receipt of a WAN Hello Packet, perform the following acceptance tests:

a) The general packet acceptance tests described in Section 2 under "General Processing of Incoming IPX Packets."

b) If the length of the packet, as described in its header, is greater than the buffer in which it was received, discard the packet and log a packetRxSmall event.

c) If the options in the variable part of the packet are ill-formed, or if they extend beyond the end of the packet, discard the packet and log a malformedOption event.

d) Compare each of the area addresses from the Area Addresses field with the set of manualAreaAddresses configured locally. Consider a match to be detected if the Address and the Mask portion are identical. If no match is detected between any pair (that is, if the local and remote system have no area address in common),

   i) If there is an adjacency and it is not in the "Up" state, generate an areaMismatch.

   ii) If the adjacency is in the "Up" state, delete the adjacency and generate an adjacencyStateChange (Down—Area Mismatch) event.

   iii) Otherwise, ignore the packet.

e) If the Circuit Type field in the received Hello packet is other than 1 or 3,

   i) If the adjacency already exists, generate an adjacencyStateChange (Down) event and delete the adjacency.

   ii) If the adjacency does not already exist, discard the packet and generate a wrongSystemType event.

Once the packet has passed the preceding tests, the router invokes a state machine comparing the router's own internal current state for the link with the value of the state field of the received packet. Figure 4-1 illustrates the state machine. Each cell of the table indicates the new state for the router's own state for the link.

	Received packet indicates "Down"	Received packet indicates "Initializing"	Received packet indicates "Up"	Link is reset by local action, or a datalink reset is detected.
Current state is "Down"	"Initializing"	"Initializing"	"Down"	"Down"
Current state is "Initializing"	"Initializing"	"Up"	"Up"	"Down"
Current state is "Up"	"Initializing"	"Up"	"Up"	"Down"

Figure 4-1: WAN Adjacency State Machine

Note: The state machine deals with the case where one end of the point-to-point link wants to bring down the adjacency and must ensure that the remote system is aware of this. This could happen in the following scenarios:

1. A circuit is deleted and re-created.

   Although NLSP sends a Hello with zero holding time when the circuit is deleted, there is no guarantee that this will get through. Thus, the remote system may be unaware that the next Hello that is received is from a new adjacency, so it will not attempt to resynchronize its LSP database.

2. When the router is reset, or an LSP with an incorrect checksum is found in memory.

   In either case, NLSP attempts to reacquire the information from its neighbors. This can be done simply if the remote system can be forced to bring down and then bring up its adjacency to the local system. Adjacency state procedures are independent of the IW2 protocol, to allow for new types of unreliable datalink media.

Figure 4-2 illustrates a typical startup scenario involving routers *A* and *B*.

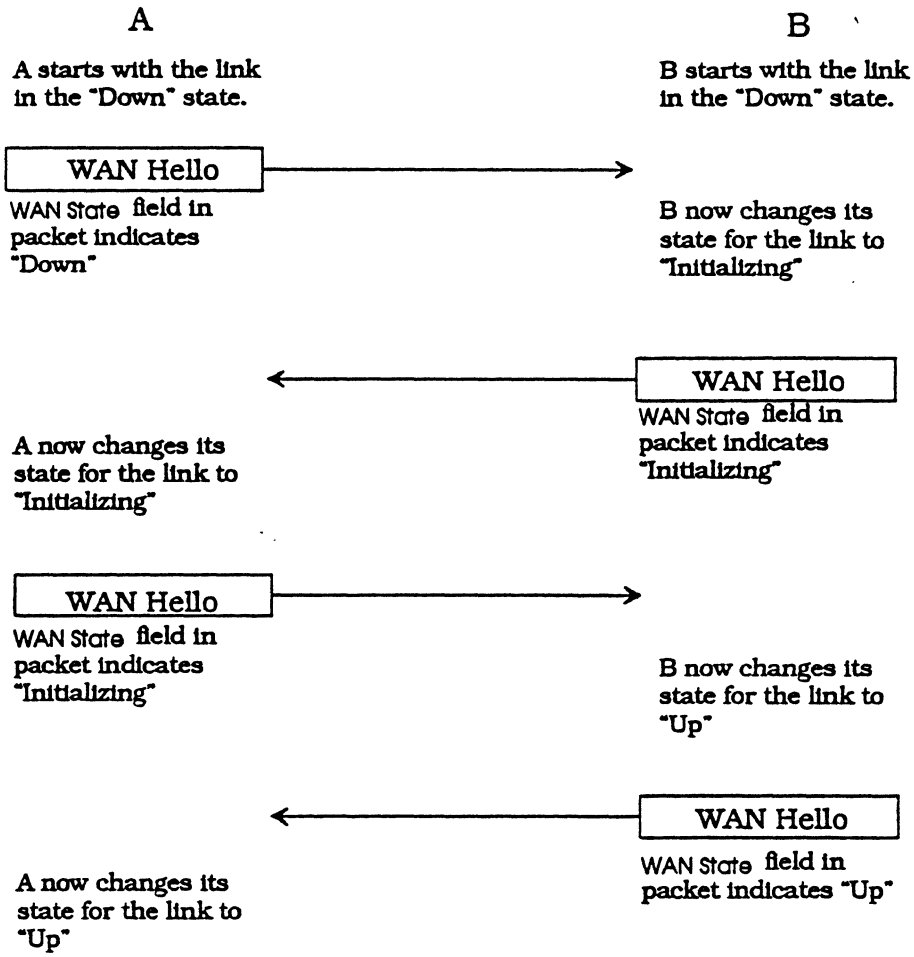


Figure 4-2: New WAN Adjacency

If the state changes from "Up" to a different state, generate an adjacencyStateChange (Down) event.

If it is a new adjacency, that is, if the state machine changes from "Initializing" to "Up," then

a) Generate an adjacencyStateChange (Up) event.

b) Transmit a WAN Hello packet.

c) Compare the Source ID field (call it *x*) of the received packet with the local systemID (call it *y*). The numeric comparison is done by considering each byte as an unsigned integer, and the first byte as most significant.

   i) If $x < y$, set the circuit's circuitID to the concatenation of the local systemID and the localCircuitID of this circuit (as in the Local Circuit ID field of WAN Hello packets sent by this router on this link).

   ii) If $x > y$, set the circuit's circuitID to the concatenation of the Source ID field of the received packet and the Local Circuit ID field of the received packet.

   Note: Do not change the value included in the Local Circuit ID field of Hello packets transmitted by this router. That value is assigned once when the circuit is created and is not subsequently changed.

iii) If $x = y$, the local systemID and local localCircuitID are used.

Note: The two values would be equal if it is some form of loopback circuit.

d) Note that the packet was "accepted."

If the adjacency already exists in the "Up" state and remains "Up," compute a CircuitID as in item c) immediately prior. Then,

a) If the computed circuitID is the same as the one previously recorded for the link, simply note that the packet was "accepted."

b) If the computed circuitID differs from the one previously recorded for the link, generate an adjacencyStateChange (Down) event, and delete the adjacency.

If, in the above steps, it was "noted that the packet was 'accepted,'"

c) Copy the adjacency areaAddressesOfNeighbor entries from the Area Addresses field of the packet.

d) Set the holdingTimer to the value of the Holding Timer field of the packet.

e) Set the neighborSystemId to the value of the Source ID field of the packet.

f) From the MTU Size field in the packet, update the actualMaxPacketSize value for the circuit; it is the lowest value of the MTU size field received on the circuit (this allows certain other packets to exceed minMTU bytes).

4.2. Maintaining Adjacencies over LAN Networks

Multipoint networks with broadcast capabilities include LANs, some Metropolitan Area Networks (MANs), such as Switched Multimegabit Data Service (SMDS), and other emerging technologies. For simplicity, the term LAN refers to broadcast networks of all types.

At various places, packet fields contain the six-byte IEEE MAC address. Sometimes decisions are made based on the six-byte IEEE MAC address reported for an incoming packet by the Datalink layer. When the medium is an IEEE 802.3, 802.4, or 802.5 LAN, the value used is an actual six-byte IEEE address. If a medium has a smaller address (for example, Omninet), it is right-justified and zero-padded. If a medium has a larger datalink address (for example, SMDS), a specification is needed for that medium to resolve values between datalink addresses and IEEE addresses.

4.2.1. Enabling LAN Circuits

When a broadcast circuit is enabled on a router, the router takes the following actions:

a) Begin sending Hello packets (page 4-5)

b) Begin accepting Hello packets from other routers on the LAN (page 4-6)

c) After waiting for a determined time interval, run the Designated Router election process (page 4-9). The interval is 2 × drBcastHelloInt.

4.2.2. Sending LAN Hello Packets

Sending Hello packets allows for routers on the broadcast circuit to discover the identity of other Level 1 routers of the same routing area on that circuit.

The LAN ID field of the Hello packet is the concatenation of a router's System ID and a one-byte Local LAN Circuit ID assigned by that router, to be unique among all the LAN circuits directly

attached to that router. If the router sending a LAN Hello believes that another router is the Designated Router for that LAN, it puts in the LAN ID field the value of the LAN ID observed in the other router's LAN Hello. Otherwise (either it believes itself to be Designated Router or perhaps it has seen no Hello packets yet), the router sending a LAN Hello uses its own systemID in the System ID field and a Local LAN Circuit ID it assigns when filling in the LAN ID field. It uses the circuitID entry to keep track of the assigned values for different circuits.

The priority is a manually configured value. It comes into play with Designated Router election.

The No Multicast bit is set to zero if the router is capable of receiving packets addressed to the selective link-level multicast addresses. Otherwise, the bit is set to one.

The variable-length fields include the following:

a) The area addresses with which the router has been configured (manualAreaAddresses).

b) The LAN addresses of the adjacencies on this circuit. The adjacency list includes only Level 1 routers within the same area. Only those adjacencies in state "Initializing" or "Up" are included. The states are described in section 4.2.5.

c) The MTU Size field must be included. It indicates the largest packet (in bytes, including IPX header but not datalink header) that the router can receive on the circuit.

A router transmits a LAN Level 1 Hello packet immediately when any circuit has been enabled. Hello packets are transmitted to the multidestination address allL1Routers. Routers listen on this address for arriving Hello packets.

A router also transmits a LAN Level 1 Hello packet when at least one second has elapsed since the last such transmission on this circuit by this router, and

d) bcastHelloInt seconds have elapsed[1] since the last periodic LAN Level 1 Hello transmission:

> The Holding Time is set to holdingTimeMultiplier × bcastHelloInt. For a Designated Router, the value of drBcastHelloInt is used instead of bcastHelloInt[1]. The Holding Time for this packet is therefore set to holdingTimeMultiplier × drBcastHelloInt seconds. This permits a failed Designated Router to be detected more rapidly.

Or

e) The contents of the next Hello to be transmitted would differ from the contents of the previous Hello transmitted by this system.

Or

f) The system has determined that it is to become or resign as Level 1 Designated Router.

To minimize the possibility of the Hello transmissions of all routers on the LAN becoming synchronized, the hello timer is only reset when a Hello is transmitted as a result of timer expiration, or on becoming or resigning as Designated Router. It is not reset if the Designated Router changes from one remote system to another.

---

[1] Jitter is applied, as described in Section 2.

4.2.3. Receiving LAN Hello Packets

Upon receipt of a LAN Hello Packet, perform the following acceptance tests:

a) The general packet acceptance tests described in Section 2 under "General Processing of Incoming IPX Packets."

b) If the source network number in the IPX header differs from this router's view of the circuit's network number, discard the packet and log a mismatchedNetworkNumber event.

c) If the source node number in the IPX header differs from the six-byte IEEE MAC address reported by the datalink layer, discard the packet and log a mismatchedNodeAddress event.

d) If the length of the packet as described in its header is greater than the buffer in which it was received, discard the packet and log a packetRxSmall event.

e) If the options in the variable part of the packet are ill-formed, or if they extend beyond the end of the packet, discard the packet and log a malformedOption event.

f) If the Circuit Type field is other than 1 or 3, discard the packet.

g) Compare each of the area addresses from the Area Addresses field with the set of manualAreaAddresses configured locally. Consider a match to be detected if the Address and the Mask portions are identical. If a match is not found between any pair (that is, if the local and remote system have no area in common), reject the adjacency and generate an areaMismatch event.

If the preceding tests succeed, the router accepts the adjacency and sets the neighborSystemType to "L1 Router."

Use the MTU size field in the packet to update the actualMaxPacketSize value for the circuit; it is the lowest value of the MTU size field received on the circuit.

If a Level 1 LAN Hello is received with the No Multicast bit set to one, the router sends future NLSP packets for that network segment to the broadcast address, where it would otherwise use the multicast address. There is no provision for returning to the multicast address.

4.2.4. Maintenance of Existing LAN Adjacencies

When a Level 1 LAN Hello is received from a router for which there is already an adjacency with

a) The adjacency neighborNICAddress equal to the six-byte IEEE MAC source address of the packet

b) The adjacency neighborSystemID equal to the Source ID field of the packet

c) The adjacency neighborSystemType equal to "L1 Router"

the router updates the adjacency's holdingTimer, priorityOfNeighbor, and areaAddressofNeighbor according to the values in the packet.

4.2.5. Detecting New LAN Adjacencies and Updating Adjacency States

When

a) A LAN Level 1 Hello is received (from router "*R*")

b) There is no adjacency for which

   i) The adjacency neighborNICAddress is equal to the six-byte IEEE MAC source address of the packet

   ii) The adjacency neighborSystemID is equal to the Source ID field of the packet

   iii) The adjacency neighborSystemType is equal to "L1 Router"

the router creates a new adjacency. However, if the Adjacency database has insufficient space to allow creating a new adjacency, the router instead merely ignores the Hello packet.

In the new adjacency, the router sets the following:

c) The neighborSystemType to "L1 Router"

d) The holdingTimer, neighborPriority, and areaAddressOfNeighbor according to the values in the packet

e) The neighborNICAddress equal to the six-byte IEEE MAC source address of the packet

The router sets the state of the adjacency to "Initializing" until it is known that the communication between this system and *R* (the source of the Hello packet) is two-way. However, R is included in future LAN Level 1 Hello packets transmitted by this system.

When R reports this router's LAN Address in its LAN Level 1 Hello packet, this router

f) Sets the adjacency's state to "Up"

g) Generates an "adjacencyStateChange (Up)" event

Figure 4-3 illustrates a typical scenario. In the figure, router B is already running and router A comes on line.

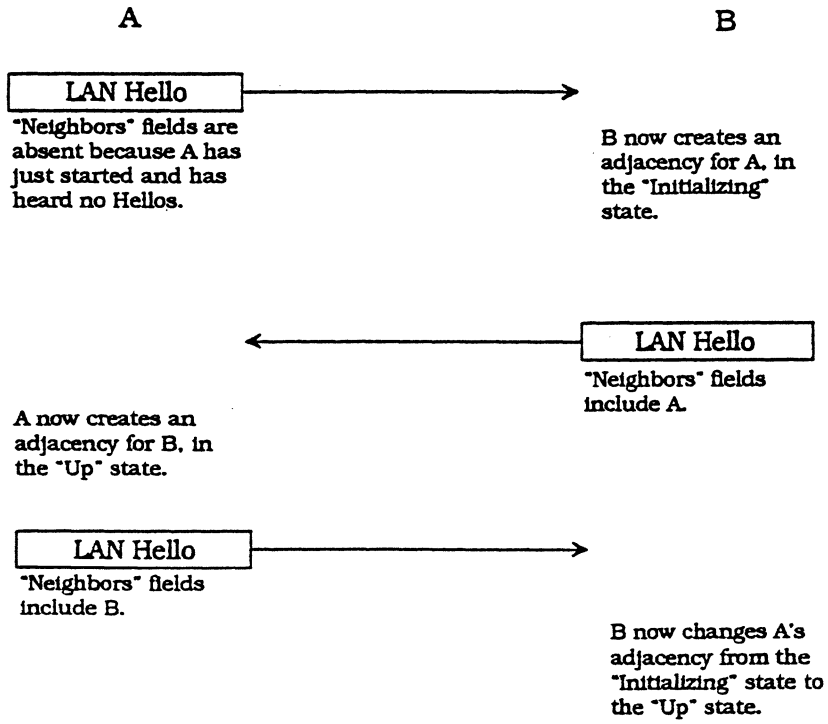


Figure 4-3: New LAN Adjacency

4.2.6. Maintaining LAN Adjacencies

The router keeps a separate holding timer (adjacency holdingTimer) for each Level 1 adjacency. The value of holdingTimer is initialized to the holding time as reported in the Holding Time field of the LAN Level 1 Hello packet. If a neighbor is not heard from in that time, the router

a) Purges it from the Adjacency database

b) Generates an "adjacencyStateChange (Down)" event

If a LAN Level 1 Hello packet is received from neighbor *N*, and this system's LAN Address is no longer in *N*'s Hello packet, this router

c) Sets the adjacency's adjacencyState to "Initializing"

d) Generates an "adjacencyStateChange (Down)" event

4.2.7. Designated Router Election

The Level 1 Designated Router is the highest priority Level 1 router on a LAN. In case of a tie, the numerically highest MAC address wins, from among the routers having highest priority. The numeric comparison is done by considering each byte as an unsigned integer, and the first byte as most significant.

When a LAN circuit changes state to "Up," a system waits 2 x drBcastHelloInt before electing a Designated Router on that circuit. However, if in this time an adjacency is formed to a system on that LAN, and that system reports that it is the Designated Router itself, the Designated Router election process is run at once.

The priority to become Designated Router on a LAN is configurable, per router per LAN attachment. The default is 44. If a system elects itself Designated Router, it raises its priority by 20. The purpose is to avoid unnecessary area-wide traffic in LSPs as Designated Router roles change for a LAN with routers that can go up and down from time to time.

The set of routers considered as candidates include the local router, together with routers from which LAN Level 1 Hello packets have been received and to which Level 1 adjacencies exist in adjacencyState "Up."

A router runs the election process whenever a LAN Level 1 Hello packet is received or transmitted. (For these purposes, transmission of one's own Hello is equivalent to receiving it.) If there has been no change to the information on which the election is performed since the last time it was run, the previous result can be assumed. The relevant information is

a) The set of router adjacency states

b) The set of router priorities (including this system's)

The LAN ID field in the LAN Level 1 Hello packets transmitted by this system is set to the value of the LAN ID field reported in the LAN Level 1 Hello packet received from the system that this system considers to be the Designated Router. The value is also used as the pseudonode ID, to enable LSPs to be issued for this system claiming connectivity to the pseudonode.

If this system determines by the election process that it is itself the Designated Router, it sets the LAN ID field to be the concatenation of the system's own ID and the locally assigned one-byte Local LAN Circuit ID.

If the system determines that it is the Designated Router for a LAN circuit, it should allo memory and other resources sufficient to generate the pseudonode LSPs for the circuit. sufficient resources are unavailable, the router takes the following steps:

a) Enter the LSP database overload state, as described in Section 5.

b) As described in Section 5, part of entering the LSP database overload state is lowering router's priority for the circuit in question. This likely causes the overloaded router to no longer be elected Designated Router for the circuit. However, it is still possible for it to be elected the Designated Router.

4.3. Adjacency Database

4.3.1. Constant Values

allL1Routers
: The LAN datalink address used to send multicast packets. The value is specific to a particular medium. A list is included in Section 2.

4.3.2. Configured Values of the Local System

manualAreaAddresses
: One to three IPX routing area addresses. Each is a 32-bit IPX network number, paired with a 32-bit mask of leading "one" bits. The mask indicates how many bits of the network number identify the area. All the manualAreaAddresses of the routers in a routing area are synonymous in identifying the area. The default area address is (area = 0, mask = 0), meaning that all networks in the reachable internetwork are in one area.

systemID
: A six-byte value assigned to a router. It must be unique in the entire internetwork.

nonBcastHelloInt
: The interval, in seconds, between NLSP Hello packets sent on WAN circuits. Default is 5 seconds.

bcastHelloInt
: The interval, in seconds, between NLSP Hello packets sent on a LAN circuit by systems other than the circuit's Designated Router. Default is 10 seconds.

drBcastHelloInt
: The interval, in seconds, between NLSP Hello packets sent on a LAN circuit by the circuit's Designated Router. Default is 3 seconds.

holdingTimeMultiplier
: The multiplier used to specify the holding time for NLSP neighbor entries as a function of the NLSP Hello interval. Default is 5.

4.3.3. Configured Values per Circuit

localMaxPacketSize
: The maximum size, in bytes, that the system supports locally on this circuit. The IPX header is included, but not the datalink header or trailer.

localHoldingTimer
This router's holding time, in seconds, sent in an NLSP Hello packet.

priority
The priority of this router to become the NLSP LAN Level 1 Designated Router on a broadcast circuit. The default is 44. If a system elects itself Designated Router, it raises its priority by 20.

4.3.4. Dynamic Values per Circuit

localCircuitID
A local one-byte identifier assigned to a circuit by the local system. The value across all directly attached WANs must be unique, and the value across all directly attached LANs must be unique.

circuitID
A seven-byte value identifying a circuit. It is derived by combining a router's systemID with that router's localCircuitID for the circuit.

actualMaxPacketSize
The maximum size, in bytes, that are sent and received on this circuit. The IPX header is included, but not the datalink header or trailer.

noMulticast
A bit recording whether a Hello packet has been received indicating that another router on a LAN circuit is incapable of receiving multicast packets. If set, packets are sent by broadcast instead of multicast.

4.3.5. Dynamic Values per Adjacency

state
A variable with three values: "Up," "Initializing," and "Down." There is a WAN state machine for the steps in establishing an adjacency, and a different state machine for LANs.

holdingTimer
The initial holding time, in seconds, for this NLSP neighbor entry as specified in the NLSP Hello packet.

neighborSystemID
The six-byte systemID of the neighboring router.

neighborNICAddress
The six-byte IEEE MAC address of the neighbor's network interface point of attachment to a LAN circuit shared with this router.

neighborSystemType
The types are "L1 router" and "L1 and L2 router." This version of the NLSP specification covers L1 operation. Provision for L2 is to ensure forward compatibility.

areaAddressesOfNeighbor
The neighbor's manual area addresses.

neighborPriority
The priority of the neighboring NLSP router for becoming the LAN Level 1 Designated Router.

4.3.6. NLSP Events

adjacencyStateChange
Indicates that the state variable has changed in value for a circuit.

packetRxSmall
A truncated packet was received, containing only a portion of the information sent.

mismatchedNodeAddress
The source node number is incorrect in a received packet.

malformedOption
A syntax check failed for a received packet.

areaMismatch
Another router reported a set of area addresses disjoint from manualAreaAddresses.

wrongSystemType
Attempt by another router to communicate outside the realm of the Level 1 routing.

mismatchedNetworkNumber
A neighbor disagrees with this router on the IPX network number assigned to a circuit.

4.4. Packet Structures

The following packet types are relevant to maintaining the Adjacency Database.

- WAN Hello (page 4-14)
- LAN Level 1 Hello (page 4-16)

These packets ride in the data portion of IPX packets. The IPX header fields are encoded as follows:

	WAN Hello	LAN Level 1 Hello
Destination Network	zero	zero
Destination Node	0xFFFFFFFFFFFF	0xFFFFFFFFFFFF
Destination Socket	0x9001	0x9001
Source Network	internalNetworkNumber of the router sending the packet	Network number of the LAN on which the packet is being transmitted
Source Node	Node number of the router sending the packet on its internalNetworkNumber (0x000000000001)	IEEE MAC address of the LAN interface through which the packet is being transmitted
Source Socket	0x9001	0x9001
Packet Type	0	0

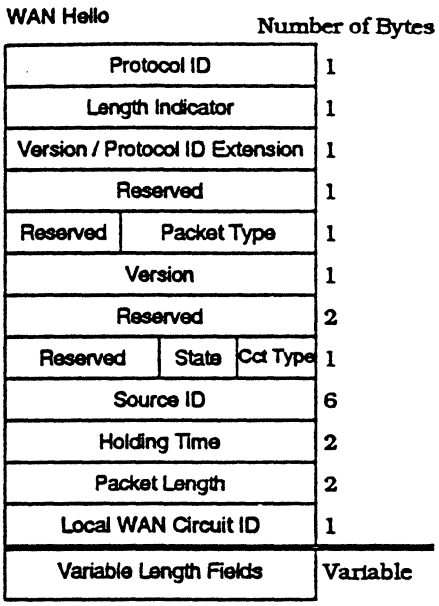
The maximum size of Hello packets on any circuit is determined by the media type and the buffer sizes used by neighbors on that circuit. Each Hello packet includes the Local MTU option, containing the sender's localMaxPacketSize for the circuit. This is the way routers communicate the packet size they are able to receive on that circuit. Every router calculates the minimal value of Local MTU for active adjacencies on each circuit. It then takes the smaller of that value and its own localMaxPacketSize for that circuit. The result is the actualMaxPacketSize value for that circuit. Hello packets can be as large as that value, but no larger.

Note: In practice, the actualMaxPacketSize value for a circuit is the circuit's maximum datalink user data payload size in most situations. There are, however, certain situations where the value is smaller. For example:

a) A bridged LAN that includes LAN segments of different media

b) A circuit on which one router uses a smaller buffer size than the circuit's maximally supported size, owing to resource limitations or misconfiguration

4.4.1. WAN Hello



Protocol ID: 0x83, identifies the NLSP routing layer.

Length indicator: The number of bytes in the fixed portion of the header (up to and including the LAN ID field).

Version/Protocol ID Extension: 1.

Reserved: 0, ignored on receipt.

Reserved (3 bits): 0, ignored on receipt.

Packet Type (5 bits): 17.

Version: 1.

Reserved: 0, ignored on receipt.

Reserved (4 bits): 0, ignored on receipt.

State (2 bits): The sending router's state associated with this link:

0 = "Up"
1 = "Initializing"
2 = "Down"

Circuit Type, abbreviated Cct Type in the diagram (2 bits):

0 = Reserved value, ignore entire packet.

1 = Level 1 routing only.

2 = Level 2 routing only (sender uses this link for Level 2 routing only).

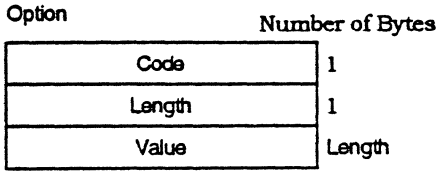
3 = Both Level 1 and Level 2 (sender is a Level 2 router and uses this link for Level 1 and Level 2 traffic).

Source ID: The system ID of the sending router.

Holding Time: Holding Timer, in seconds, to be used for the sending router.

Packet Length: The entire length of this packet, in bytes, including the NLSP header.

Local WAN Circuit ID: A unique identifier assigned to this circuit when it is created by this router.



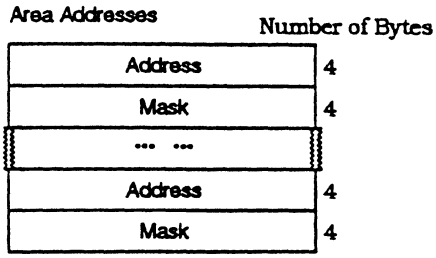
Variable Length fields: A series of optional fields, each of which has the following three-part code/length/value Option form.

Currently defined codes, and the corresponding values, are as follows:

- Area Addresses: The set of Manual Area Addresses of the sending router. This field must be present.

  Code = 0xC0. Length = Total length of the value field, in bytes; either 8, 16, or 24.

  Value = Up to three area addresses. Each area address consists of a four-byte network number, followed by a four-byte address mask. The mask contains from zero to 32 (inclusive) most-significant "one" bits to indicate which bits of the network number make up the address prefix identifying the routing area. The remaining bits are "zero." The bit-wise AND of an Address and Mask pair must be equal to Address. For example, it would be a mistake to send Address = 0x84300000 paired with Mask = 0xFF000000.

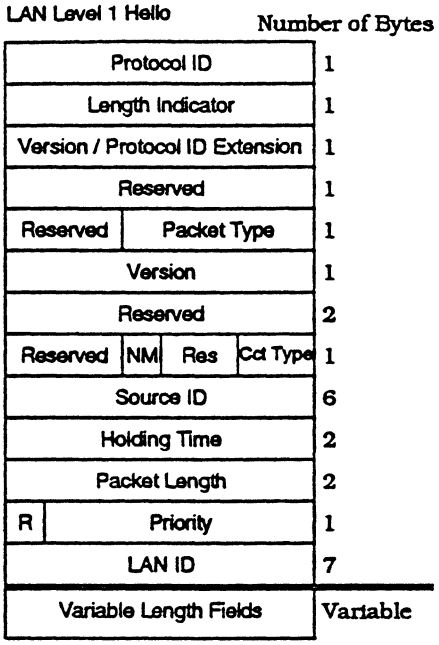


- Local MTU: indicates how large a packet the sender can transmit. This field must be present.

  Code = 0xC5. Length = 4.

  Value = The maximum number of bytes that the sending router can transmit on this interface. The count includes the IPX header, but not the datalink header.

4.4.2. LAN Level 1 Hello



Protocol ID: 0x83, identifies the NLSP routing layer.

Length indicator: The number of bytes in the fixed portion of the header (up to and including the LAN ID field.

Version/Protocol ID Extension: 1, ignored on receipt.

Reserved: 0, ignored on receipt.

Reserved (3 bits): 0, ignored on receipt.

Packet Type (5 bits): 15.

Version: 1.

Reserved: 0, ignored on receipt.

Reserved (3 bits): 0, ignored on receipt.

No Multicast, abbreviated NM in the diagram (1 bit): When set to one, indicates that the sender of the packet cannot receive traffic addressed to a multicast address; future packets on this LAN (which would otherwise be transmitted multicast) must be sent to the broadcast address.

Res (2 bits): 0, ignored on receipt.

Circuit Type, abbreviated Cct Type in the diagram (2 bits):

0 = Reserved value, ignore entire packet.

1 = Level 1 routing only.

2 = Level 2 routing only (sender uses this link for Level 2 routing only).

3 = Both Level 1 and Level 2 (sender is a Level 2 router and uses this link for Level 1 and Level 2 traffic).

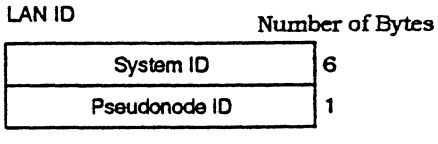
Source ID: The system ID of the sending router.

Holding Time: The Holding Timer in seconds to be used for the sending router.

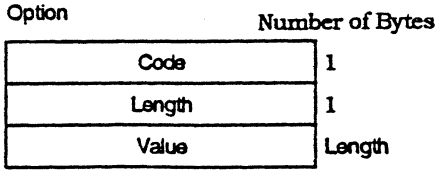
Packet Length: The entire length of this packet, in bytes, including the NLSP header.

R (1 bit): 0, ignored on receipt.

Priority (7 bits): The priority for being the LAN Level 1 Designated Router. Higher numbers have higher priority. An unsigned integer.



LAN ID: A field composed of the system ID (6 bytes) of the LAN Level 1 Designated Router, followed by a low-order Pseudonode ID byte assigned by that Designated Router. This field is copied from the Designated Router's Hello packet.



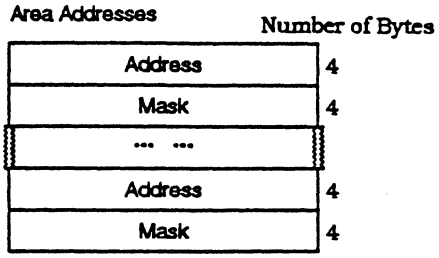
Variable Length fields: A series of optional fields, each of which has the following three-part code/length/value Option form:

Currently defined codes, and the corresponding values, are as follows:

- Area Addresses: The set of Manual Area Addresses of the sending router. This field must be present.

  Code = 0xC0. Length = Total length of the value field, in bytes; either 8, 16, or 24.

  Value = Up to three area addresses. Each area address consists of a four-byte network number, followed by a four-byte address mask. The mask contains from zero to 32 (inclusive) most-significant "one" bits to indicate which bits of the network number make up the address prefix identifying the routing area. The remaining bits are "zero." The bit-wise AND of an Address and Mask pair must be equal to Address; for example, it would be a mistake to send Address = 0x84300000 paired with Mask = 0xFF000000.

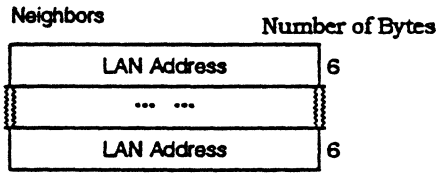


LAN Level 1 Hello packet structure defined codes and values:

- Neighbors: The set of routers on this LAN to which adjacencies of type "Level 1 Router" exist in state "Up" or "Initializing"; that is, those from which Level 1 Hello packets have been heard. This field can occur more than once

  Code = 6. Length = A multiple of six.

  Value = Each six-byte field is the IEEE 802 MAC address of the neighbor router's point of attachment to this LAN segment



- Local MTU: Indicates how large a packet the sender can transmit. This field must be present.

  Code = 0xC5. Length = 4.

  Value = The maximum number of bytes that the sending router can transmit on this interface. The count includes the IPX header, but not the datalink header.

5. Link State

Each router extracts certain information from the Adjacency database, adds locally derived information, and constructs a Link State Packet (LSP) that describes its immediate neighbors. The totality of these LSPs constructed by all the routers in the routing area make up the Link State database for the area. NLSP aims for each router to maintain a copy of the Link State database and to keep these copies synchronized with each other.

5.1. Overview of the Protocol

The Link State database is synchronized by propagating LSPs reliably from a router that observes a topology state change throughout the routing area. The format of an LSP is described on page 5-28.

If an LSP is too large to fit in lspBufferSize bytes, the router originating it splits the information into several LSPs, each identified by an LSP Number (think of this as a fragment number). Each of these LSPs propagates from the originator independently of the others. When there is a topology change, only the LSP (that is, the fragment) affected by the change is propagated. Each has an independent sequence number (incremented when the LSP content changes) to allow a recipient router to discriminate between new and outdated information.

Each fragment is complete and self-contained. From here onward the term *LSP* refers to one of these fragments, because this is the basic unit of information propagated in the routing area. When referring to the set of fragments originated by a router, the term *LSP series* is used.

There are two parts to the propagation method: flooding and receipt confirmation.

Flooding is instigated when a router detects a topology state change. The router constructs a new LSP and transmits it to each of its neighbors. On a WAN, this is a directed packet; on a LAN, it is multicast. Upon receiving an LSP, a router first decides whether it is newer; that is, whether the sequence number is higher than the one in its current copy of the database. (Other factors enter into the newness decision; complete details follow later.) If it is new, the router retransmits the LSP to all its neighbors except on the circuit over which the LSP was received.

Receipt confirmation is different for LANs and WANs. A router receiving an LSP on a WAN replies with a Partial Sequence Number Packet (PSNP). This serves as an acknowledgment for the LSP (or LSPs) and sequence numbers identified in the PSNP.

On a LAN, there is no explicit acknowledgment. A router multicasts the new LSP and (if all goes well) the other routers on the LAN receive the LSP and absorb it into their databases. To allow for the possibility that the LSP was not successfully received by all the LAN's routers, the Designated Router periodically multicasts a Complete Sequence Number Packet (CSNP) containing all the LSP identifiers and sequence numbers it has in its database for the entire routing area. (It does not send the LSPs; it sends just enough information to allow another router to detect whether it is out of step with the

Designated Router.) If all has gone well, the other routers on the LAN verify that they have the same set of LSPs and sequence numbers indicated in the CSNP.

If there is a discrepancy, what happens next depends on which router has the higher sequence number (that is, newer information) for the LSP: the Designated Router or the CSNP-recipient router. If the Designated Router has newer information, the CSNP-recipient transmits a PSNP, which acts as a request for the missing LSPs. The Designated Router receiving the PSNP replies by multicasting each requested LSP. If the CSNP-recipient has newer information, it multicasts the up-to-date LSP on the LAN, and the Designated Router updates its database (as do the other routers on the LAN).

The remainder of this chapter describes the algorithms in more detail. A number of timing considerations govern when and how often to send certain packets. To accommodate these considerations, it is convenient to describe operation of the protocol in terms of two flags. For each LSP and for each circuit over which routing messages are exchanged, a router maintains two local flags:

- *Send Routing Message* (SRMflag): When set, this flag indicates that LSP should be sent on that circuit. On a LAN, SRMflag is cleared as soon as the LSP is transmitted. On a WAN, it is cleared only on receipt of a PSNP (acknowledging the LSP) or an LSP (indicating that the neighbor was already at least as current as this router). A router regularly scans the Link State database for LSPs for which

  a) SRMflags are set

  b) The LSP was propagated no more recently than minimumLSPTransmissionInterval

  When such an LSP is found, the router transmits it on all circuits having SRMflags set, and updates the lastSent time for it.

- *Send Sequence Numbers* (SSNflag): When set, this flag indicates that information about that LSP should be included in a PSNP sent on that circuit. When the PSNP has been sent, the flag is cleared.

5.2. Generating and Checking the LSP Checksum

The checksum allows detection of transmission errors and memory corruption, to prevent both (a) the use of incorrect routing information, and (b) propagation of incorrect routing information to other routers. The router originating an LSP computes the checksum when the LSP is generated. The checksum is never modified by any other system as the LSP propagates.

The checksum is computed over all fields in the LSP after the Remaining Lifetime field. This field (and those before it) are excluded so that the LSP can be aged by systems without recomputing the checksum.

This specification makes use of the checksum function defined in ISO 8473 (Reference [ISO88]). The following subsections define the computation.

5.2.1. Symbols and Conventions

C0 and C1 are variables used in the computation.

i is the number (that is, the position) of a byte in the data block having the checksum performed; the first byte has i=1.

B[i] is the value of byte i in the block being checked.

L is the length of the data block being checked, in bytes.

X is the first byte of the calculated checksum.

Y is the second byte of the calculated checksum.

Addition is performed in one of the following modes:

- Modulo 255 arithmetic
- Eight-bit one's complement arithmetic in which, if any of the variables has the value "minus zero" (that is, 255), it is regarded as though it had the value plus zero (that is, 0)

5.2.2. Generating a Checksum

To perform a checksum on an LSP being generated, proceed as follows:

```
Construct the complete packet, with the value of the Checksum
field set to zero.

C0 ← C1 ← 0.

For i incrementing by 1 sequentially from 1 to L:
        BEGIN    -- Loop
        C0 ← C0 + B[i]
        C1 ← C1 + C0
        END      -- Loop

Calculate:
        X ← ( L - 8 ) × C0 - C1    (mod 255)
        Y ← ( L - 7 ) × ( - C0 ) + C1    (mod 255)

If X = 0 then X ← 255

If Y = 0 then Y ← 255

Place the values of X and Y in the Checksum field of the packet.
```

There is one exception: if the Remaining Lifetime field of the LSP is zero, the correct value of the checksum is zero.

5.2.3. Checking a Checksum

To verify the checksum of an arriving LSP or an LSP stored in memory, proceed as follows:

```
If either byte (or both bytes) of the Checksum field is zero,
     then the checksum is incorrect; do not proceed further.
C0 ← C1 ← 0
For i incrementing by 1 sequentially from 1 to L:
       BEGIN    -- Loop
       C0 ← C0 + B[i]
       C1 ← C1 + C0
       END      -- Loop
If C0 = C1 = 0 (mod 255) when all bytes have been processed,
     then the checksum calculation has succeeded.
Otherwise, the checksum calculation has failed.
```

There is one exception: if the Remaining Lifetime field of the LSP is zero, the correct value of the checksum is zero.

5.2.4. Partial Precomputation

The portion of the LSP covered by the checksum starts with the Source ID portion of the LSP ID field.

Partial Precomputation is an extra fault-tolerance method that a router can include as an implementation option. It is described in the next paragraph. It protects against a failure in which a router's memory of its own systemID becomes corrupted. Even if very rare, such a failure could cause wide-spread disruption if not detected. Partial Precomputation contains the failure by causing LSPs generated by the failing router to have checksum errors.

An originating router precomputes the first few steps of the checksum and saves the result for use with each LSP. Specifically, it precomputes the checksum of the systemID portion of the Source ID when the router is activated or whenever the systemID changes. When performing a checksum on a packet, then, it initializes the variables `C0` and `C1` to the saved values, and resumes the computation after the systemID portion of the Source ID field; that is, starting with the Pseudonode ID portion of the LSP ID.

5.3. The Need for Multiple LSPs

Packets for LSPs are limited in size to lspBufferSize. It may not be possible to include all the information about one's neighbors in one packet. In such cases, a router uses multiple LSPs to convey this information. (Even if the information can fit into one packet, a router may optionally use more than one LSP to convey it.) Each LSP of the

set carries the same Source ID, but sets its LSP Number individually. Each of the several LSPs is propagated independently in the routing area, allowing pipelining of activity. On the other hand, the Decision Process that builds the Forwarding database recognizes that they constitute a particular router's LSP series because they have the same Source ID.

When some event requires changing the LSP information for a router, that router reissues the one (or more) LSPs that have different contents. It is not required to reissue unchanged LSPs.

The first LSP of a series has LSP Number zero. It is treated in a special way.

a) The following fields are meaningful to the Decision Process only when they are present in the LSP number zero:

   i) The setting of the LSP Database Overload bit.

   ii) The value of the IS Type field.

   iii) The Attached Flag.

   iv) The Area Addresses option field. This option is present only in LSP zero.

   v) The Management Information option field. This option is present only in LSP zero, and must precede any Link Information options.

      Note: There must be exactly one Area Addresses option and one Management Information option in each LSP number zero. If a router receives one of these options in an LSP with a nonzero number, or if a router receives a second instance within LSP zero, these additional options are ignored.

b) When any of the preceding items are changed, a router reissues LSP zero to inform other routers of the change. Other LSPs need not be reissued.

Once a particular Option field has been assigned to a particular LSP Number, it is desirable (but not required) that it not be moved to a different LSP number. This is because moving an Option field from one LSP to another can cause temporary loss of connectivity to the entity (for example, a neighbor, service, or external route) represented by the field. This can occur if the new version of the LSP that originally contained the field (but which does not now contain it) is propagated before the new version of the other LSP (which now contains the field).

Note: In most situations, it improves performance to ensure that the number of LSPs generated by a router is close to the optimal number that would be required if the LSPs were densely packed with Link Information options. This can be accomplished by reusing space in LSPs with a lower LSP Number for new adjacencies. Fewer LSPs means consuming less processing capacity, less network bandwidth, and less router memory. However, there are situations in which a link alternates often between active and inactive. Putting such a link in a small LSP consumes less network bandwidth when flooding the LSP to other routers each time it changes. Consequently, implementers are faced with a trade-off on this issue.

If an Option field moves from one LSP to another, the SRMflags of the two updated (or new) LSPs must be set as an atomic action.

Note: If they are not set atomically, a race condition exists, in which one of the two LSPs can propagate quickly, while the other waits for an entire propagation cycle. If this occurs, entities are erroneously eliminated from the topology and routing can become unstable for a period of time potentially as large as maximumLSPGenerationInterval.

5.4. Determining Which of Two LSPs is "Newer"

At certain points in the algorithms described later, a router has two copies of an LSP with the same LSP ID and must compare the two to determine which is newer than the other, or whether the two are the same. At certain other times, the router does not have an LSP in hand, but has information about a neighbor's copy of an LSP must make the comparison with its own copy of the LSP. (The information arrives in sequence number packets from the neighbor.) These are the steps in making the comparisons:

a) If the two have different sequence numbers, the higher sequence number is considered newer.

b) If

   i) the sequence numbers are the same and

   ii) exactly one of the two has zero lifetime,

   the one with zero lifetime is considered newer.

c) If

   i) the sequence numbers are the same and

   ii) the remaining lifetimes are either both zero or both nonzero and

   iii) the checksums match,

   the two are considered the same.

d) If, on the other hand,

   i) the sequence numbers are the same and

   ii) the remaining lifetimes are either both zero or both nonzero **but**

   iii) the checksums **do not** match,

   the LSPs are considered different, and the one with the higher numerically valued checksum (treated as a four-byte unsigned number) is considered to be newer. This is the *Preference of Checksums* rule.

   Note: If two routers are misconfigured with the same systemID, and they both generate LSPs with the same sequence number, this provision at least ensures that the LSPs are treated in a consistent way by all routers.

5.5. Pseudonodes and Designated Routers

There are two types of pseudonodes: *LAN Pseudonodes* and *WAN Pseudonodes.*

LAN pseudonodes have been discussed already. They streamline NLSP operation when many routers are attached to the same LAN. There is an election process by which one of the routers is selected to be the Designated Router representing the LAN pseudonode for LSP generation.

Every NLSP router on a LAN retains in its local memory all the information needed for it to become the Designated Router, should the need arise. This local information is referred to as a *Potential Pseudonode.*

WAN pseudonodes serve a different purpose. Section 3 discusses methods by which an NLSP router can establish communication with a RIP router over a WAN (virtual) circuit. For each such circuit, a pseudonode represents the RIP routes and SAP services discovered over that circuit. The NLSP router on the circuit is the WAN Designated Router representing the WAN pseudonode. There is no election process.

Section 3 actually specifies two ways of using RIP: numbered and unnumbered. The first assigns an IPX network number to the circuit; the second does not. Depending on this choice, there is either a *Numbered WAN Pseudonode* for the circuit or an *Unnumbered WAN Pseudonode.*

Figure 5-1 illustrates the three kinds of pseudonodes.

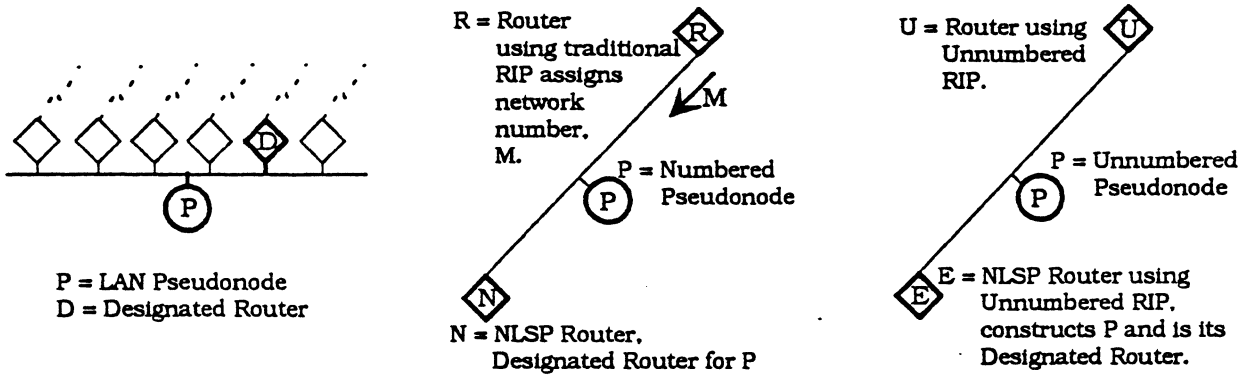


Figure 5-1: Pseudonodes

As in the numbered pseudonode case, Unnumbered RIP uses RIP and SAP on the WAN, not NLSP. As far as the traffic on the WAN is concerned, Unnumbered RIP operation really has no connection with NLSP. If—like node "E" in the figure—either or both ends of the (virtual) circuit are NLSP routers, that router constructs an unnumbered pseudonode (P) representing the WAN to the NLSP routing area (not shown in the diagram) to which it is attached.

5.6. Aging Out an LSP and Purging Superseded LSPs

When the source generates an LSP, it sets the Remaining Lifetime field to maxAge.

When a router holds an LSP before successfully transmitting it to a neighbor, it decrements the Remaining Lifetime field by the holding time. Before transmitting an LSP to a neighbor, a router decrements the Remaining Lifetime field by at least one.

When the Remaining Lifetime of an LSP in memory becomes zero, the router takes the following actions:

a) Retains only the LSP header, and changes the Remaining Lifetime field to zero

b) Updates the Packet Length and Checksum fields to reflect the changes indicated in Step a)

c) Sets the LSP header from the database when zeroAgeLifetime has expired since the Remaining Lifetime became zero

d) Records the time at which the Remaining Lifetime for the LSP became zero

e) Purges the LSP header from the database when zeroAgeLifetime has expired since the Remaining Lifetime became zero

Any time a router purges an LSP with **nonzero** Remaining Lifetime, it performs the five steps just indicated, but the retention interval in Step e) is maxAge.

Note: This could happen for one of several reasons; for example, (a) resignation as Designated Router, (b) compaction of LSPs to reduce their number, (c) receiving from another router one's own LSP from a prior incarnation (Section 5.12.3, Step b.2).

5.7. Periodic LSP Generation

The timer maximumLSPGenerationInterval governs periodic LSP generation. Within every time interval of length maximumLSPGenerationInterval, a router regenerates every LSP at least once. Jitter is applied, as described in Section 2. It is not required to synchronize regeneration of the individual LSPs.

The purpose of periodic LSP generation is to remove stale information from the routing area at a coarse time granularity. For example, suppose two large, usually disjoint campuses, A and B, temporarily link with each other using a switched network. While the link persists, the Link State information of each area flows into the other. Then the link is taken down. Fine-grained operation of NLSP does not remove B's Link State information from routers in A. In case the link is reestablished, it would be costly to reacquire all that information. However, the two campuses might not link with each other again for a very long time. So at a coarse time scale, a method is needed to purge B's Link State information from routers in A. The method is refreshing LSPs by periodic regeneration, and aging out LSPs that are not refreshed.

A second result of periodic regeneration and aging of LSPs is to handle the very rare event of an LSP corruption not being detected by the checksum check. Ultimately, the corrupted information is aged out of the LSP database.

5.8. Event-driven LSP and CSNP Generation

A router generates an LSP when an event occurs that would cause the LSP content to change. Events that can cause such a change include

- An Adjacency or Circuit Up/Down event
- A change in a circuit's cost
- A change in manualAreaAddresses
- A change in a router's systemID or name
- A change of Designated Router on a LAN
- A change of network number
- A change in SAP services or external RIP routes
- A change in the waiting status (entering or leaving an overloaded state)

When such an event occurs, the router regenerates changed LSPs with a new sequence number. If the event necessitated generation of an LSP that had not been generated previously (for example, an Adjacency Up event for an adjacency that cannot be accommodated in an existing LSP), the Sequence Number is set to one. The router then propagates the LSPs on every circuit by setting the SRMflag for each circuit. The timer maximumLSPGenerationInterval is reset.

When the local router either becomes or resigns as the Designated Router for a LAN, it generates a lanL1DesRouterChange event. In addition, when it becomes the Designated Router, it

a) Generates and transmits Level 1 pseudonode LSPs

b) Purges the pseudonode LSPs issued by the previous Designated Router (if any)

When a router resigns as the Designated Router for a LAN, it performs the actions on LSPs described on page 5-8.

There is a hold-down timer, minimumLSPGenerationInterval, on the generation of each individual LSP.

When a WAN circuit starts or restarts, that is, whenever an adjacency changes to the "Up" state

c) The router sets the SRMflag for that circuit on all LSPs

d) The router sends a complete set of CSNPs on that circuit

5.9. Generation of Level 1 Non-Pseudonode LSPs

Each router generates a Level 1 non-pseudonode LSP on its own behalf. The LSP describes the router itself and the router's adjacencies.

A Level 1 non-pseudonode LSP contains the following information in its variable-length fields:

- In the Area Address Mask option, the set of manualAreaAddresses for this router (that is, the source router for the LSP).
- In the Management Information option, configuration and version information about this router:
  - Network Number is the router's internalNetworkNumber.
  - Node Number is the router's internal node number (0x00000001).
  - IPX Version is one.
  - Router/Server Name is a textual identifier of the router originating the LSP. If the system is also a NetWare server, this field contains the server name. This is the same name that appears in the IPX WAN version 2 Information Request/Reply packets.
- The External Routes option does not appear in non-pseudonode LSPs.
- The Services Information options contain information about services that reside on the same system as the router, and that advertise themselves using SAP. All such services must advertise themselves as residing on the internalNetworkNumber.
- An instance of the Link Information option for each neighboring router, formed from the following:
  a) The set of WAN NLSP adjacencies in the "Up" state
  b) The LAN pseudonode for each directly attached active LAN circuit
  c) The WAN pseudonode for each RIP/SAP circuit for which this router is the WAN Designated Router

  Fields of the Link Information option contain values as follows:
  - The cost for the link from this router's viewpoint is reported.
  - The Neighbor ID for a LAN adjacency is the neighbor's circuitID (the neighboring pseudonode ID); that is, this value is the LAN ID of the LAN Level 1 Hello packets sent on that circuit.
  - The Neighbor ID for a WAN NLSP adjacency is the neighbor's neighborSystemID with appended zero byte (indicating non-pseudonode).
  - The Neighbor ID for a WAN pseudonode adjacency is the WAN Designated Router's (that is, this router's) systemID followed by a one-byte pseudonode ID, unique among all pseudonodes for which this router is Designated Router.

- MTU Size is the maximum number of bytes that can be transmitted by this router on this link, including the IPX header but not datalink headers.
- Delay is the period of time (in microseconds) that it takes to transmit one byte of data (excluding protocol headers) to a destination, if the media were free of other traffic. For a WAN circuit, this value is determined as part of the IW2 protocol. The value used for a LAN circuit is 100 microseconds.
- Throughput is the amount of data, in bits, that can flow through the media and be received at the other side in one second, if there is no other traffic using the interface. For a WAN circuit, this value is determined as part of the IW2 protocol. For a LAN circuit, the value is the bit rate defined for the particular media technology.

  Note: For example, if router A is attached to an X.25 network with an interface speed of 64 Kbps, and router B is attached to the same network with an interface speed of 19.2 Kbps, then both report the adjacency between them as having a throughput of 19.2 Kbps.

- Media Type contains the media code described in Figure 5-2.

Several media type codes are defined for inclusion in the Media Type field of the Link Information option. For each media type, there is a default value for the Cost field. Figure 5-2 indicates the codes for the Media Type field, and the default Cost for the currently defined media types. Additional types will be added in the future. Note that WAN media have codes with the high-order bit set; LAN media have it reset.

Media Code	*Cost*	*Description*
0x0000	Varies	Generic LAN for use when no applicable medium is defined
0x8000	Varies	Generic WAN for use when no applicable medium is defined
0x0001	25	LocalTalk
0x0002	10	Ethernet II
0x0003	10	IEEE 802.3 with IEEE 802.2 without SNAP
0x0005	10	IEEE 802.3 with IPX header immediately following the 802.3 header
0x000A	10	IEEE 802.3 with IEEE 802.2 and SNAP
0x0004	11	IEEE 802.5 with IEEE
0x000B	11	IEEE 802.5 with IEEE 802.2 and SNAP
0x0006	10	IEEE 802.4
0x0007	20	IBM PC Network II
0x0008	25	Gateway's G/Net
0x0009	10	Proteon's Pronet
0x000C	10	Racore's LANPAC
0x800D	30	ISDN
0x000E	15	ARCnet
0x000F	20	IBM PC Network II with 802.2 without SNAP
0x0010	20	IBM PC Network II with 802.2 and SNAP
0x0011	13	Corvus OmniNet at 4 Mbps
0x0012	30	Harris Adacom
0x0013	25	IP tunnel
0x0014	7	FDDI with 802.2 without SNAP
0x0017	7	FDDI with 802.2 and SNAP
0x0015	20	Commtex IVDLAN
0x0016	20	Dataco
0x0018	40	SDLC tunnel
0x0019	7	PC Office frame
0x001A	20	Hypercommunications WAIDNET
0x801B	41	PPP, asynchronous
0x801C	19	PPP, synchronous
0x801D	40	X.25
0x801E	25	IP Relay

Figure 5-2: Media Codes and Default Costs

5.10. Generation of Level 1 Pseudonode LSPs

A router generates a pseudonode LSP on behalf of each circuit for which it is a Designated Router. The LSP specifies the following information in its variable-length fields:

- The Area Address Mask option is not present. (The set of area addresses for the issuing router is already available from its own non-pseudonode LSP.)
- The Management Information option contains configuration and version information about the circuit:
  - Network Number is the IPX network number for LAN pseudonodes and for numbered WAN pseudonodes. This field is zero for unnumbered WAN pseudonodes.
  - Node Number is the IPX node number (typically, the MAC address) of the Designated Router generating the LSP on the LAN to which the LSP refers. This field is zero for WAN pseudonodes.
  - Router/Server Name gives a textual name for the circuit. It need not be present.
- The External Routes and Services Information parts contain information gleaned from (respectively) RIP and SAP packets received by this Designated Router over the (LAN or WAN) circuit represented by this pseudonode.

  Note: The RIP/SAP packets considered for this purpose are only those from RIP/SAP-only systems. NLSP implementations must be sure not to report here other NLSP routers that are sending RIP/SAP packets for backward compatibility. These "compatibility" RIP/SAP packets are recognizable because they originate from nodes that have NLSP adjacencies with this router. Adjacencies in both the "Up" and "Initializing" states are included. They are intended for consumption by RIP/SAP-only devices sharing the circuit with this router.

  If this is a WAN pseudonode, the WAN is using RIP/SAP and not NLSP. Details of RIP/SAP compatibility are in Section 7.
- An instance of the Link Information option for each neighboring router, formed from
  a) The Designated Router itself.
  b) If this is a LAN pseudonode, other NLSP routers for which an adjacency with this Designated Router exist in the "Up" state over the circuit represented by the pseudonode.

  Fields of the Link Information option contain values as follows:
  - Zero Cost.
  - The Neighbor ID for the Designated Router itself is its own systemID with appended zero byte (indicating non-pseudonode).
  - The Neighbor ID for each of the other NLSP routers is the other router's systemID with appended zero byte (indicating non-pseudonode).

- The MTU Size and Delay and Throughput fields are zero for pseudonode LSPs.
- Media Type codes are indicated in Figure 5-2.

5.11. Preparing to Initiate Transmission

When generating an LSP as in the preceding sections—periodic or event-driven, pseudonode or non-pseudonode—a router

a) Stores the LSP in its own Link State database, overwriting any previous LSP with the same LSP Number generated by this router

b) Increments the sequence number

c) Calculates the checksum

d) Sets all the SRMflags for that LSP, indicating that it is to be propagated on

- All active LAN circuits (whether or not adjacencies exist for the circuit)
- All WAN circuits having adjacencies in the "Up" state

Circuits running RIP only are excluded.

A router must ensure (by reserving resources or otherwise) that it can always store and internalize its own non-pseudonode LSP number zero. If it cannot store and internalize one of its own LSPs, it enters the overload state as described on page 5-23.

Note: It is recommended that a router ensures (by reserving resources or otherwise) that it can always store and internalize all its own LSPs: zero and nonzero, pseudonode and non-pseudonode.

Sometimes an existing LSP is retransmitted, with the same or different sequence number, but with the same information content; that is, the same variable-length part. There is no change of information content because there have been no local topology changes. When this happens, the order of information in the variable-length part must be the same as in the previously transmitted LSP.

Note: This provision allows the receiver to detect that there has been no change of information content by making a byte-wise comparison of the variable-length part. This is an efficient way to check that it is unnecessary to rerun the Decision Process. If a sequence of topology changes results in the local topology returning to some previous state, there is no requirement to preserve the ordering. Preservation is required only if there have been no changes at all.

5.12. Receipt and Propagation of LSPs

A router accepts LSPs as large as actualMaxPacketSize on each circuit. Upon receipt of an LSP on circuit C, a router first performs the following acceptance tests:

a) The general packet acceptance tests described in Section 2 under "General Processing of Incoming IPX Packets."

b) If C is a LAN, and the source datalink address of the LSP does not match the neighborNICAddress of an existing adjacency on C in the "Up" or "Initializing" state, the router discards the LSP without generating an event.

> **Note:** A LAN might contain routers from more than one routing area. Checking for existence of an adjacency ensures that only LSPs for the router's own routing area are accepted.

c) If C is a WAN and there is no adjacency on C, the router discards the LSP without generating an event.

d) If the Checksum is zero, set the Remaining Lifetime to zero and proceed.

e) If the Checksum is incorrect (including the case of either or both checksum bytes being zero, and including the zero Remaining Lifetime case described previously), log a badChecksum event and discard the packet.

f) If the Router Type field is not "Level 1" or "Level 1 and Level 2," discard the packet.

g) If the length of the packet, as described in the header, is greater than the buffer in which it was received, log a packetRxSmall event and discard the packet.

h) If the variable part of the LSP is malformed, or contains malformed options, log the malformedOption event and discard the packet.

Depending on two tests of the received LSP, there are four different ways to process the LSP. The tests are as follows:

- Examine the Remaining Lifetime field. If it contains zero, the LSP is *expired.*
- Compare the Source ID field with this router's own systemID. If they match, this is a case of the *same system ID.*

The following four subsections specify the actions to take in the four cases. These subsections use the following shorthand:

- "Send an LSP on circuit C" means
  "Clear SSNflag and set SRMflag for that LSP for C."
- "All circuits" means
  "All WANs having adjacencies in the 'Up' state, and all active LANs regardless of adjacencies."
- "Acknowledge an LSP on circuit C" means
  "Clear the SRMflag for the LSP for C. If C is a WAN, also set the SSNflag for the LSP for C."

When comparing two LSPs for newness, the test on page 5-6 is applied.

5.12.1. Expired LSP with the Same System ID

a) If an LSP with this LSPID is in the LSP database

a.1) If the local LSP is also expired

a.1.1) If the LSPs are equal, acknowledge the LSP on C.

a.1.2) If the received LSP is older, send the local LSP on C.

a.1.3) If the received LSP is newer, update the local LSP's sequence number to be one more than the incoming LSP's and send the resulting LSP on all circuits.

a.2) If the local LSP is not expired

a.2.1) If the received LSP is older, send the local LSP on C.

a.2.2) If the received LSP is newer or equal, update the sequence number to be one more than the incoming LSP and send the local LSP on all circuits.

b) If an LSP with this LSPID is **not** in the LSP database, acknowledge the LSP on C without storing it.

5.12.2. Expired LSP with a Different System ID

a) If an LSP with this LSPID is in the LSP database

a.1) If the LSPs are equal, acknowledge the LSP on C.

a.2) If the received LSP is older, send the local LSP on C.

a.3) If the received LSP is newer, store the new LSP, replacing the local copy, send it on all circuits except C, and acknowledge it on C.

b) If you do not have an LSP with this ID in the LSP database, acknowledge the LSP on C without storing it.

5.12.3. Unexpired LSP with the Same System ID

a) If there is an LSP with this LSPID in the LSP database

a.1) If the LSPs are equal, acknowledge the LSP on C.

a.2) If the received LSP is newer than the local LSP

a.2.1) Update the local LSP's sequence number to be one more than the incoming LSP and flood the local LSP.

a.2.2) If this is a non-pseudonode LSP and it is LSP number 0, and it contains the Management Information option, compare the server name and the network number fields with those parameters of the local system. If the network numbers do not match, log the duplicateLSPSystemID event. If the network numbers match but the server names do not, log the duplicateInternalNet event.

a.3) If the received LSP is older than the local LSP, send the local LSP on C.

b) If there is no LSP with this ID in the LSP database

b.1) Create an LSP with a sequence number one more than the incoming LSP.

b.2) Follow the procedures for aging an LSP with nonzero remaining lifetime on page 5-8.

5.12.4. Unexpired LSP with a Different System ID

a) If there is an LSP with this ID in the LSP database

a.1) If the LSPs are equal, acknowledge the LSP on C.

a.2) If the received LSP is older, send the local LSP on C.

a.3) If the received LSP is newer

a.3.1) Store the new LSP, replacing the local copy

a.3.2) Send the new LSP on all circuits except C and acknowledge it on C.

b) If you do not have an LSP with this ID in the LSP database

b.1) Store the incoming LSP.

b.2) Send it on all circuits except C and acknowledge it on C.

5.13. Storing a New LSP

When storing a new LSP, the router first ensures that it has enough memory to store the LSP and whatever internal data structures will be required to process the LSP. If the resources are unavailable, the LSP is ignored—it is neither stored nor acknowledged. When an LSP is ignored for this reason, the router enters the "Waiting State" (see page 5-23).

When attempting to store a new version of an existing LSP whose length has not increased, it is recommended that removing the old and storing the new occur as a single atomic action. There is no increase in the resources required. This ensures that such an LSP (which may be carrying the LSP Database Overload indication from an overloaded router) is never ignored for lack of memory resources.

5.14. Receipt of Sequence Number Packets

A router sends Sequence Number Packets (SNPs) to its immediate neighbors and they are not propagated. By sending an SNP, the router communicates the current LSP sequence numbers in its database. A PSNP reports a subset of the LSPs and sequence numbers. A complete set of CSNPs reports on the total LSP database. A complete set of CSNPs is a set whose Start LSPID and End LSPID ranges cover the complete possible range of LSPIDs. That is, every possible LSPID value appears within the range of one of the CSNPs in the set.

Upon receipt of a PSNP or CSNP on circuit C, a router first performs the following acceptance tests:

a) The general packet acceptance tests described in Section 2 under "General Processing of Incoming IPX Packets."

b) If C is a LAN, and the source data-link address of the SNP does not match the neighborNICAddress of an existing adjacency on C in the "Up" or "Initializing" state, the router discards the SNP without generating an event.

c) If C is a WAN and there is no adjacency on C, the router discards the SNP without generating an event.

d) If the length of the packet, as described in the header, is greater than the buffer in which it was received, log a packetRxSmall event and discard the packet.

e) If the variable part of the SNP is malformed, or contains malformed options, log a malformedOption event and discard the packet.

For each LSP reported in the SNP, use the method on page 5-6 to compare the reported LSP with the corresponding LSP in the local database, and take the following action for C:

Note: No check is made against the receiving router's own systemID. Refer to "Resolving LSP Confusion" later for a rationale.

e) If C is a WAN and the reported LSP is the same as the one stored in the local Link State database, clear the SRMflag.

f) If the reported LSP is older than the local LSP, clear the SSNflag and set the SRMflag.

g) If the reported LSP is newer than the database LSP

g.1) Set the SSNflag.

g.2) If C is a WAN, clear the SRMflag.

h) If no database entry exists for the LSP, and if the reported Remaining Lifetime, Checksum, and Sequence Number fields of the LSP are all nonzero

h.1) Create an LSP entry with sequence number zero.

h.2) Set the SSNflag for that entry for C.

Note: SRMflag must never be set for an LSP having a zero sequence number. Possessing a zero sequence number is semantically equivalent to having no information about that LSP. If such an LSP were propagated by setting the SRMflag, it would result in unnecessary consumption of both bandwidth and memory resources.

If the received packet is a CSNP, set the SRMflag for C for LSPs for which

i) The LSP is in this router's database.

j) The LSP has nonzero Sequence Number and nonzero Remaining Lifetime.

k) The LSPID is within the range specified by the Start LSPID and End LSPID fields of the CSNP.

l) The LSP is not mentioned in the CSNP.

Note: Because entries in the CSNP are ordered, the LSP screening process can be accomplished in a single pass through the CSNP.

5.15. Transmitting the Packets

In the preceding discussions, packets were composed and generated, flags were set, and the router made preparations to send packets. But they were not actually transmitted. The actual transmission of packets is governed by certain timers. The subsections to follow provide the details.

5.15.1. Expiration of a Complete SNP Interval

The router performs the following actions every completeSNPInterval for each LAN circuit C:

a) If this router is the Level 1 Designated Router for C, transmit a complete set of Level 1 CSNPs on C. Ignore the setting of SSNflag on the LSPs.

Note: On WAN links, CSNPs are sent only at initialization.

A complete set of CSNPs is a set whose Start LSPID and End LSPID ranges cover the complete possible range of LSPIDs. That is, every possible LSPID value appears within the range of one of the CSNPs in the set.

Note: Implementations must not compute the entire set of LSPs to report at the outset, then send more than one CSNP. Rather, each CSNP after the first should reflect those changes in the database that may have occurred since sending the previous CSNP that affects the LSP range not yet covered in this complete CSNP set.

When multiple CSNP is transmitted on a circuit, they are separated by an interval of at least 1/18 second.

Note: Minor variations are sometimes inevitable when adhering to the 1/18 second rule, owing to the timer granularity of the router. Such variations are not material.

5.15.2. Expiration of a Partial SNP Interval

The router performs the following actions every partialSNPInterval for each circuit C, with jitter applied as described in Section 2:

If either

a) C is a WAN

Or

b) C is a LAN and this router is **not** the Designated Router for C

then transmit a PSNP on C containing entries for as many LSPs with SSNflag set as fit in the packet. Then clear the SSNflag for these entries. To avoid starvation, the scan of the LSP database (for those with SSNflag set) starts with the next LSP that was not included in the previous scan. If there are no LSPs with SSNflag set, do not transmit a PSNP.

5.15.3. Expiration of Minimum LSP Transmission Interval

A router takes the following action every minimumLspTransmissionInterval, with jitter applied as in Section 2:

The router scans the LSP database and sends on each circuit C those LSPs having the SRMflag set for C. If C is a LAN, clear the SRMflag.

Where multiple LSPs are transmitted on a circuit, they must be separated by a minimum gap of 1/18 second.

Note: Minor variations are sometimes inevitable when adhering to the minimum gap rule, owing to the timer granularity of the router. Such variations are not material.

Note: In practice, it would be inefficient to scan the entire database this often, particularly when only a few LSPs had SRMflags set. Implementations would typically use additional data structures to optimize this operation.

5.15.4. Circuit Pacing to Avoid Circuit Congestion

Circuit pacing applies to transmission of LSPs and CSNPs on each circuit. This rule is in addition to the minimal 1/18 second gap.

Circuit pacing specifies that no more than paceRate CSNPs are sent per second on the circuit, and no more than paceRate LSPs. The value of paceRate is calculated based on the circuit's Throughput (in bits/second), as follows:

paceRate = Throughput / 10,000

with a minimum of 1 and a maximum of 18. For example, at 19,200 bits/second or less, paceRate is one packet per second. At 19,200 bits/second and 512-byte packets, no more than 25% of the circuit bandwidth is consumed by LSPs or CSNPs.

5.16. Determining the Latest Information

Several provisions govern manipulation of the values that appear in LSPs. The subsections to follow provide details.

5.16.1. Operation of Sequence Numbers

The Sequence Number is a four-byte unsigned value. When a system initializes, it starts with sequence number one for its own LSPs.

The intent is for the Sequence Number to be incremented monotonically. If a router restarts after being deactivated, it jumps to using sequence numbers beyond the range it used in its previous incarnation. See the next section, "Resolving LSP Confusion," for the details about how this is accomplished.

The sequence numbers a router generates for LSPs of different LSP Number are independent. The algorithm for choosing the numbers is the same, but the numbers advance separately, without being synchronized.

It may happen that a sequence number reaches the maximum value representable in the four-byte sequence-number space. The router owning the LSP, and needing to regenerate it, purges the LSP by sending the LSP with the same sequence number but zero Remaining Lifetime. This causes the LSP to be aged by all other routers. It sets a timer to regenerate the given LSP after maxAge + zeroAgeLifetime. Then the source router starts regenerating the LSP with sequence number 1.

Note: NLSP is less severe than IS-IS in this regard: here, the router need not be disabled. However, if the router is shut down manually, the user must wait maxAge + zeroAgeLifeTime to restart it. In NLSP, recovery from the overflow is automatic. Also, because the router is not shut down, only a partial loss of connectivity to the services, external routes, and Adjacencies described in the LSP may occur, rather than a total loss of all connectivity through the router.

5.16.2. Resolving LSP Confusion

This subsection does not add specification logic to the document. It shows how steps in preceding sections fit together to keep LSPs progressing monotonically in an orderly way throughout the routing area despite router restarts.

When a router restarts, losing memory of its previous state, it is possible that an LSP generated by the router in a previous incarnation (that is, by the same system but before the restart) is alive in the routing area. In this case, the collective memory of the other routers is used to advance the sequence numbers of the restarted router's LSPs into a range beyond the numbers used by the router's previous incarnation. Many of the details in processing incoming LSPs and CSNPs are motivated by the need to resolve LSP Confusion.

Suppose router A has restarted and it has immediate neighbors B and C. A sends its first LSP with sequence number one (Section 5.16.1). B still has an old LSP from A with a higher sequence number. So B, considering A to be out of date, helpfully provides A with the "newer" (actually older) LSP (Section 5.12.4, Step a.2). When A receives that LSP from B, it discovers how far its sequence number reached prior to the restart. In then advances its sequence number for the LSP to make it newer still (Section 5.12.3, Step a.2.1). When the restarted router receives its own LSP from a neighbor and the received LSP has an LSP Number not in use, it purges the LSP from the routing area (Section 5.12.3, Step b.2).

It is even possible for an LSP generated by a system in a previous incarnation to be alive in the routing area and have the same sequence number as the current LSP. To deal with this case, the comparison described on page 5-6 includes the Preference of Checksums rule. With this rule, routers detect that the LSPs are different and all routers make the same decision about which is newer. With this rule, resolution of LSP Confusion works the same way for equal and unequal sequence numbers.

Another aspect of LSP confusion is a misconfiguration in which two routers, A and B, are given the same systemID. One of the two, say A, generates an LSP number zero that all routers consider newer than B's LSP number zero. By operation of the protocol, A's LSP eventually reaches B. When this happens, B can readily detect the problem and

report A's Network Number and Router/Server Name from the Management Information field of the LSP (Section 5.12.3, Step a.2.2).

References in the preceding paragraphs cited steps in LSP processing. To ensure that LSPs are propagated to the routers that execute these steps, SNP processing must operate in a supportive way. For example, to resolve LSP confusion, a router A requires its neighbors to send A their versions of A's own LSPs. This is the reason that SNP processing does not discriminate in Section 5.14 between this router's and other routers' LSPs.

5.16.3. Synchronizing LSP Expiration

This subsection does not add specification logic to the document. It shows how steps in preceding sections fit together to make old LSPs age out of different routers' Link State databases at (approximately) the same time.

If a router detects that another router has become unreachable, the other router's LSPs are kept anyway. If this were not done, repair of a brief network partition could prompt extensive consumption of network bandwidth and computing resources to reacquire and reabsorb connectivity information that has not changed since before the partition.

Nonetheless, stale information must eventually be removed because the routers in question might not become reachable again. As long as routers in a routing area remain mutually reachable, their LSPs remain in each other's Link State databases. They are refreshed by periodic regeneration (page 5-8). If a router holds an LSP that has not been refreshed for an interval maxAge, it purges the LSP, as described on page 5-8.

To keep the routers in the routing area consistent with each other, it is not enough that they act independently in purging stale LSPs. Routers' timers are not synchronized with each other. Routing could be disrupted if there was an extended period of mismatched databases. So when the first router ages out an LSP, it conveys this fact to other reachable routers by truncating the LSP and propagating it with zero lifetime (Section 5.6 Step a).

When a router receives such a truncated LSP, it accelerates the aging of the LSP in its own database (Section 5.4 Step b and Section 5.12.4 Step a.3). Once one router has aged an LSP, all routers reachable from it purge the LSP as soon as possible.

After purging the LSP, a router keeps it in truncated form in its Link State database for a time (page 5-8). This way, unexpired versions of the same LSP cannot propagate widely. Such unexpired versions might be in the process of being propagated by a router that is aging it slower than the one instigating the purge. The expired version is considered newer than an unexpired counterpart (Section 5.4 Step b). Keeping the truncated LSP for a while acts as a lock-down on the expiration of the LSP. Including the Remaining Lifetime in Sequence Number Packets reinforces the rapid, firm spread of the expiration news (Section 5.14 Step g).

5.17. Validation of Databases

Corruption of information in router memory is possible from a failure in hardware or software. A router must not continue to operate for an extended period with corrupted routing information. Routers operate in a fail-stop manner. If a failure is detected, the router is disabled until the failure is corrected. Developers can take implementation-specific steps to guarantee this. In the absence of such a guarantee, the router performs the following actions at least every maximumLSPGenerationInterval:

a) To detect corruption of the LSP database while in memory, recheck the checksum of every LSP in the database, except those with zero Remaining Lifetime. If any checksum is incorrect

   i) Log a badIntChecksum event.

   ii) Delete the entire LSP database.

   iii) Reset every WAN adjacency to the "Down" state, as described in Section 4.

   iv) Cause the database to be reacquired.

   Note: One way to accomplish ii, iii, and iv is to disable the router and restart.

b) When the checks are completed, notify the Decision Process, even if all checksums were correct. This causes the Decision Process to be run and the Forwarding database to be recomputed. This protects against corruption of the Forwarding database in memory, which might otherwise persist undetected in memory with a stable topology.

c) Reset the timer for maximumLSPGenerationInterval with jitter applied, as described in Section 2.

5.18. Managing LSP Database Overload

If there are insufficient memory resources to store a received LSP, the receiving router's Link State Database might become inconsistent with those of other routers. The router takes the following steps to ensure that other routers do not rely on forwarding paths through the overloaded router:

a) Ignore the LSP that cannot be stored.

b) Enter the Waiting State; start a timer for waitingTime, to limit the duration of the Waiting State.

c) Generate an l1DatabaseOverload event.

d) Generate and flood the router's own LSP having zero LSP Number with the LSP Database Overload bit set.

e) Lower to zero the priority of becoming the Designated Router for every LAN cicuit for which the router is not already the Designated Router. (Circuits for

which the router is already the Designated Router are unaffected.) The lowering of priority lasts as long as the overload state, after which the priority reverts to the initial value.

Even if received LSPs can be stored, it is possible that resources become exhausted while running the Decision Process. If this happens, the router enters the Waiting State (as just described) until resources are available and waitingTime has elapsed since the last received LSP was ignored.

While in the Waiting State

a) If an LSP cannot be stored, the router ignores it and restarts the timer for waitingTime.

b) Propagation of LSPs, running the Decision Process, and forwarding data traffic proceed as normal.

c) When the waitingTime expires, the router

   i) Generates an l1DatabaseOverload (recovered) event

   ii) Clears the LSP Database Overload bit in its own Level 1 LSP with zero LSP Number and reissues it

   iii) Resumes normal operation

5.19. Link State Database

5.19.1. Configured Values

maxAge
: The time interval after which an LSP is considered to expire. Default is 7500 seconds.

zeroAgeLifetime
: The time interval that a router retains an LSP in its database once the LSP has expired: one minute. Default is 60 seconds.

maximumLSPGenerationInterval
: The maximum interval, in seconds, between the generation of the same LSP. Default is 7200 seconds.

lspBufferSize
: The maximum size of Level 1 LSPs originated by this router. Default is 512 bytes (including the IPX header but not datalink headers). Every router must support at least 512 bytes.

  Note: The end-user should be sure to configure lspBufferSize to be no larger than the smallest MTU size of any router for any circuit in the area.

minimumLSPTransmissionInterval
: The minimum interval, in seconds, between sending LSPs on a circuit. There can be two separate values: one for LANs and one for WANs. Default is 2 seconds.

minimumLSPGenerationInterval
The minimal interval, in seconds, between regenerations of the same LSP. Default is 5 seconds.

completeSNPInterval
The interval, in seconds, between generation of Complete Sequence Number Packets by a Designated Router on a LAN. Default is 20 seconds.

partialSNPInterval
The minimum interval, in seconds, between transmission of Partial Sequence Number Packets. Default is 1 second.

waitingTime
The number of seconds to remain in the waiting (LSP database overload) state before returning to normal operation. Default is 60 seconds.

5.19.2. Configured Values per Circuit

cost
The measure attached to a circuit used to make routing decisions. The Decision Process determines the smallest aggregate cost to each reachable NLSP destination. Default values are listed in Figure 5-2.

5.19.3. The LSP Database

LSPdatabase
The set of LSP packets received by this router and still in memory.

SRMflag
For each LSP and each circuit, a bit indicating that the LSP should be sent on that circuit.

SSNflag
For each LSP and each circuit, a bit indicating that the LSP should be reported in a PSNP sent on that circuit.

lastSent
For each LSP and each circuit, a record of the time at which that LSP was most recently sent on that circuit.

5.19.4. NLSP Events

lanL1DesRouterChange
The role of Designated Router on a LAN circuit is transferred from one NLSP router to another.

badChecksum
An LSP was received for which checksum verification failed.

badIntChecksum
A checksum error was detected in an LSP stored in router memory, indicating a memory corruption error.

duplicateLSPSystemID
An arriving LSP reports a systemID conflicting with this system's systemID, but with a different internal network number.

duplicateInternalNet
An arriving LSP reports an internalNetworkNumber conflicting with this system's internalNetworkNumber, and with the same systemID and different servername.

l1DatabaseOverload
The LSP database has become overloaded, and can become inconsistent with those of other routers.

enteringL1DatabaseOverload
The router state changes from normal operation to a situation in which the LSP database is overloaded.

remainingInL1DatabaseOverload
While in an overloaded state, another event occurred to prolong the LSP database overload state.

exitingL1DatabaseOverload
The router is no longer in the LSP database overload state. Normal operation is resuming.

5.20. Packet Structures

The following packet types are relevant to the Link State methods in this section.

- Level 1 LSP (page 5-28)
- Level 1 CSNP (page 5-31)
- Level 1 PSNP (page 5-34)

All are multicast when transmitted on a LAN. If a LAN Hello packet has been received with the No Multicast bit set, a router must use the broadcast address on that circuit instead of multicast.

These packets ride in the data portion of IPX packets. The IPX header fields are encoded as shown in Figure 5-3.

	LSPs	SNPs sent on a WAN	SNPs sent on a LAN
Destination Network	zero	zero	zero
Destination Node	0xFFFFFFFFFFFF	0xFFFFFFFFFFFF	0xFFFFFFFFFFFF
Destination Socket	0x9001	0x9001	0x9001
Source Network	internalNetworkNumber of the router that originally created the LSP	internalNetworkNumber of the router sending the packet	Network number of the LAN on which the packet is being transmitted
Source Node	Node number (0x000000000001) of the router that originally created the LSP on its internalNetworkNumber	Node number (0x000000000001) of the router sending the packet on its internalNetworkNumber	IEEE MAC address of the LAN interface through which the packet is being transmitted
Source Socket	0x9001	0x9001	0x9001
Packet Type	0	0	0

Figure 5-3: IPX Header Fields

These Level 1 LSPs never exceed lspBufferSize bytes in size, including the IPX header but excluding the datalink header.

The maximum size of sequence number packets on any circuit is determined by the media type and the buffer sizes used by neighbors on that circuit. Each Hello packet includes the Local MTU option, containing the sender's localMaxPacketSize for the circuit. This is the way routers communicate the packet size they are able to receive on that circuit. Every router calculates the minimal value of Local MTU for active adjacencies on each circuit. It then takes the smaller of that value and its own localMaxPacketSize for that circuit. The result is the actualMaxPacketSize value for that circuit. Sequence Number Packets can be as large as that value, but no larger.

The valid option fields in the variable parts of these packets can appear in any order.

5.20.1. Level 1 LSP

Protocol ID: 0x83, identifies the NLSP routing layer.

Length indicator: The number of bytes in the fixed portion of the header (up to and including the Router Type field).

Version/Protocol ID Extension: 1, ignored on receipt.

Reserved: 0, ignored on receipt.

Reserved (3 bits): 0, ignored on receipt.

Packet Type (5 bits): 18.

Version: 1.

Reserved: 0, ignored on receipt.

Packet Length: The entire length of this packet, in bytes, including the fixed portion of the NLSP header.

Remaining Lifetime: The number of seconds before this LSP is considered to expire.

Level 1 LSP

	Number of Bytes
Protocol ID	1
Length Indicator	1
Version / Protocol ID Extension	1
Reserved	1
Reserved | Packet Type	1
Version	1
Reserved	2
Packet Length	2
Remaining Lifetime	2
LSP ID	8
Sequence Number	4
Checksum	2
P | Attached Flag | LSP DBOL | Router Type	1
Variable Length Fields	Variable

LSP ID: A field composed of three parts:

- Source ID The system ID of the router that originated the LSP.
- Pseudonode ID Is zero if this is a non-pseudonode LSP; otherwise, it is a unique (for this Source ID) number designating this pseudonode.
- LSP Number: If a would be LSP is too large to send, the source breaks it into fragments identified by this monotonically increasing number.

LSP ID

	Number of Bytes
Source ID	6
Pseudonode ID	1
LSP Number	1

Sequence Number: The sequence number of the LSP.

Checksum: The checksum of the LSP contents from Source ID (the first part of LSP ID) to the end.

P: (one bit): zero—Indicates that this router does not support partition repair.

Attached Flag (four bits): Indicates whether the router can provide a path to other routing areas.

0 = "No," other routing areas cannot be reached through this router.

1 = "Yes," other routing areas can be reached through this router.

Other values are reserved.

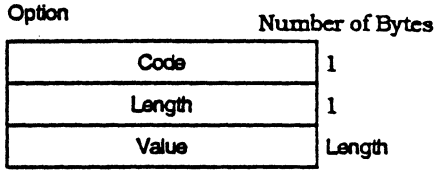
LSPDBOL: (one bit): Set to one when the LSP database is overloaded.

Router Type: (two bits):

1 = "Level 1 Router;" operation is specified in this document.

3 = "Level 1 and Level 2 Router;" accept this value from other routers, for forward compatibility.

Other values are reserved.



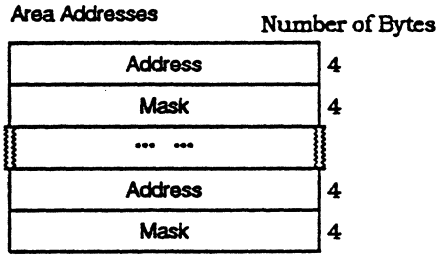
Variable Length fields: A series of optional fields, each of which has the following three-part code/length/value Option form:

Currently defined codes, and the corresponding values, are

- Area Addresses: The set of manualAreaAddresses of the sending router; not present in pseudonode LSPs.

  Code = 0xC0. Length = Total length of the value field, in bytes; either 8, 16, or 24.

  Value = Up to three area addresses. Each area address consists of a four-byte network number, followed by a four-byte address mask. The mask contains from zero to 32 (inclusive) most-significant "one" bits to indicate which bits of the network number make up the address prefix identifying the routing area. The remaining bits are "zero." The bit-wise AND of an Address and Mask pair must be equal to Address; for example, it would be a mistake to send Address = 0x84300000 paired with Mask = 0xFF000000.

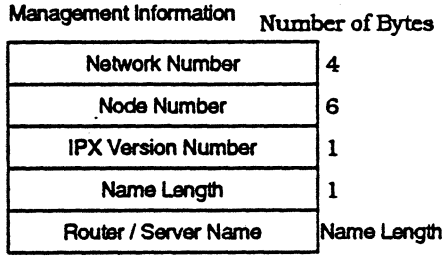


- Management Information: Several fields with information about the router originating the LSP.

  Code = 0xC1. Length = 13 to 60, or more.

  Value = The following five subfields:

  Note: To allow future versions of this protocol to add fields at the end and remain compatible with routers implementing routers implementing this version, routers receiving this option ignore any fields after those listed here.



  - Network Number: The internal IPX network number of the router generating the LSP. For a LAN pseudonode or a numbered WAN pseudonode, it is the IPX network number of the network segment that the pseudonode represents. For an unnumbered WAN pseudonode, the value is zero.

- Node Number: The internal IPX node number (0x00000001) of the router generating the LSP. For a LAN pseudonode, it is the node number (typically, the MAC address of the point of attachment) of the Designated Router generating the LSP on the LAN to which the LSP refers. For a WAN pseudonode, the value is zero.
- IPX Version Number: 1.
- Name Length: Length of the Router / Server Name field; zero if none is present.
- Router / Server Name: A string of 1 to 47 bytes identifying the router originating the LSP. If this is a pseudonode LSP, this field might not be present; but if it is, it identifies the network segment to which the LSP refers.

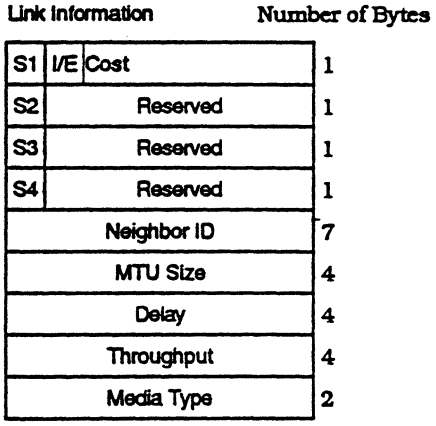
  Note: Although the server name is null-terminated in many NetWare protocols, this string is not null-terminated.

- Link Information: Several subfields with information about one adjacency of the source router. This option occurs several times, in general, once for each adjacency.

  Code = 0xC2. Length = 29 or more.

  Value = The following subfields.

  Note: To allow future versions of this protocol to add fields at the end and remain compatible with routers implementing this version, routers receiving this option ignore any fields after those listed here. When sending, use 29 as the Length. When receiving, accept any number 29 or larger.



  - S1 (1 bit): Zero, indicating that the Cost is present.
  - Internal/External, abbreviated I/E in the diagram (1 bit): Zero, indicating that the Cost is an internal metric.

    Note: With hierarchical routing, costs internal to one's own routing area are not usually comparable with those outside it. An "internal" route is chosen over an "external" one, if there is a choice. The I/E bit provides information to the Decision Process that allows the choice to be made.

  - Cost (6 bits): The cost of a link to the listed neighbor; an unsigned positive integer.
  - S2, S3, S4 (1 bit each): One, indicating that the rest of the (respective) bytes contain no information.
  - Reserved (three seven-bit fields): 0, ignored on receipt.
  - Neighbor ID: For a non-pseudonode neighbor, the neighboring router's system ID plus one byte of zero; for a pseudonode neighbor, the first six bytes are the Designated Router's system ID and the seventh byte is the unique nonzero Pseudonode ID value assigned to this pseudonode by the Designated Router.

- MTU Size: The maximum number of bytes that can be transmitted on this link by the originating router, including the IPX header but not including datalink headers. It is zero for a pseudonode LSP.
- Delay: The period of time (in microseconds) that it takes to transmit one byte of data (excluding protocol headers) to a destination, if the media is free of other traffic. It is zero for a pseudonode LSP.
- Throughput: The amount of data, in bits, that may flow through the media and be received at the other side in one second, if there is no other traffic using the interface. It is zero for a pseudonode LSP.
- Media Type: A code identifying the type of circuit; the most significant bit is one for WAN media, zero for others. See Figure 5-1 for the assigned values.

- Services Information: Describes services that advertise themselves by the SAP protocol.

  Code = 0xC3. Length = 16 to 63.

  Value = The following subfields.

  - Internal/External, abbreviated I/E in the diagram (1 bit): If the services information was received by a foreign protocol such as IPX SAP, this bit is set to 1; if the information is locally derived (for example, if the service resides on the system generating this LSP), it is 0.

  Services Information

		Number of Bytes
I/E	Hops	1
Network Number		4
Node Number		6
Socket		2
Type		2
Service Name		1 to 47

  - Hops (7 bits): The number of hops to reach the service; if I/E is zero, this field is zero also.
  - Network Number, Node Number, Socket: The IPX address at which the service is available.
  - Type: The type of service offered, see Reference [Nov92] for a partial list of defined types.
  - Service Name: The name of the service; its length is determined implicitly by the length of this option—the name is **not** null-terminated.

- External Routes: Describes routes obtained by the source router through non-NLSP protocols; for example, RIP (see Reference [Nov92]).

  Code = 0xC4. Length = $7 \times n$; $n \geq 0$.

  Value = The following subfields.

  - Internal/External, abbreviated I/E in the diagram (1 bit): 1.

  External Routes

		Number of Bytes
I/E	Hops	1
Network Number		4
Ticks		2

  - Hops: The number of hops reported by the non-NLSP protocol.
  - Network Number: The IPX network number to which this entry refers.
  - Ticks: The RIP Delay (that is, the number of RIP timer ticks) from the source router to network Network Number, as reported by the non-NLSP protocol.

5.20.2. Level 1 CSNP

Protocol ID: 0x83, identifies the NLSP routing layer.

Length indicator: The number of bytes in the fixed portion of the header (up to and including the End LSP ID field).

Version/Protocol ID Extension: 1, ignored on receipt.

Reserved: 0, ignored on receipt.

Reserved (3 bits): 0, ignored on receipt.

Packet Type (5 bits): 24.

Version: 1.

Reserved: 0, ignored on receipt.

Packet Length: The entire length of this packet, in bytes, including the fixed portion of the NLSP header.

CSNP

	Number of Bytes
Protocol ID	1
Length Indicator	1
Version / Protocol ID Extension	1
Reserved	1
Reserved | Packet Type	1
Version	1
Reserved	2
Packet Length	2
Source ID	7
Start LSP ID	8
End LSP ID	8
Variable Length Fields	Variable

Source ID: The six-byte system ID of the sending router, followed by one byte of zero.

Start LSP ID, End LSP ID: The first and last LSP in the range covered by this CSNP. Each is a field composed of three parts:

- Source ID (6 bytes) Is the systemID of the router that originated the LSP being reported.
- Pseudonode ID Is zero if this is a non-pseudonode LSP; otherwise, it is a unique (for this Source ID) number designating this pseudonode.
- LSP Number: If a would be LSP is too large to send, the source breaks it into fragments identified by this monotonically increasing number.

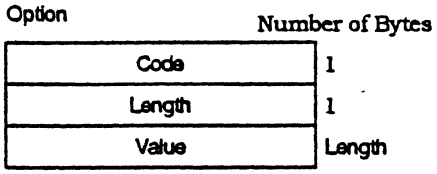
LSP ID

	Number of Bytes
Source ID	6
Pseudonode ID	1
LSP Number	1

Variable Length fields: A series of optional fields, each of which has the following three-part code/length/value Option form:

Currently defined codes, and the corresponding values, are

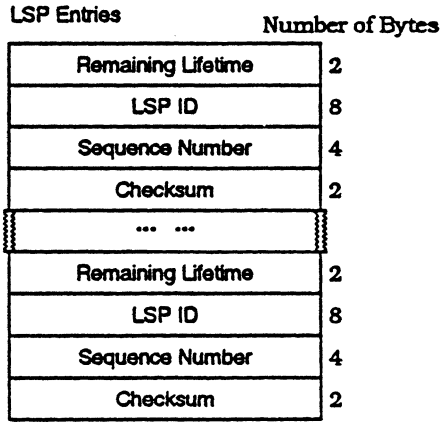
- LSP Entries: This option can appear more than once; if so, instances must be sorted in ascending LSPID order.



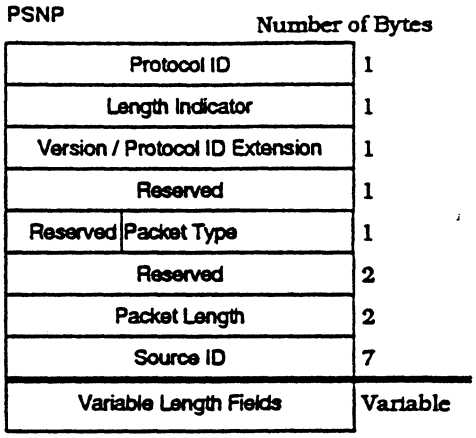
  Code = 9. Length = 16 × *n*; *n* > 0.

  Value = A list of LSP entries, sorted in ascending LSP ID order (the LSP Number byte of the LSP ID is the least significant byte). Each entry has four subfields:

  - Remaining Lifetime: Seconds remaining until the indicated LSP expires.
  - LSP ID: Identifies the LSP referred to by this LSP Entry. It has the same three-part Source ID / Pseudonode ID / LSP Number form described above.
  - Sequence Number: The sequence number of the indicated LSP.
  - Checksum: The checksum reported in the indicated LSP.



5.20.3. Level 1 PSNP



Protocol ID: 0x83, identifies the NLSP routing layer.

Length indicator: The number of bytes in the fixed portion of the header (up to and including the Source ID field).

Version/Protocol ID Extension: 1, ignored on receipt.

Reserved: 0, ignored on receipt.

Reserved (3 bits): 0, ignored on receipt.

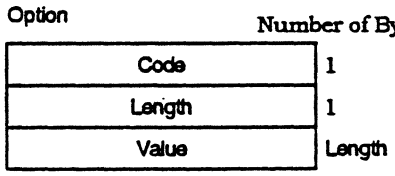
Packet Type (5 bits): 26.

Version: 1.

Reserved: 0, ignored on receipt.

Packet Length: The entire length of this packet, in bytes, including the fixed portion of the NLSP header.

Source ID: The six-byte system ID of the sending router, followed by one byte of zero.



Variable Length: A series of optional fields, each of which has the following three-part code/length/value Option form:

Currently defined codes, and the corresponding values, are

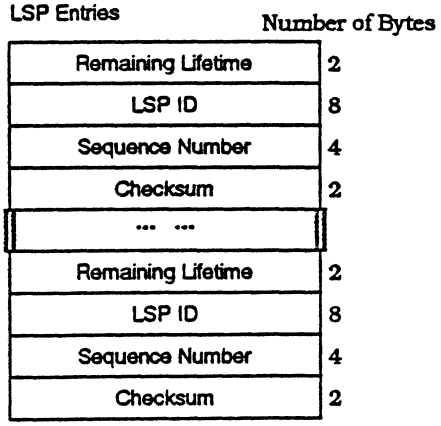
- LSP Entries: This option can appear more than once; if so, instances must be sorted in ascending LSPID order.

  Level 1 CSNP packet structure codes and values

  Code = 9. Length = $16 \times n$; $n > 0$.

  Value = A list of LSP entries, sorted in ascending LSP ID order (the LSP Number byte of the LSP ID is the least significant byte). Each entry has four subfields:

  - Remaining Lifetime: Seconds remaining until the indicated LSP expires.
  - LSP ID: Identifies the LSP referred to by this LSP Entry. It has the three-part Source ID/Pseudonode ID/LSP Number form described in the preceding subsection for CSNPs.
  - Sequence Number: The sequence number of the indicated LSP.
  - Checksum: The checksum reported in the indicated LSP.



6. Decision Process

The Link State protocol allows each router to construct a graph representing the routing area. The vertices of the graph are the nodes and pseudonodes—one for every LSP series. The arcs of the graph are the links between (pseudo)nodes. They are reported in the Link Information fields of LSPs. The Decision Process operates on this graph. It uses Dijkstra's algorithm to produce the Forwarding database. For each IPX network number in the Link State database, the Forwarding database indicates the next hop from this router toward that destination network number. The network numbers are in the LSPs—and consequently attached to nodes of the link state graph—in the Management Information field and the External Routes field.

The Decision Process is also responsible for generating equal cost paths, determining the throughput and delay characteristics of the end-to-end path, and calculating the RIP delay (Ticks) to report to RIP routers and to end nodes.

6.1. Running the Decision Process

The Decision Process runs when a change occurs to the Link State database. It is not run immediately, but rather the router waits for five seconds after detecting the change. This allows several changes clustered in time to be dealt with at once.

Implementations should ensure that the Link State database is not modified while the Decision Process is running. There is more than one way to accomplish this, and the approach chosen is influenced by the operating environment surrounding the routing software.

One approach is for the Decision Process to signal when it is running. During this time, incoming LSPs are queued but not inserted in the Link State database. If more LSPs arrive than can fit in the space allotted for the queue, the router drops any excess LSPs without acknowledging them; that is, without including their sequence numbers in any SNPs transmitted.

A second approach is to use two copies of the Link State database. When the Decision Process starts, it takes a snapshot copy of the database, keeping the snapshot static until completion of the process. Meanwhile, updates continue to be incorporated into the dynamic copy. If there is a separate memory area for each copy, the areas can alternate between the static and dynamic roles.

The first approach uses less memory, but the second approach avoids dropping incoming LSPs.

Once started, an execution of the Decision Process should not be abandoned due to new information arriving. To do so could lead to starvation; the process might never be run to completion. If an event occurs that would change the Link State database while the Decision Process is running, the five-second timer restarts when the Decision Process is completed.

6.2. Dijkstra's Algorithm in Pseudocode

The description of Dijkstra's algorithm in Section 2 is intended to appeal to your graphical intuition. Here is another description of the same algorithm, more akin to pseudocode.

START:

Keep a list containing all "unplaced" nodes, and associate a cost with each.

Start with all nodes in the unplaced list.

Set the cost of all nodes to infinite except the local node, which is set to 0.

Keep a list of nodes that are already in the "shortest path" spanning tree. (This is the Known Set of the description in Section 2.)

This set starts empty.

NEXT_NODE:

Select a node in the unplaced list with lowest cost that is not infinite.

Remove the selected node from the unplaced list and add it to the shortest-path list.

If there is no such node, the algorithm is complete and all nodes remaining in the unplaced list are unreachable.

In the unplaced list, update the costs of all nodes adjacent to the node just removed. The new cost is the minimum of

(a) the previous cost of the adjacent node, and
(b) the sum of
    (i) the cost of the node just removed, and
    (ii) the link cost of the adjacency.

goto NEXT_NODE

For example, at Step 6 in Section 2.1.7, the unplaced list looks like this, after removing V from the list:

S	40
C	50
W	infinite

Because S has the lowest cost, it is next to be removed.

6.3. Load Splitting

NLSP supports load splitting. That is, if there are equal cost paths, the traffic can be divided among them to make fuller use of the internetwork.

First, define a "maximal splitting degree." This is a number, MSD, assigned by the network administrator. It can be different for different routers. If there are equal cost paths, MSD is an upper limit on the number used for routing. If MSD=1, the router does not do load splitting. Load splitting focuses on the step in Dijkstra's algorithm where you choose which node to add to the known set.

In case of a tie to the same far node, the forwarding database has more than one entry added to it, instead of just one. How many? If the tie is among *m* links, the number of entries added is the smaller of *m* and MSD. If *m* is greater than MSD, the choice of which to add is based on the following criteria, in the following order:

(a) End-to-end path with the lowest cost

(b) End-to-end path providing the highest total Throughput

(c) End-to-end path providing the lowest total Delay

(d) End-to-end path supporting the largest MTU size

(e) First Hop node with the Lowest System ID

(f) Circuit with the lowest localCircuitID on the local router

(g) Neighbor with the lowest LAN MAC address on the remote router

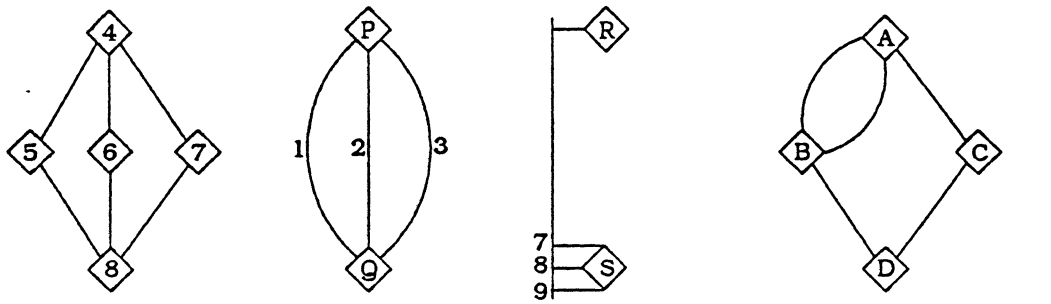
The MTU Size, Delay, and Throughput are in the Link Information of LSPs. System IDs are assigned by an administrator to each router; circuit IDs are assigned by router software. The reason for these criteria are (a) to provide the best routes, and (b) to provide deterministic routes (for reproducibility in problem resolution).

Figure 6-1 shows examples of rules (e), (f), and (g). (There are no parts (a)-(d) in the figure.) In each example, the links are of equal cost, and MSD is two.

Part (e) shows the system IDs of the routers (single-digit for simplicity). When router #4 adds #8 to the Known Set, it uses the two links to #5 and #6, omitting the link to #7.

In part (f), the localCircuitIDs are shown. When router P adds Q to the known set, it uses links #1 and #2, but not #3.

Part (g) shows two routers connected by two LANs, with S having three connections to one of the LANs. (The MAC addresses are shown as single-digit numbers for brevity.) When router R adds S to the known set, it chooses S's two lowest-numbered MAC addresses, 7 and 8.



(e) system ID *(f) circuit ID* *(g) LAN address* *(h) count nodes not links*

Figure 6-1: Load Splitting

Finally, when comparing the number of links with MSD, count the number of nodes traversed, not the number of partial paths. Part (h) of Figure 6-1 illustrates this point. When A adds D to the Known Set, it considers the number of paths to D to be two. So it splits the A-to-D load

between B and C. For the B part of that, it has already decided to split traffic between the two links.

There is an alternative acceptable implementation. Instead of focusing on the node being added, look at the set of next-hop adjacencies and prune these to, at most, MSD. With this approach, node A in Figure 6-1 part (h) splits traffic destined to D two ways rather than three.

6.4. Information Used and Not Used

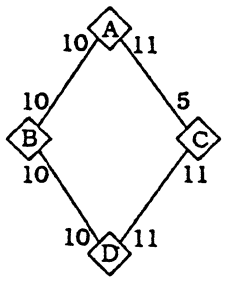
The Decision Process makes a special provision for an overloaded router; that is, for a router in the area having the LSP database overload bit set in its LSP number zero. Data traffic whose final destination is in the overloaded system itself is delivered to that system. (This leaves the door open for management action to diagnose the problem and perhaps even to reconfigure things and extricate the router from its predicament.) However, data traffic is not routed through an overloaded router to any other system—not to other routers and not to end nodes.

Implementing this provision involves the step in Dijkstra's algorithm where a node is added to the shortest-path list. At this step, the router checks whether the node being added is overloaded. If so, it adds it anyway, as normal, but after that point all outgoing links from the overloaded router are disregarded for the remainder of the Decision Process.

The Decision Process does not use a link between two routers unless each router reports a link to the other router.

Note: There is no way to determine that the links reported by the two routers are the same link. This is not a problem. Data traffic is still forwarded over the (possibly asymmetric) paths.

It is possible for the two routers to report different costs for the link. In this case, routes may be asymmetric. Figure 6-2 shows an example.



Numbers shown are link costs reported by the nearby router.

Note that A and C report different values for the same link.

The consequence is that traffic from A to D goes through B, while traffic from D to A goes through C. The total cost from A to D is 20; for D to A, the total cost is 16.

Figure 6-2: Example of an Asymmetric Route.

The IPX Network Numbers in the routing area are accumulated from the Management Information and External Route options of LSPs in the Link State database. It is possible that a Network Number is accessible both through NLSP and through RIP. That is, it appears in the Management Information option of one LSP and the External Route option of another. The Decision Process ignores the external route in this case.

6.5. Products of the Decision Process

The Forwarding database consists of a set of pairs (Network Number, Next Hop). The Next Hop indicates on which circuit, and to which neighbor on that circuit, traffic to the Network Number should be immediately forwarded by this router. If load splitting applies (page 6-4), there might be more than one Next Hop for a Network Number.

In certain cases (for example, LSP database overload, or a recent network partition), some Network Numbers might be known but unreachable.

If the Decision Process determines that certain nodes are unreachable, the router still keeps the associated LSPs in the Link State database. They persist until the LSP to which they refer expires or is deleted. The reason is that links can disappear and then reappear soon. Because a link may provide reachability to a large number of networks, it is inadvisable to delete them all only to re-create them soon after.

6.6. Routing in the Face of a LAN Partition

While running the Decision Process, a router might discover two Designated Routers for the same IPX network number. Consider Figure 6-3. The LAN with IPX Network number *N* can become partitioned if the MAC-level bridge fails. Routers *A*, *B*, and C all still consider the LAN to have IPX network number *N*. By the election process, both *A* and *B* could become designated routers for *N*.

For traffic whose final destination is network *N*, the routers *R*, *S*, and *T* have no way to know whether *A* or *B* is the route to choose. By default, they choose whichever is closer to them. Reaching the destination is problematic. There is not much that can be done to improve on this situation.

For other routing decisions, however, the situation is better in the face of this partition. Consider traffic from *R* destined for network *M*. By tracing the adjacencies, *R* readily determines that the path *R*-*S*-*B*-C-*T* will reach network *M*. Likewise, router *A* can reach network *M* by a circuitous route.

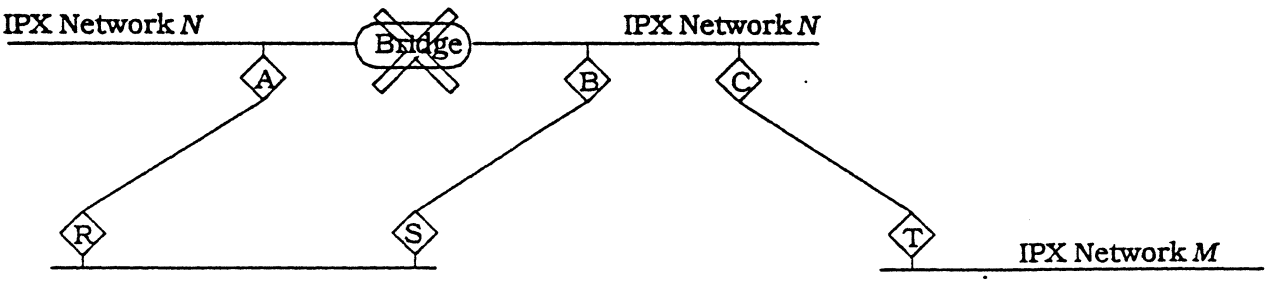


Figure 6-3: LAN Partition Example

Other hardware failure modes can cause partitions. Suppose, for example, that (instead of the bridge failing) the network interface connecting *B* to the LAN *N* were to fail. Again, both *A* and *B* could emerge as Designated Routers for *N*. But this time, *A* and C are adjacent, instead of *B*

and C. *R* and *S* can route to anywhere through *A*. In addition, the *S-B* link is used from anywhere to reach SAP services residing on system B (because they are associated with *B*'s internal network number, not with network *N*).

6.7. Routing outside the Routing Area

Although this specification encompasses Level 1 routing only, the Level 1 router must implement certain features to operate in an internetwork that includes both Level 1 and Level 2.

6.7.1. Calculating the Actual Area Address

Each router reports its manualAreaAddresses in the Area Addresses option of its LSP number zero. The Decision Process collects these values. The collected list forms the set of synonymous area address values that define and delimit the routing area. The combination is called the actualAreaAddress.

Note: Level 1 routers from other areas can share circuits with routers of this area, but they will fail to form adjacencies—an adjacency is formed only if the two routers have at least one manual area address in common. Consequently, LSPs from other routing areas do not appear in the Link State database.

Note: The collection of area addresses includes all systems in the Link State database, whether currently reachable or not. It also includes those in this router's own Level 1 LSP number zero.

If there are more than three area addresses in the collected list, the router retains three of them, as follows:

a) If the address parts are unequal, the lowest address is preferred.

b) Otherwise, the lowest mask is preferred.

For example, (0xC9000000,0xFF000000) is preferred over (0xC9000000, 0xFFFF0000).

If the preceding steps cause one of the router's own manualAreaAddresses to be dropped, the event manualAddressDroppedFromArea is generated. This is a means to detect misconfigurations.

6.7.2. Routing to an Exit Router

The key is to locate the nearest Level 2 router. This is done by examining the Attached Flag fields in the LSPs in the Level 1 Link State database. The flag is significant in each router's LSP number zero. (The value of the LSP's Router Type field is not taken into consideration.) When set, this flag indicates that the router originating it is an *Exit Router*; that is, it offers a path to other routing areas, using Level 2 routing (or other suitable method). The Decision Process determines the closest Exit Router to this system; that is, the one whose NLSP cost is smallest. Figure 6-4 illustrates an example of an exit router, E. Its Attached Flag is set to indicate that it can reach outside the routing area. It also is a Level 1 router, and (for example) conveys traffic from A to D as necessary. The links to the right of E are not in the Link State database. Exactly how E routes outside the area is not part of the NLSP specification. It could be by Level 2 routing, or it could use a different method.

Load splitting can be applied, as with other routes. The router chooses one exit router and splits the load between paths to that router. (As an implementation option, a router can split the load between two or more equally distant exit routers.)

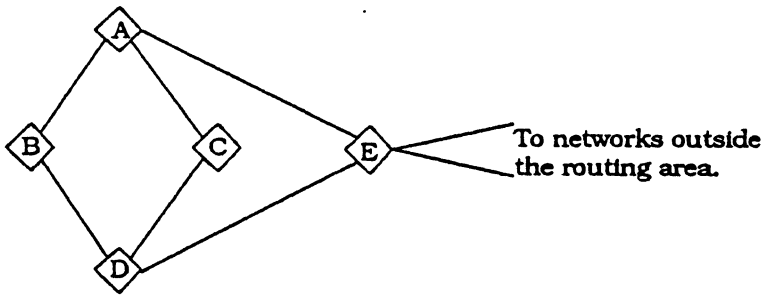


Figure 6-4: Exit Router Example.

6.7.3. Forwarding Data Packets

When the router forwards an IPX data packet, it proceeds as follows:

a) Look up the route in the Level 1 Forwarding database. If found, use the information to identify the corresponding next hop.

b) If not found, forward the packet to the nearest Exit Router (do not check whether the destination is in the area).

c) If in the preceding step there is no Exit Router accessible, discard the packet and generate an ipxOutNoRoutes event.

6.8. Decision Process Database

6.8.1. Configured Values

MSD
Maximal Splitting Degree: the maximum number of equal-cost paths the router uses to any one destination. Default is 1; no splitting.

6.8.2. Dynamic Values

actualAreaAddress
The collected manualAreaAddresses of the routers in the routing area. The combined set of up to three synonymous area addresses identifies the routing area.

6.8.3. NLSP Events

manualAddressDroppedFromArea
More than three manualAreaAddresses were detected in the area, and this router dropped one of its own when forming the actualAreaAddress.

ipxOutNoRoutes
An IPX packet could not be forwarded because the destination network number is unreachable and there is no reachable exit router.

7. RIP and SAP

RIP is the traditional routing protocol used with IPX. Like IPX itself, RIP is derived from the Xerox Network System specifications. SAP is a protocol used with IPX for a server to advertise availability of a service, and for clients to find that service (that is, to determine its IPX network address). Both use a broadcast mechanism. The two are usually implemented together.

As described in Section 2, NLSP includes RIP/SAP as its end node/router protocol. End nodes include

a) Clients that want to find services and routes

b) Servers that want to advertise their existence using SAP, and to find routes

This part of RIP/SAP support is an indispensable part of NLSP operation. It must always be active for all circuits. It includes

- Responding to RIP requests for route information
- Responding to SAP requests for service information
- Absorbing SAP information about services that reside on the same system as the router

The router-to-router part of RIP/SAP support is also included in the NLSP design, so that customers can deploy NLSP routers in the same internetwork with RIP routers. This *compatibility* part of RIP/SAP support can be activated and deactivated on a per-circuit basis. It includes

- Absorbing RIP broadcasts from RIP routers
- Absorbing SAP broadcasts transmitted by servers
- Generating RIP and SAP broadcasts for consumption by RIP routers and NetWare servers

The compatibility part of RIP support is conditionally operational, on a per-circuit basis. The end-user chooses between three alternatives, for each router attached to each link:

a) "On"—cause RIP packets to be generated and absorbed on the link.

b) "Off"—inhibit RIP from being generated or absorbed on the link.

c) "Auto"—If a RIP router is detected on the link, generate and absorb RIP packets; otherwise, don't.

The default choice is "Auto."

Likewise, the compatibility part of SAP support is operational only conditionally, on a per-link basis. Its activation or deactivation is independent of RIP's. The end-user chooses between the same three alternatives, with the same default.

The remainder of this chapter should be read in conjunction with Reference [Nov92].

In the diagrams of this section, square-shaped symbols represent RIP routers. Diamond-shaped symbols continue to represent NLSP routers, including their RIP/SAP-emulation roles.

7.1. Maintaining RIP and SAP Information

7.1.1. XRoutes and Services Defined

RIP Information consists of *External Routes*, abbreviated *XRoutes* in this chapter. Each XRoute summarizes the distance to a particular IPX network from the router reporting the XRoute. Specifically, each XRoute contains the following values:

- Network Number: The IPX network number to which this XRoute refers.
- Hops: The number of hops to reach that network. Each router traversed is counted as one hop.
- RIP Delay, or Ticks: The number of RIP timer ticks to reach that network. There are 18.21 ticks per second. The value is the time to deliver a 576-byte packet one way.

Each RIP packet sent or received conveys information about routes. When an NLSP router receives a RIP packet, the route information from the packet is absorbed into the Link State database as XRoutes, subject to the procedures specified in this chapter. When an NLSP router sends a RIP packet, it reports not only XRoutes, but also routes reachable by NLSP: the network numbers of routers and pseudonodes in the Link State database.

SAP information consists of *Services*. Each Service identifies a (potentially) reachable application service that is accessible using the IPX Network-Layer protocol. Each Service record contains the following values:

- Service Name: The name of the service. This is a text string up to 47 bytes. It identifies the service uniquely in the internetwork. Routers need not be concerned with the naming conventions.
- Service Type: The type of service offered. Novell assigns these values; for example, NetWare servers use the value 4. See Reference [Nov92] for a (partial) list of defined types and the contact point for registering new service types. Routers need not be concerned with the semantics of the Service Type value.
- IPX Network-Layer Address. The address at which the service resides.
- Hops: The number of hops to reach that service. Each router traversed is counted as one hop.

The Name/Type combination identifies a service. There can be several distinct Services having the same Name if their Types are different. If a Service appears with the same Name/Type combination as one previously known but at a different Address, it is an indication that the Service has changed location.

Each SAP packet sent or received is, in essence, a list of zero or more Services. The router does not transmit a SAP reporting a Service unless that router has a route to the network number on which the service resides.

7.1.2. Relation to Link State Database of Receiving RIP/SAP

RIP routes are absorbed by NLSP routers as they receive RIP packets from other RIP routers on directly connected networks. Those RIP routes that are kept as XRoutes are included in the pseudonode LSPs for the network on which they were received. The Designated Router for the network maintains the pseudonode LSP. Other NLSP routers on the network also retain

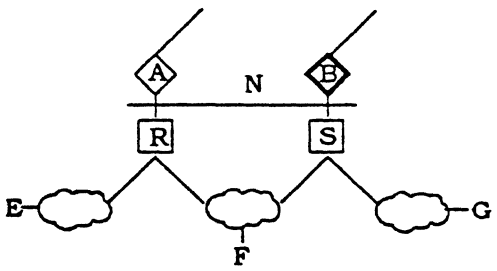
received RIP information, maintaining a *potential pseudonode*. The potential pseudonode is information that a router keeps in anticipation of possibly becoming the Designated Router and being required to produce an actual pseudonode.

When a router receives a RIP packet and is about to absorb a network number into a pseudonode (or potential pseudonode) LSP as an XRoute, it first compares that network number with the router's own manualAreaAddresses. If the network number is not within the ranges delineated by any one of the manualAreaAddresses, the XRoute is not created.

After determining that a RIP route received is not already reachable as an NLSP destination, it must be determined whether to keep that route as an XRoute in the pseudonode (or potential pseudonode) for that circuit. Exactly which XRoutes are maintained is an implementation-dependent decision (as described in Reference [Nov92], page 3-7—an implementation can keep only the best route, or all known routes, or a subset of the latter. It should be noted that keeping all known routes, or even the best route seen on each LAN, does not necessarily improve routing performance in all situations. Keeping 'not best' duplicate routes can slow convergence in the RIP when the routers are connected in loops and are attempting to delete routes. Maintaining just the 'best known' RIP routes to each RIP destination seen on all directly connected networks minimizes routing table size and reduces convergence time when deleting a route.

On each LAN, one of the NLSP routers is the Designated Router, chosen by a dynamic election procedure. Only the Designated Router propagates received RIP information in LSPs. The Designated Router is responsible for flooding the pseudonode LSP to the other NLSP routers on behalf of that network segment. For each WAN connecting an NLSP router with a RIP router, the NLSP router is the Designated Router having these responsibilities.

Other NLSP routers—those not acting as Designated Routers—also gather the RIP information, even though they do not propagate the information in LSPs they originate. First, they do so to be prepared to become the Designated Router should the need arise by failure or resignation of the current Designated Router. Second, the RIP packets are needed for the immediate address to use when forwarding data packets destined for the network number reported in the RIP. The XRoute in the pseudonode LSP does not contain the immediate address, so information from directly received RIP packets is indispensable. (Data packets are not forwarded to the Designated Router to be forwarded over the same LAN again.) Consider Figure 7-1. Network numbers E, F, and G are all attached to the pseudonode for N, which is generated by Designated Router B. However, the pseudonode does not indicate which of the RIP routers (R or S) provides access to each of the three network numbers. So when A receives data traffic from its WAN link destined for E (say), the pseudonode itself does not contain enough information to forward that traffic. To supplement the pseudonode, A keeps information it receives by RIP over the LAN. Specifically, it records that R is the path to E, that S is the path to G, and whichever of R and S is the better path to F.



A and B are NLSP routers.
B is Designated Router for N.

R and S are RIP routers.
R reports a route to networks E and F.
S reports a route to networks F and G.

Figure 7-1: Routing between NLSP and RIP

When an NLSP router is connected by a WAN circuit to a RIP router, the NLSP router constructs a WAN pseudonode (numbered or unnumbered) for the circuit and acts as the WAN Designated Router for that pseudonode. There is no election procedure. RIP and SAP packets it receives over the circuit are advertised as XRoutes and Services in the pseudonode LSP.

Incoming RIP packets from NLSP routers are ignored. Those packets are presumed to have been sent for backward compatibility. An NLSP router is recognized by its datalink address matching that of a router for which an NLSP adjacency exists in either the "Up" or the "Initializing" state.

From the Link State database, each router builds an exactly matching Link State graph, which includes the XRoutes. Each XRoute has forwarding information (hops and ticks) relative to the network where its existence was discovered. For a WAN pseudonode, the Designated Router assigns the NLSP Delay to be 15 milliseconds and the NLSP Throughput is calculated from the measurements of the Timer Request/Response exchange in Section 3:

```
start_time = time when Timer Request packet is sent, 1/18 second

end_time = time that matching Timer Response packet is received, 1/18 second

Throughput = 8 × 1,000 × 576 /
                [ { 55 × MAX ( 1, end_time - start_time) / 2 } - 15 ]
```

SAP packets and Services are like RIP packets and XRoutes. Received SAP packets report network services that are absorbed into Services Information options of pseudonode LSPs, flooded, and kept synchronized. The rules for absorbing SAPs are the same as for RIP: check the network number against manualAreaAddresses, and check that the sender is not an NLSP router. Each Service has forwarding information (hops) relative to the network where its existence was discovered.

One additional rule applies to absorbing SAP information. Recall that the router does not transmit a SAP reporting a Service unless that router has a route to the network number on which the service resides (page 7-2). To facilitate meeting this requirement, a Service is not accepted into a pseudonode (or potential pseudonode) unless either

- There is an XRoute in the same (potential) pseudonode for the network on which the Service resides

  Or

- The SAP's network number is the network number of the circuit itself (not an internal network number

Because this rule is followed everywhere, a router does not have to check each remote Service received through LSPs to make sure that there is a route to that Service (beyond the Link State graph reachability tests). Nor does it have to try and find all Services with that specific network number each time a network number is removed from the forwarding database.

7.1.3. Relation to Link State Database of Sending RIP/SAP

When sending RIP and SAP packets on a LAN, a router's behavior is the same whether or not it is the Designated Router for the LAN.

RIP routers include in the ticks value the cost of the network onto which they are broadcasting a RIP packet. This value is included in LSPs. This must be taken into account when reporting ticks in outgoing RIP packets: the value must be adjusted back to duplicating that portion in the report. For example, consider Figure 7-2. When RIP router C reports about LAN P by broadcasting a RIP packet on LAN M, it reports two ticks: one for P and one for M. The pseudonode LSP for M includes an XRoute for P showing two ticks. Then when NLSP router B broadcasts a RIP on LAN N describing P, it reports three ticks, not four. The three includes provision for N itself.

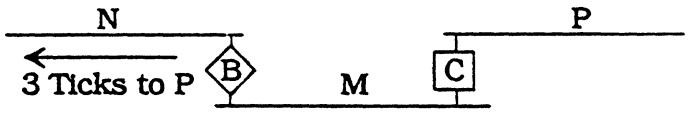


Figure 7-2: Adjusting the Ticks in RIP

There is an additional provision for the internal network numbers of routers (either RIP or NLSP). An "additional" tick is added to traverse the (conceptual) gap from the router entity within the system to the internal network number within the same system. There is no corresponding additional hop. An internal network number is considered to be 1 tick and 0 hops away from the router entity within the same system. In Figure 7-3, suppose NLSP router A has internal network number AA, NLSP router B has internal network number BB, and RIP router R has internal network number RR. When A sends a RIP on LAN N advertising AA, it reports 1 hop and 2 ticks. When it sends a RIP on LAN N advertising BB, it reports 2 hops and 3 ticks. The 3 are as follows: one for the network being transmitted onto, one for the AB link, and one "additional" tick from router B to its internal network BB. When A sends a RIP on LAN N advertising RR, it bases the ticks on the XRoute in the pseudonode representing LAN M. The tick value in the pseudonode is 2, and already contains the "additional" tick within R. So A adds one for the AP link and one for N, making 4.

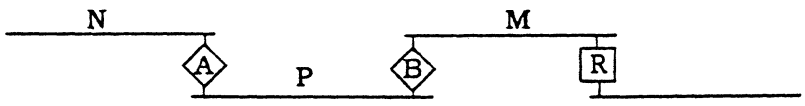


Figure 7-3: Additional Tick for Internal Networks

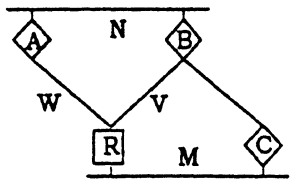
Counting hops is easier than dealing with ticks. The adjustments described for ticks do not apply. Each traversal of a router is one hop. The total hop count is reported in both RIP and SAP broadcasts. For example, when sending a SAP onto network N, router A reports 1 hop to a service residing on its internal network AA, two hops to a service on BB, three hops to a service on network M, and three hops to a service on network RR. In the latter case, the pseudonode for M shows one hop to a service on RR, because R sends a SAP for the service having hop count 1.

Each NLSP router continues to receive RIP/SAP broadcasts and to keep the information current in its local database, in its potential pseudonode. However, RIP/SAP broadcasts sent by the NLSP router are based on the XRoute information in the Link State graph built by the Decision Process. This means that RIP/SAP broadcasts by NLSP routers reflect only the Designated Router's view of that particular LAN. The broadcasts change in content only when the Designated Router detects a change. (By the Split Horizon rule described later, the broadcasts are being sent on a different LAN than the one where the information was learned.) In Figure 7-3, for example, suppose A is the Designated Router for LAN P. When B sends RIP/SAP information about P onto M, it uses information from the pseudonode built by A representing P; it does not use information from RIP/SAP packets B itself received over P (even though it absorbs that information for other purposes).

7.1.4. XRoutes, NLSP Routes, Services, and the Decision Process

The Decision Process constructs a Link State graph of the routing area. An *NLSP route* is a path in the Link State graph using only network numbers that appear in the Management Information field of LSPs. As will be seen in the following discussion, NLSP routes over other routes (which include External Route fields of LSPs).

There can be several XRoutes in LSPs for the same destination. The Link State database keeps track of them all. Figure 7-4 illustrates this situation.



Link State graph from A's viewpoint.
Circles are NLSP nodes; triangles are pseudonodes.
R appears next to nodes where its internal network number is attached as an XRoute.

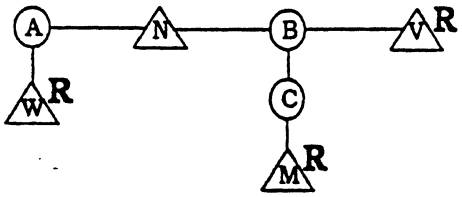
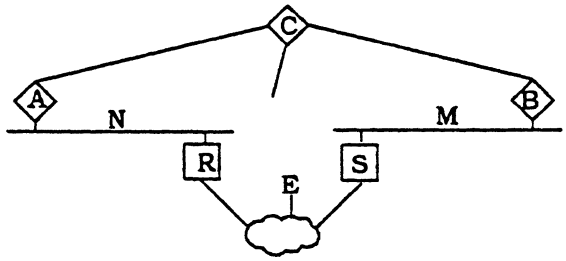


Figure 7-4: XRoutes Can Appear More than Once

The pseudonode contains the best XRoute known on that LAN to a particular network. But if multiple LANs have paths to the same network, information about that network can appear in each LAN's LSP. Consider Figure 7-5. Network E appears as an XRoute in two pseudonodes: the one for LAN N and the one for LAN M. When router C forwards data traffic destined for E, it chooses the best route based on the total tick count for the combined NLSP and RIP portions of the path. (The Decision Process calculates ticks for the NLSP portion of the path, as described on page 7-8.) In case of a tie, the total hop count is used. If there is still a tie, the choice of path is arbitrary. Accordingly, C forwards E's traffic to either A or B (or in case of tied hops and ticks, it can load split between the two).



A, B, and C are NLSP routers.

R and S are RIP routers.
Both report paths to network E..

The pseudonodes of both network N and network M report paths to E.

Figure 7-5: Multiple XRoutes for a Destination

The Decision Process determines the best route for inclusion in the Forwarding database, based on the ticks and hops. Only the portion of the Link State graph in the shortest path spanning tree is included in the paths considered to XRoute destinations.

Likewise, the router discovering SAP information determines the smallest hop count to that Service, for each network address at which it is accessible, for inclusion in the pseudonode (or potential pseudonode). The same Service name can appear more than once. For example, if a service resides on System C in Figure 7-4, it is attached to Link State nodes M, W, V, and C.

When XRoute or Service information changes, the Link State method propagates the information throughout the routing area. This implies updating the Forwarding database of each router accordingly. But the computational burden is comparatively small. The Decision Process actually consists of two parts:

a) Run Dijkstra's algorithm to build the Link State graph and spanning tree. This includes determining the costs to every NLSP destination.

b) Determine the best costs of XRoutes based on the spanning tree and the RIP costs reported at nodes of the tree.

XRoutes and Services are like leaf nodes connected to various LSP nodes of the Link State graph. So, when an XRoute or Service appears, disappears, or changes in ticks or hops, there is no structural change to the graph. A router performs only part b). Because there are no structural changes to the graph, there is no need to run Dijkstra's algorithm.

To avoid running Dijkstra's algorithm when RIP routes become unreachable, it is necessary to link together all RIP routes to the same destination (regardless of cost).

7.1.5. Building a RIP Route from the Link State Graph

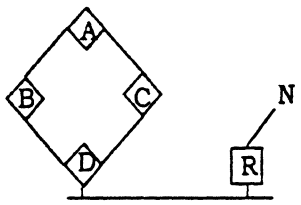
As part of the Decision Process, each router maintains hop count and RIP Delay values for each Link State graph node's *upstream route*; that is, the route from itself to that node. Call this the *Upstream RIP Delay*. The value is the sum of individual RIP Delay values of the links the route traverses. For each link, the individual RIP Delay is calculated from the NLSP Delay and Throughput values. For a LAN, the calculation is

```
MAX [ 1, ( ( 576 × 8 / Throughput) + (Delay / 1,000,000 ) ) × 18 ]
```

For a WAN, the calculation is

```
MAX [ 1, ( ( 576 × 64 / Throughput) + (2 × Delay / 1,000,000 ) ) × 18 ]
```

When summing the individual RIP Delay values, special attention is needed when load splitting is in force. It can happen that several paths have equal NLSP costs—so load splitting is applied—but with different RIP Delay values. In this case, apply the highest of the several RIP Delay values to the path. Figure 7-6 shows an example.



NLSP router A is load-splitting between B and C to reach D.

That is, AB+BD = AC+CD, with values in NLSP's metric.

The RIP Delay from A to D is MAX(AB+BD, AC+CD), where this time the values are the individual RIP Delays.

Figure 7-6: RIP Delay and Load Splitting

XRoutes maintain hop count and ticks that are relative to the LSP to which they are attached.

Using

a) The upstream RIP delays (defined previously) stored in the graph nodes

b) The values reported in XRoutes

this router can generate RIP route values to all reachable IPX networks, as follows:

- If a network is an internal route (that is, it appears in the graph as a non-XRoute), then
  a) Its RIP ticks value is the upstream RIP Delay
  b) Its RIP hop count is the number of links traversed by the downstream route (excluding WAN pseudonodes)

- If a network is an XRoute, then one starts with the downstream hop/tick values (as above) to the pseudonode where it is reported, and adds the respective hop/tick values reported in the External Routes field.

Likewise, the hops from this router to a Service is computed by adding

a) The number reported in the Services Information field

   To

b) The number of graph links (excluding pseudonodes) to the pseudonode whose LSP includes the Services Information.

7.1.6. Aging XRoutes and Services

XRoutes and Services are aged in a way similar to the one RIP routers use to age entries.

An aging timer is applied to each XRoute and Service in the local potential (and actual) pseudonode LSPs maintained by this router. When a RIP Response arrives, each route reported has its timer reinitialized if it is already represented by an XRoute. Likewise for SAP and Services.

The interval at which RIP is activated for its periodic broadcast is ripUpdate. For SAP, the corresponding value is sapUpdate.

The aging timer after which a stale XRoute is purged from its pseudonode is ripAgeMultiplier × ripUpdate. The aging timer after which a stale Service is purged from its pseudonode is sapAgeMultiplier × sapUpdate.

If the timer expires, the XRoute or Service is removed from the LSP. If an XRoute is removed from a potential pseudonode LSP, all Services residing on that network number are also removed from that potential pseudonode LSP. If this router is the Designated Router for that LSP, this change in the pseudonode LSP results in removal of the XRoute from the Link State graph. This causes a triggered update to reflect the changes to the XRoutes. As with other triggered RIP/SAP updates, this causes routers to update their respective Forwarding databases, but does not require Dijkstra's algorithm to be run.

7.2. Generating Periodic Updates

If a router (or link) were to stop operation without warning, a way is required for other routers to discover the loss of connectivity. This is accomplished by periodically broadcasting current RIP/SAP information and aging any database entries not refreshed by received broadcasts.

All NLSP routers generate RIP periodic update broadcasts every ripUpdate for its connected links supporting RIP compatibility. The default is 60 seconds, but the interval is configurable on a per-link basis.

> **Note:** The ability to configure the periodic broadcast interval and the timeout to age out RIP routes and SAP entries is a recent design, not supported by all installed systems. Older versions of IPX RIP/SAP routers use 60-second periodic timers and a four-minute timeout. All routers on any one network segment (LAN or WAN) must be configured with the same values.

The network numbers to report in these broadcasts are all the IPX network numbers that are reachable—link-state destinations (pseudonodes, internal networks) and XRoutes alike. In RIP, reporting a hop count of 16 means that the network is unreachable. Only destinations whose total hop count is less than 16 are reported in the periodic updates. (This is not affected

by a possible configured node limit greater than 16.) When sending a RIP update reporting 16 hops, the ticks value is unspecified.

If more than 50 routes are to be reported, multiple RIP packets are used, each with 50 entries or fewer. If more than seven services are to be reported, multiple SAP packets are used, each with seven entries or fewer. These packet size limitations can be relaxed by an overriding user configuration on a per-circuit basis. The value ripPacketSize determines the size (in bytes) that a RIP packet is allowed to reach. The corresponding value for SAP is sapPacketSize.

7.3. Split Horizon

RIP implements a split horizon heuristic to cut down on sending and processing redundant RIP traffic. Split horizon applies equally whether a route is learned by RIP or NLSP. There are two parts:

a) A router about to broadcast onto a particular circuit, C, does not include any information about other networks for which another router on C has a better route. For example, in Figure 7-7, suppose B is a RIP router and C is an NLSP router. When router C broadcasts a RIP packet on LAN M, it does not include information received from B about LAN N. (Otherwise, a system on M might erroneously conclude that there are two paths to N.) Moreover, if A is an NLSP router, it does not broadcast RIP packets onto N reporting routes in the network cloud Z (it does not matter whether E is a RIP or an NLSP router).

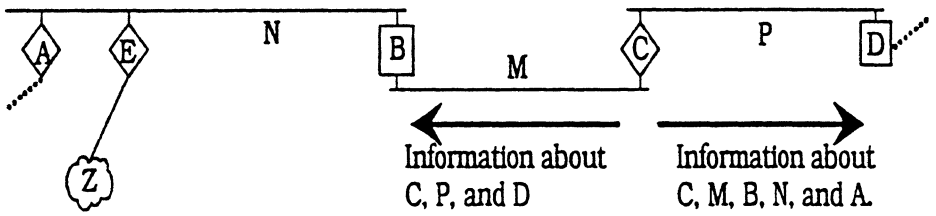


Figure 7-7: Split Horizon

A way to implement this heuristic is to ask the following questions when preparing to broadcast information about network H onto circuit J: "Is J the best next hop to H? Is J one of the candidate equal-cost best hops to H?" If the answer to either of these questions is "yes," then refrain from broadcasting information about network H onto circuit J. In the equal-cost path case, it does not matter which of the paths are actually chosen for forwarding. Figure 7-8 illustrates this. In this example, the LANs have equal cost. When A runs the Decision Process, it notes that it can reach H through either B or C with equal cost. Suppose the maximal splitting degree is one, and A chooses B as the forwarding path to H. Nonetheless, A does not broadcast RIP information about H onto J, because C is one of the candidate equal-cost best hops from A to H.

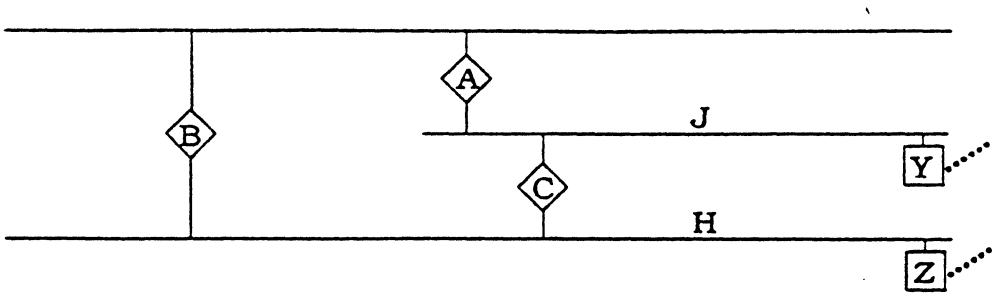


Figure 7-8: Split Horizon and Load Splitting

b) A router about to broadcast on a particular segment does not include any information about the network on which the packet is to be sent. Router C does not include information about M in its RIP broadcast on M.

The RIP Delay values in outgoing RIP reports on a link include provision for that link itself. On the other hand, hops correspond to router traversals. For example, when C transmits a RIP on M describing P, the report indicates "1 hop, 2 ticks"—one tick transits M; another transits P.

SAP broadcasts also conform with the split horizon rule.

7.4. Generating Triggered Updates

When there are changes in the routes that have been broadcast as RIP routes, or when the router discovers changes to local links, or when internally provoked events occur (for example, activating or deactivating a circuit), triggered updates of RIP routes are broadcast on circuits supporting RIP compatibility.

The form and content of triggered updates are the same as those of periodic broadcasts.

7.4.1. Changes in XRoutes and Services

When an XRoute is added to or deleted from the database, a triggered update is generated to reflect the changes. One is also generated when an XRoute's forwarding information (hops, ticks) changes.

When a router detects that a route becomes unreachable, it generates a triggered update to convey this. The cause can be (a) the Decision Process finds a network to become unreachable, or (b) the hop count reaches 16 or more.

Likewise, SAP updates are generated when Services are added or deleted from the Link State database, or there is a change in hop count as reported in LSPs.

When the router detects that a network number becomes inaccessible, it marks Services on that network as inaccessible as well, and sends triggered SAP updates accordingly.

When a change is detected, there is a hold-down timer of 0.55 seconds before transmitting the resulting RIP or SAP packet. This allows multiple changes occurring close in time to be consolidated.

7.4.2. Changes in the Link State Graph

RIP triggered updates are generated when the following occurs:

- Nodes are added to the Link State graph
- Something changes in the Link State database that changes the Hops or Ticks (a link going up or down, a cost changing, or similar change)
- Any node's upstream route first hop changes, or its upstream hop count or IPX Delay changes
- Any node becomes unreachable

Only the new or changed routes are broadcast. Even if an XRoute does not itself change, the ticks and hops to it can change when there are changes to the link-state graph between this router and the pseudonode to which the XRoute is attached.

If prior to some change a node or XRoute was within the Hop Count range for RIP (15 hops), but after the change it was outside the RIP range, it is included in at least one update with its unreachable hop count, so that neighboring RIP routers delete that route.

Likewise, SAP triggered updates are generated when an LSP having Services is added to or deleted from the Link State database, or when a Link State graph change causes the hop count for a service to cross the 16-hop threshold (in one direction or the other).

When the router detects that a network number becomes inaccessible, it marks Services on that network as inaccessible.

When a change is detected, there is a hold-down timer of 0.55 seconds before transmitting the resulting RIP or SAP packet. This allows multiple changes occurring close in time to be consolidated.

7.4.3. Circuit Activation and Deactivation

When a directly attached circuit becomes active, the router's actions depend on the configured mode for that circuit.

- If the RIP compatibility mode is "Off," there is no RIP-compatibility action to perform.
- If the RIP compatibility mode is "On,"
  a) Send the first NLSP Hello on the circuit.
  b) Transmit a complete RIP update on the circuit.
  c) Start the timer that controls periodic broadcasts.
  d) Broadcast a RIP "All Routes" request on the circuit.
- If the RIP compatibility mode is "Auto,"
  a) Send the first NLSP Hello on the circuit.
  b) Broadcast a RIP "All Routes" request on the circuit.
  c) If a RIP response is received from a non-NLSP router, the first time this happens
     i) Transmit a complete RIP update on the circuit.
     ii) Start the timer that controls periodic broadcasts.
     iii) Tag the circuit as having RIP activated.

Note: An NLSP router is recognized by its datalink address matching that of a router for which an NLSP adjacency exists in either the "Up" or the "Initializing" state.

The result of this is that with NLSP active and RIP in "Auto" mode, RIP periodic responses and triggered updates are sent only if there are other non-NLSP routers on a LAN. With RIP compatibility mode "On," RIP packets are always sent.

The same procedure is used for SAP, depending on the SAP support configuration choice for the circuit.

It is possible that another NLSP router responds to the RIP request and this router receives the response before it recognizes that the other router is an NLSP router. In this case, the adjacency is formed later, and the other router's LSPs begin to arrive. The RIP routes previously learned from that router duplicate the information in the now expanding Link State database. To prevent having that duplicate information in pseudonode LSPs, the router removes any XRoutes that were absorbed from RIP packets from that router. If the router is in "Auto" mode for that link, and if the RIPs just described had been the ones that activated RIP compatibility for the circuit, the RIP compatibility is deactivated when the XRoutes are removed.

When a directly attached circuit supporting SAP compatibility becomes active, the router broadcasts a SAP general request to discover services accessible over the circuit. It then starts the timer for generating periodic updates. As soon as any SAP packet is received from a non-NLSP router, it acts as though the periodic interval has expired and broadcasts the Service information contained in its database (with split horizon applied).

When a directly attached circuit is about to become inactive, the router sends RIP/SAP updates on that circuit indicating that destinations reached through the router through that link are now unreachable. (There is no hold-down timer applied for this part, and no interpacket gap is required.) It then generates triggered RIP/SAP updates to all the other RIP/SAP-support circuits about the routes/services that have become unreachable as a result of the deletion.

7.4.4. Router Activation and Deactivation

When a router begins operation,

a) It determines the network numbers of its directly attached RIP-support links.

b) If the system itself contains application services, it records the service names as Services in its database.

c) It performs the circuit activation procedures described on page 7-12 for each circuit.

When router operation is about to terminate, the system performs the circuit deactivation procedures described on page 7-13 for each circuit.

7.5. Receiving RIP and SAP Packets

When a valid RIP packet arrives on circuit *C*, the router takes the following steps:

a) If *C* is a LAN, and the packet's source MAC datalink address matches that of a router for which an NLSP adjacency exists, the packet is ignored.

> **Note:** The other router is an NLSP router doing RIP compatibility. Ignoring the packet prevents this router from having to process RIP routes for destinations it already knows about through LSPs.

b) If the packet is a RIP Request,

i) If the request is for a specific route (or routes), the response is returned to the requester's address. Information is returned no matter how large the hop count is.

Note: This allows end nodes to access the larger networks possible through NLSP. With RIP/SAP alone, without NLSP, the maximum supported network diameter is 15 hops.

The split horizon rule is applied. In other words, the router responds with any information normally sent in a periodic broadcast packet that would be applicable to the request.

One exception to the split horizon rule occurs if a specific RIP request is received on a network segment for the network number of the segment itself. The router responds with the requested information. Even though the information is not useful for routing, it allows a system to learn the node addresses of all routers on a local network segment.

ii) If the request is a general request (for "All Routes"), the response is returned to the requester's address only if the circuit is configured to be in "On" or "Auto" mode. If the mode is "Off," a general request is ignored. Responding to a general request is the same as generating a periodic update. Entries that would have a total reported hop count of 16 or greater are not included. This prevents other RIP routers from having to process entries that they automatically discard. Split horizon is applied.

c) If the packet is a RIP Response,

i) If the circuit has RIP compatibility configured to be "Off," the packet is ignored.

ii) The source and destination network numbers in the IPX header are compared. If they are not the same, the packet is ignored.

iii) For each route in the RIP Response, the Link State database is searched.

- If the destination network is reachable by an NLSP route (that is, by a path in the Link State graph using only network numbers that appear in the Management Information field of LSPs), the RIP route is ignored.
- If the destination is not a Link State destination, this RIP route information is searched for in the local existing database. That is, if this is the Designated Router for C, it is searched for in the pseudonode LSP; if it is not the Designated Router, it is searched for in the potential pseudonode.
  - If the route is absent, it is added.
  - If it is present from the same RIP router, changes in value are recorded and the timeout for the XRoute is refreshed.
  - If it is present from a different RIP router, and if the route received is better than the one in the database, the old information is replaced by the new.
  - If this is the Designated Router, any change is reflected in the pseudonode LSP, triggering an LSP regeneration.

A router can be maintaining more than one pseudonode (or potential pseudonode)—different ones for different directly connected network segments. When absorbing a RIP route, the router implementation has discretion. On the one hand, it can add XRoute to all the pseudonodes of networks on which RIP is heard. An alternative is to add the XRoute only to the pseudonode (or potential pseudonode) closest to the RIP destination—or several, in case of a tie. Intermediate choices are also valid. Keeping 'only the...closest' minimizes convergence time when deleting a route.

Any time a received RIP packet indicates a hop count of 16, it means that the network number cannot be reached through the router sending the packet. If this is the only path to that network number, it (the network number) is removed from the pseudonode (or potential pseudonode) and a triggered update results.

The logic to process arriving SAP packets is the same as for RIP. The router is responsible not only for absorbing the SAP packets as they are received, but also for representing SAP services detected within the router's own system. Services advertised on the router's own internal network are always accepted, and those services are added to the local LSP, the one describing this NLSP router.

As with RIP XRoutes, SAP Services discovered by SAPs arriving over directly attached circuits are maintained as part of the corresponding pseudonode LSP (or potential pseudonode). If this router is the Designated Router for that pseudonode, it (the router) floods the pseudonode LSP throughout the routing area. As with RIP, SAP packets arriving from another NLSP router (performing SAP compatibility) are ignored.

7.6. Maintaining a Proper Interpacket Gap

Certain nodes can have problems properly processing a sequence of RIP packets arriving too close together. Currently, the minimum interpacket gap time allowed for transmitting RIP packets is 55 milliseconds.

Similarly, the minimum interpacket gap time allowed for transmitting SAP packets is currently 55 milliseconds.

7.7. RIP and SAP Filters

An NLSP router can be configured on a per-link basis to filter incoming and outgoing reports of RIP routes (by network number or a pattern of numbers) and Services (by service name or name pattern). The end-user is responsible for maintaining uniformity in the configured filters. In Figure 7-9, for example, if A has different RIP/SAP filters on the LAN than B does, router C perceives radical changes in what is accessible through R if the Designated Router status changes on the LAN.

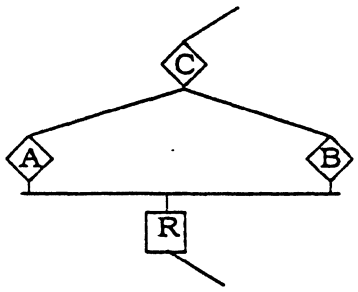


Figure 7-9: Filter Uniformity

Filtering applies to the transmission and absorption of information by the RIP/SAP protocols. It does not apply to exchange of XRoute and Service information in LSPs. Once information about a route or service is in one router's LSP, that information is propagated throughout the routing area. This means that users can control the amount of RIP/SAP traffic, but cannot necessarily prevent RIP/SAP information from reaching another region of an internetwork, if NLSP routers provide connectivity to that region.

7.8. RIP/SAP Database

7.8.1. Configured Values per Circuit

ripState
: Mode of support for RIP on the circuit: "On," "Off," or "Auto." Default is "Auto."

ripUpdate
: The RIP periodic update interval, in seconds. Default is 60 seconds.

ripAgeMultiplier
: After this many ripUpdate intervals, unrefreshed RIP information is considered expired. Default is four; it ages out after three or four minutes.

ripPacketSize
: The maximum RIP packet size used on this circuit. Default is 432 bytes (header plus 50 entries).

sapState
: Mode of support for SAP on the circuit: "On," "Off," or "Auto." Default is "Auto."

sapUpdate
: The SAP periodic update interval, in seconds. Default is 60 seconds.

sapAgeMultiplier
: After this many sapUpdate intervals, unrefreshed SAP information is considered expired. Default is four; it ages out after three or four minutes.

sapPacketSize

: The maximum SAP packet size used on this circuit. Default is 480 bytes (IPX header plus seven entries).

8. Network Management

The management of NLSP routers uses the (SNMP) running over IPX, as specified in [Wor92]. As an implementation option, the same management information can be accessed by SNMP running over IP or other Network-Layer protocol supported by the router.

This section defines the SNMP managed objects for the NLSP. The managed objects are gathered into groups. The following list itemizes the groups and characterizes the management information in each group:

System Group	Global information about the IPX protocol entity.
Circuit Group	Information about each interface on the system.
Forwarding Group	Forwarding database used to route data traffic.
Services Group	Known Services that advertise themselves using SAP.
Neighbors Group	Information about neighboring NLSP routers and the adjacencies with them.
LSP Group	Tables representing the LSP database.
Translation Group	Various mappings (for example, System ID to Server Name).
Graph Group	Representation of the network topology.

There are three MIBs, each contained in a separate subsection below:

- IPX MIB includes basic information about IPX Network-Layer operation. It is applicable to all IPX-speaking systems, both routers and end nodes. It includes information in the following groups: System, Circuit, Forwarding, and Services.
- NLSP MIB augments IPX MIB with information specific to the NLSP routing protocol. It extends the System, Circuit, and Forwarding groups, and adds the Neighbors, Translation, Graph, and LSP groups.
- RIP/SAP MIB augments IPX MIB with information specific to the RIP and SAP protocols. It extends the System and Circuit groups.

As routing protocols are extended and new ones devised, each addition augments the basic IPX MIB by adding to the existing groups or designing additional protocol-specific groups.

The managed objects defined in this section are presented in the concise MIB definition syntax specified in Reference [Ros91]. The same MIB definitions are also made available in electronic form. Every effort has been made to keep the material in this section identical to the electronic form, but in case of any discrepancy the electronic form is authoritative.

3.1. IPX MIB

```
IPX DEFINITIONS ::= BEGIN

-- This MIB defines the management information for a system using the IPX
-- protocol. The MIB consists of four groups:
--
--   1. System Group - contains general information about all instances
--                     of IPX on the system
--
--   2. Circuit Group - contains information about all circuits used by
--                      IPX on the system
--
--   3. Forwarding Group - contains generic routing information that
--                         must be provided by any IPX routing protocol.
--
--   4. Services Group - contains information about all known services.
--
-- The MIB is designed to support multiple instances of the IPX
-- protocol on one system via a system instance identifier which is the
-- primary index for every table in this MIB.
--
-- This MIB is designed to provide a basic framework for the management
-- of systems implementing the IPX protocol. Additional MIBs may be
-- created (especially in the area of IPX routing protocols) to contain
-- more specific information. Whenever possible, these additional MIBs
-- should follow the format of this IPX MIB. Information in these MIBs
-- should be linked to this MIB via the use of the system instance
-- identifier mentioned above.


IMPORTS
        enterprises, Counter
                FROM RFC1155-SMI
        OBJECT-TYPE
                FROM RFC-1212
        PhysAddress
                FROM RFC-1213;

novell  OBJECT IDENTIFIER ::= { enterprises 23 }
experimental  OBJECT IDENTIFIER ::= { novell 4 }
ipx     OBJECT IDENTIFIER ::= { experimental 11 }

-- Groups

ipxSystem OBJECT IDENTIFIER ::= {ipx 1}
ipxCircuit OBJECT IDENTIFIER ::= {ipx 2}
ipxForwarding OBJECT IDENTIFIER ::= {ipx 3}
ipxServices OBJECT IDENTIFIER ::= {ipx 4}

-- Types

NetNumber ::= OCTET STRING (SIZE(4))

-- System Group
--  This group contains global information about each instance of IPX
--  running on one system.

-- System Table
--  This table contains one entry for each instance of IPX running on
--  the system.

ipxSysTable OBJECT-TYPE
  SYNTAX     SEQUENCE OF IPXSysEntry
  ACCESS     not-accessible
  STATUS     mandatory
  DESCRIPTION "The IPX System table."
  ::= {ipxSystem 1}

ipxSysEntry OBJECT-TYPE
  SYNTAX     IPXSysEntry
  ACCESS     not-accessible
  STATUS     mandatory
  DESCRIPTION "Each entry corresponds to one instance of IPX running
          on the system."
  INDEX      {ipxSysInstance}
  ::= {ipxSysTable 1}

IPXSysEntry ::= SEQUENCE {
              ipxSysInstance
                INTEGER,
              ipxSysExistState
                INTEGER,
              ipxSysIntNetNumExists
                INTEGER,
              ipxSysIntNetNum
                NetNumber,
              ipxSysName
                OCTET STRING,
              ipxSysMaxPathSplits
                INTEGER,
              ipxSysMaxHops
                INTEGER,
              ipxSysVersionMajor
                INTEGER,
              ipxSysVersionMinor
                INTEGER,
              ipxSysInReceives
                Counter,
              ipxSysInTooManyHops
                Counter,
              ipxSysInHdrErrors
                Counter,
              ipxSysInUnknownSockets
                Counter,
              ipxSysInFiltered
                Counter,
              ipxSysInCompressDiscards
                Counter,
              ipxSysInDiscards
                Counter,
              ipxSysInDelivers
                Counter,
              ipxSysNETBIOSPackets
                Counter,
              ipxSysForwPackets
                Counter,
              ipxSysOutRequests
                Counter,
              ipxSysOutNoRoutes
                Counter,
              ipxSysOutFiltered
                Counter,
              ipxSysOutCompressDiscards
                Counter,
              ipxSysOutMalformedRequests
                Counter,
              ipxSysOutDiscards
                Counter,
              ipxSysOutPackets
                Counter,
              ipxSysCircCount
                Counter,
              ipxSysDestCount
                Counter,
              ipxSysServCount
                Counter,
              ipxSysResourceFailures
                Counter,
```

```
xSysConfigSockets
 Counter,
xSysMaxOpenSockets
 Counter
xSys[illegible]SocketFails
 Coun[illegible]


 OBJECT-TYPE
 INTEGER
 read-write
 mandatory
ON "The unique identifier of the instance of IPX to which this
corresponds. This value may be written only when
ting a new entry in the table."
try 1}

te OBJECT-TYPE
 INTEGER {
  off(1),
  on(2)
 }
 read-write
 mandatory
ON "The validity of this entry in the IPX system table.
ing this field to off indicates that this entry may be
ted from the system table at the IPX implementation's
retion."
try 2}

umExists OBJECT-TYPE
 INTEGER {
  no(1),
  yes(2)
 }
 read[illegible]
 mand[illegible]y
ON "Indicates whether this instance of IPX has an internal
work number."
ntry 3}

lum OBJECT-TYPE
 NetNumber
 read-write
 mandatory
ON "The IPX internal network number of this instance of IPX.
s value is undefined if the value of
rSysIntNetNumExists is no."
ntry 4}

OBJECT-TYPE
 OCTET STRING (SIZE(0..48))
 read-write
 mandatory
ION "The readable name for this system."
ntry 5}

thSplits OBJECT-TYPE
 INTEGER (1..32)
 read-write
 mandatory
ION "The maximum number of paths with equal routing metric
ue which this instance of the IPX may split
ween when forwarding packets."
Entry 6}

ps C[illegible]T-TYPE
 INTEGER
 read-write
 mandatory
 DESCRIPTION "The maximum number of hops a packet may take."
 ::= {ipxSysEntry 7}

ipxSysVersionMajor OBJECT-TYPE
 SYNTAX    INTEGER
 ACCESS    read-only
 STATUS    mandatory
 DESCRIPTION "The major version number of IPX supported."
 ::= {ipxSysEntry 8}

ipxSysVersionMinor OBJECT-TYPE
 SYNTAX    INTEGER
 ACCESS    read-only
 STATUS    mandatory
 DESCRIPTION "The minor version number of IPX supported."
 ::= {ipxSysEntry 9}

ipxSysInReceives OBJECT-TYPE
 SYNTAX    Counter
 ACCESS    read-only
 STATUS    mandatory
 DESCRIPTION "The total number of IPX packets received, including
        those received in error."
 ::= {ipxSysEntry 10}

ipxSysInTooManyHops OBJECT-TYPE
 SYNTAX    Counter
 ACCESS    read-only
 STATUS    mandatory
 DESCRIPTION "The number of IPX packets discarded due to exceeding the
        maximum hop count."
 ::= {ipxSysEntry 11}

ipxSysInHdrErrors OBJECT-TYPE
 SYNTAX    Counter
 ACCESS    read-only
 STATUS    mandatory
 DESCRIPTION "The number of IPX packets discarded due to errors in
        their headers, including any IPX packet with a size less
        than the minimum of 30 bytes."
 ::= {ipxSysEntry 12}

ipxSysInUnknownSockets OBJECT-TYPE
 SYNTAX    Counter
 ACCESS    read-only
 STATUS    mandatory
 DESCRIPTION "The number of IPX packets discarded because the
        destination socket was not open."
 ::= {ipxSysEntry 13}

ipxSysInFiltered OBJECT-TYPE
 SYNTAX    Counter
 ACCESS    read-only
 STATUS    mandatory
 DESCRIPTION "The number of incoming IPX packets discarded due to
        filtering."
 ::= {ipxSysEntry 14}

ipxSysInCompressDiscards OBJECT-TYPE
 SYNTAX    Counter
 ACCESS    read-only
 STATUS    mandatory
 DESCRIPTION "The number of incoming IPX packets discarded due to
        decompression errors."
 ::= {ipxSysEntry 15}

ipxSysInDiscards OBJECT-TYPE
 SYNTAX    Counter
 ACCESS    read-only
 STATUS    mandatory
```

```
  DESCRIPTION "The number of IPX packets received but discarded due to
          processing decision."
  ::= {ipxSysEntry 16}

ipxInDelivers OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The total number of IPX packets delivered locally,
          including packets from local applications."
  ::= {ipxSysEntry 17}

ipxSysNETBIOSPackets OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "."
  ::= {ipxSysEntry 18}

ipxSysForwPackets OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of IPX packets forwarded."
  ::= {ipxSysEntry 19}

ipxSysOutRequests OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of IPX packets supplied locally for
          transmission, not including any packets counted in
          ipxForwPackets."
  ::= {ipxSysEntry 20}

ipxSysOutNoRoutes OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of IPX packets discarded because no route was
          found."
  ::= {ipxSysEntry 21}

ipxSysOutFiltered OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of outgoing IPX packets discarded due to
          filtering."
  ::= {ipxSysEntry 22}

ipxSysOutCompressDiscards OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of outgoing IPX packets discarded due to
          compression errors."
  ::= {ipxSysEntry 23}

ipxSysOutMalformedRequests OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of IPX packets supplied locally that contained
          errors in their structure."
  ::= {ipxSysEntry 24}

ipxSysOutDiscards OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of outgoing IPX packets discarded due to
          processing decision."
  ::= {ipxSysEntry 25}

ipxSysOutPackets OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The total number of IPX packets transmitted."
  ::= {ipxSysEntry 26}

ipxSysCircCount OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of circuits known to this instance of IPX."
  ::= {ipxSysEntry 27}

ipxSysDestCount OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of currently reachable destinations known to
          this instance of IPX."
  ::= {ipxSysEntry 28}

ipxSysServCount OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of services known to this instance of IPX."
  ::= {ipxSysEntry 29}

ipxSysResourceFailures OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times this instance of the IPX has been
          unable to obtain needed resources (memory, etc.)"
  ::= {ipxSysEntry 30}

ipxSysConfigSockets OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The configured maximum number of IPX sockets that may be
          open at one time."
  ::= {ipxSysEntry 31}

ipxSysMaxOpenSockets OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The maximum number of IPX sockets actually open at one
          time by this system."
  ::= {ipxSysEntry 32}

ipxSysOpenSocketFails OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of IPX socket open calls which failed."
  ::= {ipxSysEntry 33}


-- Circuit Group
-- This group contains management information for each circuit known
-- to this system.
```

```
-- Circuit Table
-- The Circuit table contains an entry for each circuit known to the
-- system.

   rcTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF IPXCircEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The Circuit table."
  ::= {ipxCircuit 1}

ipxCircEntry OBJECT-TYPE
  SYNTAX    IPXCircEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry corresponds to one circuit known to the
          system."
  INDEX     {
          ipxCircSysInstance,
          ipxCircIndex
          }
  ::= {ipxCircTable 1}

IPXCircEntry ::= SEQUENCE {
              ipxCircSysInstance
                INTEGER,
              ipxCircIndex
                INTEGER,
              ipxCircExistState
                INTEGER,
              ipxCircOperState
                INTEGER,
              ipxCircIfIndex
                INTEGER,
              ipxCircName
                OCTET STRING,
              ipxCircInfo
                OCTET STRING,
              ipxCircType
                INTEGER,
              ipxCircLocalMaxPacketSize
                INTEGER,
              ipxCircCompressState
                INTEGER,
              ipxCircCompressSlots
                INTEGER,
              ipxCircCompressedSent
                INTEGER,
              ipxCircUncompressedSent
                INTEGER,
              ipxCircMediaType
                INTEGER,
              ipxCircNetNumber
                NetNumber,
              ipxCircStateChanges
                Counter,
              ipxCircInitFails
                Counter,
              ipxCircDelay
                INTEGER,
              ipxCircThroughput
                INTEGER
              }

ipxCircSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of IPX
          to which this entry corresponds. This value may be
          written only when creating a new entry in the table."
  ::= {ipxCircEntry 1}

ipxCircIndex OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The identifier of this circuit, unique within the
          instance of IPX. This value may be written
          only when creating a new entry in the table."
  ::= {ipxCircEntry 2}

ipxCircExistState OBJECT-TYPE
  SYNTAX    INTEGER {
              off(1),
              on(2)
              }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The validity of this circuit entry. A circuit with this
          value set to off may be deleted From the table at the
          IPX implementation's discretion."
  ::= {ipxCircEntry 3}

ipxCircOperState OBJECT-TYPE
  SYNTAX    INTEGER {
              off(1),
              on(2)
              }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The operational state of the circuit."
  ::= {ipxCircEntry 4}

ipxCircIfIndex OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The value of ifIndex for the interface used by this
          circuit. This value may be written only when creating
          a new entry in the table."
  ::= {ipxCircEntry 5}

ipxCircName OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(0..20))
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The readable name for the circuit."
  ::= {ipxCircEntry 6}

ipxCircInfo OBJECT-TYPE
  SYNTAX    OCTET STRING
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "Additional readable information for the circuit. The
          content of this field is implementation defined."
  ::= {ipxCircEntry 7}

ipxCircType OBJECT-TYPE
  SYNTAX    INTEGER {
              other(1),
              broadcast(2),
              ptToPt(3),
              wanRIP(4),
              unnumberedRIP(5),
              dynamic(6),
              wanWS(7)
              }
  ACCESS    read-write
  STATUS    mandatory
```

```
  DESCRIPTION "The type of the circuit."
  ::= {ipxCircEntry 8}

ipxCircLocalMaxPacketSize OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The maximum size (including header), in bytes, that the
          system supports locally on this circuit."
  ::= {ipxCircEntry 9}

ipxCircCompressState OBJECT-TYPE
  SYNTAX    INTEGER {
              off(1),
              on(2)
            }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The compression state on this circuit. This value may
          be written only when creating a new entry in the table."
  ::= {ipxCircEntry 10}

ipxCircCompressSlots OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The number of compression slots available on this
          circuit. This value may be written only when creating a
          new entry in the table."
  ::= {ipxCircEntry 11}

ipxCircCompressedSent OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of compressed packets sent."
  ::= {ipxCircEntry 12}

ipxCircUncompressedSent OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of packets sent without being compressed
          even though compression was turned on for this circuit."
  ::= {ipxCircEntry 13}

ipxCircMediaType OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(2))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The media type used on this circuit."
  ::= {ipxCircEntry 14}

ipxCircNetNumber OBJECT-TYPE
  SYNTAX    NetNumber
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The IPX network number to which this circuit is bound."
  ::= {ipxCircEntry 15}

ipxCircStateChanges OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times the circuit has changed state."
  ::= {ipxCircEntry 16}

ipxCircInitFails OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times that initialization of this
          circuit has failed."
  ::= {ipxCircEntry 17}

ipxCircDelay OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The period of time, in milliseconds, that it takes to
          transmit one byte of data, excluding protocol headers,
          to a destination on the other end of the circuit, if
          the circuit is free of other traffic."
  ::= {ipxCircEntry 18}

ipxCircThroughput OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The amount of data, in bits per second, that may flow
          through the circuit if there is no other traffic."
  ::= {ipxCircEntry 19}


-- Forwarding Group
-- This group provides a representation of the forwarding database used
-- by all instances of IPX on the system.

-- Destination Table
-- The Destination table contains information about all known
-- destinations.

ipxDestTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF IPXDestEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The Destination table contains information about all
          known destinations."
  ::= {ipxForwarding 1}

ipxDestEntry OBJECT-TYPE
  SYNTAX    IPXDestEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry corresponds to one destination."
  INDEX     {
          ipxDestSysInstance,
          ipxDestNetNum
          }
  ::= {ipxDestTable 1}

IPXDestEntry ::= SEQUENCE {
              ipxDestSysInstance
                INTEGER,
              ipxDestNetNum
                NetNumber,
              ipxDestProtocol
                INTEGER,
              ipxDestCost
                INTEGER,
              ipxDestHopCount
                INTEGER,
              ipxDestNextHopCircIndex
                INTEGER,
              ipxDestNextHopNICAddress
                OCTET STRING,
              ipxDestNextHopNetNum
                NetNumber,
              ipxDestType
                INTEGER
```

```
        }

ipxDestSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of IPX
        to which this row corresponds. This value may be written
        only when creating a new entry with ipxDestProtocol
        equal to static."
  ::= {ipxDestEntry 1}

ipxDestNetNum OBJECT-TYPE
  SYNTAX    NetNumber
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The IPX network number of the destination. This value
        may be written only when creating a new entry with
        ipxDestProtocol equal to static."
  ::= {ipxDestEntry 2}

ipxDestProtocol OBJECT-TYPE
  SYNTAX    INTEGER {
              other(1),
              local(2),
              rip(3),
              nlsp(4),
              static(5)
            }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The routing protocol from which knowledge of this
        destination was obtained. This value may be written only
        when creating a new entry with ipxDestProtocol equal
        to static."
  ::= {ipxDestEntry 3}

ipxDestCost OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The cost to reach this destination. The meaning of the
        cost value is dependent on the routing protocol (i.e.
        Cost = delay in ticks for RIP, Cost = total path default
        cost for NLSP, etc.). This value may be written only
        when creating a new entry with ipxDestProtocol equal
        to static."
  ::= {ipxDestEntry 4}

ipxDestHopCount OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The number of hops necessary to reach the destination.
        This value may be written only when creating a new entry
        with ipxDestProtocol equal to static."
  ::= {ipxDestEntry 5}

ipxDestNextHopCircIndex OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the circuit used to reach the
        next hop. This value may be written only when creating
        a new entry with ipxDestProtocol equal to static."
  ::= {ipxDestEntry 6}

ipxDestNextHopNICAddress OBJECT-TYPE
  SYNTAX    OCTET STRING
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The NIC address of the next hop. This value may be
        written only when creating a new entry with
        ipxDestProtocol equal to static."
  ::= {ipxDestEntry 7}

ipxDestNextHopNetNum OBJECT-TYPE
  SYNTAX    NetNumber
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The IPX network number of the next hop. This value may
        be written only when creating a new entry with
        ipxDestProtocol equal to static."
  ::= {ipxDestEntry 8}

ipxDestType OBJECT-TYPE
  SYNTAX    INTEGER {
              unknown(1),
              nlspLevel1Router(2),
              router(4),
              network(5)
            }
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "Indicates whether the destination is a network or a
        router."
  ::= {ipxDestEntry 9}


-- Services Group
-- The Services group contains management information for all known
-- services.

-- Services Table
--  This table contains the services information indexed by service
--  name and type.

ipxServTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF IPXServEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The table of services, indexed by name and type."
  ::= {ipxServices 1}

ipxServEntry OBJECT-TYPE
  SYNTAX    IPXServEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry corresponds to one service."
  INDEX     {
        ipxServSysInstance,
        ipxServName,
        ipxServTypeValue
        }
  ::= {ipxServTable 1}

IPXServEntry ::= SEQUENCE {
            ipxServSysInstance
              INTEGER,
            ipxServName
              OCTET STRING,
            ipxServTypeValue
              OCTET STRING,
            ipxServType
              INTEGER,
            ipxServProtocol
              INTEGER,
            ipxServNetNum
              NetNum,
            ipxServNode
```

```
            OCTET STRING,
          ipxServSocket
            OCTET STRING
          }

ipxServSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of IPX
        to which this entry corresponds. This value may be
        written only when creating entries with the value of
        ipxServProtocol equal to static."
  ::= {ipxServEntry 1}

ipxServName OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(1..48))
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The service name. This value may be written only when
        creating entries with the value of ipxServProtocol
        equal to static."
  ::= {ipxServEntry 2}

ipxServTypeValue OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(2))
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The service type's hexadecimal value. This value may be
        written only when creating entries with the value of
        ipxServProtocol equal to static."
  ::= {ipxServEntry 3}

ipxServType OBJECT-TYPE
  SYNTAX    INTEGER {
              unknown(1)
            }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The service type. This value may be written only when
        creating entries with the value of ipxServProtocol
        equal to static."
  ::= {ipxServEntry 4}

ipxServProtocol OBJECT-TYPE
  SYNTAX    INTEGER {
              other(1),
              local(2),
              nlsp(4),
              static(5),
              sap(6)
            }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The protocol from which knowledge of this service was
        obtained. This value may be written only when creating
        entries with the value of ipxServProtocol equal to
        static."
  ::= {ipxServEntry 5}

ipxServNetNum OBJECT-TYPE
  SYNTAX    NetNumber
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The IPX network number portion of the IPX address of the
        service. This value may be written only when creating
        entries with the value of ipxServProtocol equal to
        static."
  ::= {ipxServEntry 6}

ipxServNode OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(6))
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The node portion of the IPX address of the service. This
        value may be written only when creating entries with the
        value of ipxServProtocol equal to static."
  ::= {ipxServEntry 7}

ipxServSocket OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(2))
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The socket portion of the IPX address of the service.
        This value may be written only when creating entries with
        the value of ipxServProtocol equal to static."
  ::= {ipxServEntry 8}


-- Destination Services Table
-- This table contains the services information indexed by address,
-- name, and type.

ipxDestServTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF IPXDestServEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The table of services, indexed by address, name,
        and type."
  ::= {ipxServices 2}

ipxDestServEntry OBJECT-TYPE
  SYNTAX    IPXDestServEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry corresponds to one service."
  ::= {ipxDestServTable 1}

IPXDestServEntry ::= SEQUENCE {
            ipxDestServSysInstance
              INTEGER,
            ipxDestServNetNum
              NetNumber,
            ipxDestServNode
              OCTET STRING,
            ipxDestServSocket
              OCTET STRING,
            ipxDestServName
              OCTET STRING,
            ipxDestServTypeValue
              OCTET STRING,
            ipxDestServType
              INTEGER,
            ipxDestServProtocol
              INTEGER
            }

ipxDestServSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of IPX
        to which this entry corresponds. This value may be
        written only when creating entries with the value of
        ipxDestServProtocol equal to static."
  ::= {ipxDestServEntry 1}

ipxDestServNetNum OBJECT-TYPE
  SYNTAX    NetNumber
  ACCESS    read-write
```

```
  STATUS      mandatory
  DESCRIPTION "The IPX network number portion of the IPX address of the
          service. This value may be written only when creating
          entries with the value of ipxDestServProtocol equal to
          static."
  ::= {ipxDestServEntry 2}

ipxDestServNode OBJECT-TYPE
  SYNTAX      OCTET STRING (SIZE(6))
  ACCESS      read-write
  STATUS      mandatory
  DESCRIPTION "The node portion of the IPX address of the service. This
          value may be written only when creating entries with the
          value of ipxDestServProtocol equal to static."
  ::= {ipxDestServEntry 3}

ipxDestServSocket OBJECT-TYPE
  SYNTAX      OCTET STRING (SIZE(2))
  ACCESS      read-write
  STATUS      mandatory
  DESCRIPTION "The socket portion of the IPX address of the service.
          This value may be written only when creating entries with
          the value of ipxDestServProtocol equal to static."
  ::= {ipxDestServEntry 4}

ipxDestServName OBJECT-TYPE
  SYNTAX      OCTET STRING (SIZE(1..48))
  ACCESS      read-write
  STATUS      mandatory
  DESCRIPTION "The service name. This value may be written only when
          creating entries with the value of ipxDestServProtocol
          equal to static."
  ::= {ipxDestServEntry 5}

ipxDestServTypeValue OBJECT-TYPE
  SYNTAX      OCTET STRING (SIZE(2))
  ACCESS      read-write
  STATUS      mandatory
  DESCRIPTION "The servie type's hexadecimal value. This value may be
          written only when creating entries with the value of
          ipxDestServProtocol equal to static."
  ::= {ipxDestServEntry 6}

ipxDestServType OBJECT-TYPE
  SYNTAX      INTEGER {
                unknown(1)
              }
  ACCESS      read-write
  STATUS      mandatory
  DESCRIPTION "The service type. This value may be written only when
          creating entries with the value of ipxDestServProtocol
          equal to static."
  ::= {ipxDestServEntry 7}

ipxDestServProtocol OBJECT-TYPE
  SYNTAX      INTEGER {
                other(1),
                local(2),
                nlsp(4),
                static(5),
                sap(6)
              }
  ACCESS      read-write
  STATUS      mandatory
  DESCRIPTION "The protocol from which knowledge of this service was
          obtained. this value may be written only when the value
          of ipxDestServProtocol is equal to static."
  ::= {ipxDestServEntry 8}

END
```

8.2. NLSP MIB

```
NLSP DEFINITIONS ::= BEGIN

-- This MIB defines the management information for the NLSP protocol
-- running in an IPX enviroment. It provides information in addition
-- to that contained in the IPX MIB itself. All tables in this MIB are
-- linked to an instance of IPX via the system instance identifier as
-- defined in the IPX MIB.


IMPORTS
  enterprises, Counter
        FROM RFC1155-SMI
  OBJECT-TYPE
        FROM RFC-1212
  PhysAddress
        FROM RFC-1213;

novell OBJECT IDENTIFIER ::= {enterprises 23}
experimental OBJECT IDENTIFIER ::= {novell 4}
nlsp  OBJECT IDENTIFIER ::= {experimental 9}


-- Groups

nlspSystem      OBJECT IDENTIFIER ::= {nlsp 1}
nlspCircuit     OBJECT IDENTIFIER ::= {nlsp 2}
nlspForwarding  OBJECT IDENTIFIER ::= {nlsp 3}
nlspNeighbors   OBJECT IDENTIFIER ::= {nlsp 4}
nlspTranslation OBJECT IDENTIFIER ::= {nlsp 5}
nlspGraph       OBJECT IDENTIFIER ::= {nlsp 6}
nlspLSP         OBJECT IDENTIFIER ::= {nlsp 8}


-- Types

SystemID  ::= OCTET STRING (SIZE(6))
NLSPID    ::= OCTET STRING (SIZE(7))
NetNumber ::= OCTET STRING (SIZE(4))


-- System Group
--  This group contains global information about each instance of NLSP
--  running on one system.

-- System Table
--  This table contains an entry for each instance of NLSP running on
--  the system.

nlspSysTable OBJECT-TYPE
  SYNTAX      SEQUENCE OF NLSPSysEntry
  ACCESS      not-accessible
  STATUS      mandatory
  DESCRIPTION "The NLSP system table."
  ::= {nlspSystem 1}

nlspSysEntry OBJECT-TYPE
  SYNTAX      NLSPSysEntry
  ACCESS      not-accessible
  STATUS      mandatory
  DESCRIPTION "Each entry corresponds to one instance of NLSP running
          on the system."
  INDEX       {nlspSysInstance}
  ::= {nlspSysTable 1}

NLSPSysEntry ::= SEQUENCE {
                  nlspSysInstance
                    INTEGER,
```

```
        nlspSysState
          INTEGER,
        nlspSysID
          SystemID,
        nlspSysMinNonBcastLSPTransInt
          INTEGER,
        nlspSysMinBcastLSPTransInt
          INTEGER,
        nlspSysMinLSPGenInt
          INTEGER,
        nlspSysMaxLSPGenInt
          INTEGER,
        nlspSysMaxLSPAge
          INTEGER,
        nlspSysBcastHelloInt
          INTEGER,
        nlspSysNonBcastHelloInt
          INTEGER,
        nlspSysDRBcastHelloInt
          INTEGER,
        nlspSysHoldTimeMultiplier
          INTEGER,
        nlspSysCompSNPInt
          INTEGER,
        nlspSysPartSNPInt
          INTEGER,
        nlspSysWaitTime
          INTEGER,
        nlspSysOrigL1LSPBufSize
          INTEGER,
        nlspSysVersion
          INTEGER,
        nlspSysCorrLSPs
          Counter,
        nlspSysL1Overloaded
          INTEGER,
        nlspSysL1DbaseOverloads
          Counter,
        nlspSysMaxSeqNums
          Counter,
        nlspSysSeqNumSkips
          Counter,
        nlspSysTransmittedLSPs
          Counter,
        nlspSysReceivedLSPs
          Counter,
        nlspSysOwnLSPPurges
          Counter,
        nlspSysVersionErrors
          Counter,
        nlspSysIncorrectPackets
          Counter,
        nlspSysNearestL2DefaultExists
          INTEGER,
        nlspSysNearestL2DefaultRouter
          SystemID
        }

nlspSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP to which this
        corresponds.  This value links the instance of NLSP to an
        instance of IPX running on the system  (i.e. the value
        of nlspSysInstance should be the same as a value of
        ipxSysInstance).  This value may be written only when
        creating a new entry in the table."
  ::= {nlspSysEntry 1}

nlspSysState OBJECT-TYPE
  SYNTAX    INTEGER {
              off(1),
              nlspLevel1Router(2)
              }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "Indicates the operational state of this instance of NLSP."
  ::= {nlspSysEntry 2}

nlspSysID OBJECT-TYPE
  SYNTAX    SystemID
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The system ID for this instance of NLSP."
  ::= {nlspSysEntry 3}

nlspSysMinNonBcastLSPTransInt OBJECT-TYPE
  SYNTAX    INTEGER (1..30)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The minimum interval, in seconds, between transmission
        of LSPs on a non-broadcast circuit."
  ::= {nlspSysEntry 4}

nlspSysMinBcastLSPTransInt OBJECT-TYPE
  SYNTAX    INTEGER (1..30)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The minimum interval, in seconds, between transmission
        of LSPs on a broadcast circuit."
  ::= {nlspSysEntry 5}

nlspSysMinLSPGenInt OBJECT-TYPE
  SYNTAX    INTEGER (1..30)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The minimum interval, in seconds, between the generation
        of the same LSP."
  ::= {nlspSysEntry 6}

nlspSysMaxLSPGenInt OBJECT-TYPE
  SYNTAX    INTEGER (1..50000)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The maximum interval, in seconds, between the generation
        of the same LSP."
  ::= {nlspSysEntry 7}

nlspSysMaxLSPAge OBJECT-TYPE
  SYNTAX    INTEGER (1..50000)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The value, in seconds, placed in the lifetime field of
        LSPs generated by this instance of the IPX Router."
  ::= {nlspSysEntry 8}

nlspSysBcastHelloInt OBJECT-TYPE
  SYNTAX    INTEGER (1..100)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The interval, in seconds, at which NLSP Hellos will be
        sent on a broadcast circuit, if this system is not the
        designated router."
  ::= {nlspSysEntry 9}

nlspSysNonBcastHelloInt OBJECT-TYPE
  SYNTAX    INTEGER (1..100)
  ACCESS    read-write
  STATUS    mandatory
```

```
  DESCRIPTION "The interval, in seconds, at which NLSP Hellos will be
        sent on a non-broadcast circuit."
  ::= {nlspSysEntry 10}

nlspSysDRBcastHelloInt OBJECT-TYPE
  SYNTAX    INTEGER (1..100)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The interval, in seconds, at which the designated
        router sends NLSP Hellos on a broadcast circuit."
  ::= {nlspSysEntry 11}

nlspSysHoldTimeMultiplier OBJECT-TYPE
  SYNTAX    INTEGER (2..20)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The holding time multiplier used to specify the holding
        time for NLSP neighbor entries as a function of the NLSP
        Hello interval."
  ::= {nlspSysEntry 12}

nlspSysCompSNPInt OBJECT-TYPE
  SYNTAX    INTEGER (1..600)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The interval, in seconds, between generation of Complete
        Sequence Number Packets by a designated router on a
        broadcast circuit."
  ::= {nlspSysEntry 13}

nlspSysPartSNPInt OBJECT-TYPE
  SYNTAX    INTEGER (1..60)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The minimum interval, in seconds, between transmission of
        Partial Sequence Number Packets."
  ::= {nlspSysEntry 14}

nlspSysWaitTime OBJECT-TYPE
  SYNTAX    INTEGER (1..300)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The number of seconds to delay in the waiting state
        before entering the on state."
  ::= {nlspSysEntry 15}

nlspSysOrigL1LSPBufSize OBJECT-TYPE
  SYNTAX    INTEGER (512..4096)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The maximum size of Level 1 LSPs and SNPs originated by
        this instance of the IPX Router."
  ::= {nlspSysEntry 16}

nlspSysVersion OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The version number of this instance of NLSP."
  ::= {nlspSysEntry 17}

nlspSysCorrLSPs OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of corrupt LSPs detected."
  ::= {nlspSysEntry 18}

nlspSysL1OverLoaded OBJECT-TYPE
  SYNTAX    INTEGER {
              no(1),
              yes(2)
            }
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "Indicates whether the NLSP Level 1 database is overloaded."
  ::= {nlspSysEntry 19}

nlspSysL1DbaseOloads OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times the NLSP Level 1 LSP database has
        become overloaded."
  ::= {nlspSysEntry 20}

nlspSysMaxSeqNums OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times the router has attempted to exceed
        NLSP's maximum sequence number."
  ::= {nlspSysEntry 21}

nlspSysSeqNumSkips OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times a sequence number skip has occurred."
  ::= {nlspSysEntry 22}

nlspSysTransmittedLSPs OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of LSPs transmitted by this system."
  ::= {nlspSysEntry 23}

nlspSysReceivedLSPs OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of LSPs received by this system."
  ::= {nlspSysEntry 24}

nlspSysOwnLSPPurges OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times a zero-aged copy of the router's own
        LSP has been received from some other node."
  ::= {nlspSysEntry 25}

nlspSysVersionErrors OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times that a received NLSP packet was
        rejected because its version number was invalid."
  ::= {nlspSysEntry 26}

nlspSysIncorrectPackets OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times that an incorrectly formatted NLSP
        packet was received."
  ::= {nlspSysEntry 27}

nlspSysNearestL2DefaultExists OBJECT-TYPE
```

```
    SYNTAX    INTEGER {
                no(1),
                yes(2)
              }
    ACCESS    read-only
    STATUS    mandatory
    DESCRIPTION "Indicates whether this instance of the IPX Router knows
            of a NLSP Level 2 router that currently can reach other
            areas using the default metric."
    ::= {nlspSysEntry 28}

nlspSysNearestL2DefaultRouter OBJECT-TYPE
    SYNTAX    SystemID
    ACCESS    read-only
    STATUS    mandatory
    DESCRIPTION "The system ID of the nearest NLSP Level 2 router that
            currently can reach other areas using the default
            metric. The value is undefined if the value of
            nlspSysNearestL2DefaultExists is no."
    ::= {nlspSysEntry 29}


-- System Area Address Table
-- The System Area Address table contains the area addresses configured
-- for NLSP.

nlspSysAreaTable OBJECT-TYPE
    SYNTAX    SEQUENCE OF NLSPSysAreaEntry
    ACCESS    not-accessible
    STATUS    mandatory
    DESCRIPTION "The System Area Address table contains the area addresses
            configured for NLSP."
    ::= {nlspSystem 2}

nlspSysAreaEntry OBJECT-TYPE
    SYNTAX    NLSPSysAreaEntry
    ACCESS    not-accessible
    STATUS    mandatory
    DESCRIPTION "Each entry in the table corresponds to one NLSP
            System Area Address."
    INDEX     {
            nlspSysAreaSysInstance,
            nlspSysAreaNet,
            nlspSysAreaMask
          }
    ::= {nlspSysAreaTable 1}

NLSPSysAreaEntry ::= SEQUENCE {
                    nlspSysAreaSysInstance
                      INTEGER,
                    nlspSysAreaNet
                      OCTET STRING,
                    nlspSysAreaMask
                      OCTET STRING
                   }

nlspSysAreaSysInstance OBJECT-TYPE
    SYNTAX    INTEGER
    ACCESS    read-write
    STATUS    mandatory
    DESCRIPTION "The unique identifier of the instance of NLSP and IPX
            (via ipxSysInstance) to which this row corresponds."
    ::= {nlspSysAreaEntry 1}

nlspSysAreaNet OBJECT-TYPE
    SYNTAX    OCTET STRING (SIZE(4))
    ACCESS    read-write
    STATUS    mandatory
    DESCRIPTION "The network address portion of the area address."
    ::= {nlspSysAreaEntry 2}

nlspSysAreaMask OBJECT-TYPE
    SYNTAX    OCTET STRING (SIZE(4))
    ACCESS    read-write
    STATUS    mandatory
    DESCRIPTION "The mask portion of the area address."
    ::= {nlspSysAreaEntry 3}


-- Actual Area Address Table
-- The Actual Area Address table contains the area addresses actually
-- used by NLSP.

nlspActAreaTable OBJECT-TYPE
    SYNTAX    SEQUENCE OF NLSPActAreaEntry
    ACCESS    not-accessible
    STATUS    mandatory
    DESCRIPTION "The Actual Area Address table contains the area addresses
            actually used by NLSP."
    ::= {nlspSystem 3}

nlspActAreaEntry OBJECT-TYPE
    SYNTAX    NLSPActAreaEntry
    ACCESS    not-accessible
    STATUS    mandatory
    DESCRIPTION "Each entry in the table corresponds to one NLSP
            Actual Area Address."
    INDEX     {
            nlspActAreaSysInstance,
            nlspActAreaNet,
            nlspActAreaMask
          }
    ::= {nlspActAreaTable 1}

NLSPActAreaEntry ::= SEQUENCE {
                    nlspActAreaSysInstance
                      INTEGER,
                    nlspActAreaNet
                      OCTET STRING,
                    nlspActAreaMask
                      OCTET STRING
                   }

nlspActAreaSysInstance OBJECT-TYPE
    SYNTAX    INTEGER
    ACCESS    read-write
    STATUS    mandatory
    DESCRIPTION "The unique identifier of the instance of NLSP and IPX
            (via ipxSysInstance) to which this row corresponds."
    ::= {nlspActAreaEntry 1}

nlspActAreaNet OBJECT-TYPE
    SYNTAX    OCTET STRING (SIZE(4))
    ACCESS    read-write
    STATUS    mandatory
    DESCRIPTION "The network address portion of the area address."
    ::= {nlspActAreaEntry 2}

nlspActAreaMask OBJECT-TYPE
    SYNTAX    OCTET STRING (SIZE(4))
    ACCESS    read-write
    STATUS    mandatory
    DESCRIPTION "The mask portion of the area address."
    ::= {nlspActAreaEntry 3}


-- Circuit Group
-- This group contains the NLSP information for each circuit known
-- to this system.
```

```
-- Circuit Table
-- The Circuit table contains an entry containing the NLSP information
-- for each circuit known to the system.

   rcTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF NLSPCircEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The Circuit table."
  ::= {nlspCircuit 1}

nlspCircEntry OBJECT-TYPE
  SYNTAX    NLSPCircEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry corresponds to one circuit known to the
          system."
  INDEX     {
          nlspCircSysInstance,
          nlspCircIndex
          }
  ::= {nlspCircTable 1}

NLSPCircEntry ::= SEQUENCE {
                nlspCircSysInstance
                  INTEGER,
                nlspCircIndex
                  INTEGER,
                nlspCircState
                  INTEGER,
                nlspCircPace
                  INTEGER,
                nlspCircHelloTimer
                  INTEGER,
                nlspCircL1DefaultCost
                  INTEGER,
                nlspCircL1DesRouterPriority
                  INTEGER,
                nlspCircL1CircID
                  OCTET STRING,
                nlspCircL1DesRouter
                  SystemID,
                nlspCircLANL1DesRouterChanges
                  Counter,
                nlspCircNeighChanges
                  Counter,
                nlspCircRejNeighbors
                  Counter,
                nlspCircOutPackets
                  Counter,
                nlspCircInPackets
                  Counter,
                nlspCircActualMaxPacketSize
                  INTEGER
                }

nlspCircSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP and IPX
          (via ipxSysInstance) to which this entry corresponds.
          This value may be written only when creating a new
          entry in the table."
  ::= {nlspCircEntry 1}

   CircIndex OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The identifier of this circuit, unique within the
          instance of NLSP.  This value may be written
          only when creating a new entry in the table."
  ::= {nlspCircEntry 2}

nlspCircState OBJECT-TYPE
  SYNTAX    INTEGER {
              off(1),
              on(2)
              }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "Indicates whether NLSP information may be sent/received
          over this circuit."
  ::= {nlspCircEntry 3}

nlspCircPace OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The maximum pace, in packets per second, at which NLSP
          packets may be sent on this circuit."
  ::= {nlspCircEntry 4}

nlspCircHelloTimer OBJECT-TYPE
  SYNTAX    INTEGER (1..100)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The interval, in seconds, between NLSP Hello packets
          sent on this circuit."
  ::= {nlspCircEntry 5}

nlspCircL1DefaultCost OBJECT-TYPE
  SYNTAX    INTEGER (1..63)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The NLSP default cost of this circuit for Level 1
          traffic."
  ::= {nlspCircEntry 6}

nlspCircL1DesRouterPriority OBJECT-TYPE
  SYNTAX    INTEGER (1..127)
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The priority for becoming the NLSP LAN Level 1
          Designated Router on a broadcast circuit."
  ::= {nlspCircEntry 7}

nlspCircL1CircID OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(7))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The NLSP ID for this circuit."
  ::= {nlspCircEntry 8}

nlspCircL1DesRouter OBJECT-TYPE
  SYNTAX    SystemID
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The system ID of the NLSP LAN Level 1 Designated Router
          on this circuit."
  ::= {nlspCircEntry 9}

nlspCircLANL1DesRouterChanges OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times the NLSP LAN Level 1 Designated
          Router has changed on this circuit."
  ::= {nlspCircEntry 10}
```

```
nlspCircNeighChanges OBJECT-TYPE
  SYNTAX     Counter
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The number of times a NLSP neighbor state change has
          occurred on this circuit."
  ::= {nlspCircEntry 11}

nlspCircRejNeighbors OBJECT-TYPE
  SYNTAX     Counter
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The number of times that a NLSP neighbor has been
          rejected on this circuit."
  ::= {nlspCircEntry 12}

nlspCircOutPackets OBJECT-TYPE
  SYNTAX     Counter
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The number of NLSP packets sent on this circuit."
  ::= {nlspCircEntry 13}

nlspCircInPackets OBJECT-TYPE
  SYNTAX     Counter
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The number of NLSP packets received on this circuit."
  ::= {nlspCircEntry 14}

nlspCircActualMaxPacketSize OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The actual maximum packet size (including header),
          in bytes, that has been used on this circuit."
  ::= {nlspCircEntry 15}


-- Forwarding Group
-- This group contains NLSP forwarding information in addition to that
-- contained in the IPX forwarding group.

-- Destination Table
-- The Destination table contains additional NLSP forwarding
-- information about all destinations learned about via NLSP.

nlspDestTable OBJECT-TYPE
  SYNTAX     SEQUENCE OF NLSPDestEntry
  ACCESS     not-accessible
  STATUS     mandatory
  DESCRIPTION "The Destination table contains information about all
          known destinations learned about via NLSP."
  ::= {nlspForwarding 1}

nlspDestEntry OBJECT-TYPE
  SYNTAX     NLSPDestEntry
  ACCESS     not-accessible
  STATUS     mandatory
  DESCRIPTION "Each entry corresponds to one destination."
  INDEX      {
          nlspDestSysInstance,
          nlspDestNetNum
          }
  ::= {nlspDestTable 1}

NLSPDestEntry ::= SEQUENCE {
              nlspDestSysInstance
                INTEGER,
              nlspDestNetNum
                NetNumber,
              nlspDestID
                NLSPID,
              nlspDestEstDelay
                INTEGER,
              nlspDestEstThroughput
                INTEGER,
              nlspDestNextHopID
                NLSPID
              }

nlspDestSysInstance OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP and IPX
          (via ipxSysInstance) to which this row corresponds."
  ::= {nlspDestEntry 1}

nlspDestNetNum OBJECT-TYPE
  SYNTAX     NetNumber
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The IPX network number of the destination."
  ::= {nlspDestEntry 2}

nlspDestID OBJECT-TYPE
  SYNTAX     NLSPID
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The destination NLSP ID (6-octet system ID plus 1-octet
          pseudo-node ID)."
  ::= {nlspDestEntry 3}

nlspDestEstDelay OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The estimated delay, in milliseconds, to reach the
          destination."
  ::= {nlspDestEntry 4}

nlspDestEstThroughput OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The estimated throughput, in bits per second, to the
          destination."
  ::= {nlspDestEntry 5}

nlspDestNextHopID OBJECT-TYPE
  SYNTAX     NLSPID
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The NLSP ID (6-octet system ID plus 1-octet pseudo-node
          ID) of the next hop."
  ::= {nlspDestEntry 5}


-- NLSP Neighbors Group
-- This group contains management information for each neighboring
-- NLSP router known to the system.

-- NLSP Neighbors Table
-- This table contains an entry for each neighboring NLSP router
-- known to the system.

nlspNeighTable OBJECT-TYPE
  SYNTAX     SEQUENCE OF NLSPNeighEntry
```

```
  ACCESS     not-accessible
  STATUS     mandatory
  DESCRIPTION "The NLSP Neighbors table."
  ::= {nlspNeighbors 1}

nlsp...eighEntry OBJECT-TYPE
  SYNTAX     NLSPNeighEntry
  ACCESS     not-accessible
  STATUS     mandatory
  DESCRIPTION "Each entry corressponds to one neighboring NLSP router
          known to the system."
  INDEX      {
          nlspNeighSysInstance,
          nlspNeighCircIndex,
          nlspNeighIndex
          }
  ::= {nlspNeighTable 1}

NLSPNeighEntry ::= SEQUENCE {
                    nlspNeighSysInstance
                      INTEGER,
                    nlspNeighCircIndex
                      INTEGER,
                    nlspNeighIndex
                      INTEGER,
                    nlspNeighState
                      INTEGER,
                    nlspNeighNICAddress
                      OCTET STRING,
                    nlspNeighSysType
                      INTEGER,
                    nlspNeighSysID
                      SystemID,
                    nlspNeighName
                      OCTET STRING,
                    nlspNeighUsage
                      INTEGER,
                    nlspNeighHoldTimer
                      INTEGER,
                    nlspNeighRemainingTime
                      INTEGER,
                    nlspNeighPriority
                      INTEGER
                    }

nlspNeighSysInstance OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP and IPX
          (via ipxSysInstance) to which this row corresponds."
  ::= {nlspNeighEntry 1}

nlspNeighCircIndex OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The identifier of the parent circuit of this neighbor
          within this instance of the NLSP and IPX."
  ::= {nlspNeighEntry 2}

nlspNeighIndex OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The identifier for this NLSP neighbor entry, unique
          within the parent circuit."
  ::= {nlspNeighEntry 3}

nlspNeighState OBJECT-TYPE
  SYNTAX     INTEGER {
                initializing(1),
                up(2),
                failed(3),
                down(4)
                }
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The state of the connection to the neighboring NLSP
          router."
  ::= {nlspNeighEntry 4}

nlspNeighNICAddress OBJECT-TYPE
  SYNTAX     OCTET STRING
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The NIC Address of the neighboring NLSP router."
  ::= {nlspNeighEntry 5}

nlspNeighSysType OBJECT-TYPE
  SYNTAX     INTEGER {
                unknown(1),
                nlspLevel1Router(2)
                }
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The type of the neighboring NLSP router."
  ::= {nlspNeighEntry 6}

nlspNeighSysID OBJECT-TYPE
  SYNTAX     SystemID
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The neighboring NLSP router's system ID."
  ::= {nlspNeighEntry 7}

nlspNeighName OBJECT-TYPE
  SYNTAX     OCTET STRING (SIZE(0..48))
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The readable name for the neighboring NLSP router."
  ::= {nlspNeighEntry 8}

nlspNeighUsage OBJECT-TYPE
  SYNTAX     INTEGER {
                undefined(1),
                level1(2)
                }
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The usage of the connection to the neighboring NLSP
          router."
  ::= {nlspNeighEntry 9}

nlspNeighHoldTimer OBJECT-TYPE
  SYNTAX     INTEGER (1..65535)
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The initial holding time, in seconds, for this NLSP
          neighbor entry as specified in the NLSP Hello packet."
  ::= {nlspNeighEntry 10}

nlspNeighRemainingTime OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The remaining time to live, in seconds, for this NLSP
          neighbor entry."
  ::= {nlspNeighEntry 11}
```

```
nlspNeighPriority OBJECT-TYPE
  SYNTAX    INTEGER (1..127)
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The priority of the neighboring NLSP router for
        becoming the LAN Level 1 Designated router if the value
        of nlspNeighSysType is nlspLevel1Router."
  ::= {nlspNeighEntry 12}


-- Translation Group
-- The translation group contains tables providing mappings between
-- network numbers, NLSP system IDs, and router names.

-- NLSP ID Mapping Table
-- This table maps NLSP system IDs to router names and IPX network
-- numbers.

nlspIDMapTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF NLSPIDMapEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "This table maps NLSP system IDs to router names and
        IPX network numbers."
  ::= {nlspTranslation 1}

nlspIDMapEntry OBJECT-TYPE
  SYNTAX    NLSPIDMapEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry maps one NLSP system ID to its corresponding
        router name and IPX network number."
  INDEX     {
        nlspIDMapSysInstance,
        nlspIDMapID
       }
  ::= {nlspIDMapTable 1}

NLSPIDMapEntry ::= SEQUENCE {
                    nlspIDMapSysInstance
                      INTEGER,
                    nlspIDMapID
                      NLSPID,
                    nlspIDMapServerName
                      OCTET STRING,
                    nlspIDMapNetNum
                      NetNumber
                   }

nlspIDMapSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP and IPX
        (via ipxSysInstance) to which this row corresponds."
  ::= {nlspIDMapEntry 1}

nlspIDMapID OBJECT-TYPE
  SYNTAX    NLSPID
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The NLSP ID (6-octet sytem ID plus the pseudo-node ID)."
  ::= {nlspIDMapEntry 2}

nlspIDMapServerName OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(0..48))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The readable name corresponding to this NLSP ID."
  ::= {nlspIDMapEntry 3}

nlspIDMapNetNum OBJECT-TYPE
  SYNTAX    NetNumber
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The IPX network number corresponding to this NLSP ID."
  ::= {nlspIDMapEntry 4}


-- IPX Network Number Mapping Table
-- This table maps IPX network numbers to router names and NLSP IDs.

nlspNetMapTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF NLSPNetMapEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "This table maps IPX network numbers to router names
        and NLSP IDs."
  ::= {nlspTranslation 2}

nlspNetMapEntry OBJECT-TYPE
  SYNTAX    NLSPNetMapEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry maps one IPX network number to its
        corresponding router name and NLSP ID."
  INDEX     {
        nlspNetMapSysInstance,
        nlspNetMapNetNum
       }
  ::= {nlspNetMapTable 1}

NLSPNetMapEntry ::= SEQUENCE {
                    nlspNetMapSysInstance
                      INTEGER,
                    nlspNetMapNetNum,
                      NetNumber,
                    nlspNetMapServerName
                      OCTET STRING,
                    nlspNetMapID,
                      NLSPID
                   }

nlspNetMapSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP and IPX
        (via ipxSysInstance) to which this row corresponds."
  ::= {nlspNetMapEntry 1}

nlspNetMapNetNum OBJECT-TYPE
  SYNTAX    NetNumber
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The IPX network number."
  ::= {nlspNetMapEntry 2}

nlspNetMapServerName OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(0..48))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The router name corresponding to the IPX network number."
  ::= {nlspNetMapEntry 3}

nlspNetMapID OBJECT-TYPE
  SYNTAX    NLSPID
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The NLSP ID corresponding to the IPX network number."
```

```
    ::= {nlspNetMapEntry 4}


-- Name Mapping Table
-- This table maps router names to their corresponding IPX network
-- number and NLSP ID.

nlspNameMapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NLSPNameMapEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION "This table maps router names to the corresponding IPX
                 network number and NLSP ID."
    ::= {nlspTranslation 3}

nlspNameMapEntry OBJECT-TYPE
    SYNTAX      NLSPNameMapEntry
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION "Each entry maps one router name to its corresponding
                 IPX network number and NLSP ID."
    INDEX       {
                 nlspNameMapSysInstance,
                 nlspNameMapServerName
                }
    ::= {nlspNameMapTable 1}

NLSPNameMapEntry ::= SEQUENCE {
                        nlspNameMapSysInstance
                          INTEGER,
                        nlspNameMapServerName
                          OCTET STRING,
                        nlspNameMapNetNum
                          NetNumber,
                        nlspNameMapID
                          NLSPID
                       }

nlspNameMapSysInstance OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION "The unique identifier of the instance of NLSP and IPX
                 (via ipxSysInstance) to which this row corresponds."
    ::= {nlspNameMapEntry 1}

nlspNameMapServerName OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(0..48))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION "The readable name for this system."
    ::= {nlspNameMapEntry 2}

nlspNameMapNetNum OBJECT-TYPE
    SYNTAX      NetNumber
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION "The IPX network number corresponding to the router name."
    ::= {nlspNameMapEntry 3}

nlspNameMapID OBJECT-TYPE
    SYNTAX      NLSPID
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION "The NLSP ID corresponding to the router name.  This value
                 is undefined if the value of nlspSysState is off."
    ::= {nlspNameMapEntry 4}


-- Graph Group
-- The Graph group provides a representation of the network topology.
-- The group is optional.

-- Node Table
-- The node table contains an entry for each node in the graph.

nlspNodeTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NLSPNodeEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION "The node table contains an entry for each node in the
                 graph."
    ::= {nlspGraph 1}

nlspNodeEntry OBJECT-TYPE
    SYNTAX      NLSPNodeEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION "Each entry corresponds to one graph node."
    INDEX       {
                 nlspNodeSysInstance,
                 nlspNodeNLSPID
                }
    ::= {nlspNodeTable 1}

NLSPNodeEntry ::= SEQUENCE {
                        nlspNodeSysInstance
                          INTEGER,
                        nlspNodeID
                          NLSPID,
                        nlspNodeNetNum
                          NetNumber,
                        nlspNodeType
                          INTEGER,
                        nlspNodeEstDelay
                          INTEGER,
                        nlspNodeEstThroughput
                          INTEGER,
                        nlspNodeMaxPacketSize
                          INTEGER,
                        nlspNodeCost
                          INTEGER,
                        nlspNodeOverload
                          INTEGER,
                        nlspNodeDesRouter
                          INTEGER
                       }

nlspNodeSysInstance OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION "The unique identifier of the instance of NLSP and IPX
                 (via ipxSysInstance) to which this row corresponds."
    ::= {nlspNodeEntry 1}

nlspNodeNLSPID OBJECT-TYPE
    SYNTAX      NLSPID
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION "The NLSP ID for this node."
    ::= {nlspNodeEntry 2}

nlspNodeNetNum OBJECT-TYPE
    SYNTAX      NetNumber
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION "The IPX network number of this node."
    ::= {nlspNodeEntry 3}
```

```
nlspNodeType OBJECT-TYPE
  SYNTAX     INTEGER {
               unknown(1),
               nlspLevel1Router(2),
               router(4),
               network(5)
             }
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The type of system the node represents."
  ::= {nlspNodeEntry 4}

nlspNodeEstDelay OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The estimated delay, in milliseconds, to reach the
          destination represented by this node."
  ::= {nlspNodeEntry 5}

nlspNodeEstThroughput OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The estimated throughput, in bits per second, to the
          destination represented by this node."
  ::= {nlspNodeEntry 6}

nlspNodeMaxPacketSize OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The maximum packet size, in bytes, that can be sent to
          the destination represented by this node."
  ::= {nlspNodeEntry 7}

nlspNodeCost OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The cost to reach this node."
  ::= {nlspNodeEntry 8}

nlspNodeOverload OBJECT-TYPE
  SYNTAX     INTEGER {
               no(1),
               yes(2)
             }
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "Indicates whether this node is overloaded."
  ::= {nlspNodeEntry 9}

nlspNodeDesRouter OBJECT-TYPE
  SYNTAX     INTEGER {
               no(1),
               yes(2)
             }
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "Indicates whether this node is a designated router."
  ::= {nlspNodeEntry 10}


-- Link Table
-- This table contains the entries for all of the links in the graph.

nlspLinkTable OBJECT-TYPE
  SYNTAX     SEQUENCE OF NLSPLinkEntry
  ACCESS     not-accessible
  STATUS     mandatory
  DESCRIPTION "The Link table contains entries for all of the links in
          the graph."
  ::= {nlspGraph 2}

nlspLinkEntry OBJECT-TYPE
  SYNTAX     NLSPLinkEntry
  ACCESS     not-accessible
  STATUS     mandatory
  DESCRIPTION "Each entry corresponds to one link."
  INDEX      {
               nlspLinkSysInstance,
               nlspLinkNLSPID,
               nlspLinkIndex
             }
  ::= {nlspLinkTable 1}

NLSPLinkEntry ::= SEQUENCE {
                    nlspLinkSysInstance
                      INTEGER,
                    nlspLinkNLSPID
                      NLSPID,
                    nlspLinkIndex
                      INTEGER,
                    nlspLinkNeighNLSPID
                      NLSPID,
                    nlspLinkCost
                      INTEGER,
                    nlspLinkMaxPacketSize
                      INTEGER,
                    nlspLinkThroughput
                      INTEGER,
                    nlspLinkDelay
                      INTEGER,
                    nlspLinkMediaType
                      OCTET STRING
                  }

nlspLinkSysInstance OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP and IPX
          (via ipxSysInstance) to which this row corresponds."
  ::= {nlspLinkEntry 1}

nlspLinkNLSPID OBJECT-TYPE
  SYNTAX     NLSPID
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The NLSP ID (6-byte system ID plus 1-octet pseudo-node
          ID) of the node to which this link belongs."
  ::= {nlspLinkEntry 2}

nlspLinkIndex OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The unique value identifying the link within the node."
  ::= {nlspLinkEntry 3}

nlspLinkNeighNLSPID OBJECT-TYPE
  SYNTAX     NLSPID
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The NLSP ID (6-byte system ID plus 1-octet pseudo-node
          ID) of the neighboring node."
  ::= {nlspLinkEntry 4}

nlspLinkCost OBJECT-TYPE
```

```
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The cost to use this link."
  ::= {nlspLinkEntry 5}

nlspLinkMaxPacketSize OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The maximum size, in bytes, of a packet that may be sent
          over this link."
  ::= {nlspLinkEntry 6}

nlspLinkThroughput OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The link's maximum throughput, in bits per second."
  ::= {nlspLinkEntry 7}

nlspLinkDelay OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The delay, in milliseconds, on this link."
  ::= {nlspLinkEntry 8}

nlspLinkMediaType OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(2))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The media type of this link."
  ::= {nlspLinkEntry 9}

-- Path Table
-- This table allows the path(s) that a packet may take to reach a
-- destination to be reconstructed. The entries in this table
-- represent those links that are one hop closer to the source and
-- would be used for the minimum cost path(s) to reach the
-- destination.

nlspPathTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF NLSPPathEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The path table."
  ::= {nlspGraph 3}

nlspPathEntry OBJECT-TYPE
  SYNTAX    NLSPPathEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each row in this table represents a link to a node that
          is one hop closer to the source and would be used for
          the minimum cost path(s) to reach the destination."
  ::= {nlspPathTable 1}

NLSPPathEntry ::= SEQUENCE {
                    nlspPathSysInstance
                      INTEGER,
                    nlspPathDestNLSPID
                      NLSPID,
                    nlspPathLinkIndex
                      INTEGER
                  }

nlspPathSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP and IPX
          (via ipxSysInstance) to which this row corresponds."
  ::= {nlspPathEntry 1}

nlspPathDestNLSPID OBJECT-TYPE
  SYNTAX    NLSPID
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The NLSP ID (6-octet system ID plus 1-octet pseudo-node
          ID) of this destination."
  ::= {nlspPathEntry 2}

nlspPathLinkIndex OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The unique value identifying this link within the
          destination node."
  ::= {nlspPathEntry 3}


-- LSP Group
-- The LSP group provides a representation of NLSP's LSP database. This
-- group is optional.

-- LSP Header Table
-- The LSP header table contains summary information about each LSP
-- in the database as well as an OCTET STRING containing the entire
-- LSP header.

nlspLSPTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF NLSPLSPEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The LSP header table."
  ::= {nlspLSP 1}

nlspLSPEntry OBJECT-TYPE
  SYNTAX    NLSPLSPEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry corresponds to one LSP's header."
  ::= {nlspLSPTable 1}

NLSPLSPEntry ::= SEQUENCE {
                    nlspLSPSysInstance
                      INTEGER,
                    nlspLSPID
                      OCTET STRING,
                    nlspLSPLifetime
                      INTEGER,
                    nlspLSPSeqNum
                      INTEGER,
                    nlspLSPChecksum
                      INTEGER,
                    nlspLSPRouterType
                      INTEGER,
                    nlspLSPOverload
                      INTEGER,
                    nlspLSPHeader
                      OCTET STRING
                  }

nlspLSPSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The unique identifier for the instance of NLSP and IPX
```

```
        (via ipxSysInstance) to which this entry corresponds."
  ::= {nlspLSPEntry 1}

nlspLSPID OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(8))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The value that uniquely identifies this LSP."
  ::= {nlspLSPEntry 2}

nlspLSPLifetime OBJECT-TYPE
  SYNTAX    INTEGER (0..65535)
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of seconds prior to the expiration of the
        LSP."
  ::= {nlspLSPEntry 3}

nlspLSPSeqNum OBJECT-TYPE
  SYNTAX    INTEGER (0..255)
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The sequence number of the LSP."
  ::= {nlspLSPEntry 4}

nlspLSPChecksum OBJECT-TYPE
  SYNTAX    INTEGER (0..65535)
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The checksum value of the LSP."
  ::= {nlspLSPEntry 5}

nlspLSPRouterType OBJECT-TYPE
  SYNTAX    INTEGER {
              unknown(1),
              nlspLevel1Router(2)
            }
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The type of the router that sent the LSP."
  ::= {nlspLSPEntry 6}

nlspLSPOverload OBJECT-TYPE
  SYNTAX    INTEGER {
              no(1),
              yes(2)
            }
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "Indicates whether the sending router's LSP database is
        overloaded."
  ::= {nlspLSPEntry 7}

nlspLSPHeader OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(24))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The complete LSP header."
  ::= {nlspLSPEntry 8}


-- LSP Options Table
--   The LSP options table is used to obtain each option contained in
--   an LSP.

nlspLSPOptTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF NLSPLSPOptEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The LSP Options table."
  ::= {nlspLSP 2}

nlspLSPOptEntry OBJECT-TYPE
  SYNTAX    NLSPLSPOptEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry corresponds to one option from an LSP."
  ::= {nlspLSPOptTable 1}

NLSPLSPOptEntry ::= SEQUENCE {
                    nlspLSPOptSysInstance
                      INTEGER,
                    nlspLSPOptLSPID
                      OCTET STRING,
                    nlspLSPOptIndex
                      INTEGER,
                    nlspLSPOptCode
                      INTEGER,
                    nlspLSPOptLength
                      INTEGER,
                    nlspLSPOptValue
                      OCTET STRING
                  }

nlspLSPOptSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of NLSP and IPX
        (via ipxSysInstance) to which this entry corresponds."
  ::= {nlspLSPOptEntry 1}

nlspLSPOptLSPID OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(8))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The value that uniquely identifies the LSP."
  ::= {nlspLSPOptEntry 2}

nlspLSPOptIndex OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The value that uniquely identifies this option within the
        LSP."
  ::= {nlspLSPOptEntry 3}

nlspLSPOptCode OBJECT-TYPE
  SYNTAX    INTEGER (0..255)
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The code that identifies the type of the option."
  ::= {nlspLSPOptEntry 4}

nlspLSPOptLength OBJECT-TYPE
  SYNTAX    INTEGER (0..255)
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The length of the option's value field."
  ::= {nlspLSPOptEntry 5}

nlspLSPOptValue OBJECT-TYPE
  SYNTAX    OCTET STRING (SIZE(0..255))
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The option's value field."
  ::= {nlspLSPOptEntry 6}

END
```

8.3. RIP/SAP MIB

```
RIPSAP DEFINITIONS ::= BEGIN

-- This MIB defines the management information for the RIP and SAP
-- protocols running in an IPX enviroment. It provides information in
-- addition to that contained in the IPX MIB itself. All tables in this
-- MIB are linked to an instance of IPX via the system instance
-- identifier as defined in the IPX MIB.


IMPORTS
  enterprises, Counter
        FROM RFC1155-SMI
  OBJECT-TYPE
        FROM RFC-1212
  PhysAddress
        FROM RFC-1213;

novell OBJECT IDENTIFIER ::= {enterprises 23}
experimental OBJECT IDENTIFIER ::= {novell 4}
ripsap OBJECT IDENTIFIER ::= {experimental 10}


-- Groups

ripsapSystem  OBJECT IDENTIFIER ::= {ripsap 1}
ripsapCircuit OBJECT IDENTIFIER ::= {ripsap 2}


-- Types

NetNumber ::= OCTET STRING (SIZE(4))


-- System Group
-- This group contains global information about each instance of
-- RIP/SAP running on one system.

-- System Table
-- This table contains an entry for each instance of RIP/SAP
-- running on the system.

ripsapSysTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF RIPSAPSysEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The RIP/SAP system table."
  ::= {ripsapSystem 1}

ripsapSysEntry OBJECT-TYPE
  SYNTAX    RIPSAPSysEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "Each entry corresponds to one instance RIP/SAP
          Router running on the system."
  INDEX     {ripsapSysInstance}
  ::= {ripsapSysTable 1}

RIPSAPSysEntry ::= SEQUENCE {
                ripsapSysInstance
                  INTEGER,
                ripsapSysRIPState
                  INTEGER,
                ripsapSysRIPIncorrectPackets
                  Counter,
                ripsapSysSAPState
                  INTEGER,
                ripsapSysSAPIncorrectPackets
                  Counter
                }

ripsapSysInstance OBJECT-TYPE
  SYNTAX    INTEGER
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "The unique identifier of the instance of RIP/SAP to
          which this row corresponds. This value links the
          instance of RIP/SAP to an instance of IPX running on the
          system (i.e. the value of nlspSysInstance should be the
          same as a value of ipxSysInstance). This value may be
          written only when creating a new entry in the table."
  ::= {ripsapSysEntry 1}

ripsapSysRIPState OBJECT-TYPE
  SYNTAX    INTEGER {
                off(1),
                on(2)
                }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "Indicates the operational state of this instance of RIP."
  ::= {ripsapSysEntry 2}

ripsapSysSAPState OBJECT-TYPE
  SYNTAX    INTEGER {
                off(1),
                on(2)
                }
  ACCESS    read-write
  STATUS    mandatory
  DESCRIPTION "Indicates the operational state of this instance of SAP."
  ::= {ripsapSysEntry 3}

ripsapSysRIPIncorrectPackets OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times that an incorrectly formatted RIP
          packet was received."
  ::= {ripsapSysEntry 4}

ripsapSysSAPIncorrectPackets OBJECT-TYPE
  SYNTAX    Counter
  ACCESS    read-only
  STATUS    mandatory
  DESCRIPTION "The number of times that an incorrectly formatted SAP
          packet was received."
  ::= {ripsapSysEntry 5}


-- Circuit Group
-- This group contains RIP and SAP management information for each
-- circuit known to this system.

-- Circuit Table
-- The Circuit table contains an entry for each circuit known to the
-- system.

ripsapCircTable OBJECT-TYPE
  SYNTAX    SEQUENCE OF RIPSAPCircEntry
  ACCESS    not-accessible
  STATUS    mandatory
  DESCRIPTION "The Circuit table."
  ::= {ripsapCircuit 1}

ripsapCircEntry OBJECT-TYPE
  SYNTAX    RIPSAPCircEntry
  ACCESS    not-accessible
```

```
  STATUS     mandatory
  DESCRIPTION "Each entry corresponds to one circuit known to the
          system."
  INDEX      {
          ripsapCircSysInstance,
          ripsapCircIndex
          }
  ::= {ripsapCircTable 1}

RIPSAPCircEntry ::= SEQUENCE {
                ripsapCircSysInstance
                  INTEGER,
                ripsapCircIndex
                  INTEGER,
                ripsapCircRIPState
                  INTEGER,
                ripsapCircRIPPace
                  INTEGER,
                ripsapCircRIPUpdate
                  INTEGER,
                ripsapCircRIPAgeMultiplier
                  INTEGER,
                ripsapCircRIPPacketSize
                  INTEGER,
                ripsapCircRIPOutPackets
                  Counter,
                ripsapCircRIPInPackets
                  Counter,
                ripsapCircSAPState
                  INTEGER,
                ripsapCircSAPPace
                  INTEGER,
                ripsapCircSAPUpdate
                  INTEGER,
                ripsapCircSAPAgeMultiplier
                  INTEGER,
                ripsapCircSAPPacketSize
                  INTEGER,
                ripsapCircSAPGetNearestServerReply
                  INTEGER,
                ripsapCircSAPOutPackets
                  Counter,
                ripsapCircSAPInPackets
                  Counter,
                }

ripsapCircSysInstance OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The unique identifier of the instance of RIP/SAP and IPX
          (via ipxSysInstance) to which this entry corresponds.
          This value may be written only when creating a new entry in
          the table."
  ::= {ripsapCircEntry 1}

ripsapCircIndex OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The identifier of this circuit, unique within the
          instance of RIP/SAP.  This value corresponds to the circuit
          identifier found in ipxCircIndex.  This value may be
          written only when creating a new entry in the table."
  ::= {ripsapCircEntry 2}

  CircRIPState OBJECT-TYPE
SYNTAX     INTEGER {
                off(1),
                on(2)
                }
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "Indicates whether RIP information may be sent/received
          over this circuit."
  ::= {ripsapCircEntry 3}

ripsapCircRIPPace OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The maximum pace, in packets per second, at which RIP
          packets may be sent on this circuit."
  ::= {ripsapCircEntry 4}

ripsapCircRIPUpdate OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The RIP periodic update interval, in seconds."
  ::= {ripsapCircEntry 5}

ripsapCircRIPAgeMultiplier OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The holding multiplier for information received in RIP
          periodic updates."
  ::= {ripsapCircEntry 6}

ripsapCircRIPPacketSize OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The RIP packet size used on this circuit."
  ::= {ripsapCircEntry 7}

ripsapCircRIPOutPackets OBJECT-TYPE
  SYNTAX     Counter
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The number of RIP packets sent on this circuit."
  ::= {ripsapCircEntry 8}

ripsapCircRIPInPackets OBJECT-TYPE
  SYNTAX     Counter
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The number of RIP packets received on this circuit."
  ::= {ripsapCircEntry 9}

ripsapCircSAPState OBJECT-TYPE
  SYNTAX     INTEGER {
                off(1),
                on(2)
                }
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "Indicates whether SAP information may be sent/received
          over this circuit."
  ::= {ripsapCircEntry 10}

ripsapCircSAPPace OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The maximum pace, in packets per second, at which SAP
          packets may be sent on this circuit."
  ::= {ripsapCircEntry 11}
```

```
ripsapCircSAPUpdate OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The SAP periodic update interval, in seconds."
  ::= {ripsapCircEntry 12}

ripsapCircSAPAgeMultiplier OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The holding multiplier for information received in
          SAP periodic updates."
  ::= {ripsapCircEntry 13}

ripsapCircSAPPacketSize OBJECT-TYPE
  SYNTAX     INTEGER
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "The SAP packet size used on this circuit."
  ::= {ripsapCircEntry 14}

ripsapCircSAPGetNearestServerReply OBJECT-TYPE
  SYNTAX     INTEGER {
               no(1),
               yes(2)
               }
  ACCESS     read-write
  STATUS     mandatory
  DESCRIPTION "Indicates whether to respond to SAP get nearest server
          requests received on this circuit."
  ::= {ripsapCircEntry 15}

ripsapCircSAPOutPackets OBJECT-TYPE
  SYNTAX     Counter
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The number of SAP packets sent on this circuit."
  ::= {ripsapCircEntry 16}

ripsapCircSAPInPackets OBJECT-TYPE
  SYNTAX     Counter
  ACCESS     read-only
  STATUS     mandatory
  DESCRIPTION "The number of SAP packets received on this circuit."
  ::= {ripsapCircEntry 17}

END
```

9. Comparison with IS-IS

The basis of NLSP's design is the ISO OSI IS-IS standard (Reference ISO92). This section summarizes the key differences between NLSP and IS-IS. The current version of NLSP, as described in this document, is the basis for comparison.

9.1. Terminology in the Specification Document

IS-IS	*NLSP*
Intermediate System.	Router.
End System.	End node. (Examples of end nodes are desktop client workstations and nonrouting servers.)
Protocol Data Unit.	Packet.
Subnetwork.	Network.
Network.	Internetwork.

9.2. Addressing Issues

IS-IS	*NLSP*
The Network-Layer address, or NSAP, is variable in length, up to 20 bytes. For routing purposes, it has a variable-length Area Address, a one- to eight-byte ID field, and a one-byte Selector. The Selector identifies a software entity within the system.	The Network-Layer address is fixed at 12 bytes. It consists of a four-byte network number, a six-byte node number, and a two-byte socket field. The socket identifies a software entity within the system.
Each router has at least one area address. The maximum number of area addresses for an area is three or more (the number is configured uniformly in an area).	NLSP allows area addresses to be used. The maximum number of area addresses for an area is three. The default area address if all networks are in one area is (area = 0, mask =0).
Each Manual Area Address is represented as a length-preceded NSAP prefix; that is, the leading bytes of the NSAP. It has one-byte granularity.	Each Manual Area Address is represented as pair of four-byte numbers. The first is an IPX network number; the second is a mask of leading "one" bits indicating how long the area identifier is. It is variable-length with one-bit granularity.
Each router has a unique system ID. The system ID is 1 to 8 bytes (the size is configured uniformly in an area).	Each router has a unique system ID. The system ID is 6 bytes. Each router also has a unique internal network number and a textual name.

9.3. Routing Issues

IS-IS	*NLSP*
Systems use information about links to routers and end nodes to determine routes to routers and to end nodes.	Systems determine routes to IPX network numbers, and determine services reachable at those network numbers.
Includes Level 1 and Level 2 routing.	Includes specification of Level 1 routing and (for forward compatibility) supports routing to the nearest Level 2 router.
A router supports one to four routing metrics.	There is one routing metric: "Cost."
LSPs are sent only on LANs having adjacencies with the sending router.	LSPs are sent on a LAN even if there are no adjacencies on the circuit in question.
When a LAN circuit changes state to "Up," a system waits 2 x bcastHelloInt before electing a Designated Router on that circuit.	When a LAN circuit changes state to "Up," a system waits 2 x drBcastHelloInt before electing a Designated Router on that circuit. However, if in this time an adjacency is formed to a system on that LAN, and that system reports that it is the Designated Router itself, the Designated Router election process is run at once.
Priority is not affected by overload status.	Priority of becoming Designated Router is reduced when in LSP overload state.
The priority to become Designated Router on a LAN is configurable, per router per LAN attachment.	The priority to become Designated Router on a LAN is configurable, per router per LAN attachment. The default is 44. If a system elects itself Designated Router, it raises its priority by 20.
Level 1 LSPs describe links to other routers, and to end nodes.	Level 1 LSPs describe links to other routers, to external routes, and to services. NLSP also includes a management information field that contains the router's IPX name and internal network number.
Does not include extra measurements or management information.	Includes Throughput, Delay, MTU Size, and Media Type in LSPs. The first three of these also enter into load-splitting decisions.

Each LSP IS Neighbors option can contain multiple neighbors.	Each LSP Link Information option describes one neighbor and the link connecting to it.
The maximum size of sequence number packets and hello packets is configurable.	The maximum size of sequence number packets and hello packets is determined dynamically per circuit.
Comparison of LSPs for newness considers sequence number and remaining lifetime.	In addition to sequence number and remaining lifetime, comparison of LSPs for newness considers the checksum. The "preference of checksums" rule makes resolving LSP confusion operate the same way for equal and unequal sequence numbers.
On wraparound of an LSP sequence number, the router is disabled for a period.	On wraparound of an LSP sequence number, the specific LSP is purged for a period, but the router can (optionally) continue operation.
No backward compatibility needed.	Includes backward compatibility features for RIP and SAP. Includes WAN pseudonodes.
The originating lifetime of an LSP is 20 minutes.	The originating lifetime of an LSP is 120 minutes.

9.4. End Node Support

IS-IS	*NLSP*
Uses the ES-IS protocol.	Uses RIP and SAP.
Can route data traffic reliably to an end node attached to a partitioned LAN.	Cannot route data traffic reliably to an end node attached to a partitioned LAN, because NLSP routes to IPX networks (not individual end nodes).

9.5. Datalink Issues

IS-IS	*NLSP*
Assumes that a router detects remote start-up of a WAN link. Also assumes that failure of data delivery results in circuit disconnect.	Includes a state machine in establishing WAN circuits, to support unreliable datagram datalinks.

Hello messages are padded to maximum configured size (or one byte less) to ensure adjacencies are formed only between routers which can communicate fully.	MTU Size is included in Hello messages, enabling packets to be sent as large as possible, without overrunning the receiver's buffer.
Specifies pacing of packet transmissions to avoid overrunning receivers' capacity to handle traffic bursts.	Pacing of PDU transmissions vary in detail from IS-IS.
Uses multicast on a LAN.	Uses multicast on a LAN, with provision to revert to broadcast if there is a router which does not support multicast.
Includes specification of operation over X.25 and IEEE 802.x networks.	Includes specification of operation over X.25 and IEEE 802.x networks, but also operates over other LAN media, and over other WAN media using IW2 (of which X.25 support is one example).

9.6. System Integrity Issues

IS-IS	*NLSP*
If a router receives an LSP with a bad checksum, it does an area-wide purge of the LSP.	If a router receives an LSP with a bad checksum, it generates an event and discards the LSP.
Includes optional authentication specification.	Does not include authentication.
Partial precomputation of LSP checksums is required.	Partial precomputation of LSP checksums is optional.

9.7. Packet Format and Framing

IS-IS	*NLSP*
Packets are transmitted directly over the datalink layer.	Packets are transmitted with IPX headers.
Packets have a fixed part and a set of variable-length option fields.	Packet formats are very closely based on those of IS-IS. The fixed part of the NLSP packet is the same length as the IS-IS packet. Some fields that are reserved in IS-IS are used in NLSP; for example, the State field in the WAN Hello packet and the No Multicast bit in the LAN Hello packet. Some fields used in IS-IS are reserved in NLSP; for example, the IS-IS ID Length and Maximum Area Addresses fields do not apply to NLSP because these parameters are fixed.

LSP buffer size is 1492 bytes.	LSP buffer size is 512 bytes, but may be configured if done consistently throughout the routing area.

9.8. System Management

IS-IS	*NLSP*
System Management is specified in GDMO notation.	System Management is specified in concise MIB form for SNMP.

10. References

[ANS91] ANSI, "Integrated Services Digital Network (ISDN)—Digital Subscriber Signaling System No. 1 (DSS1)—Signaling Specification for Frame Relay Bearer Service," ANSI T1.617-1991, June 1991.

[All92] M. Allen, RFC 1362, "Novell IPX Over Various WAN Media (IPXWAN)," Novell, Inc., September 1992.

[Bra92] T. Bradley, C. Brown, and A. Malis, "Multiprotocol Interconnect over Frame Relay," RFC 1294, January 1992.

[DEC90] Digital Equipment Corp., Northern Telecom, and StrataCom, "Frame Relay Specification with Extension Based on Proposed T1S1 Standards," September, 1990.

[Mat92] S. Mathur and M. Lewis, "Compressing IPX Headers over WAN Media (CIPX)," Internet Draft, December 1992.

[Nov92] Novell, Inc., "IPX Router Specification," June 16, 1992, Part Number 107-000029-001.

[Per92] R. Perlman, "Interconnections: Bridges and Routers," Addison-Wesley, 1992.

[ISO88] International Organization for Standardization (ISO), "Information processing systems—Data communications—Protocol for providing the connectionless-mode network service", ISO 8473, 1988.

[ISO92] International Organization for Standardization (ISO), "Information technology—Telecommunications and information exchange between systems—Intermediate system to Intermediate system intradomain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)", ISO 10589, 1992-04-30.

[Mal92] A. Malis, D. Robinson, and R. Ullman, "Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode", RFC 1356, August 1992.

[Pos80] Postel, J.B. "User Datagram Protocol." RFC 768, August 28, 1980.

[Pos81] Postel, J.B. "Internet Protocol," RFC 791, September 1981.

[Pos81a] J.B. Postel, "Internet Control Message Protocol," RFC 792, September 1981.

[Ros91] M.T. Rose and K. McCloghrie, eds., "Concise MIB definitions," RFC 1212, March 1991.

[Sim92] W. Simpson, "The Point-to-Point Protocol (PPP) for the Transmission of Multiprotocol Datagrams over Point-to-Point Links", RFC 1331, May 1992.

[Sim92a] W. Simpson, "The PPP Internetwork Packet Exchange Control Protocol (IPXCP) Compromise Version", Internet Draft, December 1992.

[Wor92] R. Wormley and S. Bostock, "SNMP over IPX," RFC 1298, February 1992.

[Xer81] Xerox Corp., "Internet Transport Protocols," XSIS 028112, December 1981.

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z