The ATM Forum

Technical Committee

Private Network-Network Interface Specification Version 1.0 (PNNI 1.0)

af-pnni-0055.000
March 1996

Acknowledgements

The PNNI SWG was chaired by Mike Goguen. Doug Dykeman was the editor for the PNNI Routing related sections of the document, and Rao Cherukuri for PNNI Signalling. The official meeting minutes were taken by Jason Jeffords.

The following people made significant technical contributions to the PNNI specification:

Masuma Ahmed
Anthony Alles
Francois Audet
Tomas Berggren
Caralyn Brown
David Bucciero
Ross Callon
Rao Cherukuri
Ashok Chippa
Rob Coltun
David Cypher
Roger Dev
R. Dianda
John Drake
Tim Dwight
Doug Dykeman
John Elwell
Robert Epley
Norman Finn
Mike Goguen
Dan Grossman
Joel Halpern
S. Kamran Hasan
Tim Hefel
Juha Heinanen
Tom Helstern
Michael Hluchyj
Eric Hoffman
Heinrich Hummel
Nick Huslak
Jason Jeffords
Martin Johnsson
Claude Kawa
John Keene
Walter Knitl
Paul Koning
Garry Kump
Whay Lee
Fong-Ching Liaw
Sandrine Masson
Keith McCloghrie
Dave McDysan
John Moy
Raj Nair
Maryann Perez
Drew Perkins
Bernhard Petri
Paul Phillips
Russell Pretty
Tony Przygienda
John Rutemiller
Abdallah Saleh
Bill Salkewicz
Hal Sandick
Shirish Sathaye
Ross Schibler
Jean-Bernard Schmitt
Stephen Shew
F. Y. Simon
Kumar Sivarajan
Ethan Spiegel
Ed Sullivan
Hiroshi Suzuki
George Swallow
Ted Tedijanto
Kamlesh Tewani
Marek Tomaszewski
Joachim Wagner
James Watt
David Wells
Malcolm Wiles

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Overview

This document defines the PNNI protocol for use between private ATM switches, and between groups of private ATM switches. The abbreviation PNNI stands for either Private Network Node Interface or Private Network-to-Network Interface, reflecting these two possible usages. PNNI includes two categories of protocols:

- A protocol is defined for distributing topology information between switches and clusters of switches. This information is used to compute paths through the network. A hierarchy mechanism ensures that this protocol scales well for large world-wide ATM networks. A key feature of the PNNI hierarchy mechanism is its ability to automatically configure itself in networks in which the address structure reflects the topology. PNNI topology and routing is based on the well-known link-state routing technique.
- A second protocol is defined for signalling, that is message flows used to establish point-to-point and point-to-multipoint connections across the ATM network. This protocol is based on the ATM Forum UNI signalling, with mechanisms added to support source routing, crankback, and alternate routing of call setup requests in case of connection setup failure.

Use of this specification by public networks that wish to do so is not precluded.

1.2 Reference Models

The following is the a reference model for a Switching System:

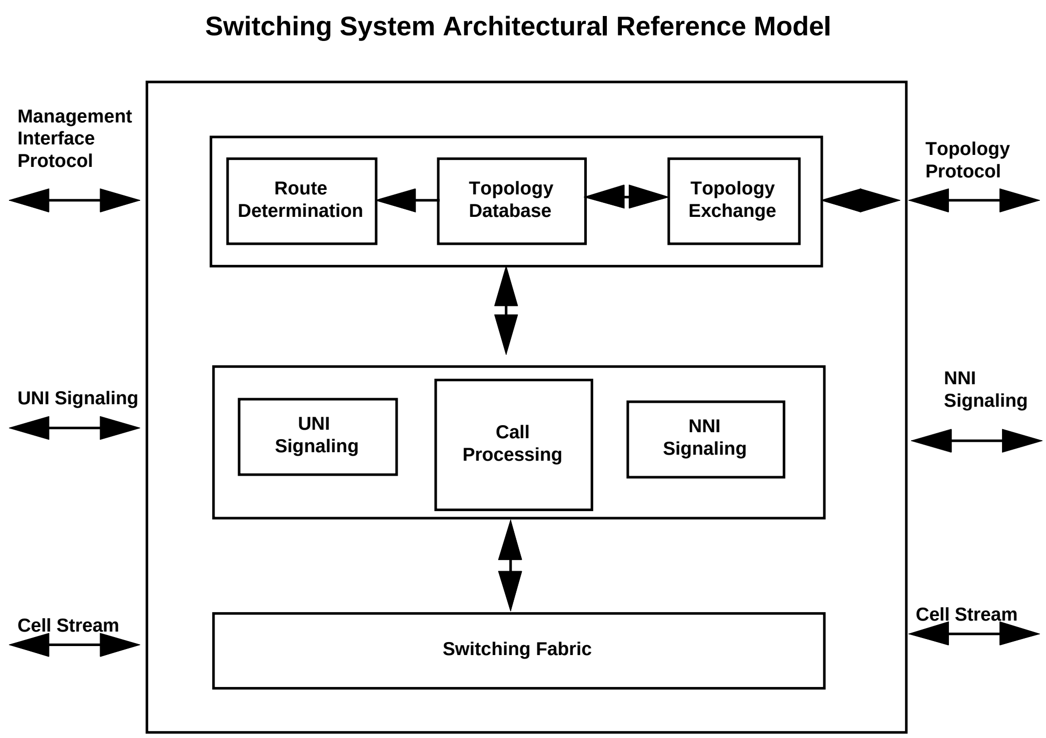


Figure 1-1: Switching System Architectural Reference Model

1.3 PNNI Status

The current document is a draft version of the Phase 1 PNNI specification. PNNI Phase 1 is designed to be compatible with the ATM Forum UNI 3.1 specification.

Phase 1 of PNNI has the following characteristics:

- Supports all UNI 3.1 and some UNI 4.0 capabilities.
- Scales to very large networks.
- Supports hierarchical routing.
- Supports QoS.
- Supports multiple routing metrics and attributes.
- Uses source routed connection setup.
- Operates in the presence of partitioned areas.
- Provides dynamic routing, responsive to changes in resource availability.
- Separates the routing protocol used within a peer group from that used among peer groups.
- Interoperates with external routing domains, not necessarily using PNNI.
- Supports both physical links and tunneling over VPCs.
- Supports soft PVPC/PVCCs.
- Supports anycast.

1.4 Document Organization

The PNNI specification is organized into four main sections:

- Chapter 3 includes descriptive text covering the PNNI routing protocol.
- Chapter 4 includes descriptive text covering the PNNI signalling protocol.
- Chapter 5 provides the detailed definition of the PNNI routing protocol.
- Chapter 6 provides the detailed definition of the PNNI signalling protocol.

Annexes are formal clarifications of material in the document, and are part of the specification. Appendixes are included to help clarify the specification, but are not part of the formal PNNI protocol.

1.5 References

ATM Forum	Traffic Management Specification Version 4.0
ATM Forum	User-Network Interface Specification Version 3.1
ATM Forum	User-Network Interface Signalling Specification Version 4.0
ATM Forum	ILMI Specification Version 4.0
IEEE 802-1990	IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture
IETF RFC 1573	"Evolution of the Interface Group of MIB-II", K. McCloghrie and F. Kastenholz, January 1994.
IETF RFC 1695	"Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2", M. Ahmed and K. Tesink, August 1994.

IETF RFC 1902	"Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", SNMPv2 Working Group, January 1996.
IETF RFC 1903	"Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", SNMPv2 Working Group, January 1996.
IETF RFC 1904	"Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", SNMPv2 Working Group, January 1996.
ITU-T I.150 (1993)	B-ISDN Asynchronous Transfer Mode Functional Characteristics
ITU-T I.361 (1993)	B-ISDN ATM Layer Specification
ITU-T I.363 (1993)	B-ISDN ATM Adaptation Layer (AAL) Specification
ITU-T Q.2110 (1994)	B-ISDN SAAL Service Specific Connection Oriented Protocol (SSCOP)
ITU-T Q.2130 (1994)	B-ISDN SAAL Service Specific Coordination Function (SSCF) for Support of Signalling at the User-Network Interface
ITU-T Q.2931 (1995)	B-ISDN DSS2 User-Network Interface (UNI) Layer 3 Specification for Basic Call/Connection Control
ITU-T Q.2951 (1995)	Stage 3 Description for Number Identification Supplementary Services Using B-ISDN DSS2 Basic Call
ITU-T Q.2962 (Note 1)	B-ISDN DSS2 Connection Characteristics Negotiation During Establishment Phase
ITU-T Q.2971 (Note 1)	B-ISDN DSS2 UNI Layer 3 Specification for Point-to-Multipoint Call/Connection Control

Note 1: This document is in course of preparation for publication during 1996.

Only these specific versions of the referenced documents, and the specific versions of the documents referenced from them etc., are included. Other, future, versions of these specifications may also be applicable, as may from time to time be determined by the ATM Forum Technical Committee.

2. Terminology

2.1 Abbreviations

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
AFI	Authority and Format Identifier
ATM	Asynchronous Transfer Mode
AvCR	Available Cell Rate
AW	Administrative Weight
BGP	Border Gateway Protocol
B-ICI	B-ISDN Inter Carrier Interface
CAC	Connection Admission Control
CBR	Constant Bit Rate
CDV	Cell Delay Variation
CLP	Cell Loss Priority
CLR	Cell Loss Ratio
CLR_0	Cell Loss Ratio objective for CLP=0 traffic
CRM	Cell Rate Margin
CTD	Cell Transfer Delay
DSP	Domain Specific Part
DTL	Designated Transit List
ES	End System
ESI	End System Identifier
FSM	Finite State Machine
GCAC	Generic Connection Admission Control
ICR	Initial Cell Rate
IDI	Initial Domain Identifier
IDP	Initial Domain Part
IDRP	Inter Domain Routing Protocol
ILMI	Interim Local Management Interface
IE	Information Element
ID	Identifier
IG	Information Group
LGN	Logical Group Node
LSB	Least Significant Bit
maxCR	Maximum Cell Rate
maxCTD	Maximum Cell Transfer Delay
MBS	Maximum Burst Size
MCR	Minimum Cell Rate
MIB	Management Information Base
MSB	Most Significant Bit
NNI	Network-to-Network Interface
NSAP	Network Service Access Point
OSPF	Open Shortest Path First
PCR	Peak Cell Rate
PG	Peer Group
PGL	Peer Group Leader
PGLE	Peer Group Leader Election
PTSE	PNNI Topology State Element
PTSP	PNNI Topology State Packet
PNNI	Private Network-to-Network Interface

PVC	Permanent Virtual Circuit
PVCC	Permanent Virtual Channel Connection
PVPC	Permanent Virtual Path Connection
QoS	Quality of Service
RAIG	Resource Availability Information Group
RCC	Routing Control Channel
RDF	Rate Decrease Factor
RIF	Rate Increase Factor
SAAL	Signalling ATM Adaptation Layer
SCR	Sustainable Cell Rate
SSCOP	Service Specific Connection Oriented Protocol
SSCS	Service Specific Convergence Sublayer
SVC	Switched Virtual Circuit
SVCC	Switched Virtual Channel Connection
TBE	Transit Buffer Exposure
TLV	Type, Length, Value
UBR	Unspecified Bit Rate
ULIA	Uplink Information Attribute
UNI	User to Network Interface
VBR	Variable Bit Rate
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VF	Variance Factor
VP	Virtual Path
VPC	Virtual Path Connection
VPI	Virtual Path Identifier

2.2 Definitions

Address Prefix	A string of 0 or more bits up to a maximum of 152 bits that is the lead portion of one or more ATM addresses.
Adjacency	The relationship between two communicating neighboring peer nodes.
Aggregation Token	A number assigned to an outside link by the border nodes at the ends of the outside link. The same number is associated with all uplinks and induced uplinks associated with the outside link. In the parent and all higher-level peer group, all uplinks with the same aggregation token are aggregated.
Alternate Routing	A mechanism that supports the use of a new path after an attempt to set up a connection along a previously selected path fails.
Ancestor Node	A logical group node that has a direct parent relationship to a given node (i.e., it is the parent of that node, or the parent's parent, ...).
ATM Anycast Capability	The ability to allow an application to request a point-to-point connection to a single ATM end system that is part of an ATM group.
Border Node	A logical node that is in a specified peer group, and has at least one link that crosses the peer group boundary.

Bypass	A bypass represents the connectivity between two ports in the complex node representation. A bypass is always an exception.
Child Node	A node at the next lower level of the hierarchy which is contained in the peer group represented by the logical group node currently referenced. This could be a logical group node, or a physical node.
Child Peer Group	A child peer group of a peer group is any one containing a child node of a logical group node in that peer group. A child peer group of a logical group node is the one containing the child node of that logical group node.
Common Peer Group	The lowest level peer group in which a set of nodes is represented. A node is represented in a peer group either directly or through one of its ancestors.
Complex Node Representation	A collection of nodal state parameters that provide detailed state information associated with a logical node.
Connection Scope	The level of routing hierarchy within which a given connection request to a group address is constrained.
Crankback	A mechanism for partially releasing a connection setup in progress which has encountered a failure. This mechanism allows PNNI to perform alternate routing.
Default Node Representation	A single value for each nodal state parameter giving the presumed value between any entry or exit to the logical node and the nucleus.
Designated Transit List	A list of node and optionally link Ids that completely specify a path across a single PNNI peer group.
Dijkstra’s Algorithm	An algorithm that is sometimes used to calculate routes given a link and nodal state topology database.
Domain	Synonymous with PNNI Routing Domain.
DTL Originator	The first lowest-level node within the entire PNNI routing domain to build the initial DTL stack for a given connection.
DTL Terminator	The last lowest-level node within the entire PNNI routing domain to process the connection (and thus the connection’s DTL).
End System	A system on which connection termination points are located.
Entry Border Node	The node which receives a call over an outside link. This is the first node within a peer group to see this call.
Exception	A connectivity advertisement in a PNNI complex node representation that represents something other than the default node representation.

Exit Border Node	The node that will progress a call over an outside link. This is the last node within a peer group to see this call.
Exterior	Denotes that an item (e.g., link, node, or reachable address) is outside of a PNNI routing domain.
Exterior Link	A link which crosses the boundary of the PNNI routing domain. The PNNI protocol does not run over an exterior link.
Exterior Reachable Address	An address that can be reached through a PNNI routing domain, but which is not located in that PNNI routing domain.
Exterior Route	A route which traverses an exterior link.
Foreign Address	An address or address prefix that does not match any of a given node's summary addresses.
Hello Packet	A type of PNNI Routing packet that is exchanged between neighboring logical nodes.
Hierarchically Complete Source Route	A stack of DTLs representing a route across a PNNI routing domain such that a DTL is included for each hierarchical level between and including the current level and the lowest visible level in which the source and destination are reachable.
Hop by Hop Route	A route that is created by having each switch along the path use its own routing knowledge to determine the next hop of the route, with the expectation that all switches will choose consistent hops such that the call will reach the desired destination. PNNI does not use hop-by-hop routing.
Horizontal Link	A link between two logical nodes that belong to the same peer group.
Induced Uplink	An uplink "A" that is created due to the existence of an uplink "B" in the child peer group represented by the node that created uplink "A". Both "A" and "B" share the same upnode, which is higher in the PNNI hierarchy than the peer group in which uplink "A" is seen.
Inside Link	Synonymous with horizontal link.
Instance ID	A subset of an object's attributes which serve to uniquely identify a MIB instance.
Interior	Denotes that an item (e.g., link, node, or reachable address) is inside of a PNNI routing domain.
Internal Reachable Address	An address of a destination that is directly attached to the logical node advertising the address.
Leadership Priority	The priority with which a logical node wishes to be elected peer group leader of its peer group. Generally, of all nodes in a peer group, the one with the highest leadership priority will be elected as peer group leader.

Level	Level is the position in the PNNI hierarchy at which a particular node or peer group exists. A level that has a smaller numerical value implies greater topology aggregation, and is hence called a 'higher level' in the PNNI hierarchy throughout this document. Conversely, a level that has a larger numerical value implies less topology aggregation, and is hence called a 'lower level' in the PNNI hierarchy throughout this document.
Link	Synonymous with logical link.
Link Aggregation Token	See Aggregation Token.
Link Attribute	A link state parameter that is considered individually to determine whether a given link is acceptable and/or desirable for carrying a given connection.
Link Constraint	A restriction on the use of links for path selection for a specific connection.
Link Metric	A link parameter that requires the values of the parameter for all links along a given path to be combined to determine whether the path is acceptable and/or desirable for carrying a given connection.
Link State Parameter	Information that captures an aspect or property of a link.
Logical Group Node	An abstract representation of a lower level peer group as a single point for purposes of operating at one level of the PNNI routing hierarchy.
Logical Link	An abstract representation of the connectivity between two logical nodes. This includes individual physical links, individual virtual path connections, and parallel physical links and/or virtual path connections.
Logical Node	A lowest-level node or a logical group node.
Logical Node ID	A string of bits that unambiguously identifies a logical node within a routing domain.
Lowest Level Node	A leaf in the PNNI routing hierarchy; an abstraction representing a single instance of the PNNI routing protocol. Lowest-level nodes are created in a switching system via configuration. They are not created dynamically.
Membership Scope	The level of routing hierarchy within which advertisement of a given address is contrained.
MIB Attribute	A single piece of configuration, management, or statistical information which pertains to a specific part of the PNNI protocol operation.

MIB Instance	An incarnation of a MIB object that applies to a specific part, piece, or aspect of the PNNI protocol's operation.
MIB Object	A collection of attributes that can be used to configure, manage, or analyze an aspect of the PNNI protocol's operation.
Native Address	An address or address prefix that matches one of a given node's summary addresses.
Neighbor Node	A node that is directly connected to a particular node via a logical link.
Network Management Entity (NM)	The body of software in a switching system that provides the ability to manage the PNNI protocol. NM interacts with the PNNI protocol through the MIB.
Nodal Attribute	A nodal state parameter that is considered individually to determine whether a given node is acceptable and/or desirable for carrying a given connection.
Nodal Constraint	A restriction on the use of nodes for path selection for a specific connection.
Nodal Metric	A nodal parameter that requires the values of the parameter for all nodes along a given path to be combined to determine whether the path is acceptable and/or desirable for carrying a given connection.
Nodal State Parameter	Information that captures an aspect or property of a node.
Node	Synonymous with logical node.
Non-Branching Node	A node that cannot currently support additional branching points for point-to-multipoint calls.
Nucleus	The interior reference point of a logical node in the PNNI complex node representation.
Null	A value of all zeros.
Outlier	A node whose exclusion from its containing peer group would significantly improve the accuracy and simplicity of the aggregation of the remainder of the peer group topology.
Outside Link	A link to a lowest-level outside node. In contrast to an inside link (i.e., horizontal link) or an uplink, an outside link does not form part of the PNNI topology, and is therefore not used in path computation.
Outside Node	A node which is participating in PNNI routing, but which is not a member of a particular peer group.
Parent Node	The logical group node that represents the containing peer group of a specific node at the next higher level of the hierarchy.

Parent Peer Group	The parent peer group of a peer group is the one containing the logical group node representing that peer group. The parent peer group of a node is the one containing the parent node of that node.
Path Constraint	A bound on the combined value of a topology metric along a path for a specific connection.
Path Scope	The highest level of PNNI hierarchy used by a path.
Peer Group	A set of logical nodes which are grouped for purposes of creating a routing hierarchy. PTSEs are exchanged among all members of the group.
Peer Group Identifier	A string of bits that is used to unambiguously identify a peer group.
Peer Group Leader	A node of a peer group that performs the extra work of collecting, aggregating, and building data that will be suitable to represent the entire peer group as a single node. This representation is made available in the parent node.
Peer Group Level	The number of significant bits in the peer group identifier of a particular peer group.
Peer Node	A node that is a member of the same peer group as a given node.
Physical Link	A real link which attaches two switching systems.
PNNI Protocol Entity	The body of software in a switching system that executes the PNNI protocol and provides the routing service.
PNNI Routing Control Channel	VCCs used for the exchange of PNNI routing protocol messages.
PNNI Routing Domain	A group of topologically contiguous systems which are running one instance of PNNI routing.
PNNI Routing Hierarchy	The hierarchy of peer groups used for PNNI routing.
PNNI Topology State Element	A collection of PNNI information that is flooded among all logical nodes within a peer group.
PNNI Topology State Packet	A type of PNNI Routing packet that is used for flooding PTSEs among logical nodes within a peer group.
Port	The point of attachment of a link to a node.
Port Identifier	The identifier assigned by a logical node to represent the point of attachment of a link to that node.
Private ATM Address	A twenty-octet address used to identify an ATM connection termination point.

Reachable Address Prefix	A prefix on a 20 octet ATM address indicating that all addresses beginning with this prefix are reachable.
Read-Only (RO)	Attributes which are read-only can not be written by Network Management. Only the PNNI Protocol entity may change the value of a read-only attribute. Network Management entities are restricted to only reading such read-only attributes. Read-only attributes are typically for statistical information, including reporting result of actions taken by auto-configuration.
Read-Write (RW)	Attributes which are read-write can not be written by the PNNI protocol entity. Only the Network Management Entity may change the value of a read-write attribute. The PNNI Protocol Entity is restricted to only reading such read-write attributes. Read-write attributes are typically used to provide the ability for Network Management to configure, control, and manage a PNNI Protocol Entity's behavior.
Restricted Transit Node	A node that is to be used for transit by a call only in restricted circumstances. It is free from such restriction when it is used to originate or terminate a call.
Routing Computation	The process of applying a mathematical algorithm to a topology database to compute routes. There are many types of routing computations that may be used. The Dijkstra algorithm is one particular example of a possible routing computation.
Routing Constraint	A generic term that refers to either a topology constraint or a path constraint.
Scope	A scope defines the level of advertisement for an address. The level is a level of a peer group in the PNNI routing hierarchy.
Source Route	As used in this document, a hierarchically complete source route.
Split System	A switching system which implements the functions of more than one logical node.
Spoke	In the complex node representation, this represents the connectivity between the nucleus and a specific port.
Summary Address	An address prefix that tells a node how to summarize reachability information.
Switching System	A set of one or more physical devices that act together as a single PNNI network management entity. A switching system contains one or more lowest-level nodes and, when it is acting as a PGL, one or more LGNs.
Topology State Parameter	A generic term that refers to either a link parameter or a nodal parameter.
Topology Aggregation	The process of summarizing and compressing topology information at a hierarchical level to be advertised at the level above.

Topology Attribute	A generic term that refers to either a link attribute or a nodal attribute.
Topology Constraint	A topology constraint is a generic term that refers to either a link constraint or a nodal constraint.
Topology Database	The database that describes the topology of the entire PNNI routing domain as seen by a node.
Topology Metric	A generic term that refers to either a link metric or a nodal metric.
Uplink	Represents the connectivity from a border node to an upnode.
Upnode	The node that represents a border node’s outside neighbor in the common peer group. The upnode must be a neighboring peer of one of the border node’s ancestors.

2.3 Notation and Conventions

string	A dot delimited string such as “A.2.3.1” represents a node ID or a peer group ID.
<*string*>	A string enclosed in brackets “< >“ represents an ATM address. Specifically, it is the address of the entity identified by the string.
P<*string*>	A string enclosed by the notation “P< >“ represents a prefix of an ATM address.
PG(*string*)	A string enclosed by the notation “PG( )” represents a peer group ID.

3. PNNI Routing Description

In this chapter an introduction to PNNI routing is provided. A detailed definition of the protocols is given in Chapter 5. In case of discrepancies, the material in Chapter 5 has precedence over this chapter.

The functions of the PNNI routing protocol include:

- Discovery of neighbors and link status.
- Synchronization of topology databases.
- Flooding of PTSEs.
- Election of PGLs.
- Summarization of topology state information.
- Construction of the routing hierarchy.

The remainder of this chapter provides a bottom up description of the PNNI routing protocol.

3.1 Physical Network

PNNI routing applies to a network of lowest-level nodes. Figure 3-1 illustrates such a network, consisting of 26 interconnected lowest-level nodes shown as small circles.

Data passes through lowest-level nodes to other lowest-level nodes and to end systems. End systems are points of origin and termination of connections. End systems are not shown in Figure 3-1. For purposes of route determination, end systems are identified by the 19 most significant octets of ATM End System Addresses. The selector octet (the last octet) is not used for purposes of PNNI route determination but may be used by end systems.

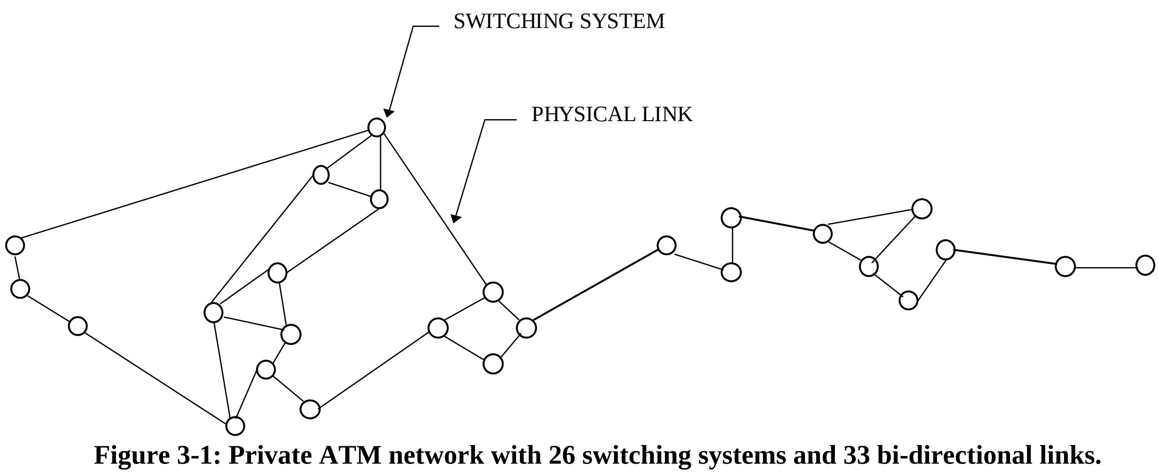


Figure 3-1: Private ATM network with 26 switching systems and 33 bi-directional links.

Each arc in Figure 3-1 represents a “physical link” attaching two switching systems. A port is the attachment point of a link to a lowest-level node within a switching system. Physical links are duplex (traffic may be carried in either direction). However, physical link characteristics may be different in each direction, either because the capacities are different or because existing traffic loads differ. Each physical link is therefore identified by two sets of parameters, one for each direction. Such a set consists of a transmitting port Identifier plus the node ID of the lowest-level node containing that port. Note that PNNI port IDs (as described in Section 5.3.4) may be different from equipment specific port identifiers.

3.2 The Lowest Hierarchical Level

If the PNNI protocol supported only the flat network representation depicted in Figure 3-1, then each lowest-level node would have to maintain the entire topology of the network, including information for every physical link in the network and reachability information for every node in the network. Although feasible for small networks, this would create enormous overhead for larger networks. The PNNI routing hierarchy is designed to reduce this overhead while providing for efficient routing.

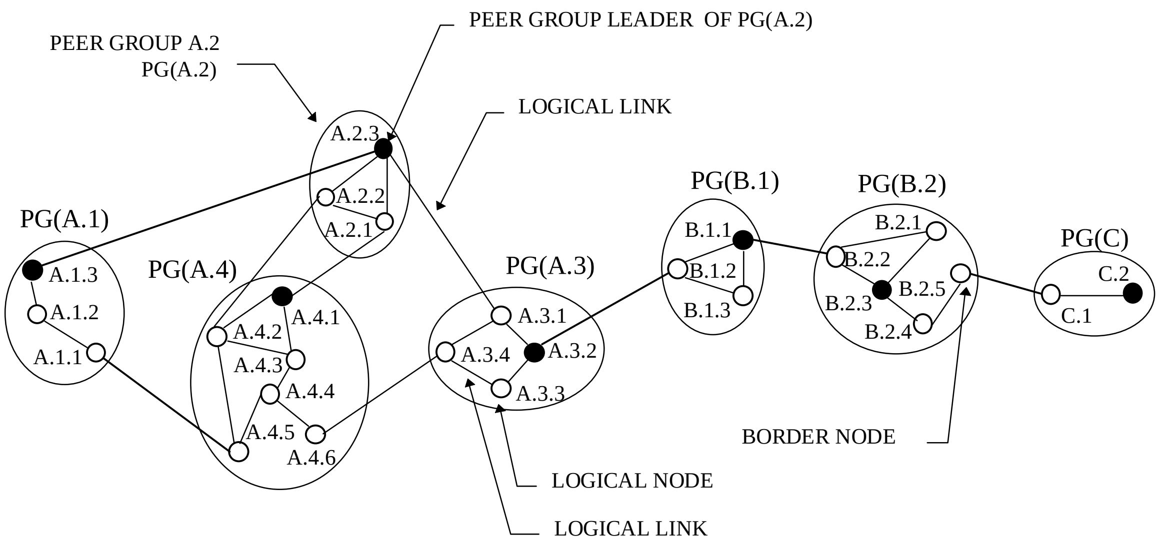


Figure 3-2: Partially configured PNNI hierarchy showing lowest level nodes.

This section begins with the description of the PNNI routing hierarchy by focusing on the lowest hierarchical level. Note that partial connectivity does exist before the entire PNNI routing hierarchy has been established; the entire PNNI routing hierarchy must be present before full connectivity between all nodes is achieved.

3.2.1 Peer Groups and Logical Nodes

The PNNI hierarchy begins at the lowest level where the lowest-level nodes are organized into peer groups. A “logical node” in the context of the lowest hierarchy level is a lowest-level node. For simplicity, logical nodes are often denoted as “nodes”. A peer group (PG) is a collection of logical nodes, each of which exchanges information with other members of the group, such that all members maintain an identical view of the group. Logical nodes are uniquely and unambiguously identified by “logical node IDs”.

In the example (see Figure 3-2) the network is organized into 7 peer groups A.1, A.2, A.3, A.4, B.1, B.2, and C. To avoid later confusion between nodes and peer groups, peer groups are represented in the figures by PG(). For example PG(A.3) denotes peer group A.3. Node and peer group numbering, such as A.3.2 and A.3, is for identification purposes when reading this document. It is an abstract representation that reflects the hierarchical structure being described. For example the node denoted by A.3.2 is located in PG(A.3), i.e. in peer group A.3.

A peer group is identified by its “peer group identifier”. Peer group IDs are specified at configuration time. Neighboring nodes exchange peer group IDs in “Hello packets” (see Section 5.6 for more information on the Hello protocol). If they have the same peer group ID then they belong to the same peer group. If the exchanged peer group IDs are different, then the nodes belong to different peer groups.

A “border node” has at least one link that crosses the peer group boundary. Hence neighboring nodes with different peer group IDs are border nodes of their respective peer groups. In the presence of certain errors or failures, peer groups can partition, leading to the formation of multiple peer groups with the same peer group ID.

The peer group ID is defined as a prefix of at most 13 octets on an ATM End System Address. Thus the peer group ID can default to a prefix on the address(s) of one or more nodes belonging to the peer group in question, but this is not a requirement.

3.2.2 Logical Links And Their Initialization

Logical nodes are connected by “logical links”. Between lowest level nodes, a logical link is either a physical link (such as those shown in Figure 3-1) or a VPC between two lowest-level nodes. Links between lowest level nodes in the same peer group are not aggregated. For example if two physical links were to connect the same pair of lowest-level nodes in Figure 3-1 then they would be represented by two separate logical links in Figure 3-2. Logical links inside a peer group are “horizontal links” whereas links that connect two peer groups are “outside links”.

When a logical link becomes operational, the attached nodes initiate an exchange of information via a well-known VCC used as a “PNNI Routing Control Channel” (RCC). Hello packets sent periodically by each node on this link specify the ATM End System Address, node ID, and its port ID for the link. In this way the Hello protocol makes the two neighboring nodes known to each other. As already mentioned, the PNNI Hello protocol also supports the exchange of peer group IDs so that neighboring nodes can determine whether they belong to a same peer group or to different peer groups.

The Hello protocol runs as long as the link is operational. It can therefore act as a link failure detector when other mechanisms fail.

3.2.3 Information Exchange in PNNI

Each node exchanges Hello packets with its immediate neighbors and thereby determines its local state information. This state information includes the identity and peer group membership of the node’s immediate neighbors, and the status of its links to the neighbors. Each node then bundles its state information in “PNNI Topology State Elements” (PTSEs), which are reliably flooded throughout the peer group.

PTSEs are the smallest collection of PNNI routing information that is flooded as a unit among all logical nodes within a peer group. A node’s topology database consists of a collection of all PTSEs received, which represent that node’s present view of the PNNI routing domain. In particular the topology database provides all the information required to compute a route from the given node to any address reachable in or through that routing domain.

3.2.3.1 Nodal Information

Every node generates a PTSE that describes its own identity and capabilities, information used to elect the peer group leader (see Section 3.2.4), as well as information used in establishing the PNNI hierarchy (see Section 3.3.1). This is referred to as the nodal information.

3.2.3.2 Topology State Information

PTSEs contain, among other things, “topology state parameters” (i.e. “link state parameters”, which describe the characteristics of logical links, and “nodal state parameters”, which describe the characteristics of nodes).

Topology state parameters are classified as either attributes or metrics. An attribute is considered individually when making routing decisions. For example a security “nodal attribute” could cause a given path to be refused. A metric on the other hand is a parameter whose effect is cumulative along a path. For example, delay “metrics” add up as one progresses along a given path. See Section 5.8 for more details.

Certain topology state information, especially that related to bandwidth, is rather dynamic. On the other hand, other topology state information, such as Administrative Weight, may be relatively static. There is no distinction between dynamic and static topology state parameters in the flooding mechanism for PNNI topology distribution.

3.2.3.3 Reachability Information

Reachability information consists of addresses and address prefixes which describe the destinations to which calls may be routed. This information is advertised in PTSEs by nodes in the PNNI routing domain.

Internal and exterior reachability information is logically distinguished based on its source. PNNI routing may not be the only protocol used for routing in an ATM network. Exterior reachability is derived from other protocol exchanges outside this PNNI routing domain. Internal reachability represents local knowledge of reachability within the PNNI routing domain. The primary significance of this distinction is that exterior reachability information shall not be advertised to other routing protocols or routing domains (for fear of causing routing loops across routing domains). Manual configuration can be used to create internal or exterior reachability information with corresponding effects on what is advertised to other routing protocols or domains.

Exterior reachable addresses may also be used to advertise connectivity to otherwise independent PNNI routing domains.

3.2.3.4 Initial Topology Database Exchange

When neighboring nodes, at either end of a logical link being initialized through the exchange of Hellos, conclude that they are in the same peer group, they proceed to synchronize their “topology databases”. Database synchronization is the exchange of information between neighbor nodes resulting in the two nodes having identical topology databases. The topology database includes detailed topology information about the peer group in which the logical node resides plus more abstract topology information representing the remainder of the PNNI routing domain. The way in which this higher level information flows into the peer group is described in subsequent Sections 3.3 to 3.5.

During a topology database synchronization, the nodes in question first exchange PTSE header information, i.e. they advertise the presence of PTSEs in their respective topology database. When a node receives PTSE header information that advertises a more recent PTSE version than the one it has or advertises a PTSE that it does not yet have, it requests the advertised PTSE and updates its topology database with the subsequently received PTSE. If a newly initialized node connects to a peer group then the ensuing database synchronization reduces to a one way topology database copy.

A link is advertised via PTSE transmissions only after the database synchronization between the respective neighboring nodes has successfully completed. In this way the link state parameters are distributed to all topology databases in the peer group containing that link. As we shall see in the following section, flooding is the mechanism used for advertising links.

3.2.3.5 Flooding

Flooding is the reliable hop-by-hop propagation of PTSEs throughout a peer group. It ensures that each node in a peer group maintains an identical topology database (Section 5.8.3). Flooding is the advertising mechanism in PNNI.

In essence, the flooding procedure is as follows. PTSEs are encapsulated within “PNNI topology state packets” (PTSPs) for transmission. When a PTSP is received its component PTSEs are examined. Each PTSE is acknowledged by encapsulating information from its PTSE header within an “Acknowledgment Packet”, which is sent back to the sending neighbor. If the PTSE is new or of more recent origin than the node’s current copy, it is installed in the topology database and flooded to all neighbor nodes except the one from which the PTSE was received. A PTSE sent to a neighbor is periodically retransmitted until acknowledged.

Flooding is an ongoing activity, i.e. each node issues PTSPs with PTSEs that contain updated information. The PTSEs contained in topology databases are subject to aging and get removed after a predefined duration if they are not refreshed by new incoming PTSEs. Only the node that originally originates a particular PTSE can reoriginate that PTSE. PTSEs are reissued both periodically and on an event driven basis.

3.2.4 Peer Group Leader

Section 3.3 describes how a peer group is represented in the next hierarchical level by a single node called a “logical group node”. The functions needed to perform this role are executed by a node, called the “peer group leader”, that is a member of the peer group being represented. There is at most one active peer group leader (PGL) per peer group; more precisely at most one per partition in the case of a partitioned peer group.

Apart from its specific role in aggregation and distribution of information for maintaining the PNNI hierarchy, the PGL does not have any special role in the peer group. For all other functions, e.g., connection establishment, it acts like any other node.

A “peer group leader election” (PGLE) process determines which node takes on this leader function. The criterion for election is a node’s “leadership priority”. The node with highest leadership priority in a peer group becomes leader of that peer group. Note that the election process is a continuously running protocol. When a node becomes active with a leadership priority higher than that of the current PGL, the election process transfers peer group leadership to the newly activated node. When a PGL is removed or fails, the node with the next highest leadership priority becomes PGL. Finally, if a race condition between multiple nodes with identical leadership priority occurs then that node with the highest node ID becomes PGL. Note that when a node is elected PGL, its leadership priority is increased to ensure stability.

Internal operation of a peer group does not require having a peer group leader. Full connectivity within a peer group can be achieved without a peer group leader. A PNNI Routing Domain configured as a single peer group can achieve full connectivity even without a peer group leader.

A degenerate form of a peer group is one containing a single node. The peer group leader of a single node peer group is the node itself. This could occur through configuration, or as a result of failures.

3.3 One Hierarchical Level Up

This section describes how peer groups are represented in the next hierarchical level.

3.3.1 Logical Group Nodes

A “logical group node” is an abstraction of a peer group for the purpose of representing that peer group in the next PNNI routing hierarchy level. For example, in Figure 3-3, logical group node A.2 represents peer group A.2 in the next higher level peer group A. Figure 3-3 shows one way that the peer groups in Figure 3-2 can be organized into the next level of peer group hierarchy.

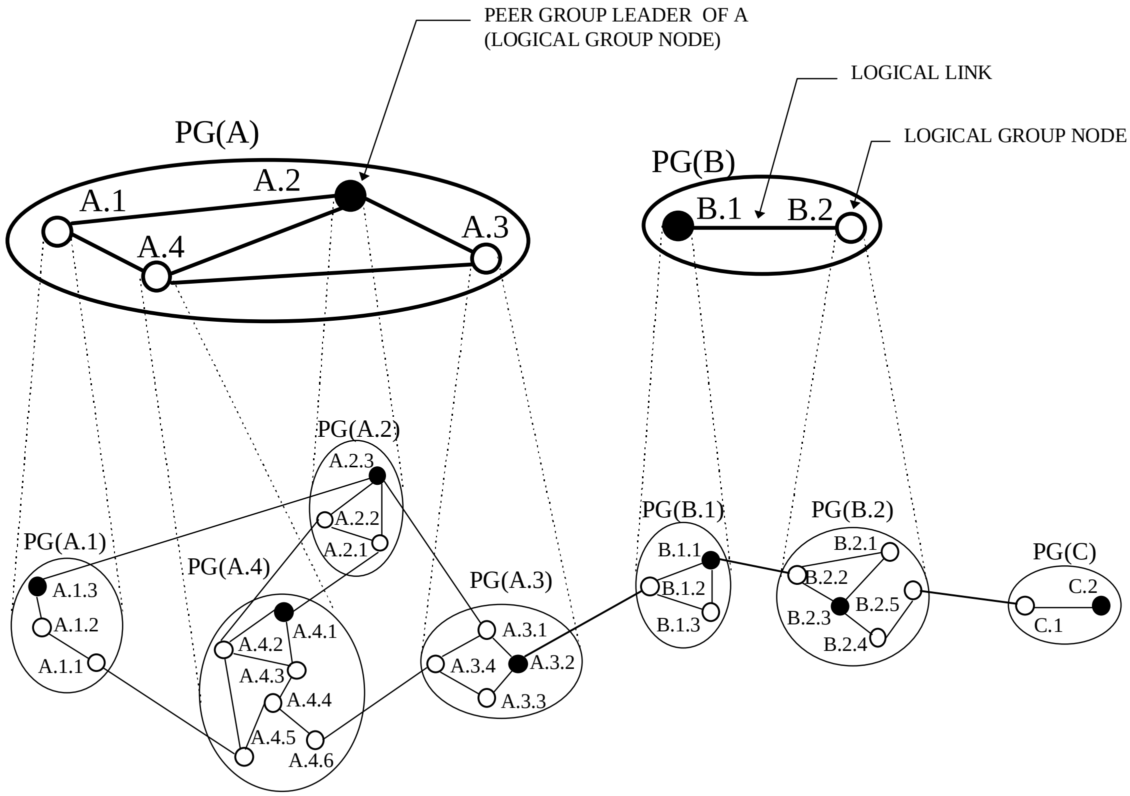


Figure 3-3: Partially configured PNNI hierarchy showing lowest hierarchical levels.

The functions of the logical group node and the peer group leader of its child peer group are closely related. In this specification the functions of these two nodes are considered to be executed in the same system. Therefore, the interface between those two components is not specified.

The functions of a logical group node include aggregating and summarizing information about its child peer group and flooding that information into its own peer group. A logical group node also passes information received from its peer group to the PGL of its child peer group for flooding. A logical group node does not participate in PNNI signalling.

A logical group node is identified by a node ID which by default contains the peer group ID of the peer group that the node is representing.

A logical group node is addressable by a unique ATM End System Address that may, for example, correspond to the address of the lowest-level node in the same switching system but with a different Selector value.

The manner in which a peer group is represented depends on the policies and algorithms of the peer group leader. Thus given two potential peer group leaders that implement the same policies and algorithms, the representation of the peer group does not depend on which of the two is elected.

Observe that logical group nodes in Figure 3-3 are organized into peer groups. For example, logical nodes A.1, A.2, A.3 and A.4 are organized into peer group A. This higher level peer group is a peer group in the sense of Section 3.2, the only difference being that each of its nodes represents a separate lower level peer group. Consequently, peer group A has a peer group leader (logical group node A.2) chosen by the leader election process. Note the functions that define peer group leader of A are located in node A.2, which is in turn implemented on the switching system containing lowest-level node A.2.3.

Peer group A is called the “parent peer group” of peer groups A.1, A.2, A.3 and A.4. Conversely, peer groups A.1, A.2, A.3 and A.4 are called “child peer groups” of peer group A.

A parent peer group is identified by a peer group ID that must be shorter in length than its child peer group IDs. Any node capable of becoming peer group leader must be configured with its parent peer group ID.

The length of a peer group ID indicates the level of that peer group within the PNNI hierarchy. One refers to this length as the “level indicator”. PNNI levels are not dense, in the sense that not all levels will be used in any specific topology. For example a peer group with an ID of length "n" bits may have a parent peer group whose ID ranges anywhere from 0 to n-1 bits in length. Similarly, a peer group with an ID of length "m" bits may have a child peer group whose identifier ranges anywhere from m+1 to 104 bits in length (104 is the maximum peer group ID length and corresponds to 13 octets).

3.3.2 Feeding Information Up The Hierarchy

A logical group node represents an entire underlying peer group. The associated peer group leader, as a member of the underlying peer group, has received complete topology state information from all nodes in the peer group. This provides the peer group leader with all of the required information to instantiate the logical group node. Conceptually this may be thought of as the peer group leader feeding information up to the logical group node it instantiates. This upward flow includes two types of information: reachability and topology aggregation. Reachability refers to summarized address information (see Section 3.5) needed to determine which addresses can be reached through the lower level peer group. Topology aggregation refers to the summarized topology information (see Section 3.3.8) needed to route into and across this peer group.

There is a filtering function inherent in the summarization process that propagates only the information needed by the higher levels. PTSEs never flow up the hierarchy. Instead the summarized information is advertised within PTSEs originated by the logical group node and flooded to its peers.

3.3.3 Feeding Information Down The Hierarchy

Section 3.3.2 described how feeding information up the PNNI routing hierarchy is necessary for creating the hierarchy itself and for distributing routing information about child peer groups. Conversely feeding information down the hierarchy is necessary to allow nodes in the lower level peer groups to route to all destinations reachable via the PNNI routing domain. Route computation uses this information to select routes to destinations.

Each logical group node feeds information down to its underlying peer group. The information fed down consists of all PTSEs it originates or receives via flooding from other members of the LGN’s peer group. Each PTSE that flows down to a peer group leader is flooded across that peer group. This gives every node in a peer group a view of the higher levels into which it is being aggregated. In summary, PTSEs flow horizontally through a peer group and downward into and through child peer groups.

3.3.4 Uplinks (Revisiting The Hello Protocol)

When neighbor nodes conclude from the Hello protocol that they belong to different peer groups, they become border nodes. For example nodes A.3.4 and A.4.6 are border nodes in Figure 3-4. Links between border nodes in different peer groups are called outside links. There is no database exchange across outside links; the only PNNI protocol flows are for the Hello protocol.

Border nodes extend the Hello protocol across outside links to include information (the nodal hierarchy list) about their respective higher level peer groups, and the logical group nodes representing them in these peer groups. This information allows the border nodes to determine the lowest level peer group common to both border nodes. For example in Figure 3-4, the border nodes A.3.4 and A.4.6 identify that they have peer group A in common.

The mechanisms described above allow each node to know the complete topology (including nodes and links) within its peer group, as well as the complete (summarized) topology of the higher level parent peer group, and grand-parent peer group etc. In order for the node to realize which (border) nodes have connectivity to which higher level nodes, the border nodes must advertise links to those higher level nodes. These are called uplinks. The node at the other end of the uplink, the upnode, is always a neighboring peer of one of its ancestor nodes (in the two-level case depicted in Figure 3-4, a neighboring peer of its parent node).

For certain Hello states, a border node must include an Uplink Information Attribute (ULIA) information group in the Hello packets its sends on each outside link to neighboring border nodes. The ULIA sent to an outside neighbor encloses a complete set of information about the reverse direction resources to be advertised in the uplink advertisement that the receiving outside neighbor generates.

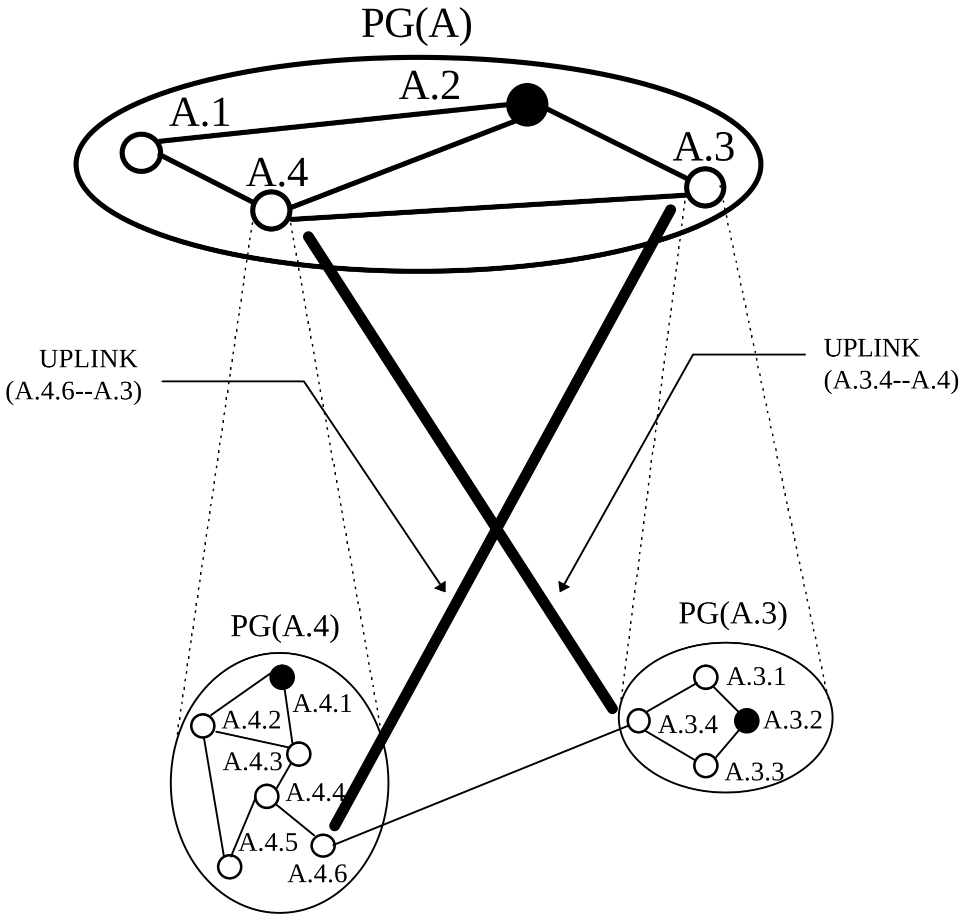


Figure 3-4: Uplinks.

The nodal hierarchy list provides the border nodes with the information necessary for them to determine their common higher level peer groups, and to identify the higher level nodes to which the border nodes will declare connectivity. For example in Figure 3-4, border node A.3.4 recognizes that its neighbor A.4.6 is represented by logical group node A.4 in the common parent peer group A. Consequently A.3.4 advertises an uplink between itself and upnode A.4. The uplink is denoted by (A.3.4--A.4). Similarly node A.4.6 has an uplink (A.4.6--A.3) to the remote higher level node (or upnode) A.3.

Border nodes advertise their uplinks in PTSEs flooded in their respective peer groups. This enables all nodes in the peer group to update their topology databases with the uplinks. It also gives the peer group leaders reachability information that must be fed up the hierarchy since uplinks help create the higher level peer group.

Topology state parameters for both directions of an uplink are included in the uplink PTSE since the upnode does not advertise a PTSE for the downward direction. Topology state parameters for the reverse direction of the uplink are exchanged in Hello packets on the outside link.

3.3.5 PNNI Routing Control Channels

Neighboring PNNI nodes have a routing control channel for the exchange of PNNI routing information. Neighboring nodes at their lowest level of the PNNI routing hierarchy use a reserved VCC for their routing control channel. The routing control channel between logical group nodes is an SVCC. The information required to establish this SVCC is derived from the uplink advertisements in the peer group represented by the logical group node.

For example, to establish the RCC between A.3 and A.4 the following steps are taken. Peer group leader A.3.2 receives the PTSE describing the uplink (A.3.4--A.4) flooded by A.3.4. A.3.2 extracts two key pieces of information from this PTSE, the ID of the common peer group that was previously identified as peer group A by the Hello protocol executing across (A.3.4--A.4.6), and the ATM End System Address of upnode A.4. Node A.3.2 passes this information to its logical group node A.3. From this information, A.3 deduces that it belongs to the same peer group as A.4. It also deduces that A.4 is a neighbor and it therefore needs to have an RCC to A.4.

Since A.3 has the ATM End System Address used to reach A.4, it has sufficient information to set up the SVCC. Similarly logical group node A.4 has enough information to set up the SVCC to A.3. Note that the protocol ensures that only one of them will initiate the SVCC establishment.

Note that at the time the SVCC for the RCC is being established, the nodes in both neighboring peer groups have all information required to route the SVCC. In particular, each peer group has its own internal topology information and knows of the existence and identity of uplinks and upnodes relevant to the routing of the SVCC-based RCC.

3.3.6 Revisiting Initial Topology Database Exchange

Logical group nodes at higher levels have behaviors similar to those of lower level nodes. For example, in Figure 3-3 when the SVCC-based RCC is established between nodes A.3 and A.4, the Hello protocol is activated across it. When A.3 and A.4 confirm that they are in the same peer group they will execute an initial topology database exchange across the RCC. Note A.3 and A.4 already know they belong to a same peer group otherwise they would not have set-up the RCC; the Hello protocol simply confirms the association.

The initial topology database exchange between logical group nodes A.3 and A.4 will synchronize the databases of these two nodes. Note that information specific to the child peer groups of A.3 and A.4 is not part of these databases. Thus the initial topology database exchange between nodes A.3 and A.4 only includes the PTSEs flooded in peer group A. Since PTSEs flow horizontally and downwards (Section 3.3.3), the database exchange will also include PTSEs fed into peer group A by A.2, the peer group leader of A. This assumes that the peer group leader of A exists before link (A.3--A.4) comes up. If that is not the case, no higher-level information will be available in the peer group at that time.

3.3.7 Horizontal Links

Horizontal links are logical links between nodes in the same peer group. Thus not only is (A.3.4--A.3.3) a horizontal link but (A.3--A.4) is also one (see Figure 3-4). This latter horizontal link is in the higher level peer group A and is created as a consequence of the uplinks derived in Section 3.3.4. Hellos are sent to the neighbor LGN over the SVCC-based RCC to exchange port IDs and status for horizontal links.

Horizontal links are not advertised until a successful exchange of Hellos and completion of database synchronization has occurred between neighboring nodes over the RCC. PTSEs describing the new link can now be flooded within the peer group containing the link, i.e. within peer group A, and downwards to the child peer groups.

A horizontal link between a pair of LGNs represents the connectivity between those two nodes for routing purposes. Logical group nodes are responsible for assigning port IDs to horizontal links, as well as other links attached to the node.

3.3.8 Topology Aggregation

Topology aggregation is the notion of reducing nodal as well as link information to achieve scaling in a large network. It is not only motivated by the need for complexity reduction but also to hide the topology internals of peer groups in the interest of security.

3.3.8.1 Link Aggregation

Link aggregation refers to the representation of some set of links between the same two peer groups by a single logical link (see Section 5.10.3.1). For example in Figure 3-3, the link connecting node A.2 to A.4 represents the aggregation of the two links (A.2.1--A.4.1), and (A.2.2--A.4.2).

Logical group nodes are responsible for link aggregation. A logical group node examines all of the uplink advertisements from its child peer group to a specific upnode. All uplinks to the same upnode with the same aggregation token, as the result of configuration, are aggregated into a single link. This link could be either a horizontal link, if the upnode is a peer of the logical group node, or an induced uplink otherwise (see Section 3.4.2).

3.3.8.2 Nodal Aggregation

Nodal aggregation is the process of representing a child peer group by a logical group node in its parent peer group.

The “complex node representation” is used to represent the result of this aggregation in the parent peer group. It is also permissible to use the complex node representation to model a lowest-level node.

The simplest complex node representation is a symmetric star topology with a uniform radius. The center of the star is the interior reference point of the logical node, and is referred to as the nucleus. The logical connectivity between the nucleus and a port of the logical node is referred to as a spoke. The concatenation of two spokes represents traversal of a symmetric peer group. The symmetric star topology is used as the “default node representation”, which consists of a single value for each nodal state parameter giving a presumed value between any entry or exit of the logical node and the nucleus, in either direction.

Usually peer groups are not symmetric. For example they may contain “outliers”, i.e. nodes whose removal would significantly improve the peer group symmetry. This asymmetry can be modeled by a set of “exceptions.” Exceptions can be used to represent particular ports whose connectivity to the nucleus is significantly different from the default. Additionally, an exception can be used to represent connectivity between two ports, i.e., a bypass, that is significantly better than that implied by traversing the nucleus.

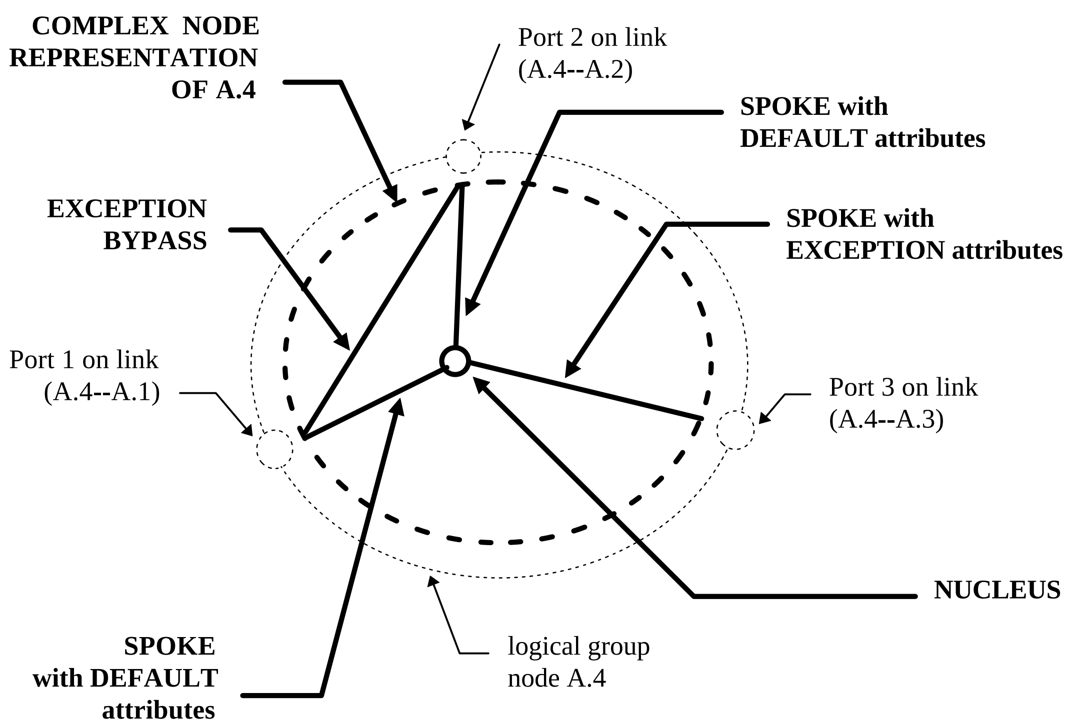


Figure 3-5: Complex node representation of LGN A.4 from Figure 3-3.

The complex node representation is illustrated in the following example. Consider peer group A.4 in Figure 3-3 and its summarization into the complex node represented in Figure 3-5. The nucleus represents the inside of the logical group node A.4. Each spoke emanating from the nucleus is associated with a port of the logical group node A.4. Figure 3-5 includes three spokes, one for each port. The three ports relate to Figure 3-3 as follows:

- Port 1 represents the port on link (A.4--A.1).
- Port 2 represents the port on link (A.4--A.2).
- Port 3 represents the port on link (A.4--A.3).

Note that the spokes for ports 1 and 2 use the default attributes, while the spoke for port 3 is an exception. One possible cause for the exception would be if nodes A.4.1 through A.4.5 are closely clustered whereas A.4.6 is a distant outlier.

Node traversal can also be modeled by an exception that bypasses the nucleus. For example (Figure 3-5) the Exception Bypass joining port 1 to port 2 corresponds to traversing A.4 when going between nodes A.1 and A.2. This exception could be caused by a very high speed bypass link between nodes A.4.2 and A.4.5

Node A.4 distributes its complex node representation via flooding of PTSEs to A.1, A.2 and A.3 The PTSEs are then passed down to the respective peer group leaders (Section 3.3.3) who in turn flood them across their own groups with the result that the topology databases of all nodes in peer groups A.1, A.2, A.3, and A.4 contain the complex node representation of A.4.

Routing across a logical group node corresponds to choosing a concatenation of spokes and/or bypasses. There is never a case where more than two spokes are included in such a concatenation, or there would be a loop. The concatenation must be selected to meet the resource requirements of the call. For example (Figures 3-3 and 3-5) assume logical link (A.1--A.2) is congested so that node A.3.4 routes to a node inside peer group A.1 either via logical links (A.3--A.4), (A.4--A.1) or (A.3--A.2), (A.2--A.4), (A.4--

A.1). Furthermore assume that the two paths are equivalent from the routing criteria point of view (plausible since both traverse similar number of links). Then path selection corresponds to choosing the best of three possible complex node traversals (Figure 3-5):

1. Concatenation of the two spokes with default attributes.
2. Concatenation of the Exception spoke and a spoke with default attributes.
3. Exception Bypass.

If the best is Exception Bypass then (A.3--A.2), (A.2--A.4), (A.4--A.1) is the preferred route. Node A.3.4 will therefore use the link (A.4.2--A.4.5) when routing to a node inside peer group A.1.

Routing to an internal reachable address in a logical group node corresponds to choosing a spoke or a concatenation of bypasses and a spoke to the nucleus with acceptable attributes.

3.4 Completing The PNNI Routing Hierarchy

In this section the PNNI routing hierarchy description is completed by focusing on the recursive nature of the peer groups.

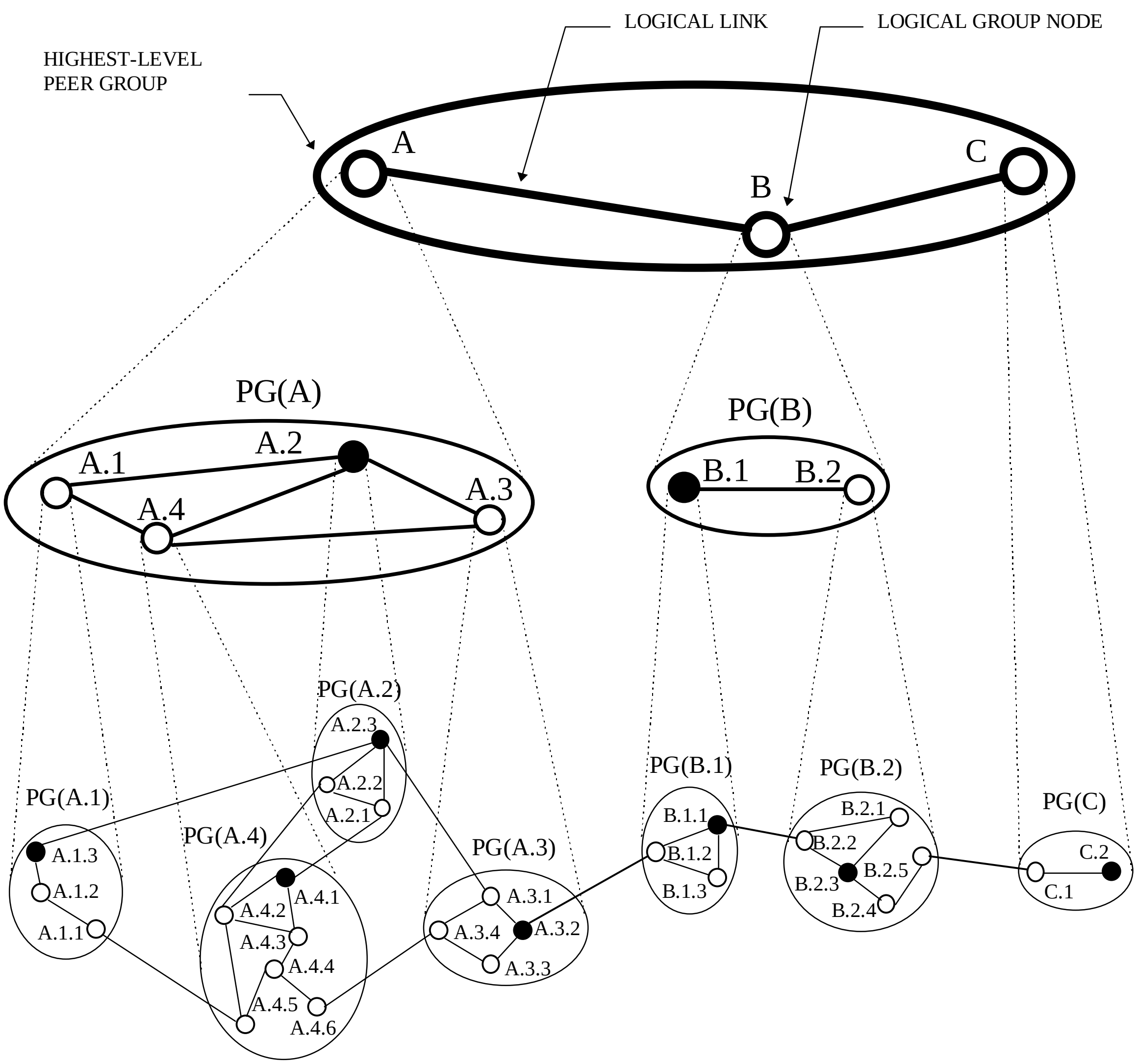


Figure 3-6: Example of a complete PNNI hierarchically configured network.

3.4.1 Progressing To The Highest Level Peer Group

The PNNI routing hierarchy example of the previous sections is still incomplete. The higher level peer groups (Figure 3-3) do not exhibit connectivity among each other. Figure 3-6 shows one possible completion of the hierarchy. Completion is achieved by creating ever higher levels of peer groups until the entire network is encompassed in a single highest level peer group. In the example of Figure 3-6 this is achieved by configuring one more peer group containing logical group nodes A, B and C. Node A represents peer group A which in turn represents peer groups A.1, A.2, A.3, A.4 and so on. Another possible configuration would be if peer groups B and C were aggregated into a peer group BC which was then aggregated with peer group A to form the highest level peer group. The network designer controls the hierarchy via configuration parameters that define the logical nodes and peer groups.

The hierarchical structure is very flexible. The upper limit on successive, child/parent related, peer groups is given by the maximum number of ever shorter address prefixes that can be derived from longest 13 octet address prefix. This equates to 104, more than enough since even international networks can be more than adequately configured with less than 10 levels of ancestry.

3.4.2 Uplinks And Horizontal links Revisited

The PNNI routing hierarchy allows asymmetries in the sense that for a given lower level peer group its parent peer group can simultaneously be a grandparent or great-grandparent peer group to some other lower level peer group. For example, the lower level peer group C, in Figure 3-6, is directly represented in the highest level peer group by logical group node C whereas lower level peer groups B.1 and B.2 are first grouped into the parent peer group B before being represented at the highest level by logical group node B.

The uplinks in this scenario are shown in Figure 3-7. The creation of the four uplinks (B.2.5--C), (C.1--B), (B.1.1--B.2) and (B.2.2--B.1) are covered in Section 3.3.4. However, uplink (B.2--C) is different in that it is derived from uplink (B.2.5--C). This is called an “induced uplink.”

When peer group leader B.2.3 receives the PTSE (flooded by B.2.5) describing the uplink (B.2.5--C) it passes the common peer group ID (the highest level peer group in this case) and the ATM End System Address of the upnode C to its LGN B.2. From this information, B.2 recognizes that node C is not a member of peer group B. It therefore derives a new uplink (B.2--C) that expresses the reachability to C from the perspective of B.2. Since B.2 represents B.2.5, (B.2--C) is a higher level representation of (B.2.5--C).

If another link were to connect B.2.1 to C.1, then uplink (B.2--C) could represent the aggregation of two lower level uplinks, namely (B.2.5--C) and (B.2.1--C). This would be determined by the aggregation tokens in the two uplinks.

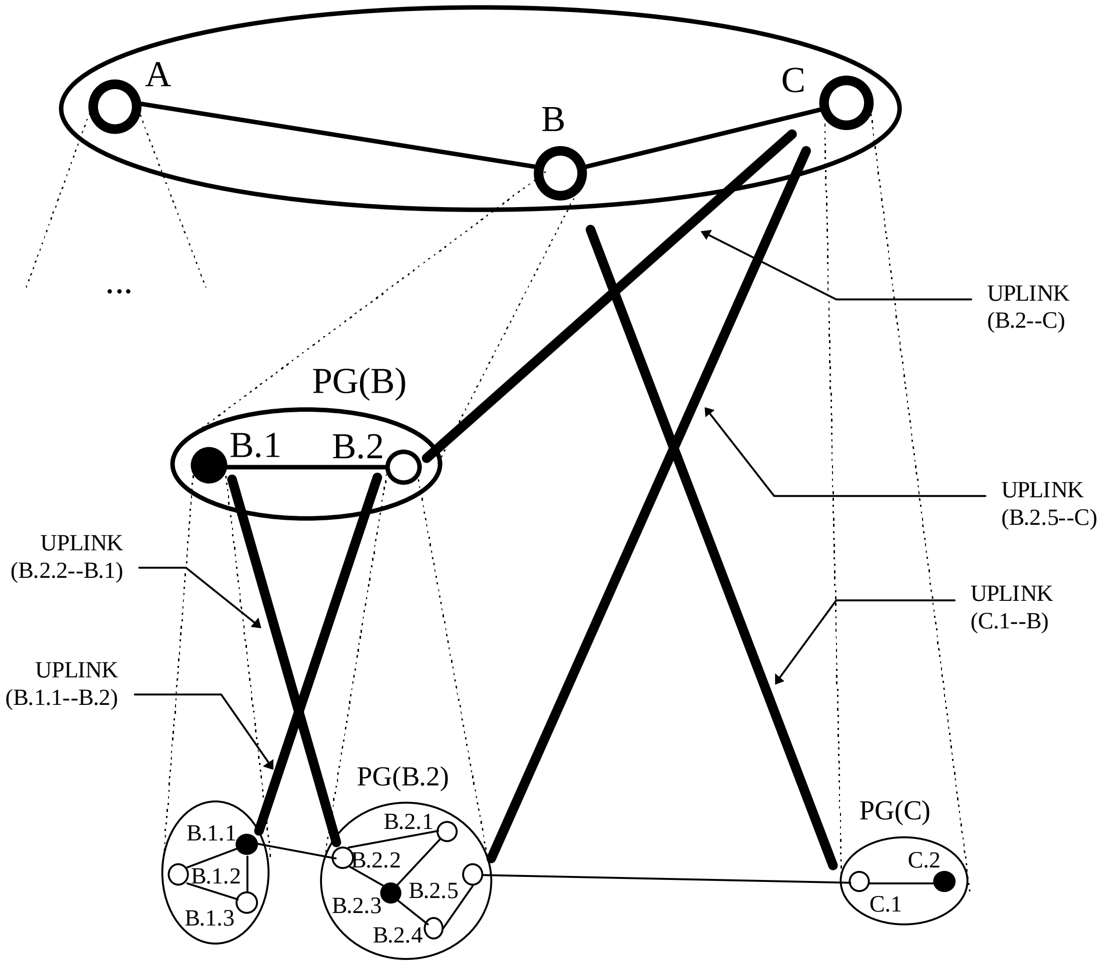


Figure 3-7: Uplinks revisited.

Induced uplinks may also be derived from induced uplinks at the next lower level. In the example, if there were additional peer groups between peer group B and the highest level peer group, there would be additional uplinks derived to upnode C.

An SVCC-based RCC is established between LGNs B and C as follows. After creating the uplink (B.2--C), B.2 floods its peer group B with a PTSE specifying the new uplink. On receiving the uplink information, peer group leader B.1 passes the common peer group ID (the highest level peer group in this case) and the ATM End System Address of the upnode C to its LGN B. Node B recognizes from this that node C is located in its peer group and now has sufficient information to set up the SVCC-based RCC between LGNs B and C.

- Node B chooses a path to node C through one of the border nodes advertising an uplink to C. In this example, B.2 is the only border node advertising an uplink to upnode C, and therefore the SVCC will transit border node B.2.

3.4.3 Recursion in the Hierarchy

The creation of a PNNI routing hierarchy can be viewed as the recursive generation of peer groups, beginning with a network of lowest-level nodes and ending with a single top-level peer group encompassing the entire PNNI routing domain. The hierarchical structure is determined by the way in which peer group IDs are associated with logical group nodes by configuration. From this perspective the key element of a PNNI routing hierarchy is the peer group structure.

Generally, the behavior of a peer group is independent of its level. Thus the descriptions in Section 3.3 apply to all peer groups. However, the highest level peer group differs in that it does not need a peer group leader since there is no parent peer group in which to represent it.

3.5 Address Summarization & Reachability

Address summarization reduces the amount of addressing information which needs to be distributed in a PNNI network and thus contributes to scaling in large networks. It consists of using a single “reachable address prefix” to represent a collection of end system and/or node addresses that begin with the given prefix. Reachable address prefixes can be either summary addresses or foreign addresses.

A “summary address” associated with a node is an address prefix that is either explicitly configured at that node or that takes on some default value. A “foreign address” associated with a node is an address which does not match any of the node’s summary addresses. By contrast a “native address” is an address that matches one of the node’s summary addresses.

These concepts are clarified in the example depicted in Figure 3-8 which is derived from Figure 3-6. The attachments to nodes A.2.1, A.2.2 and A.2.3 represent end systems. The alphanumeric associated with each end system represents that end system’s ATM address. For example <A.2.3.2> represents an ATM address, and P<A.2.3>, P<A.2>, and P<A> represent successively shorter prefixes of that same ATM address.

An example list of configured summary addresses for each node in peer group A.2 is as follows:

Table 3-1: Configured Summary Addresses

Summary Addrs at A.2.1	Summary Addr at A.2.2	Summary Addr at A.2.3
P<A.2.1> (configured)	P<Y.1> (configured)	P<A.2.3> (configured)
P<Y.2> (configured)	P<Z.2> (configured)	

Other summary addresses could have been chosen, for example P<Y.1.1> instead of P<Y.1> at node A.2.2 or P<W> at node A.2.1. But P<A.2> could not have been chosen (instead of P<A.2.1> or P<A.2.3>) as default summary address at nodes A.2.1 and A.2.3 since a remote node selecting a route would not be able to differentiate between the end systems attached to A.2.3 and the end systems attached to A.2.1. Note that for the chosen summary address list at A.2.1, P<W.1.1.1> is a foreign address since it does not match any of the summary addresses of node A.2.1.

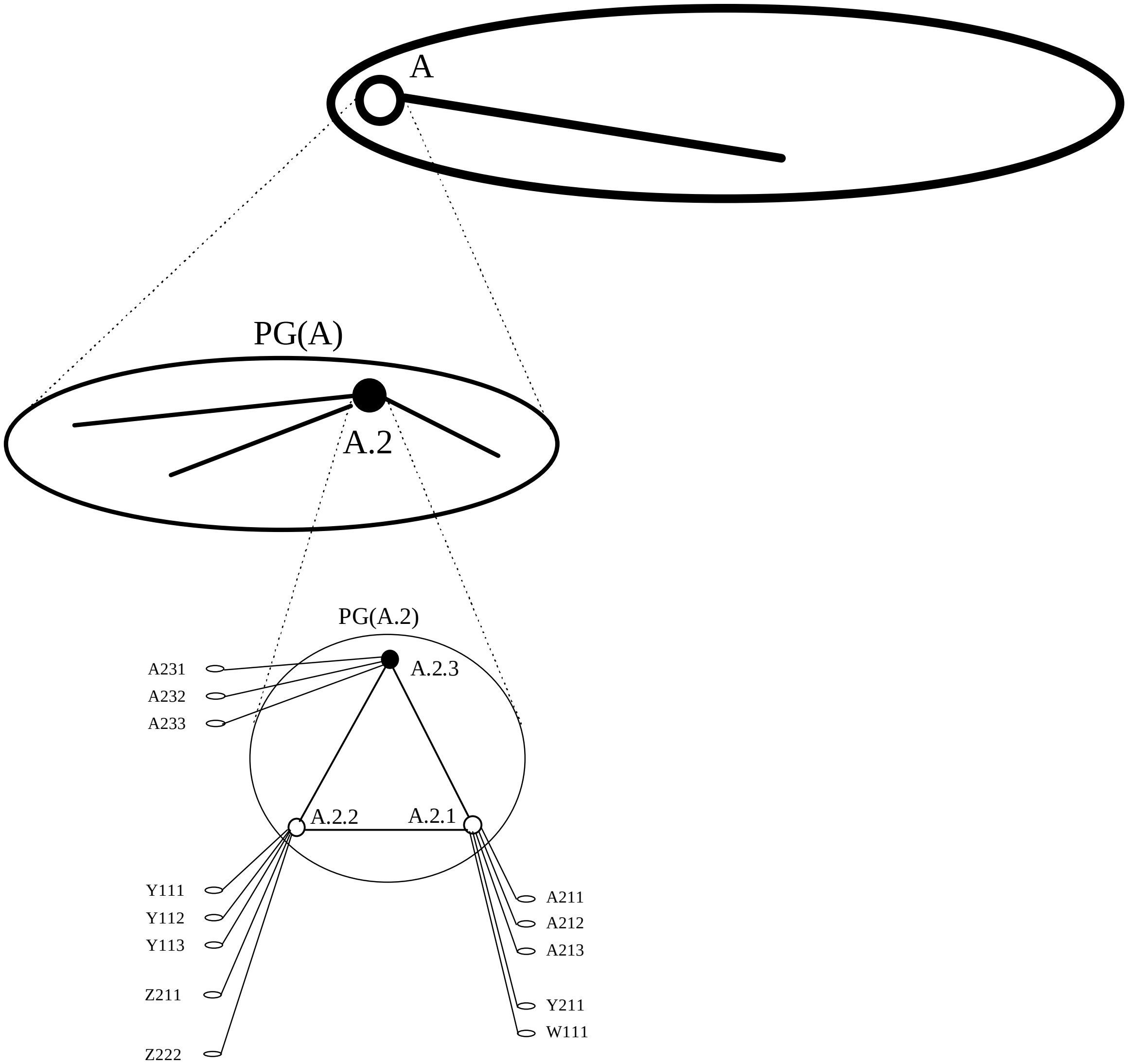


Figure 3-8: Address summarization in a PNNI hierarchy.

LGN A.2 requires its own list of summary addresses. Here again there are different alternatives. As PG(A.2) is the ID of peer group A.2 it is reasonable to include P<A.2> as a default summary address. Furthermore since summary addresses P<Y.1> and P<Y.2> can be further summarized by P<Y>, it also makes sense to configure P<Y> as a summary address in this list:

Table 3-2: Summary Address List

Summary Addr List of LGN A.2
P<A.2> (default) P<Y> (configured)

P<Z.2> is not included in the summary address list of LGN A.2 so as to demonstrate what happens in such a situation. It should be noted that every node in peer group A.2 having the potential to become peer group leader should have the same summary address list in its LGN configuration.

The following table shows the reachable address prefixes advertised by each node in peer group A.2:

Table 3-3: Advertised Reachable Address Prefixes

Reachable Addr Prefixes flooded by node A.2.1	Reachable Addr Prefixes flooded by node A.2.2	Reachable Addr Prefixes flooded by node A.2.3
P<A.2.1>	P<A.2.2>	P<A.2.3>
P<Y.2>	P<Y.1>	
P<W.1.1.1>	P<Z.2>	

As expected, node A.2.1 floods its summary addresses plus its foreign address whereas nodes A.2.2 and A.2.3 only issue summary addresses since they lack foreign addressed end systems.

Reachability information, i.e., reachable address prefixes, are fed throughout the PNNI routing hierarchy so that all nodes can reach the end systems with addresses summarized by these prefixes. A filtering is associated with this information flow to achieve further summarization wherever possible, i.e. LGN A.2 attempts to summarize every reachable address prefix advertised in peer group A.2 by matching it against all summary addresses contained in its list (see Table 3-2). For example when LGN A.2 receives (via PGL A.2.3) the reachable address prefix P<Y.1> issued by node A.2.2 (see Table 3-1) and finds a match with its configured summary address P<Y>, LGN A.2 achieves a further summarization by advertising its summary address P<Y> instead of the longer reachable address prefix P<Y.1>.

There is another filtering associated with advertising of reachability information to limit the distribution of reachable address prefixes. By associating an "suppressed summary address" with the address(es) of end system(s), advertising by an LGN of that summary address is inhibited. This option allows some addresses in the lower level peer group to be hidden from higher levels of the hierarchy, and hence other peer groups. This feature can be implemented for security reasons, making the presence of a particular end system address unknown outside a certain peer group.

Reachable address prefixes that cannot be further summarized by the LGN A.2 are advertised unmodified. For example when LGN A.2 receives the reachable address prefix P<Z.2> issued by A.2.2, the match against all its summary addresses (Table 3-2) fails, consequently LGN A.2 advertises P<Z.2> unmodified. Note that LGN A.2 views P<Z.2> as foreign since the match against all its summary addresses failed, even though P<Z.2> is a summary address from the perspective of node A.2.2. The resulting reachability information advertised by LGN A.2 is listed in Table 3-4:

Table 3-4: Advertised Reachability Information

Reachability information advertised by LGN A.2.
P<A.2>
P<Y>
P<Z.2>
P<W.1.1.1>

Note that the reachability information advertised by node A.2.3 is different than that advertised by LGN A.2 as shown in Tables 3-3 and 3-4, even though node A.2.3 is PGL of peer group A.2. The reachability information advertised by LGN A.2 is the only reachability information about this peer group available outside of the peer group.

The relationship between LGN A and peer group leader A.2 is similar to the relationship between LGN A.2 and peer group leader A.2.3. If LGN A is configured without summary addresses, then it would advertise all reachable address prefixes that were flooded across peer group A into the highest peer group (including the entire list in Table 3-4). On the other hand if LGN A is configured with the default summary address P<A> (default because the ID of peer group A is PG(A)) then it will attempt to further summarize every reachable address prefix beginning with P<A> before advertising it. For example it will advertise the summary address P<A> instead of the address prefix P<A.2> (see Table 3-4) flooded by LGN A.2.

The ATM addresses of the logical nodes are subject to the same summarization rules as end system addresses.

The reachability information (reachable address prefixes) issued by a specific PNNI node is advertised across and up successive (parent) peer groups, then down and across successive (child) peer groups to eventually reach all PNNI nodes lying outside the specified node.

3.5.1 Address Scoping

Reachability information advertised by a logical node always has a scope associated with it. The scope denotes a level in the PNNI routing hierarchy, and it is the highest level at which this address can be advertised or summarized. If an address has a scope indicating a level lower than the level of the node, the node will not advertise the address. If the scope indicates a level that is equal to or higher than the level of the node, the address will be advertised in the node's peer group.

When summarizing addresses, the address to be summarized with the highest scope will determine the scope of the summary address. The same rule applies to group addresses, i.e. if two or more nodes in a peer group advertise reachability to the same group address but with different scope, their parent node will advertise reachability to the group address with the highest scope.

Note that rules related to address suppression take precedence over those for scope.

3.5.1.1 An Example of Address Summarization with Address Scoping

In this section we present a very simple and artificial example, but one that demonstrates most of the possible default cases of address summarization. Suppose we have the very simple network shown in Figure 3-9 consisting three levels of hierarchy and of only two nodes. Suppose also that the number of addresses required to prevent suppression is configured to 1.

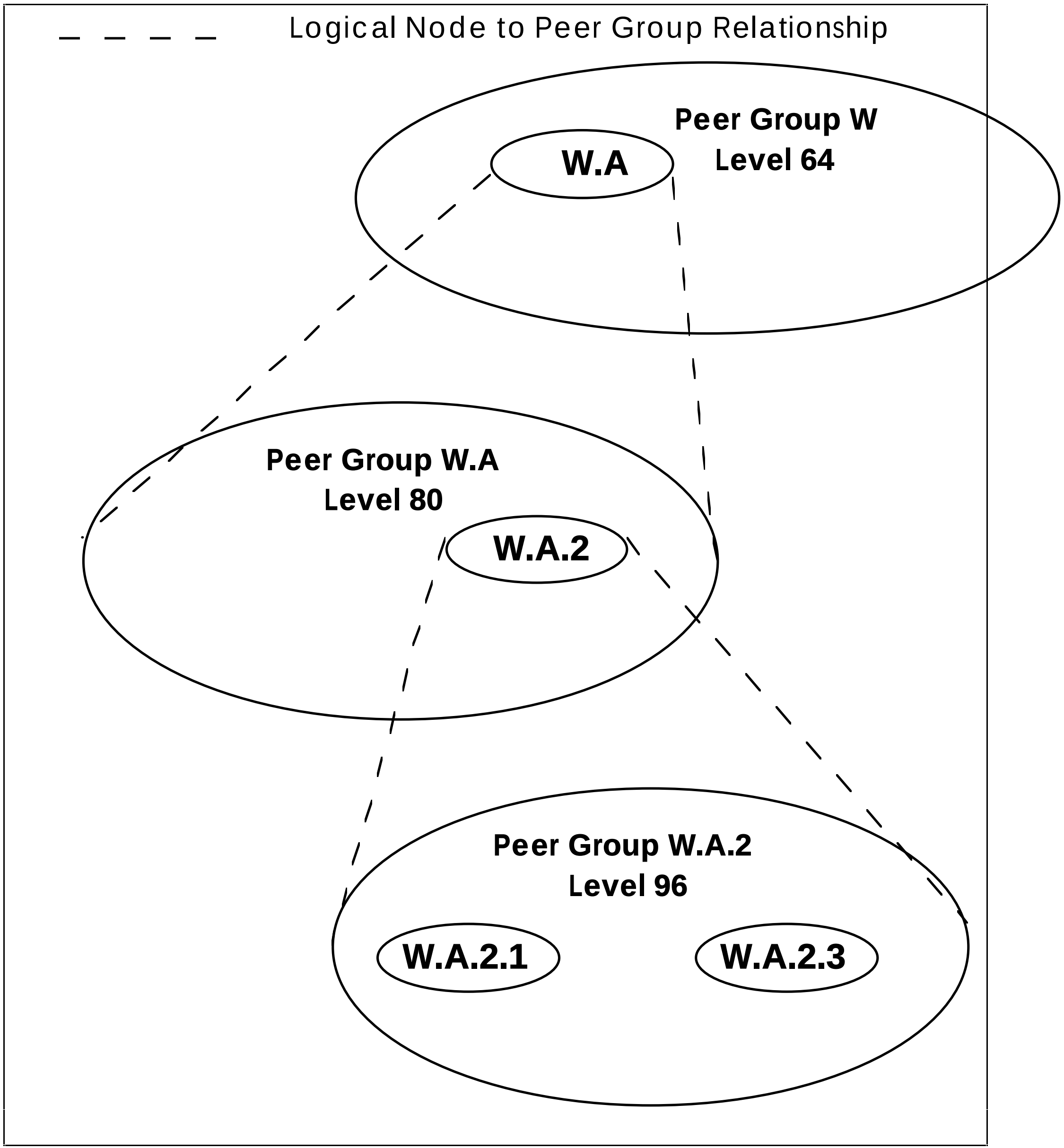


Figure 3-9: Sample Network

Suppose Node W.A.2.1 which is at level 96 has the following reachable addresses:

W.A.2.1.ddd
W.A.2.1.eee
W.A.2.3.fff
W.F.2.7.zzz
A.N.Y.C.ast1 Scope 80
A.N.Y.C.ast2 Scope 104
A.N.Y.C.ast3 Scope 96
A.N.Y.C.ast4 Scope 72

The node would advertise these as follows:

W.A.2.1
W.A.2.3.fff

W.F.2.7.zzz
A.N.Y.C.ast1 Scope 80
A.N.Y.C.ast3 Scope 96
A.N.Y.C.ast4 Scope 72

W.A.2.1 is the node's summary address by default. The addresses which match this are not advertised. The address A.N.Y.C.ast2 was not advertised since its scope, 104, is lower than the node's level.

Node W.A.2.3 is also at level 96 and has the following addresses attached.

W.A.2.3.aaa
W.A.2.3.bbb
W.A.2.3.ccc
W.F.1.5.xxx
W.F.2.7.yyy
A.N.Y.C.ast3 Scope 80

The node would advertise these as follows:

W.A.2.3
W.F.1.5.xxx
W.F.2.7.yyy
A.N.Y.C.ast3 Scope 80

Node W.A.2 is at level 80. The Peer Group Leader of Peer Group W.A.2 (this same physical node but operating at level 96) has received the addresses advertised by the two nodes. These are fed to the logical group node W.A.2 which summarizes them as follows:

W.A.2
W.F.1.5.xxx
W.F.2.7.yyy
W.F.2.7.zzz
A.N.Y.C.ast1 Scope 80
A.N.Y.C.ast3..Scope 80
A.N.Y.C.ast4 Scope 72

W.A.2 is the default summary address for this node. All address which match it are suppressed. The address A.N.Y.C.ast3 is advertised because one of the two instances of this had sufficient scope. Node logical group node W.A is at level 64. It again summarizes the addresses as follows:

W.A
W.F.1.5.xxx
W.F.2.7.yyy
W.F.2.7.zzz

Its default summary address is W.A. None of the group address are advertised since they all have a scope greater than level 64.

3.6 Single Node Perspective

This section derives the perspective that a lowest-level node has of the network. This view is informative since it corresponds to the network description contained in the node's topology database(s). Figure 3-10 shows this view for all the nodes contained in the lowest level peer group A.3 of the PNNI routing hierarchy depicted in Figure 3-6. The view is the same for all four nodes A.3.1 to A.3.4 of the group because flooding within peer group A.3 ensures that the topology databases of all its members are identical.

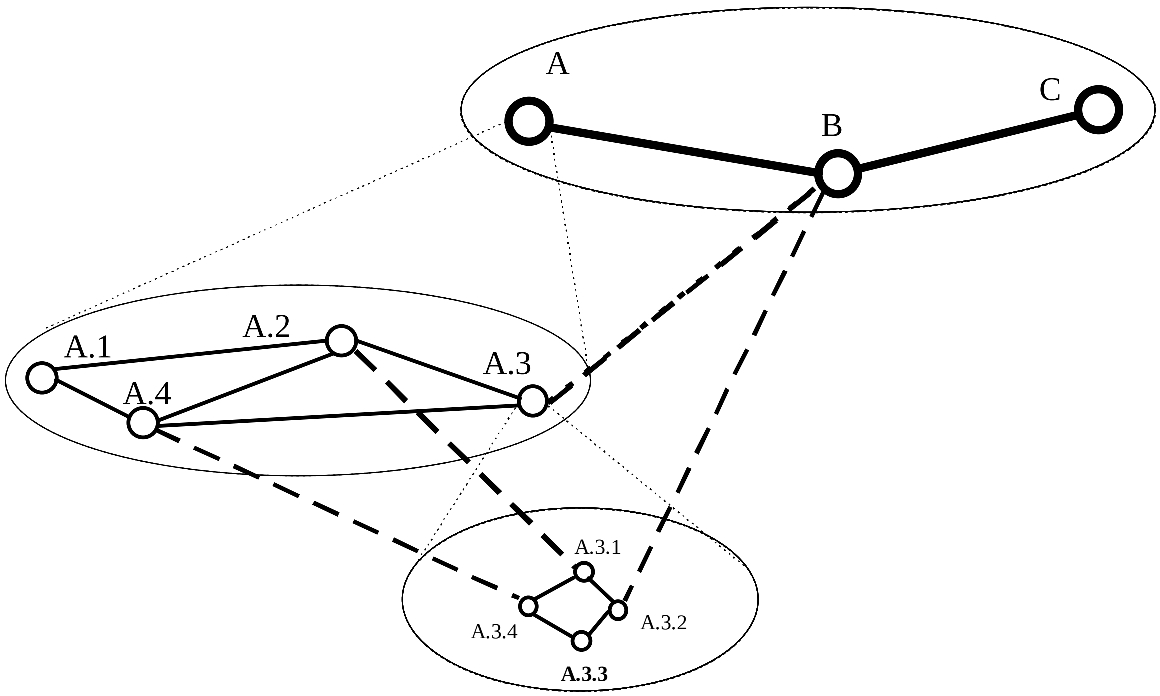


Figure 3-10: View of the network from nodes A.3.3, A.3.2, A.3.1, and A.3.4.

The view shown in Figure 3-10 includes all ancestor peer groups of peer group A.3, i.e. peer group A and the highest level peer group. Generalizing, the view that a peer group has of the rest of the network corresponds to that group's parent peer group and parent peer group of the parent and so on. This is true for any peer group at any hierarchical level.

This example focuses on node A.3.3 and traces how its 'view of the world' evolves. To simplify the scenario it is assumed that the neighbors of A.3.3 are active before A.3.3 comes up. When A.3.3 initializes, two new links appear, (A.3.3--A.3.4) and (A.3.3--A.3.2). The Hello protocols on these new links inform A.3.3 of its neighbors A.3.2 and A.3.4.

Node A.3.3 can now obtain the entire topology database(s) from its neighbors via the initial topology database exchange and via flooding.

The single node perspective of A.3.3 is formed from the PTSEs in the topology database as follows. Node A.3.3 learns the topology of peer group A.3 from information in the PTSEs originated by its peers in peer group A.3. Connectivity to the upnode A.4 in parent peer group A (Figure 3-6) is learned from the uplink (A.3.4--A.4) that was created and flooded by border node A.3.4. Similarly, connectivity to upnode A.2 is learned from the uplink (A.3.1--A.2), that was created and flooded by border node A.3.1. There is no uplink from any node in peer group A.3 to node A.1. Uplinks only provide information about connectivity between lower level peer groups and neighboring upnodes (e.g., A.4 and A.2). Node A.3.3's complete view of the nodes A.1, A.2, and A.4 and the connectivity among them is learned from the PTSEs originated by those nodes.

Connectivity to upnode B (Figure 3-6) is obtained by the uplink (A.3.2--B) created by node A.3.2. Node A.3.3's complete view of the nodes B and C and the connectivity between them is learned from the PTSEs originated by those nodes.

Topology aggregation is implicit in all nodes of Figure 3-10 that do not belong to peer group A.3, the group to which A.3.3 belongs. For example the complex node representation shown in Figure 3-5 is associated with LGN A.4.

Topology aggregation is also implicit in all links of Figure 3-10 that do not belong to peer group A.3. The example demonstrates this for link (A.2--A.4) which represents the aggregation of links (A.2.1--A.4.1) and (A.2.2--A.4.2), contained in Figure 3-6.

In its topology database A.3.3 will have advertisements describing the address prefixes reachable through each node. For example, the reachable address prefixes associated with the end systems attached to nodes A.2.1, A.2.2, A.2.3 (Figures 3-6 and 3-8) are summarized in advertisements from logical group node A.2.

3.7 Path Selection

ATM is a connection-oriented networking technology. That means that a path selected by PNNI for the establishment of a virtual connection or virtual path will remain in use for a potentially extended period of time. This means that the consequences of an inefficient routing decision will affect a connection for as long as that connection remains open. Thus it is critical that PNNI select paths carefully.

ATM allows a user to specify, when setting up a call, Quality of Service (QoS) and bandwidth parameter values that an ATM network must be able to guarantee for that call. Call establishment consists of two operations: (1) the selection of a path, and (2) the setup of the connection state at each point along that path. Path selection is done in such a way that the path chosen appears to be capable of supporting the QoS and bandwidth requested, based on currently available information. The processing of the call setup at each node along the path confirms that the resources requested are in fact available. (If they are not, then crankback occurs, which causes a new path to be computed if possible; thus the final outcome is either the establishment of a path satisfying the request, or refusal of the call.)

To some extent efficient QoS-sensitive path selection is still a research issue. Also, some known algorithms for QoS-sensitive path selection in the presence of multiple constraints require consideration of multiple independent link parameters and are relatively expensive computationally. It is therefore very important that PNNI allows flexibility in the choice of QoS-sensitive path selection algorithms.

There are two basic routing techniques that have been used in networking: source routing and hop-by-hop routing. In source routing, the originating (or source) system selects the path to the destination and other systems on the path obey the source's routing instructions. In hop-by-hop routing, each system independently selects the next hop for that path, which results in progress toward the destination provided that the decisions made at each hop are sufficiently consistent

In principle, either routing technique can be applied to connection-oriented networks, such as ATM. However, hop-by-hop routing does have some disadvantages in this type of network .

The first is the creation of routing loops. There are several possible causes of routing loops:

1. Inconsistency in routing decisions when switches use different routing algorithms
2. Inconsistency in routing databases among the switches (typically due to changes in topology information that have not fully propagated yet)

Inconsistency in routing decisions leads to a fundamental constraint of hop-by-hop routing, which is that the path selection must be fully specified, and all systems must implement it exactly as specified.

Further, due to inconsistency among the routing databases it is also possible for paths to be followed as a consequence of the individual hop by hop decisions that are far from optimal yet contain no loops. Any connection established with such an inefficient path will use that path for as long as that connection remains open.

Another disadvantage of hop-by-hop routing is that it replicates the cost of the path selection at each system. While this is less serious for connection-oriented networks (where the cost only occurs at connection setup) the QoS-sensitive path selection of PNNI may be far more costly.

In source routing, the source is responsible for selecting the path to the destination. It does this based on its local knowledge of the network topology. Since only one database is involved, loops are not possible, nor will the paths be inefficient due to database inconsistency. Furthermore, since only the source selects the path , the algorithm used need not be the same in every system, and the cost of executing it is only incurred once. In addition, it is much easier to do path selection based on specialized considerations, because there is no requirement for the algorithms to be consistent among all the nodes.

To avoid the difficulties stated above of using hop by hop routing with connection oriented service, and to gain the advantages of source routing, PNNI uses source routing for all connection setup requests. This implies that the first node in a peer group selects the entire path across that peer group. The path is encoded as a Designated Transit List (DTL) which is explicitly included in the connection setup request. The DTL specifies every node used in transit across the peer group, and may optionally also specify the logical links which are used between those nodes. If a node along the path is unable to follow the DTL for a specific connection setup request due to lack of resources, then the node must refuse that request and must crankback the request to the node that created the DTL.

Since PNNI allows for multi-level hierarchical routing, the originating switch selects a path to the destination containing all the detail of the hierarchy known to it. This is called a hierarchically complete source route. Such a path is not a fully detailed source route because it does not contain the details of the path outside the originator’s peer group. Instead, those portions of the path are abstracted as a sequence of logical group nodes to be transited. When the call setup arrives at the entry switching system of a peer group, that switching system is responsible for selecting a (lower level) source route describing the transit across that peer group. Naturally, the path used to cross a lower level peer group must be consistent with the higher level path (i.e., must reach the “next hop” destination specified by the higher level path).

The use of source routing implies that only one node is involved in the actual selection of the path followed across any one peer group by any one specific connection setup request. This implies that it is not necessary for each node in a peer group to use the same path selection algorithm. Rather, each node may use whatever path is felt most appropriate given the capabilities of that node and the constraints of the call. Different nodes may use different path selection algorithms for a variety of reasons, such as transition to a new software release, for networks in which nodes have significantly different processing capabilities, or in multi-vendor networks. Similarly, the path selection algorithm used for one level of the PNNI hierarchy may be different from the path selection algorithm used for a different level (subject to the constraint that path selection at any particular lower level must be consistent with the path selected by higher levels).

PNNI therefore does not specify any single required algorithm for path selection. Rather, each implementation is free to use whatever path selection algorithm is felt appropriate. One acceptable path selection algorithm is provided in Appendix H.

3.7.1 Generic Connection Admission Control

The idea of a generic connection admission control (CAC) came from the realization that, since CAC is not to be standardized, it is not feasible to perform the actual CAC to determine if a switching system will admit a new connection.

For each connection request, a path is computed based on information stored in the node's topology database, together with the connection's service category, traffic characteristics, and QoS requirements. The path is chosen to satisfy the connection's end-to-end traffic and QoS requirements as well as to meet network efficiency criteria. During connection setup, each switching system along the chosen path performs CAC to ensure that the connection can be accommodated without jeopardizing QoS guarantees to the existing connections. How a switching system does CAC is not subject for standardization. If a switching system accepts the connection, its ability to accept future connections may change. This will trigger origination of new PTSE instances describing this node's updated resource availability if the changes are significant (see Section 5.8.5.2). The determination of a 'significant' change is algorithmically defined as controlled by configuration parameters. The values of the configuration parameters are chosen to balance the frequency of PTSE updates against the possibility of a failed call attempt due to old information.

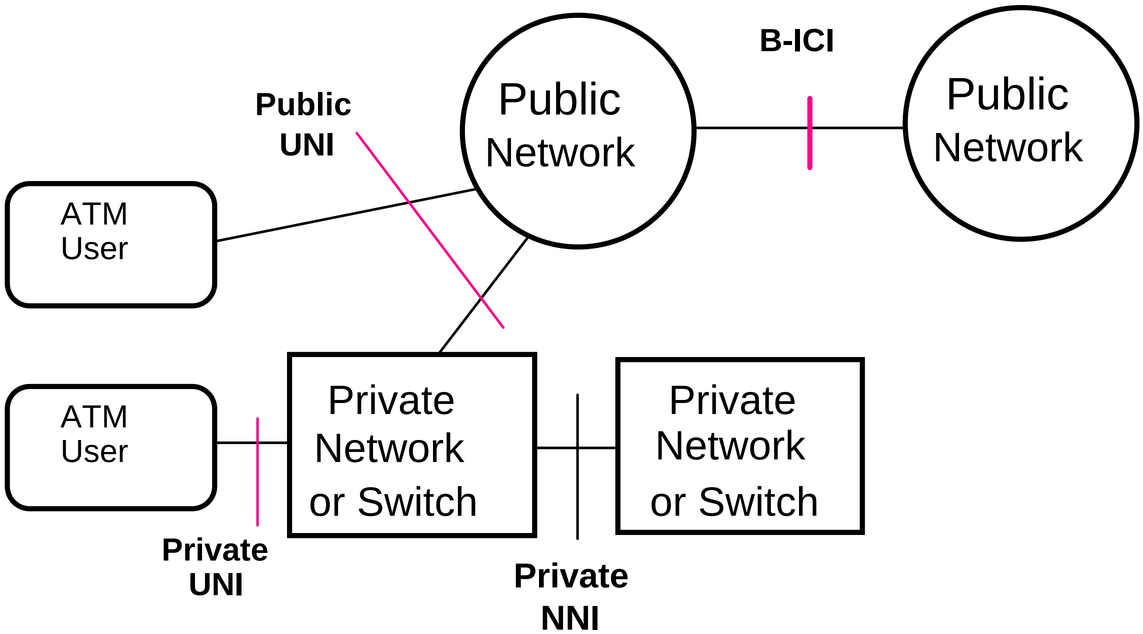
A generic CAC is needed in the path selection process to determine if a link or node is likely to have enough resources available to support the proposed connection. A generic CAC should include a link or node if the switching system is likely to accept the proposed connection. Conversely, the generic CAC should exclude a link or node if the switching system is likely to reject the new connection. In essence, a generic CAC predicts the outcome of the actual CAC performed at a switching system.

Using a generic CAC, each switching system is free to use any CAC but is required to advertise a set of topology state parameters carrying some information that can be used by the generic CAC to make its prediction. A generic CAC and the set of topology state parameters supporting it have to be flexible enough to allow any CAC to be approximated reasonably well.

4. PNNI Signalling Description

4.1 Introduction

ATM signalling is a set of protocols used for call/connection establishment and clearing over ATM interfaces. The interfaces of interest to the ATM Forum are illustrated in Figure 4-1.



Where:

Public UNI: the user-network interface between an ATM user and a public ATM network
Private UNI: the user-network interface between an ATM user and a private ATM network
B-ICI: the network-network interface between two public networks or switching systems. B-ICI stands for B-ISDN Inter Carrier Interface.
Private-NNI: the network-network interface between two private networks or switching systems.

Figure 4-1: ATM Interfaces

This chapter provides a description of the signalling procedures over the PNNI interface. These procedures enable calls to be setup across a private ATM network that is executing the PNNI protocol.

4.2 Overview

PNNI signalling is based on a subset of UNI 4.0 signalling. It does not support some UNI 4.0 signalling features such as proxy signalling, leaf initiated join capability or user to user supplementary service, but adds new features which pertain to the use of PNNI routing for dynamic call setup. In addition, PNNI signalling differs from UNI 4.0 signalling in that it is symmetric.

PNNI signalling makes use of the information gathered by PNNI routing. Specifically, it uses the route calculations derived from the reachability, connectivity, and resource information dynamically maintained by PNNI routing. These routes are calculated as needed from the node's view of the current topology.

In order to meet the requirements, PNNI signalling uses four additional features beyond those defined for UNI signalling. Designated Transit Lists (DTLs) are used to carry hierarchically-complete source routes. Crankback and alternate routing allows for attempting alternate paths to cope with inaccurate information. Associated signalling is used for PNNI operation over virtual path connections. Soft permanent VPCs/VCCs (soft PVPC/PVCCs) are supported.

4.3 Associated Signalling

By default (if no VPCs are configured for use as logical links on the interface), the signalling channel on VPI=0 controls all of the virtual paths on the physical interface. PNNI also supports signalling and routing over multiple virtual path connections (VPCs) to multiple destinations through a single physical interface. In this situation, each VPC configured for use as a logical link has a signalling channel associated with it. Virtual channels within these VPCs are controlled only by the associated signalling channel of that particular VPC, i.e., the default signalling channel (on VPI=0) on the same physical interface does not control virtual channels within VPCs used as logical links, but does control all the remaining virtual channels and virtual paths on the physical link. This use of the VPI=0 signalling channel is a special case of the Non-Associated Signalling capability defined in Q.2931.

4.4 Designated Transit Lists

In processing a call, PNNI signalling may request a route from PNNI routing. PNNI specifies routes using DTLs. A DTL is a complete path across a peer group, consisting of a sequence of Node IDs and optionally Port IDs traversing the peer group. It is provided by the source node (i.e., DTL originator) or an entry border node to a peer group. A hierarchically complete source route represents a route across a PNNI routing domain that includes each hierarchical routing level between the current level and the lowest visible level in which the source and destination are reachable. This is expressed as a sequence of DTLs ordered from lowest to highest peer group level and organized as a stack (i.e., a last in, first out data structure). The DTL at the top of the stack is the DTL corresponding to the lowest level peer group.

4.5 Crankback

When creating a DTL, a node uses the currently available information about resources and connectivity. That information may be inaccurate for a number of reasons. These reasons include hierarchical aggregation and changes in resource availability due to additional calls which have been placed since the information was produced. Therefore a call being processed according to the DTL may be blocked along its specified route. Crankback and alternate routing is a mechanism for adapting to this situation short of clearing the call back to the source. When the call cannot be processed according to the DTL, it is cranked back to the creator of that DTL with an indication of the problem. This node may choose an alternate path over which to progress the call or may further crankback the call. An alternate path must obey all received higher-level DTLs, and must avoid the blocked node(s) or link(s).

4.6 Soft PVPCs and PVCCs

A PVPC or PVCC is a permanent virtual path connection or virtual channel connection. The qualifier “permanent” signifies that it is established administratively (i.e., by network management) rather than on demand (i.e., by the use of signalling across the UNI). A soft PVPC or PVCC is one where the establishment within the network is done by signalling. By configuration the switching system at one end of the soft PVPC or PVCC initiates the signalling for this. These procedures are specified in detail in Annex C.

The network management system provisions one end of the Soft PVPC or PVCC with the address identifying the egress interface from the network. The calling end has the responsibility for establishing, releasing, and (optionally) re-establishing the call.

4.7 An Example of Connection Setup with Crankback

In this section, a detailed example of connection setup including crankback and alternate routing is presented. The network used in this example is shown in Figure 4-2. It is assumed that the link between A.3.3 and B.1.2 is blocked (though the PNNI routing information says that it is OK), and that the link between A.3.1 and B.2.3 cannot support the high peak cell rate requested by the call.

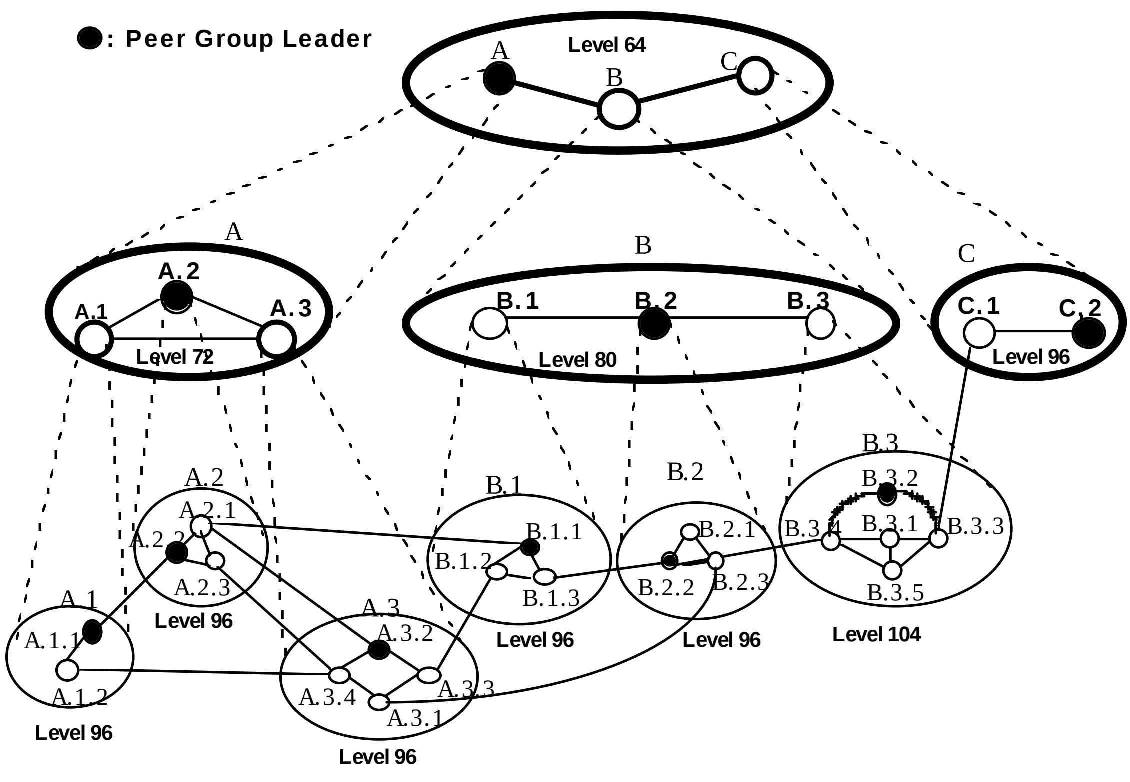


Figure 4-2: A Sample Hierarchical Network

Consider the detailed picture of peer groups A.[1--3] and B.[1--3] shown in Figure 4-2. Assume a connection is to be setup from an end-system A.1.2.x attached to switching system A.1.2 to an end system B.3.3.y attached to switching system B.3.3. A.1.2.x sends the setup request to A.1.2 as a normal UNI SETUP. A.1.2 examines its view of the world. The destination address B.3.3.y is reported as reachable through node B. Note that B as it appears in the name of the peer group may be a different bit string than the B that is part of the ATM-endpoint address of the destination. That is why the advertisement from B explicitly includes the prefixes of the reachable addresses. As mentioned above, it is likely that the PTSE from node B is actually announcing that it can reach an address prefix describing multiple end systems, which include end system B.3.3.y.

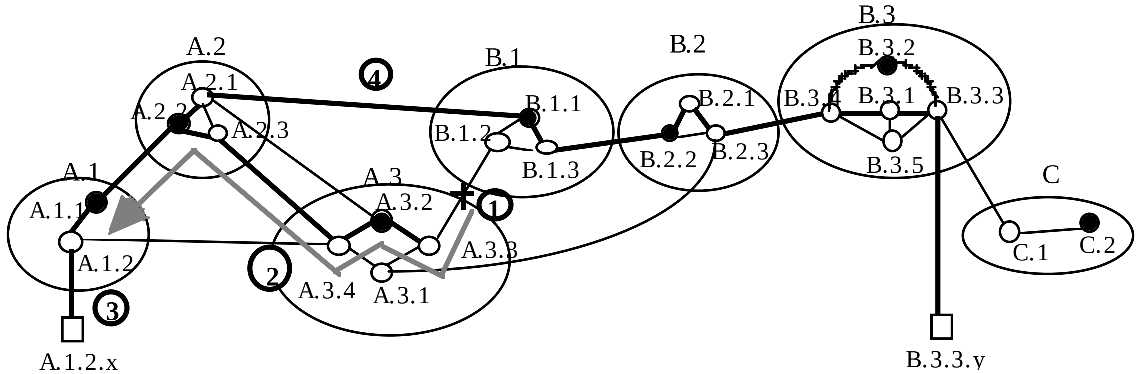


Figure 4-3: An Example of Crankback and Alternate Routing

Examining the topology, A.1.2 finds that the path is either (A.1.2, A.1.1, A.2, A.3, B), (A.1.2, A.1.1, A.2, B), or (A.1.2, A.3, B). Let us assume that on the basis of metrics and policy, A.1.2 chooses (A.1.2, A.1.1, A.2, A.3, B). We will then build three DTLs, in a stack:

DTL:[A.1.2, A.1.1], pointer-2
DTL:[A.1, A.2, A.3] pointer-1
DTL:[A, B] pointer-1

Each DTL lists nodes that the call setup needs to visit at a given hierarchical level (with the current node included). The current transit pointer following each DTL specifies which element in the list is currently being visited at that level, except that the top DTL is advanced by one to indicate which node will be visited next. Note that the transit pointer is not actually encoded as an index, but rather as an octet offset. For clarity, it is shown simply as an index in this section. The DTL lists are ordered as a stack of DTLs (i.e., the top DTL is acted on; when the end of the top DTL is reached, it is removed from the call request and the next DTL is examined).

Before forwarding the call setup, A.1.2 stores the content of the SETUP message, in case alternate routing will be required. A.1.2 then forwards the call setup to his neighbor A.1.1. A.1.1 examines the top DTL and notices the DTL pointing to its own Node ID. A.1.1 looks for the next entry in the top DTL, but finds it exhausted. Looking at the next DTL, the next destination is A.2. Since A.1.1 is not A.2 (i.e., A.1.1 is not in the peer group summarized into logical node A.2), it starts looking to see how to get to A.2. A.1.1 finds that an immediate neighbor is in A.2, so it removes the top DTL and advances the current transit pointer in the next DTL:

DTL:[A.1, A.2, A.3] pointer-2
DTL:[A, B] pointer-1

Note that A.1.1 did not add any DTLs and did not make any routing decisions, so A.1.1 will not be involved in alternate routing and does not need to make any copies of the SETUP message before forwarding the call setup to A.2.

A.2.2 looks at the top DTL, and sees that the current destination is A.2. Since A.2.2 is in A.2, it looks at the next entry in the DTL, and starts routing to A.3. Analyzing the topology, A.2.2 finds that the path is through A.2.3, so it pushes that DTL onto the list:

DTL:[A.2.2, A.2.3] pointer-2
DTL:[A.1, A.2, A.3] pointer-2
DTL:[A, B] pointer-1

Since A.2.2 added a DTL to the stack, and may be involved in alternate routing, A.2.2 stores the contents of the received SETUP message and the contents of the DTL that it added on top of the stack, before sending the packet to A.2.3.

A.2.3 first determines that the top DTL target has been reached, and then that the DTL is exhausted. Finally, A.2.3 notices that the next destination on the next DTL is a neighbor, so A.2.3 removes the top DTL and advances the current transit pointer in the next DTL:

DTL:[A.1, A.2, A.3] pointer-3
DTL:[A, B] pointer-1

The setup arrives at A.3.4. Since A.3.4 is in A.3 the target has been reached. That leaves the target as B. A.3.4 builds a route to B, through A.3.2 and A.3.3, and pushes a new DTL on the list:

DTL:[A.3.4, A.3.2, A.3.3] pointer-2
DTL:[A.1, A.2, A.3] pointer-3
DTL:[A, B] pointer-1

Since A.3.4 added a DTL to the stack, and may be involved in alternate routing, A.3.4 stores the contents of the SETUP message before progressing the call to A.3.2.

A.3.2 receives the call setup, advances the current transit pointer, and forwards the setup request to A.3.3. A.3.2 will not be involved in alternate routing, and so does not need to keep any copies of the SETUP message. A.3.3 receives the call setup, and decides to forward the SETUP message to his neighbor in B. Before forwarding the SETUP message, A.3.3 removes the top two DTLs, as both have been exhausted and the SETUP message is departing from A.3.3 and A.3:

DTL:[A, B] pointer-2

At this point, the call setup is blocked, as A.3.3 (the preceding node) finds out either from the CAC internal to A.3.3, or from a RELEASE or a RELEASE COMPLETE message containing a Crankback IE with blocked transit type #2, “call or party has been blocked at the succeeding end of this interface”, as indicated by the circle labeled 1. The crankback procedure now begins, with A.3.3 sending back out the preceding interface a RELEASE message. The Crankback IE in the RELEASE message indicates that the call was blocked at the link between A.3.3 and B, as determined from the DTLs from the received SETUP message, and that the Crankback Level is 96 (the level of peer group A.3).

The RELEASE message is received at A.3.2, which did not create any DTLs for this call, so crankback proceeds further. After or while tearing down the succeeding virtual channel, the RELEASE message is sent out the preceding interface in the direction of A.3.4.

The RELEASE message is received at A.3.4, which specified a DTL for this call at level 96, so A.3.4 attempts to do alternate routing. After eliminating the blocked link from consideration, two different paths are found across A.3 to B. One option is through A.3.1 and on to B, the other is through A.3.2, A.3.3, A.3.1, and on to B. However, the link between A.3.1 and B cannot support the high peak cell rate requested in the original SETUP message, so A.3.4 decides to continue the crankback procedure rather than to try one of the other options, as better paths may exist further back along the original path. A.3.4 sends another RELEASE message back through the preceding interface, but changes the blocked transit identifier to show that the link from A.3 to B is blocked, and also changes the Crankback Level to 72 (the level of A.1's peer group) as indicated by the circle labeled 2.

The RELEASE message is received at A.2.3, which did not create any DTLs for this call, so crankback proceeds further (in the direction of A.2.2).

The RELEASE message is received at A.2.2, which did create a DTL for this call. A.2.2 examines the crankback level in the crankback IE, and determines that the level is higher than the level of the DTL that it created, so crankback proceeds further (in the direction of A.1).

The RELEASE message is received at A.1.1, which did not create any DTLs for this call, so crankback proceeds further (in the direction of A.1.2).

The RELEASE message is received at A.1.2. A.1.2 sees that the Crankback Level matches the level of a DTL it specified for this call, and finds a copy of the corresponding SETUP message contents. Alternate routing is attempted, as indicated by the circle numbered 3, and A.1.2 decides to try the path (A.1.2, A.1.1, A.2, B), resulting in the following stack of DTLs:

DTL:[A.1.2, A.1.1], pointer-2
DTL:[A.1, A.2], pointer-1
DTL:[A, B], pointer-1

A.1.2 then forwards the call setup to A.1.1. A.1.1 removes the top DTL from the stack, advances the current transit pointer in the next DTL, and forwards the call to A.2.2. A.2.2 finds a path through A.2.1 to B, pushes a new DTL onto the stack, and forwards the call setup to A.2.1:

DTL:[A.2.2, A.2.1], pointer-2
DTL:[A.1, A.2], pointer-2
DTL:[A, B], pointer-1

A.2.1 receives the setup and notes that B is a neighbor. He removes the top two DTLs from the stack, advances the current transit pointer in the next DTL, and forwards the call setup to B.1.1 as indicated by the circle numbered 4 in Figure 4-3:

DTL:[A, B], pointer-2

B.1.1 receives the setup, and sees that the current DTL destination has been reached. B.1.1 must now build a new source route to descend to the final destination. He notices that the path will go through B.2 to B.3, so he pushes that on. He then calculates that the path through B.1 is simply to B.1.3, so he pushes that on, giving:

DTL:[B.1.1, B.1.3] pointer-2
DTL:[B.1, B.2, B.3] pointer-1
DTL:[A, B], pointer-2

B.1.1 then forwards the setup to B.1.3. In the usual fashion, after checking, B.1.3 removes the top DTL, advances the DTL pointer, and hands it to his neighbor in B.2. B.2.2 notices that he is the current target, and then finds a path through B.2 to B.3, and pushes that on the list, giving:

DTL:[B.2.2, B.2.1, B.2.3] pointer-2

DTL:[B.1, B.2, B.3] pointer-2
DTL:[A, B], pointer-2

B.2.2 then forwards the setup to B.2.1.

B.2.1 looks at the top DTL, advances it, and forwards the setup to B.2.3. B.2.3 removes the DTL, advances the pointer, and forwards the setup to his neighbor in B.3. B.3.4 builds a new DTL, giving:

DTL:[B.3.4, B.3.1, B.3.3] pointer-2
DTL:[B.1, B.2, B.3] pointer-3
DTL:[A, B], pointer-2

B.3.1 looks at the top DTL, advances the current transit pointer, and forwards the setup to B.3.3. B.3.3 determines that it is the DTL terminator for the DTL stack since:

- all DTLs are at the end and
- B.3.3 is a lowest-level node, and
- B.3.3 has reachability to the destination.

B.3.3 then strips the DTL stack from the SETUP message, and forwards the message across the UNI to the destination.

4.8 Supporting Anycast with Scoping

Reachability information advertised by a logical node, including that to group addresses, has scope associated with it (see Section 3.5.1). Scope serves two purposes. First, it defines the highest level at which a given reachable address (or prefix) can be advertised. Second, it defines the “service” scope for a given group address used for anycast purposes, i.e., the highest level peer group containing the group address outside of which a calling party will not be allowed to establish connections to.

For each anycast connection request, the user indicates at the UNI, using the connection scope selection IE, the routing scope of the connection, i.e., how far out it is allowing the network to search for the anycast address. Using a local mapping, the first node translates the value of the connection scope selection IE into a PNNI routing level and uses it to compute a route to the anycast address. The PNNI routing level is not carried in PNNI signalling (but the connection scope as specified by the calling user is carried, so it will be available if the call progresses to other routing domains). However, as the example below shows, the DTL IEs carry enough information for entry border nodes to determine the appropriate destination in the case where there are multiple anycast group addresses advertised in its peer group with different scopes.

Consider the network configuration shown in Figure 4-2 and assume the following:

- B.2.1 advertises reachability to group address G1 with scope of 80,
- B.2.3 also advertises reachability to G1, but with scope of 64.

Using the address scoping rules discussed in Section 3.5.1, B.2 and B will also advertise reachability to G1, and the scope of advertisement will be 64.

Suppose a user at A.1.1 requests an anycast connection to G1 with connection scope selection that gets translated locally into a PNNI routing level of 64. Suppose also that from A.1.1’s point of view, no nodes in peer group A advertise reachability to G1 (there may be nodes in peer groups A.2 or A.3 advertising reachability to G1, but with scope no larger than their respective peer group). Consequently, A.1.1 computes a route to B and generates the following DTLs:

DTL:[A.1.1, A.1.2] pointer-2

DTL:[A.1, A.3] pointer-1
DTL:[A, B] pointer-1

The DTLs will route the connection to B.1.2 where the called party address (i.e., G1) is used for the first time to compute lower-level routes. In this case, B.1.2 will see the reachability advertisement from B.2 to G1 and computes the following DTLs:

DTL:[B.1.2, B.1.3] pointer-2
DTL:[B.1, B.2] pointer-1
DTL:[A, B] pointer-2

When the connection setup reaches B.2.2, the called party address is again used to compute lower-level routes. In this case, there are multiple advertisements for G1 seen by B.2.2, namely from B.2.1 and B.2.3. B.2.2 infers from the highest level DTL (i.e., [A, B]) that the calling party is within the peer group containing A and B, but outside peer group B. Therefore, B.2.2 chooses to go to B.2.3, instead of B.2.1.

5. PNNI Routing Specification

This chapter provides the detailed specification of the PNNI routing protocol. The chapter begins with operational procedures that apply to the entirety of the protocol. Sections 5.2 through 5.5 provide specifications of the elements of the system. Sections 5.5 through 5.13 are the procedures to implement PNNI. Section 5.14 contains all of the packet formats.

5.1 General Operational Procedures

5.1.1 Timers

Timers that trigger transmission of messages must be jittered. This is done to prevent unintentional synchronization of transmissions, which could result in network instability.

Jittering is accomplished by applying a new random fractional variance to the time out value each time a timer is reset. The range of the fractional variance is plus or minus 25% of the nominal value.

5.1.2 Packet Transmission

Except for Hello packets (see Section 5.6.2.2) all packets are encoded according to the protocol version given by the Version field of the hello data structure.

5.1.3 Packet Validation

For any received packet, if the packet length in the PNNI packet header exceeds the received data length, the packet shall be discarded without further processing.

If the packet type in the PNNI packet header is not recognized the entire packet shall be discarded.

Any other parsing error in type or length field (including the case where the length is not a multiple of four) can be handled by ignoring the offending element, the enclosing element or by ignoring the entire packet at the discretion of the implementation.

If a packet is received with an unsupported version (see Section 5.6.1), the packet must be discarded, optionally with a management notification. For packets other than Hello packets, any packet received with a supported Version different from the expected value is to be discarded.

5.2 Addressing

5.2.1 ATM Addressing Convention

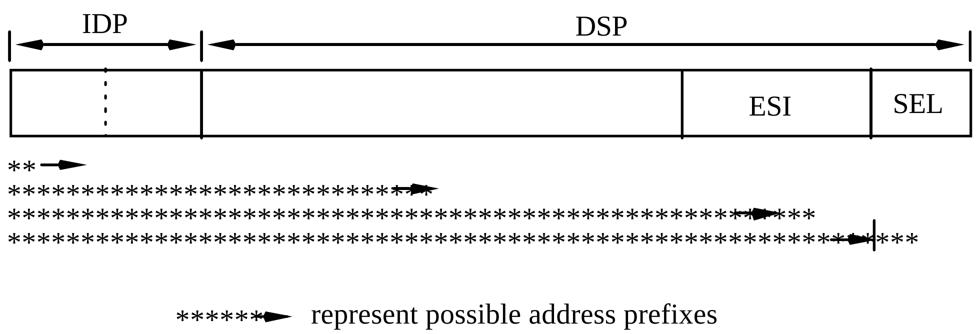


Figure 5-1: NSAP Address Structure and Address Prefixes

Addressing and identification of components of the PNNI routing hierarchy are based on use of ATM End System Addresses and/or prefixes applied to ATM End System Addresses. ATM End System Addresses are in turn modeled after NSAP addresses as illustrated in Figure 5-1.

ATM End System Addresses are 20 octets long. PNNI routing only operates on the first 19 octets of the ATM address. The selector (20th) octet has only local significance to the end system and is therefore ignored by PNNI routing. NSAPs are usually assigned by an authority which imposes substructure on the address. PNNI never uses, examines, or interprets the additional substructure if it is present, with one significant exception. The first octet, the Authority and Format Identifier (AFI), is used by PNNI to distinguish between individual addresses, which identify single ATM end systems, or group addresses, which identify one or more ATM end systems. The valid individual and group values of the AFI are given in Table A5-1 of UNI signalling 4.0. The substructure of the NSAP address (IDP, DSP, ESI, and SEL) is included here for completeness, but is not important to routing operations. For further information on the substructure of ATM End System Addresses, see Section 3.1 of UNI Signalling 4.0. Guidelines for assigning ATM Group addresses are provided in Annex 5 of UNI Signalling 4.0.

A prefix of a ATM End System Address is the first “p” bits of that address. The value of p may vary from zero to 152 bits. A prefix with zero length summarizes “all” ATM End System Addresses in one advertisement; PNNI routing will direct calls for which there is no more specific match to systems which advertise such a zero length prefix (i.e. the ‘default’ route). Prefixes with lengths greater than zero and less than 152 are used to summarize some portion of the addressing domain where shorter prefixes summarize greater portions than do longer prefixes (i.e. longer prefixes are more specific).

A set of possible address prefixes are represented in Figure 5-1 by the set of ‘starred’ (****>) arrows. Note that the arrows, like the prefixes, are always applied to the left most bits of an address. The prefix length determines how far these arrows stretch to the right.

PNNI operates in a topologically hierarchical environment. The structure of the hierarchy is defined by the peer group IDs (see Section 5.3.2) used in the routing domain. Address assignment has a hierarchy that should generally correspond to the topological hierarchy, for proper scaling. This allows address summarization where an address prefix represents reachability to all addresses that begin with the stated prefix. Addresses that are exception cases in this general hierarchy are then described by longer prefixes.

5.2.2 Node Addresses

Nodes actively taking part in PNNI routing are addressed using ATM End System Addresses.

A single switching system may contain multiple nodes. Each of these nodes requires a unique ATM End System Address. This address is used for establishment of SVCCs used in PNNI. One way for a switching system to generate unique addresses for the nodes it instantiates is by using different values of the selector.

5.2.3 End System Addresses

ATM End System Addresses are 20 octets. PNNI routing, as stated above, operates on the first 19 octets. The selector octet is only used to distinguish destinations reachable at the same interface.

Generally PNNI will advertise reachability to end systems using prefixes on ATM End System Addresses to form summaries. If an end system attached to a node does not fit into one of the node's configured summaries it will be necessary for the node to make an explicit advertisement for that end system which will necessarily carry a full 152-bit (all 19-octets) prefix length. True summaries of end system reachability will have prefix lengths less than 152 bits. Where the addressing hierarchy follows the topological hierarchy, it is possible to advertise reachability to a large number of end systems using a single prefix. Shorter prefixes (i.e. lower prefix length values) summarize greater numbers of addresses and vice versa.

There may be a need to calculate routes to end systems which are located in networks supporting only E.164 addresses. For this reason, PNNI allows end system reachability to be advertised via prefixes of E.164 addresses using the embedded E.164 NSAP format. Although PNNI has been designed for use in a private ATM network, it is likely that PNNI might also be used internally in public ATM networks, or offered as an external interface to other public and/or private ATM networks.

Since summaries can enclose other summaries, it is possible that there are many summaries which match, to varying degrees, a given destination address. PNNI route computation will always direct calls to a logical node that is advertising the best match for the given destination that has a wide enough scope for the call. The best match is defined to be the matching (summary) advertisement with the longest prefix.

5.3 Identifiers and Indicators

5.3.1 Level Indicators

PNNI entities (nodes, links, and peer groups) occur at various hierarchical levels. The level specifies a bit string length, and ranges from 0 to 104.

Given two entities, where one is an ancestor of the other, the ancestor is a higher-level entity and will have a smaller level indicator than the other. It should be noted that it is not meaningful to directly compare levels for entities where neither is an ancestor of the other.

The level indicator is absolute in the sense that it specifies the exact number of significant bits used for the peer group ID.

PNNI levels are not "dense", in the sense that not all levels will be used in any specific topology. A peer group with an ID of length "n" bits may have a parent peer group whose identifier ranges anywhere from

0 to n-1 bits in length. Similarly, a peer group with an ID of length "m" bits may have a child peer group whose identifier ranges anywhere from m+1 to 104 bits in length.

5.3.2 Peer Group Identifiers

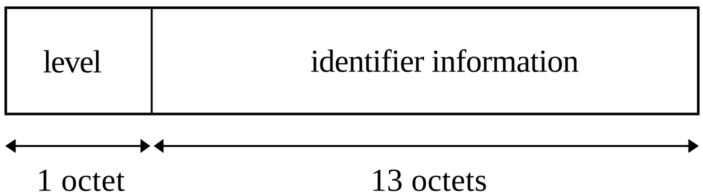


Figure 5-2: Peer Group Identifier 14 octet encoding

A peer group identifier is a string of bits between zero and 104 bits (13 octets) in length. Peer group identifiers must be prefixes of ATM End System Addresses such that the organization that administers the peer group has assignment authority over that prefix. For example, if an organization is given an n-bit prefix it may assign peer group identifiers with length n or greater, but not less than n.

Peer group identifiers are encoded using 14 octets; a 1 octet (8 bit) level indicator followed by 13 octets (104 bits) of identifier information. The value of the level indicator must be between zero and 104 (bits). The value sent in the identifier information field must be encoded with the 104 - n right-most bits set to zero, where n is the level.

5.3.3 Node Identifiers

The Node Identifier is twenty-two octets in length, and consists of a one octet level indicator followed by a twenty one octet opaque value, i.e., these twenty one octets have no implied internal structure. This opaque value must be unique within the entire routing domain. The level of a node is the same as the level of its containing peer group. A node receiving a node identifier in a PNNI packet must make no other assumptions about the internal structure. Two methods for creating node IDs are currently defined, however other methods of generating node IDs are not precluded.

For nodes which do not represent a peer group:

- the level indicator specifies the level of the node’s containing peer group;
- the second octet takes the value 160; This helps distinguish this case from the case below since an encoded peer group ID can not begin with this value.
- the remainder of the node ID contains the twenty-octet ATM End System Address of the system represented by the node.

For a logical group node which represents a child peer group A.2 in its parent peer group A:

- the level indicator specifies the level of the peer group containing the LGN (i.e., the level of peer group A);
- the next 14 octets are the encoded peer group ID of child peer group A.2.
- the next 6 octets contain the End System Identifier (ESI) of the physical system implementing the logical group node functionality;

Note: The ESI contained in this field allows multiple nodes at the same level to be distinguished in case the lower level peer group is partitioned.

- the last octet of the node ID is zero.

During the operation of the PNNI routing protocol, nodes operating at a particular level will receive PTSPs which include PTSEs from other nodes within their peer group at the same level, and from higher level peer groups. The source node of each PTSP is specified by including the source node ID in each PTSP. Systems implementing PNNI routing are required to accept such incoming packets regardless of the format of the opaque value part of the node ID.

Two PNNI routing packets are from the same node if and only if they contain identical 22-octet source node Ids. For purposes of ordering node IDs the first octet of the node ID is the most significant octet.

The node ID of a node is not allowed to change while the node is operational, i.e. while the node has any adjacencies, hello FSMs in any state other than Down, or any topology database entries.

5.3.4 Port Identifiers and Logical Links

A port ID is a 32 bit number assigned by a node to unambiguously identify a point of attachment of a logical link to that node. Port IDs are only meaningful in the context of the assigning node, identified by its node ID.

The values zero and 0xFFFFFFFF are reserved and may not be used to identify specific ports.

A logical link is identified by the node ID of either node at the end of that link and the port ID assigned by that node.

ATM logical links are duplex having potentially different characteristics in each direction. For links internal to a peer group each end node advertises the port ID and outbound link characteristics. The nodes advertising each end of the link exchange their port IDs with each other. This is accomplished during the exchange of PNNI Hello packets.

For uplinks the advertising border node assigns the local port ID and advertises the link characteristics in both directions along with the upnode’s node ID (see Section 5.10.2).

Each advertised link from a node must have a unique port ID within the context of that node.

5.3.5 Aggregation Tokens

An Aggregation Token is a 4 octet identifier. It serves, along with the remote node ID to identify uplinks which are to be aggregated at the next level of the hierarchy. These also serve to bind links of the higher level to those of the next lower level.

The scope of significance of an Aggregation Token is limited to pair-wise logical group nodes. All links between a pair of logical group nodes with the same value of the Aggregation Token must be advertised as one logical link. The same token value may be used between other pairs of nodes without confusion of semantics. However, for ease and flexibility of administration, the space has been chosen to be large enough that globally unique token values can be assigned.

The nodes at each end of an outside link agree on the Aggregation Token advertised in the associated uplinks using the algorithm in Section 5.10.3.1.

The Aggregation Token is included in the PTSEs which describe uplinks as aggregation and binding information. The Aggregation Token is included in the PTSEs which describe Horizontal Links as binding information.

5.3.6 Scope of Addresses

Addresses and address prefixes used in PNNI can have scopes associated with them. Both individual and group addresses have membership scopes, which are used to determine the scope of advertisement of reachable addresses (see Section 5.9.1). For group addresses, scope can also be used for connection scope control, which allows a calling user to constrain a point-to-point connection request using the ATM Anycast capability to group members within a specified level of routing hierarchy (see Sections 5.13, 6.4.5.23, 6.5.2.1, and Annex B).

For PNNI, the scope of reachable addresses is specified by a level indicator. At the UNI, including when using ILMI address registration (in the ILMI object atmfAddressOrgMemberScope), the scope may be indicated by one of fifteen levels of organizational scope, as defined in Section A5.2 of UNI Signalling 4.0. A network specific mapping is used to translate between the organizational scope indicated across the UNI and PNNI routing level indicators. The mapping applies to both membership scope, which is passed across the UNI via ILMI address registration and is then advertised with the corresponding reachable address in PTSEs as a PNNI routing level, and to connection scope, which is passed across the UNI in a SETUP message. The mapping shall be a configurable object of the network. The default values for this mapping are as follows:

Table 5-1: Default UNI Scope to PNNI Level Mappings

UNI scope	PNNI Routing Level Indicator
1-3	96
4-5	80
6-7	72
8-10	64
11-12	48
13-14	32
15 (global)	0

5.4 Logical Links

Logical links are an abstract representation of the connectivity between two logical nodes. This includes individual physical links, individual virtual path connections, uplinks, and aggregations of logical links. There may be parallel logical links between two logical nodes. If two lowest-level nodes are connected by a virtual path connection (VPC) which is to be used for PNNI, each of them must know (by configuration/management) that the VPC exists and is to be used for PNNI.

5.5 PNNI Routing Control Channels

PNNI Routing Control Channels (RCCs) are VCCs used for the exchange of PNNI routing protocol packets (Hellos, PTSP, PTSE acks, etc.) between logical nodes that are logically or physically adjacent. There are three cases:

1. For the exchange of the PNNI routing protocol over physical links, a reserved VCC with VPI=0 and VCI=18 will be used.

2. For the exchange of the PNNI routing protocol over a virtual path connection with VPI=V, the PNNI routing protocol exchange will take place over the PNNI VCC within the VPC, that is VPI=V and VCI=18.
3. For the exchange of PNNI routing protocol messages between Logical Group Nodes, an SVCC is established that is dedicated for that purpose and the VPI and VCI are assigned by signalling in the normal way for SVCCs.

5.5.1 Specification of the AAL used by the PNNI Routing Control Channel

This section specifies the ATM adaptation layer (AAL) used by the PNNI routing control channel (RCC). PNNI protocol packets are encapsulated in AAL-Service Data Units (SDU).

The AAL used by the RCC is AAL Type 5 (AAL5). AAL Type 5 is composed of two sublayers, a common part and a service specific part. The Common Part AAL protocol provides unassured information transfer and a mechanism for detecting corruption of AAL SDUs used by the PNNI routing protocol (recovery of corrupted or lost AAL-SDUs is handled by the PNNI routing protocol above the AAL). AAL Type 5 Common Part shall be used to support the PNNI RCC. The AAL Type 5 Common Part Protocol is specified in ITU-T Recommendation I.363. The RCC uses the null Service Specific Convergence Sublayer (SSCS). The null SSCS is described in the AAL Type 5 section of ITU-T Recommendation I.363.

The PNNI RCC is used in Message Mode as described in ITU-T Recommendation I.363. This means that only a complete PNNI packet may be encapsulated in an AAL-SDU. In addition, each AAL-SDU shall contain only one complete PNNI packet.

5.5.2 Traffic Contract for RCCs over Physical Links

The RCC between two lowest level nodes connected via a physical link shall use the following default traffic descriptor:

- Service category is nrt-VBR
- PCR(CLP=0+1) = RCCPeakCellRate
- SCR(CLP=0) = RCCSustainableCellRate
- MBS(CLP=0) = RCCMaximumBurstSize
- Tagging applied.
- Frame discard allowed.

5.5.3 Traffic Contract for RCCs over VPCs

The RCC between two lowest level nodes connected via a VPC shall have by default the same service category as the VPC, and use the following default traffic descriptor:

CBR:
- PCR = RCCPeakCellRate

VBR (rt and nrt):
- PCR = RCCPeakCellRate
- SCR = RCCSustainableCellRate (Note 1)
- MBS = RCCMaximumBurstSize (Note 1)
- Tagging (Note 2)

- Frame discard allowed.

ABR
- PCR = PCR for VPC
- MCR = 0.005 * MCR for VPC

UBR
- PCR = PCR for VPC

Note 1: The SCR and MBS values apply to the same CLP substream as the SCR parameter of the VPC.
Note 2: When the SCR parameter applies to the CLP=0 substream the default is that tagging is applied. Otherwise the default is that tagging is not applied.

5.5.4 SVCC-Based RCC Connection Establishment Overview

When establishing an SVCC to be used as a PNNI routing control channel, the LGN provides all call setup parameter values necessary for SVCC establishment using local configuration information. When a call is received to establish an RCC, the call setup parameters in the SETUP message may be rejected if the indicated values cannot be supported by this node. However, the values of local PNNI configuration parameters are not considered in making the decision to accept or reject the call. All PNNI implementations are required to support the default values stated below.

The LGN shall request non-real-time VBR service. If that is not available and real-time VBR is available, then that shall be requested in a new connection request. If real-time VBR is not available and CBR is, CBR shall be requested. If CBR is not available and ABR is, ABR shall be requested. If no other service category is available, UBR shall be requested. A service category is considered to be not available either if no route can be found with that service category, or if the call setup with that service category fails due to problems with service category , traffic parameters, or QoS parameters.

The call and connection procedures for SVCC setup are specified in Section 6.5. The following discussion details the content of the SETUP, CONNECT, RELEASE and RELEASE COMPLETE messages specifically for the case of an RCC. If a required information element is not present, the call must be rejected. Note that other information elements described in Section 6.4 of this specification may also be present. The coding standard used is discussed in Section 6.4.5.1 of this document.

5.5.4.1 SETUP Message Contents

5.5.4.1.1 AAL Parameters

The AAL parameters information element (see Section 6.4.5.8) must be used in the SETUP message. Table 5-2 lists the parameters to be used.

Table 5-2: AAL Parameters

Field	Value
AAL Type	5 (for AAL5)
Forward Maximum CPCS-SDU Size	8192 octets.
Backward Maximum CPCS-SDU Size	Same as Forward Maximum CPCS-SDU Size
SSCS Type	0 (Null SSCS)

5.5.4.1.2 ATM Traffic Descriptor

The ATM traffic descriptor information element (see Section 6.4.5.9) must be present. The parameters necessary to establish an RCC are included in the table below. It is possible that the calling end system may wish to create a connection with a different cell rate from the default values given in Table 5-3. In this case the called party may reject the call because it cannot support the requested connection.

When the called party rejects a call based on User-Cell Rate parameters the following actions may be taken by the calling party. If, in response to a SETUP message, a calling LGN receives a RELEASE COMPLETE message, or a CALL PROCEEDING message followed by a RELEASE message, with Cause code (Cause #37, User Cell Rate not available), it may examine the diagnostic field of the Cause I.E. and re-attempt the call after selecting smaller values for the parameter(s) indicated. If the RELEASE COMPLETE or RELEASE message is received with Cause code (Cause #73, Unsupported combination of traffic parameters), it may try other combinations permitted by Section 6.4.5.9.

Table 5-3: ATM User Cell Rate/ATM Traffic Descriptor

Field	Value
Forward Peak Cell Rate (CLP=0+1)	RCCPeakCellRate (for nrt-VBR, rt-VBR, CBR) (note 1)
Forward Sustainable Cell Rate (CLP=0)	RCCSustainableCellRate (for nrt-VBR, rt-VBR)
Forward Maximum Burst Size (CLP=0)	RCCMaximumBurstSize (for nrt-VBR, rt-VBR)
Backward Peak Cell Rate (CLP=0+1)	RCCPeakCellRate (for nrt-VBR, rt-VBR, CBR) (note 1)
Backward Sustainable Cell Rate (CLP=0)	RCCSustainableCellRate (for nrt-VBR, rt-VBR)
Backward Maximum Burst Size (CLP=0)	RCCMaximumBurstSize (for nrt-VBR, rt-VBR)
Best Effort Indicator	(included for UBR only)
Frame Discard Forward	Frame discard allowed (in SETUP message)
Frame Discard Backward	Frame discard allowed (in CONNECT message)
Tagging Forward	Tagging requested (in SETUP message, for nrt-VBR, rt-VBR)
Tagging Backward	Tagging requested (in CONNECT message, if “Tagging supported” was received in SETUP message, for nrt-VBR, rt-VBR)

Note 1 - For ABR and UBR service categories, the default Forward and Backward Peak Cell Rates (CLP=0+1) shall be equal to the line rate.

5.5.4.1.3 Broadband Bearer Capability

This information element (see Section 6.4.5.10) must be used in the SETUP message sent by the calling party. It is used to indicate what kind of network connection is desired. This specification recommends using Service Category non-real time VBR (nrt-VBR). When the calling party receives an indication from the network that Service Category nrt-VBR is not supported via a RELEASE or RELEASE COMPLETE message with cause #57 (bearer capability not authorized), cause #58 (bearer capability not presently available), or cause #65 (bearer capability not implemented), then Service Category rt-VBR must be tried. If that is not available, then Service Category CBR must be tried. If that is not available, then ABR must be tried. If that is not available, the UBR must be tried.

In the signalling message the Service Category is encoded by using the value BCOB-X for Bearer Class and using the values defined for that Service Category in the ATM Transfer Capability field of this information element.

Table 5-4: Broadband Bearer Capability

Field	**Value**
Bearer Class	BCOB-X
ATM Transfer Capability	Non-real time VBR Real time VBR Constant Bit Rate Available Bit Rate Unspecified Bit Rate
Susceptibility to clipping	0 (Not susceptible to clipping)
User plane connection configuration	0 for point-to-point

5.5.4.1.4 Broadband Low Layer Information

This information element (see Section 6.4.5.12) must be used in the SETUP message. It is used to indicate the protocol type carried in the connection. This encoding uses the ATM Forum's allocated 24-bit OUI with the PID indicating the PNNI Routing Control Channel.

Table 5-5: Broadband Low Layer Information

Field	**Value**
User information layer 3 protocol	11 (ISO/IEC TR 9577)
ISO/IEC TR 9577 Initial Protocol Identifier	64 (SNAP Identifier - 0x80, spread over 2 octets, left justified)
Continued from previous Octet	Continued (Ext bit is set to 1)
SNAP ID	0x80 (indicates SNAP and PID follow)
SNAP Organizational Unit Identifier	0x 00 A0 3E (ATM Forum OUI)
PID	**0x000A**

5.5.4.1.5 QoS Parameter

This information element (see Section 6.4.5.28) must be used in the SETUP message sent by the calling party.

Table 5-6: QoS Parameter

Field	**Value**
QoS Class Forward	RCCQoSClass
QoS Class Backward	RCCQoSClass

5.5.4.1.6 Extended QoS Parameters

When the service category used for establishment of the PNNI routing control channel is non-real-time VBR, real-time VBR, or CBR, an Extended QoS parameters information element must be included in the SETUP message sent by the calling party. By default no fields shall be included for the Cell loss ratio or Cell delay variation.

Table 5-7: Extended QoS Parameters

Field	Value
Origin	0 (for Originating user) (for nrt-VBR, rt-VBR, CBR)

5.5.4.1.7 End-to-End Transit Delay

By default the End-to-end transit delay information element shall not be included in the SETUP message.

5.5.4.1.8 Called Party Number

This information element (see Section 6.4.5.15) must be used in the SETUP message sent by the calling party. All ATM addresses used to set up the RCC use the ATM Forum UNI ATM End System Address Format of 20 octets.

Table 5-8: Called Party Number (ATM End System Addresses)

Coding Standard	0
I.E. Instruction Field	0
Length of Called Party	20 octets
Type of Number	0 (for Unknown)
Addressing/Numbering Plan	2 (binary 0010) (for ATM end system address)

5.5.4.1.9 Calling Party Number

This information element (see Section 6.4.5.18) must be used in the SETUP message sent by the calling party PNNI.

5.5.4.1.10 Connection Identifier

This information element must be present in the first message returned by the succeeding side (either the CALL PROCEEDING or CONNECT message) and indicates the VPI/VCI value assigned. Optionally, the Preceding side may include the information element in SETUP message to indicate a preferred VPI/VCI value for the connection (see Section 6.4.5.22).

5.5.4.1.11 DTL

This information element must be present in the SETUP message sent by the calling party (see Section 6.4.6.4).

5.5.4.2 CONNECT Message Contents

The CONNECT message is formatted by the called party. It is received by the calling party. It is used primarily to confirm the connection.

5.5.4.2.1 AAL Parameters

The called party may include the AAL Parameter information element in the CONNECT message.

If this information element is returned by the network, it should be checked to ensure that the parameters are unchanged or values that can be accepted. If not, the call should be released.

5.5.4.2.2 Broadband Low Layer Information

The called party may include a B-LLI information element in the CONNECT message.
If this information element is returned from the called party, it must be unchanged from the coding requested in the SETUP message. If not, the call must be released.

5.5.4.2.3 Connection Identifier

This information element indicates what VPI/VCI values have been assigned as specified in Section 6.5.2.2.

5.5.4.3 RELEASE and RELEASE COMPLETE Message Contents

PNNI defines the use of the following Cause Codes to indicate why a RCC is being released by a PNNI endpoint. The following encoding is used when explicitly releasing a RCC:

Table 5-9: Cause I.E.

Coding Standard	0	(ITU-TS), used for cause 16 and 31
	3	ATM Forum, used for cause 53
I.E. Instruction Field	0	(not significant)
Location	0	(User)
Cause Value	16	Normal call clearing
	31	Normal, Unspecified
	53	Call Cleared due to change in Peer Group Leader (this is a PNNI-specific cause code; see Section 5.5.6)

5.5.5 SVCCs when One End is not an LGN

After the initial exchange of Hellos, a lowest-level node may find that its neighbor is not in the same peer group, i.e., an outside neighbor. Upon examining the neighbor’s nodal hierarchy list (see Section 5.6.2.3), the lowest-level node discovers that it is in the peer group of one of its neighbor’s ancestors, i.e., the lowest-level node is a peer with the appropriate LGN as determined from the list.

Under such circumstances, the lowest-level node must be prepared to communicate over an SVCC with the neighboring peer LGN. It must, following the rules for SVCC initiator, either accept or initiate the SVCC. It must then use the procedures specified for operation over an SVCC, as described for SVCCs between logical group nodes. In this situation lowest level nodes may be required to aggregate outside links into horizontal links.

All references in this specification to an SVCC between logical group nodes intrinsically include this case as well.

An example of this is depicted in the following figure. Node A is a lowest-level node, which has outside links to nodes B.1, B.2, and B.3, none of which are in node A's peer group. However, from the nodal hierarchy lists exchanged on these outside links, these nodes discover that node A is a peer of the LGN representing the peer group in which B.1, B.2, and B.3 reside. Therefore an SVCC-based RCC must be established between nodes A and B.

In addition, node A is required to aggregate the outside links into a set of one or more horizontal links depending on the aggregation token values associated with those outside links. For example, assume there are two different aggregation token values being used in the diagram: i) for the links from B.1 to A and B.2 to A, and ii) for the links from B.3 to A. Given these aggregation token values, two horizontal links will be declated between nodes A and B: i) which aggregates the links from B.1 to A and B.2 to A, and ii) which aggregates the links from B.3 to A.

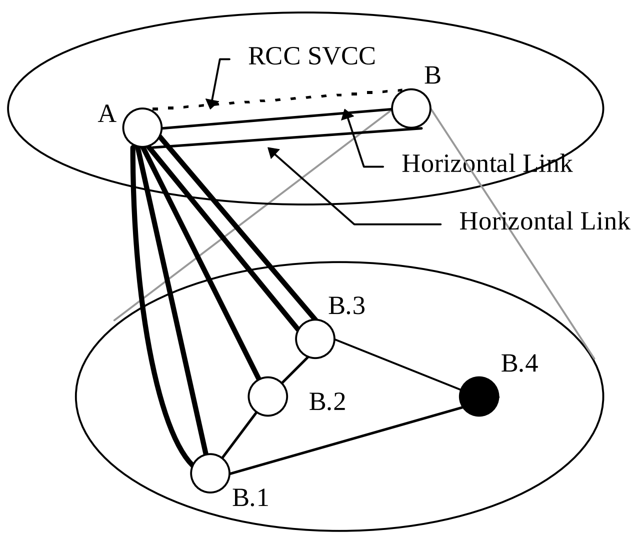


Figure 5-3: An SVCC-Based RCC and horizontal links between a Lowest-Level Node and an LGN.

5.5.6 Establishing and Maintaining SVCCs between Logical Group Nodes

5.5.6.1 SVCC Establishment

The SVCC-based RCCs between neighboring LGNs are established based on information gathered by PNNI.

The SVCC will traverse a border node which is declaring an uplink to the appropriate logical group node. Where there are many border nodes advertising uplinks to the same logical group node, any of the border nodes may be selected.

Each LGN finds out about the address of neighboring LGNs from the uplinks announced by border nodes in the peer groups that the LGN represents. Specifically, the called party address used must be the ATM End System Address for the neighboring LGN at the other end of the uplink which the SVCC is intended to cross, as advertised by the border node. For future flexibility, note that there is no requirement that different border nodes report the same ATM End System Address for the neighboring LGN.

In order to facilitate the routing of the SVCC Setup prior to the advertisement of the horizontal links between the LGNs, special requirements are put on the DTL that the LGN generates. Specifically, the DTL on the bottom of the stack must contain only two Logical Node Ids, that of the LGN that is initiating the SVCC, and that of the LGN that is the target of the SVCC. The Logical Port ID of the initiating LGN is irrelevant. In addition, the Logical Port ID of the last node (i.e., the border node) in all lower DTLs must specify a Logical Port ID that the border node has advertised in an uplink IG to the proper upnode. This Logical Port ID must not be zero.

5.5.6.2 Inconsistent Information and Partitioned Neighbor Peer Groups

Normally there is a single SVCC between a pair of adjacent peer groups. However, in some cases the border nodes within a particular peer group, with links to the same neighboring peer group, may be advertising inconsistent information. Thus, some border nodes may be advertising an uplink to one logical group node (which is alleged to represent a particular neighboring peer group), while other border nodes may be advertising an uplink to a different logical group node (which is alleged to represent the same neighboring peer group).

There are at least two reasons why this situation may occur: (i) The neighboring peer group may be partitioned, for example due to physical link and/or node failures, or (ii) The PGL in the neighboring peer group may have changed, and the border nodes may differ with respect to how rapidly they find out about and advertise this change.

In general, it is not possible to determine which of these situations has occurred. It is therefore necessary to operate with the assumption that the neighboring peer group is partitioned, and therefore to open SVCCs to each alleged neighboring LGN. If in fact there has been a change in the neighboring PGL (and one of the advertisements is out of date), then it will not be possible to actually succeed in opening an SVCC to the corresponding LGN.

5.5.6.3 Detailed Mechanisms for Establishment and Maintenance of SVCCs

The following describes the mechanism executed by a particular logical group node, here called ThisLGN, for setting up SVCCs to neighboring LGNs of ThisLGN.

When an uplink advertisement containing upnode X has reached ThisPGL and is processed by ThisLGN (ThisPGL denotes the peer group leader of ThisLGN’s child peer group):

A.1 If node X is at a higher level than the level of the peer group of ThisLGN, then announce an uplink to X by originating an appropriate PTSE in ThisLGN’s peer group.

A.2 Otherwise, if ThisLGN has an SVCC open to node X, then do nothing.

A.3 Else, If ThisLGN’s node ID is numerically smaller than the node ID of node X, then do nothing.

A.4 Else, if node X is at a lower level than ThisLGN’s peer group, or is at the same level but is in a different peer group than ThisLGN, then an error condition has occurred and nothing should be done. Note that this error condition is likely to occur if ThisLGN’s peer group has just changed, ThisPGL has just been elected, or ThisPGL’s peer group has just healed a partition. Otherwise this should not occur and indicates an error in the border node advertising the uplink.

A.5 Else (X is in the same peer group as ThisLGN; there is no SVCC currently open to X, and ThisLGN has the numerically larger node ID): Start timer InitialLGNSVCTimer with value InitialLGNSVCTimeout. Note that this timer must be jittered. When this timer expires, open an inter-LGN SVCC to the ATM address of X.

A.6 If the SVCC setup attempt succeeds, then use the SVCC as a PNNI Routing Control Channel within ThisLGN's peer group (beginning with exchange of Hellos, and other defined PNNI protocols).

When ThisLGN establishes the SVCC-based RCC, it is the DTL Originator and generates the first DTL:

DTL:[ThisLGN, Node X] pointer-1

The SETUP is then (logically) passed to ThisPGL and it generates the DTL to the border node.

DTL:[ThisPGL, ... BorderNode1] pointer-1

ThisPGL is then responsible for alternate routing when crankback occurs. Given the puropse of this SVCC, all practical routes should be attempted.

If the SVCC setup attempt fails, then start the RetryLGNSVCTimer timer with value RetryLGNSVCTimeout. Note that this timer must be jittered. If a successful SVCC arrives from the same peer LGN (X) while the timer is running, cancel the timer.

If ThisLGN detects the presence of two or more SVCCs to the same neighboring LGN then:

B.1 If ThisLGN's node ID is numerically smaller than the neighboring LGN's node ID, then ThisLGN shall continue to use all SVCCs. Hello packets must be transmitted on all SVCCs. Database exchange packets and PTSPs may be transmitted on any of the SVCCs. Packets received on any of the SVCCs are acted on independent of which SVCC they were received on. Note that this implies that there is only a single LGN Hello FSM for multiple SVCCs.

B.2 If ThisLGN's node ID is numerically larger than the neighboring LGNs node ID, then choose one SVCC to leave open. Close the other SVCC(s) with cause number 16 "normal call clearing".

If ThisPGL ceases to be PGL:

C.1 All PTSEs originated by ThisLGN shall be flushed.

C.2 ThisLGN clears the SVCCs to all of its neighboring LGNs by sending RELEASE messages with CAUSE IEs indicating cause number 53 "call cleared due to change in PGL".

If an existing SVCC to a neighboring LGN is closed:

D.1 If ThisLGN receives a RELEASE message with cause code number 53 "call cleared due to change in PGL", that relates to a particular SVCC to neighbor node X, then the respective higher level link(s) shall be removed by carrying out the following actions. The event LinkDown shall be triggered in the SVCC-based RCC Hello FSM to upnode X (see Section 5.6.3.1), and the LinkDown event shall be triggered in all associated LGN horizontal link Hello FSMs (see Section 5.6.3.2). Start the RetryLGNSVCTimer timer with value RetryLGNSVCTimeout.

D.2 Else, if the cause code indicates that the call was cleared due to an error, and if upnode X is still being advertised as the destination of uplinks originated by one or more border nodes, and another SVCC is not opened to X, and ThisLGN has a numerically larger node ID than upnode X, then attempt to re-establish this SVCC to upnode X immediately and go to Step A.6. Otherwise do nothing.

If the RetryLGNSVCTimer expires:

E.1. If the upnode X is still being advertised as the destination of uplinks originated by one or more border nodes, and if X is in the same peer group as ThisLGN, there is no SVCC currently open to

X, and ThisLGN has the numerically larger node ID, then retry an SVCC setup to X and go to step A.6.

E.2. Otherwise do nothing.

Note: The failure of an SVCC between neighboring LGNs may be caused by failure of a single component internal to either of the neighboring peer groups. This does not necessarily imply any significant loss of connectivity between peer groups. Thus, if the SVCC fails this MUST NOT have any immediate effect on announcement of the corresponding higher level link between logical group nodes. Rather, there must be an attempt to re-establish the SVCC prior to changing the announced status of the link.

See Section 5.6.3 for further details.

5.6 The Hello Protocol

PNNI Hello packets are exchanged between neighbor nodes. This is done over RCCs.

Hellos are sent over all physical links and VPCs in order to discover and verify the identity of neighbor nodes and to determine the status of the links to those nodes.

In addition, Hellos are also sent over all SVCCs used as RCCs. These Hellos are used to verify the identity of the neighbor node, the status of the horizontal links to that node, and to determine the status of the RCC.

References to the hello data structures that appear in the Hello FSM refer to the hello data structures described in the appropriate sections (see Sections 5.6.2.1.1, 5.6.3.1.1, 5.6.3.2.1) based on the contexts.

5.6.1 PNNI Version Negotiation

A PNNI implementation generally supports a range of protocol versions. In protocol messages, version fields are unsigned integers. A node indicates by the `newest' and `oldest' version supported fields in all packets the range of supportable versions. All versions in this range are supported by the advertiser. An additional field, the 'protocol version', contains the version used to produce the packet. Once negotiation has taken place, this will reflect the result of that negotiation. This negotiation consists of using the lower of the highest locally supportable version, and the highest version supportable by the neighbor. This negotiation is only done as part of the Hello FSM. If that version is not locally supportable, the adjacency can not be started, and management should be notified. The explicit 'oldest' supported version field can be used by the other side to detect this incompatibility, so that it may also notify management of the problem.

5.6.2 Hellos Over Physical Links and VPCs

5.6.2.1 The Hello State Machine

Between two lowest-level neighbor nodes each physical link or VPC has its own instance of the hello protocol. An instance of the hello protocol is made up of a hello data structure and a Hello state machine.

For lowest-level neighbor nodes with parallel physical links and/or VPCs between them, there will be multiple instances of the Hello protocol. However, for the purposes of database synchronization and flooding of PTSEs, there is only one instance of the neighboring peer data structure and associated neighbor peer state machine. In order to describe the interaction between the multiple Hello conversations and the single neighboring peer conversation (for database synchronization and flooding procedures), reference is made to a neighboring peer state machine and to its events AddPort and DropPort. The event AddPort indicates that a new inside link to the neighboring peer node has come up (i.e., has reached the Hello state 2-Way Inside). The event DropPort indicates that a link to the neighboring peer has gone down (i.e., has fallen out of the Hello state 2-Way Inside). The description of the Hello state machine includes indication of when the events AddPort and DropPort must be triggered, thus describing the interaction between the multiple Hello conversations and the corresponding neighboring peer conversation.

5.6.2.1.1 The Hello Data Structure

There is a single hello data structure for each of this node's physical ports and for each logical port (each Virtual Path Connection for which this node is an endpoint).

Each hello data structure consists of the following items:

State
: The operational status of a link. This is described in more detail in Section 5.6.2.1.2.

Port ID
: A number assigned by the node that identifies which physical port and which virtual path connection, if any, is described by this hello data structure.

Remote Node ID
: The Node ID of the neighbor node on the other end of the link. The Remote Node ID is learned when Hellos are received from the neighbor.

Remote Peer Group ID
: The peer group ID of the neighbor node on the other end of the link. The Remote Peer Group ID is learned when Hellos are received from the neighbor.

Remote Port ID
: The neighbor's port ID for this link. The Remote Port ID is learned when Hellos are received from the neighbor. When the Remote Port ID is not known, its value must be set to zero.

HelloInterval
: The amount of time, in seconds, between Hellos that the node sends out over this link, in the absence of event-triggered Hellos.

Hello Timer
: An interval timer that fires every HelloInterval seconds. Whenever the timer fires, the node transmits a Hello over this link.

InactivityFactor
: The amount of time, in multiples of the HelloInterval declared by the neighbor in its Hellos, before the node will consider the link down, if the neighbor's Hellos cease to arrive.

Inactivity Timer
: A single shot timer whose firing indicates that no Hellos have been received from this neighbor recently. The initial value of the Inactivity Timer must be set to the InactivityFactor times the HelloInterval from the most recent Hello received from the neighbor node.

Version
: The current version of the Hello protocol being used for communication with this neighbor. If no acceptable version number has been derived, this field will be zero.

For outside links (i.e., links to nodes in other peer groups) the following additional items are included:

Transmitted ULIA Sequence Number
: The sequence number from the most recently transmitted ULIA.

Received ULIA Sequence Number
: The sequence number from the most recently received ULIA or cleared if the last received hello did not contain the ULIA.

Received Nodal Hierarchy Sequence Number
: The sequence number from the most recently received nodal hierarchy list or cleared if the last received hello did not contain a nodal hierarchy list.

Upnode Node ID
: The node ID of the upnode. The Upnode Node ID is learned when Hellos containing a sufficiently complete nodal hierarchy list are received from the neighbor node.

Common Peer Group ID
: The peer group ID of the common peer group of the border node and its outside neighbor. The Common Peer Group ID is learned when Hellos containing a sufficiently complete nodal hierarchy list are received from the neighbor node.

Upnode ATM End System Address
: The ATM End System Address of the upnode. The Upnode ATM End System Address is learned when Hellos containing a sufficiently complete nodal hierarchy list are received from the neighbor node.

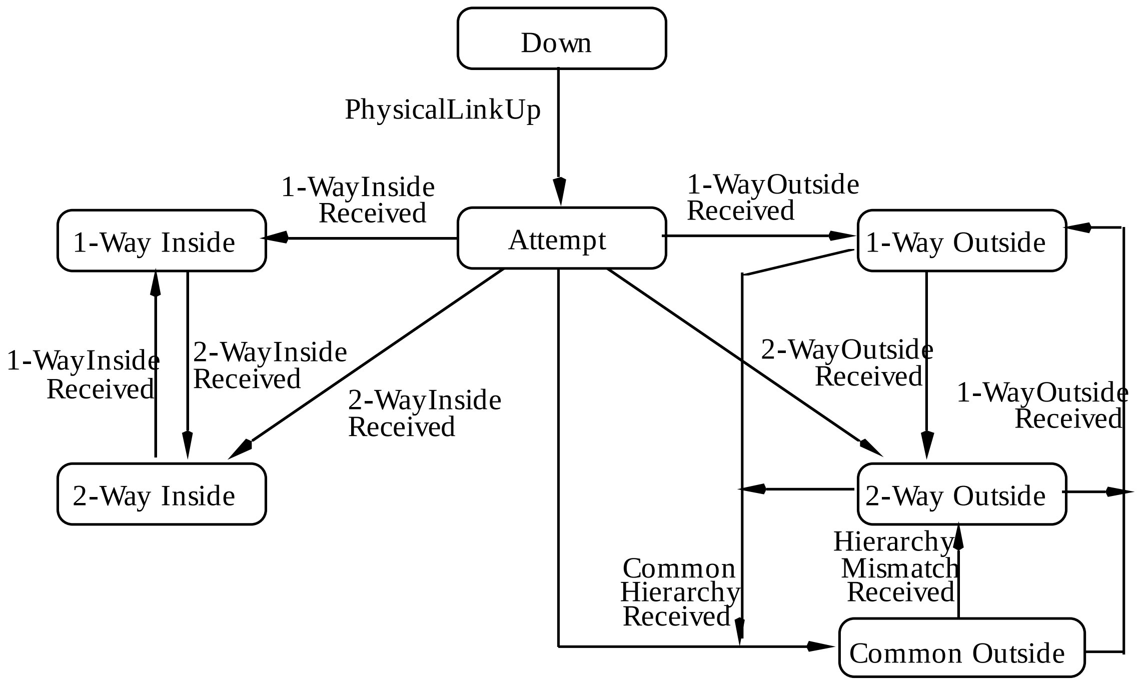
Derived Link Aggregation Token
: The algorithmically derived value of the link aggregation token for this link.

Configured Link Aggregation Token
: This node’s configured link aggregation token for this link.

Remote Link Aggregation Token
: The link aggregation token received in Hellos from the neighbor at the other end of the link.

5.6.2.1.2 Hello States

The states that Hello FSM may attain are described in this section. Figure 5-4 shows a diagram of the possible state changes. The arcs are labeled with the events that cause each state change. These events are described in Section 5.6.2.1.3. For a detailed description of the state changes and the actions involved with each state change, see Section 5.6.2.1.4.



In addition to the state transitions pictured,

Event LinkDown always forces Down State,
Event InactivityTimer always forces Attempt State,
Event HelloMismatchReceived always forces Attempt State.

Figure 5-4: Hello State Changes

Down
: The initial state of the Hello FSM. This state is also reached when lower-level protocols have indicated that the link is not usable. No PNNI routing packets will be sent or received over such a link.

Attempt
: This state indicates that either no Hellos or Hellos with mismatch information have been received recently from the neighbor. In this state, attempts are made to contact the neighbor by periodically sending the neighbor Hellos with period HelloInterval.

1-Way Inside
: In this state, Hellos have recently been received from the neighbor and it has been established that both nodes are members of the same peer group, but the remote node ID and remote port ID in the neighbor's Hellos were set to zero.

2-Way Inside

In this state, Hellos have recently been received from the neighbor indicating that both nodes are members of the same peer group and including the correct remote node ID and remote port ID fields. When this state is reached, it indicates that bi-directional communication over this link between the two nodes has been achieved. Database summary packets, PTSE Request packets, PTSPs, and PTSE acknowledgment packets can only be transmitted over links that are in the 2-Way Inside state. For physical links and VPCs, only those links that are in the 2-Way Inside state can be advertised by this node in PTSEs as horizontal links.

1-Way Outside

This state indicates that Hellos have recently been received from the neighbor and that the neighbor node belongs to a different peer group, but the remote node ID and remote port ID in the neighbor’s Hellos were set to zero. In this state and in the 2-Way Outside state, the node searches for a common peer group that contains both this node and the neighbor node.

2-Way Outside

In this state, Hellos have recently been received from the neighbor indicating that the neighbor node belongs to a different peer group and including the correct remote node ID and remote port ID fields, but with a nodal hierarchy list that does not include any common peer group. In this state and in the 1-Way Outside state, the node searches for a common peer group that contains both this node and the neighbor node.

Common Outside

When this state is reached, it indicates that a common level of the routing hierarchy has been found, in addition to achieving full bi-directional communication between the two nodes. Links that have reached the Common Outside state can be advertised in PTSEs as uplinks to the upnode.

5.6.2.1.3 Events causing Hello state changes

State changes can be brought about by several possible events associated with operation of the Hello protocol. These events are shown as the labeled arcs in Figure 5-4. A detailed explanation of the state changes and actions taken after an event occurs is given in Section 5.6.2.1.4.

LinkUp

Lower layer protocols have indicated that the link is operational. If autoconfiguration is enabled for the link, the LinkUp event is generated when the ILMI autoconfiguration has completed with an ATM interface type of Private NNI.

1-WayInsideReceived

A Hello has been received from the neighbor in which the peer group ID matches the peer group ID of this node and Remote Node ID and Remote Port ID are set to zero. Additionally, if the Version, Remote Node Id, Remote Port ID and Remote Peer Group Id fields in the hello data structure are not equal to zero, they must match, respectively, the Protocol Version, Node Id of Source, Port ID and Peer Group ID from the received Hello.

2-WayInsideReceived

A Hello has been received from the neighbor in which the remote node ID and remote port ID correctly identify this node’s node ID and this port ID, and the peer group ID matches this node’s peer group ID. Additionally, if the Version, Remote Node Id, Remote Port ID and Remote Peer Group Id fields in the hello data structure are not equal to zero, they must match, respectively, the Protocol Version, Node Id of Source, Port ID and Peer Group ID from the received Hello.

1-WayOutsideReceived

A Hello has been received from the neighbor including a peer group ID that does not match this node’s own peer group ID, and with remote node ID and remote port ID fields that are set to zero. Additionally, if the Version, Remote Node Id, Remote Port ID and Remote Peer Group Id fields in the hello data structure are not equal to zero, they must match, respectively, the Protocol Version, Node Id of Source, Port ID and Peer Group ID from the received Hello.

2-WayOutsideReceived

A Hello has been received from the neighbor including:

(i) remote node ID and remote port ID fields that correctly reflect this node's node ID and port ID for this link, and
(ii) a peer group ID that does not match this node's own peer group ID, and
(iii) the Common Peer Group ID, the Upnode Node Id and Upnode ATM End System Address in the hello data structure are set to zero, and
(iv) one of the following is true:
 A - Either the nodal hierarchy list, aggregation token or ULIA is not present
 B - the nodal hierarchy list is present but does not contain a common peer group.

Additionally, if the Version, Remote Node Id, Remote Port ID and Remote Peer Group Id fields in the hello data structure are not equal to zero, they must match, respectively, the Protocol Version, Node Id of Source, Port ID and Peer Group ID from the received Hello.

CommonHierarchyReceived

A Hello has been received from the neighbor including:

(i) remote node ID and remote port ID fields that correctly reflect this node's node ID and port ID for this link, and
(ii) a peer group ID that does not match this node's own peer group ID, and
(iii) a ULIA, and
(iv) an Aggregation token, and
(v) a nodal hierarchy list containing either:
- a sequence number that matches the received Nodal Hierarchy Sequence Number, and the Common Peer Group ID has already been set in the hello data structure, or
- a common peer group; and the Common Peer Group ID, the Upnode Node Id and Upnode ATM End System Address in the hello data structure are set to zero or these fields match, respectively, the lowest common peer group in the received nodal hierarchy list and the Node ID and ATM End System Address associated with that common peer group ID.

Additionally, if the Version, Remote Node Id, Remote Port ID and Remote Peer Group Id fields in the hello data structure are not equal to zero, they must match, respectively, the Protocol Version, Node Id of Source, Port ID and Peer Group ID from the received Hello.

HelloMismatchReceived

A Hello has been received from the neighbor in which at least one of the Version, originating node ID, peer group ID, and port ID is different from the Protocol Version, remote node ID, remote peer group ID, or remote port ID, respectively, in the hello data structure.

Alternatively, a Hello has been received in which the remote node ID and/or remote port ID are different from this node's own node ID or this node's port ID for the receiving link, respectively, and are not both set to zero. In the state Attempt, only the second alternative is considered. The HelloMismatchReceived event takes precedence over the other events.

HierarchyMismatchReceived

A Hello has been received from the neighbor that:

(i) includes a remote node ID and remote port ID fields that correctly reflect this node's node ID and port ID for this link, and
(ii) includes a peer group ID that does not match this node's own peer group ID, and
(iii) does not meet the criteria for HelloMismatchReceived, and
(iv) either:
- there is no nodal hierarchy list, or

- no ULIA is found, or
- no aggregation token is found, or
- a new nodal hierarchy list sequence number has been received and either
  - the sequence of peer group levels is not strictly descending in value, or
  - no common peer group is found, or
  - the lowest common peer group ID found is different from the Common Peer Group ID in the hello data structure, or
  - in the information triple that includes the Common Peer Group ID, a higher level node ID and/or higher level ATM End System Address that does not match the Upnode Node ID or Upnode ATM End System Address in the hello data structure, respectively.

HelloTimerExpired
: The Hello Timer has expired.

InactivityTimerExpired
: The Inactivity Timer has expired. This means that no Hellos have been received recently from the neighbor.

LinkDown
: Lower layer protocols indicate that this link is not usable. If ILMI autoconfiguration completes with an ATM interface type not equal to Private NNI, a LinkDown event is generated. Note that the loss of ILMI connectivity does not generate any event.

5.6.2.1.4 Description of The Hello State Machine

The finite state machine is a two dimensional table with States across the top of the table and the Events down the left side. Each pairing of event and state cross at a "cell" in the table. The cell shows what state transition occurs and the action to take. For example, for the event and state pair of "LinkUp" and "Down" the cell reads "Hp1, Attempt". "Attempt " is the new state and "Hp1" is the Action to be taken. The actions definitions can be found following table.

Table 5-10: Hello FSM

	States						
Events	Down	Attempt	1-Way Inside	2-Way Inside	1-Way Outside	2-Way Outside	Common Outside
Link Up	Hp1, Attempt	Hp0, Attempt	Hp0, 1WayIn	Hp0, 2WayIn	Hp0, 1WayOut	Hp0, 2WayOut	Hp0, Common
1-Way Inside Received	Hp0, Down	Hp2, 1WayIn	Hp12, 1WayIn	Hp10, 1WayIn	FSM_Err	FSM_Err	FSM_Err
2-Way Inside Received	FSM_Err	Hp3 , 2WayIn	Hp4 , 2WayIn	Hp12, 2WayIn	FSM_Err	FSM_Err	FSM_Err
1-Way Outside Received	Hp0, Down	Hp5 , 1WayOut (Note 1)	FSM_Err	FSM_Err	Hp12 , 1WayOut	Hp13 , 1WayOut	Hp14 , 1WayOut
2-Way Outside Received	FSM_Err	Hp5, 2WayOut (Note 1)	FSM_Err	FSM_Err	Hp12, 2WayOut	Hp12, 2WayOut	FSM_Err
Common Hierarchy Received	FSM_Err	Hp6, Common (Note 1)	FSM_Err	FSM_Err	Hp7, Common	Hp7, Common	Hp20, Common
Hello Mismatch Received	FSM_Err	Hp0, Attempt	Hp8, Attempt	Hp16, Attempt	Hp8, Attempt	Hp8, Attempt	Hp17, Attempt
Hierarchy Mismatch Received	FSM_Err	FSM_Err	FSM_Err	FSM_Err	FSM_Err	FSM_Err	Hp11, 2WayOut
Hello Timer Expired	FSM_Err	Hp15, Attempt	Hp15, 1WayIn	Hp15, 2WayIn	Hp15, 1WayOut	Hp15, 2WayOut	Hp15, Common
Inactivity Timer Expired	FSM_Err	FSM_Err	Hp8, Attempt	Hp16, Attempt	Hp8, Attempt	Hp8, Attempt	Hp17, Attempt
Link Down	Hp0, Down	Hp9, Down	Hp9, Down	Hp18, Down	Hp9, Down	Hp9, Down	Hp19, Down

Note 1: Nodes that are not capable of becoming border nodes must use action Hp0 and remain in the Attempt state.

FSM_ERR Represents an internal implementation error.

Hp0

Action: Do nothing.

Hp1

Action: Send a Hello out over the link and start the Hello Timer, enabling the periodic sending of Hellos.

Hp2

Action: Start the Inactivity Timer for the link. Set the Remote Node ID, Remote Peer Group ID, and Remote Port ID in the hello data structure to the Node ID, Peer Group ID, and Port

ID listed in the received Hello. Calculate the lower of the received newest version supported and the local newest version supported. Record this as the Version number. Send a Hello to the neighbor and restart the Hello Timer.

Hp3

Action: Start the Inactivity Timer for the link. Set the Remote Node ID, Remote Peer Group ID, and Remote Port ID in the hello data structure to the Node ID, Peer Group ID, and Port ID listed in the received Hello. Calculate the lower of the received newest version supported and the local newest version supported. Record this as the Version number. Send a Hello to the neighbor and restart the Hello Timer. Invoke the corresponding neighboring peer state machine with the event AddPort.

Hp4

Action: Restart the Inactivity Timer. Invoke the corresponding neighboring peer state machine with the event AddPort.

Hp5

Action: Start the Inactivity Timer for the link. Set the Remote Node ID, Remote Peer Group ID, and Remote Port ID in the hello data structure to the Node ID, Peer Group ID, and Port ID listed in the received Hello. Calculate the lower of the received newest version supported and the local newest version supported. Record this as the Version number. Send a Hello to the neighbor including a nodal hierarchy list and all outgoing service category metrics information groups describing this link, and restart the Hello Timer.

Hp6

Action: Start the Inactivity Timer for the link. Set the Remote Node ID, Remote Peer Group ID, and Remote Port ID in the hello data structure to the Node ID, Peer Group ID, and Port ID listed in the received Hello. Find the lowest common peer group in the nodal hierarchy list received in the neighbor’s Hello and in this node’s own nodal hierarchy list, and set the Upnode Node ID, Common Peer Group ID, and Upnode ATM End System Address to the values contained in the corresponding information triple from the received nodal hierarchy list. Calculate the lower of the received newest version supported and the local newest version supported. Record this as the Version number. Send a Hello to the neighbor including a nodal hierarchy list and all outgoing ULIAs describing this link, and restart the Hello Timer. The link may now be advertised by this node in PTSEs as an uplink to the upnode.

Hp7

Action: Restart the Inactivity Timer, since a Hello has been seen from the neighbor. Find the lowest common per group in the nodal hierarchy list received in the neighbor’s Hello and in this node’s own nodal hierarchy list, and set the Upnode Node ID, Common Peer Group ID, and Upnode ATM End System Address to the values contained in the corresponding information triple from the received nodal hierarchy list. The link may now be advertised by this node in PTSEs as an uplink to the upnode.

Hp8

Action: The Inactivity Timer is disabled and the Version, remote node ID, remote peer group ID, remote port ID, Upnode ID, common peer group ID, Upnode ATM End System Address fields, Received ULIA Sequence Number and the Received Nodal Hierarchy Sequence Number in the hello data structure are cleared. Send a Hello to the neighbor and restart the Hello Timer.

Hp9

Action: The Hello and Inactivity timers are disabled and the Version, remote node ID, remote peer group ID, remote port ID, Upnode Node ID, common peer group ID, Upnode ATM End System Address, Received ULIA Sequence Number and the Received Nodal Hierarchy Sequence Number fields in the hello data structure are cleared.

Hp10

Action: Restart the Inactivity Timer. Send a Hello to the neighbor and restart the Hello Timer. Invoke the neighboring peer state machine with the event DropPort.

Hp11

Action: Restart the Inactivity Timer. The Upnode ID, common peer group ID and Upnode ATM End System Address fields in the hello data structure are cleared. This link must be removed from any PTSEs originated by this node in which it has been advertised as an uplink

Hp12

Action: Restart the Inactivity Timer for the link since a Hello has been received from the neighbor.

Hp13

Action: Restart the Inactivity Timer. Clear the Received ULIA Sequence Number and the Received Nodal Hierarchy Sequence Number from the Hello data structure. Send a Hello to the neighbor including a nodal hierarchy list and all outgoing ULIAs describing this link, and restart the Hello Timer.

Hp14

Action: Restart the Inactivity Timer. The Upnode ID, common peer group ID, Upnode ATM End System Address fields, Received ULIA Sequence Number and the Received Nodal

Hierarchy Sequence Number in the hello data structure are cleared. Send a Hello to the neighbor including a nodal hierarchy list and all outgoing ULIAs describing this link, and restart the Hello timer. This link must be removed from any PTSEs originated by this node in which it has been advertised as an uplink.

Hp15

Action: Send a Hello to the neighbor and restart the Hello Timer.

Hp16

Action: The Inactivity Timer is disabled and the Version, remote node ID, remote peer group ID, remote port ID, Upnode ID, common peer group ID, Upnode ATM End System Address fields, Received ULIA Sequence Number and the Received Nodal Hierarchy Sequence Number in the hello data structure are cleared. Send a Hello to the neighbor and restart the Hello Timer. The neighboring peer state machine must be invoked with the event DropPort.

Hp17

Action: The Inactivity Timer is disabled and the Version, remote node ID, remote peer group ID, remote port ID, Upnode ID, common peer group ID, Upnode ATM End System Address fields, Received ULIA Sequence Number and the Received Nodal Hierarchy Sequence Number in the hello data structure are cleared. Send a Hello to the neighbor and restart the Hello Timer. This link must be removed from any PTSEs originated by this node in which it has been advertised as an uplink.

Hp18

Action: The Hello and Inactivity timers are disabled and the Version, remote node ID, remote peer group ID, remote port ID, Upnode Node ID, common peer group ID, Upnode ATM End System Address, Received ULIA Sequence Number and the Received Nodal Hierarchy Sequence Number fields in the hello data structure are cleared. The neighboring peer state machine must be invoked with the event DropPort.

Hp19

Action: The Hello and Inactivity timers are disabled and the Version, remote node ID, remote peer group ID, remote port ID, Upnode Node ID, common peer group ID, Upnode ATM End System Address, Received ULIA Sequence Number and the Received Nodal Hierarchy Sequence Number fields in the hello data structure are cleared. This link must be removed from any PTSEs originated by this node in which it has been advertised as an uplink.

Hp20

Action: Restart the Inactivity Timer. If the ULIA sequence number does not match the received ULIA Sequence Number in the hello data structure, then the uplink advertisement corresponding to this link must be updated with the received ULIA and re-originated immediately (subject to PTSE holddown).

5.6.2.2 Sending Hellos

Hellos are sent over each link in order to establish that bi-directional communication has been achieved. When the Version field of the hello data structure is zero, Hellos are sent encoded according to the newest version supported by this implementation. Otherwise the recorded Version is used. In either case, the version used is encoded in the Protocol Version field of the Hello packet.

In all states other than Down, Hellos are transmitted periodically, every HelloInterval seconds. In addition, event-triggered Hellos are sent in the following cases (subject to the hold-down timer):

- Upon every state change except the following:
    - 1-Way Inside to 2-Way Inside,
    - 1-Way Outside to 2-Way Outside,
    - 1-Way Outside to Common Outside,
    - 2-Way Outside to Common Outside, and
    - Common Outside to 2-Way Outside.
- On an outside link, whenever a significant change occurs in the ULIA for this outside link.
- On an outside link, whenever a change occurs in the node's nodal hierarchy list.
- On an outside link, whenever a change occurs in the link aggregation token for this outside link.

The period between successive transmission of Hellos may not be less than MinHelloInterval. The use of a hold-down timer prevents a node from sending Hellos at unacceptably high rates. If multiple event triggered hellos are deferred because of the hold down timer, then only one hello is transmitted containing the most current information for all IGs when the hold down timer expires. Whenever event-triggered Hellos are transmitted, the Hello Timer is restarted so that the next Hello is scheduled after HelloInterval seconds subject to the usual jitter.

When the Hello state is Attempt, Hellos must have their remote node ID and remote port ID fields set to zero as stored in the hello data structure. When the Hello state is 1-Way Inside, 2-Way Inside, 1-Way Outside, 2-Way Outside, or Common Outside, Hellos must include as the remote node ID and the remote port ID the neighbor node's node ID and port ID, as learned from the Hellos received over the same link and stored in the hello data structure.

The nodal hierarchy list, describing all of the node's higher level node IDs, peer group IDs, and LGN ATM End System Addresses as received from the PGL in its Higher Level Binding information, is included in Hellos when the state is 1-Way Outside, 2-Way Outside, or Common Outside. If multiple nodes claim to be peer group leader of this peer group, advertise the information from the one chosen locally as peer group leader. For higher-level peer groups, if multiple nodes claim to be PGL, advertise the information from the one selected by the ancestor LGN in that peer group.

Whenever a change occurs in the number or content of known higher levels, as expressed in the nodal hierarchy list, the sequence number of the nodal hierarchy list must be incremented and an event-triggered Hello must be sent. Additionally, whenever a change occurs in the node ID, peer group ID, or ATM address at the lowest level, the sequence number of the nodal hierarchy list must be incremented and an event-triggered Hello must be sent. When sending the first instance of the nodal hierarchy list to any neighbor, the value of the sequence number must be greater than zero.

Each time a Hello is transmitted including the nodal hierarchy list, the list must include all higher levels that are known at that time. If no higher levels of hierarchy are known, then an empty nodal hierarchy list must be included. Note that when the hierarchy is still coming up, the number of levels included in the nodal hierarchy list may increase with each successive Hello.

A ULIA IG that encompasses all outgoing Resource Availability IGs is included in Hellos when the state is either 1-WayOutside, 2-WayOutside or CommonOutside.

The Configured Link Aggregation Token, which is used to identify uplinks that may be aggregated to a particular induced uplink or a particular horizontal link, is included in Hellos when the state is either 1-WayOutside, 2-WayOutside or CommonOutside.

5.6.2.2.1 Originating and sending ULIAs in Hellos on Outside Links

For certain Hello states, a border node must include a ULIA IG in the Hello packets it sends on each outside link to neighboring border nodes. The ULIA itself does not specify any link state information, but rather is used to bundle other IGs which do. The flexibility provided by this mechanism allows a border node to dictate the exact link state information (IGs) advertised by its neighbor without requiring the neighbor to understand or interpret the individual IGs. In this fashion, portions of the network can be upgraded (e.g. to provide new security or policy features in new and unknown IGs), and have the correct link state information advertised throughout the hierarchy.

Significant change in the ULIA contents is indicated by a change to the ULIA sequence number. The originator only increments the ULIA sequence number when the change to the ULIA contents is significant.

Since the border nodes do not interpret the individual IGs within ULIAs received in Hello packets, any change to the ULIA sequence number indicates a significant change, and must therefore trigger an update to the corresponding uplink PTSE. Re-origination of a new instance of an uplink PTSE in response to this triggered update is subject to the PTSE hold down timer.

A border node must not generate new sequence numbers for the ULIA it is sending in Hellos unless a significant change has occurred for one or more of the IGs contained in the ULIA. If no significant change has occurred since the last ULIA was sent in a Hello packet, then the (possibly modified) ULIA with the previous sequence number must continue to be sent in Hellos. Any insignificant changes to the bundled IGs by definition is not important enough to incur the processing expense of propagating the change hierarchically and therefore must not carry a change in the sequence number. Whenever a new ULIA is formed in response to a significant change, the sequence number must be incremented.

A significant change event for any IG that was previously bundled into a ULIA and sent in a Hello packet triggers the building of a new ULIA, and the sending of a new Hello packet to the corresponding outside neighbor. When the new Hello packet is sent, it will include the new ULIA and new sequence number. When forming the new ULIA in response to a significant change to one or more of the bundled IGs, the border node inserts the most recent and accurate link state information for all bundled IGs, whether significant change has occurred for them or not. In this way, a significant change to any one IG causes the latest information for all IGs bundled in the same ULIA to be advertised.

5.6.2.3 Receiving Hellos

If a Hello is received with a Protocol Version that is not supported, the packet must be discarded. If the range of supported versions indicated in the Hello packet does not overlap with the range of supported versions of this node, a management notification may be logged. See Section 5.1.3.

If a Hello is received with a top level unknown TLV with the mandatory tag bit set, the hello must be discarded. In addition, if either the Hello Interval or Port ID in the received hello packet is set to zero, the packet is also invalid and must be discarded.

The neighbor node is identified by the node ID found in the Hello's header. If the Remote Node ID, Remote Peer Group ID, and Remote Port ID in the hello data structure have not yet been specified, then they must be set to the received Hello's originating node ID, peer group ID, and port ID, respectively. If the Version field in the hello data structure is zero, calculate the lower of the received newest version supported and the local newest version supported. Record this as the Version number.

If the received Hello contains a new instance of the nodal hierarchy list, as indicated by a new sequence number, then the nodal hierarchy list must be searched for the lowest level peer group that both nodes have in common. Note that the neighbor's node ID, peer group ID, and ATM End System Address must be considered logically to be the lowest level component of the nodal hierarchy list, even though it does not explicitly appear in the list.

If a node finds a match with its own peer group ID, there is no uplink (see Section 5.5.5). Rather this node will become a neighboring peer of the LGN representing the neighbor border node in the common peer group. An SVCC will be established and one or more horizontal links will be advertised as detailed in Section 5.5.6.

The received aggregation token is processed as described in Section 5.10.3.1.

Each received Hello causes the Hello state machine to be executed with one of the following events, depending on the contents of the received Hello:

1-WayInsideReceived
2-WayInsideReceived
1-WayOutsideReceived
2-WayOutsideReceived
CommonHierarchyReceived
HelloMismatchReceived
HierarchyMismatchReceived

5.6.2.3.1 Processing ULIAs received in Hellos on outside links

When a border node receives a Hello packet from a neighboring border node on an outside link, it must determine whether to (re-)originate the corresponding uplink PTSE. No change to the uplink PTSE need be made if the sequence number for the ULIA received in the most recent Hello packet is the same as that received with the previous ULIA (in the previous Hello packet). The sequence number comparison at the receiver is used to determine whether significant change to the contained link state information IGs has occurred. Any change in the ULIA sequence number field when compared with the previously received ULIA sequence number constitutes a significant change that, subject to hold down, will cause the receiver to re-originate the corresponding uplink PTSE. Note that if the uplink advertisement is re-originated for other reasons, the most recently received ULIA must be used in composing it.

5.6.3 The LGN Hello Protocol

In the following discussion, we refer to Logical Group Node neighbors, and do so for the ease of reading the text (see Section 5.5.5). It must be noted however, that when the hierarchy has unequal depths it is possible that a lowest level border node is peer to a LGN. In that case an SVCC will exist between that border node and the LGN, and all of the following text applies.

The Hello protocol between LGNs serves two purposes. One is to monitor the status of the SVCC used as a PNNI routing control channel between the two LGNs to increase robustness. The second is to communicate and agree upon the horizontal links which they will mutually advertise.

SVCCs will often traverse multiple links. The failure of one of these links will result in a failed SVCC. Such a failure may only represent a small change in the connectivity between the two LGNs. If a link internal to one of the two represented Peer Groups fails, it may represent no change in connectivity at all. Therefore, maintaining the state of the RCC between the LGNs and maintaining the states of the links between the LGNs must be independent functions in order to ensure the stability of the hierarchy. While the two functions are coupled into the same hello message, they are decoupled temporally, that is, separate sets of timers pertain to the two functions. Failure of the RCC monitoring function will cause the SVCC to be re-established. No change of state due to this event will occur in the state machines associated with the logical links until their own timer has expired.

5.6.3.1 SVCC-Based RCC Hello Protocol

The protocol used to verify the communications link between two LGNs is very close to the protocol between lowest-level neighbors, and uses the same packet type. However, unlike lowest-level neighbor nodes, LGN neighbors will have a single PNNI routing control channel between them. This SVCC-based RCC is used for the exchange of all PNNI routing packets between the LGN neighbors, including PTSPs and other packets used to maintain database synchronization as well as Hellos. The Hello protocol used to monitor the status of the SVCC triggers the AddPort and DropPort events in the neighboring peer state machine that control database synchronization between the LGNs. This is similar to the relationship between the Hello protocol and the neighboring peer state machines run between lowest-level neighbors. The event AddPort in the neighboring peer state machine is triggered when the Hello state machine for the SVCC reaches the state 2-Way Inside. The event DropPort in the neighboring peer state machine is triggered when the Hello state machine for the SVCC falls out of the 2-Way Inside state.

Once the SVCC is declared up by the signalling protocol, a Hello protocol instance is initiated. This protocol is essentially the same as the protocol that runs between lowest-level neighbors, with a few modifications:

1. A port value of 0xFFFFFFFF is always used in the Port ID field in Hello messages. SVCCs between LGNs do not have port IDs assigned to them. If a Hello is received in which the Port ID does not take the value of 0xFFFFFFFF, the event HelloMismatchReceived is triggered.
2. SVCC-based RCCs are always inside one peer group. The events 1-WayOutsideReceived, 2-WayOutsideReceived, CommonHierarchyReceived, and HierarchyMismatchReceived cannot be triggered for SVCCs between LGNs. Instead, if a Hello is received in which the Peer Group ID is not the same as this node's peer group ID, the event HelloMismatchReceived is triggered.
3. The node ID in received Hellos must be equal to the value in the corresponding uplink PTSE. If it is not equal the event HelloMismatchReceived is triggered.

   For the LGN that is the calling party for the SVCC-based RCC, that uplink PTSE is necessarily received before the SVCC is initiated. For the LGN that is the called party, there is a race condition between the arrival of the call setup from the neighbor and the uplink PTSE. If the called-party LGN receives a SETUP from a node which it has yet to recognize as a neighbor, the called-party LGN must accept the call, but ignore any hellos until an uplink PTSE is received indicating that node as a neighbor. The binding between the uplink PTSE and the SVCC-based RCC is the node ID in Hellos received over the SVCC-based RCC and the Upnode ID in the uplink PTSE.

4. An SVCIntegrityTimer is set in the Down, Attempt and One-way states. If the timer expires, the SVCC-based RCC is declared down and the SVCC is released.
5. For the LGN that is the called party, a HelloMismatchReceived event is handled by returning to the ATTEMPT state. For the LGN that is the calling party, a HelloMismatchReceived event is handled by releasing the SVCC and re-establishing it, as described in Section 5.5.6. The situation should also be logged and trapped to network management.
6. Failure of the SVCC indicated from lower levels (ATM, PHY, Signalling) is treated as a LinkDown Event. Procedures to re-establish the SVCC are followed, as described in Section 5.5.6.

5.6.3.1.1 The SVCC-Based RCC Hello Data Structure

There is a single hello data structure for each SVCC-based RCC to a neighboring node.

Each instance of this data structure consists of the following items:

State
: The operational status of a link. This is described in more detail in Section 5.6.2.1.2. Note that only the inside states will apply.

Remote Node ID
: The Node ID of the neighbor node on the other end of the link.

Remote Port ID
: The neighbor's port ID for this link. When the Remote Port ID has not been received, its value is zero.

HelloInterval
: The amount of time, in seconds, between Hellos that the node sends out over this link in the absence of event-triggered Hellos.

Hello Timer
: An interval timer that fires every HelloInterval seconds. Whenever the timer fires, the node transmits a Hello over this link.

InactivityFactor
: The amount of time, in multiples of the HelloInterval declared by the neighbor in its Hellos, before the node will consider the link down, if the neighbor's Hellos cease to arrive.

Inactivity Timer
: A single shot timer whose expiration indicates that no Hellos have been received from this neighbor recently. The initial value of the Inactivity Timer must be set to the Inactivity Factor times the Hello Interval from the most recent Hello packet received from the neighbor node.

SVCIntegrityTime
: The amount of time in seconds this node will wait for an SVCC-based RCC to reach 2-Way Inside state before releasing it.

SVCIntegrityTimer
: The timer used to determine when to consider the SVCC unusable and therefore release it.

HorizontalLinkInactivityTime
: The amount of time in seconds a node will continue to advertise a horizontal link for which it has not received and processed the LGN horizontal link IG.

Horizontal Link Inactivity Timer
: A single shot timer whose expiration indicates that no LGN horizontal links IGs have been received from this neighbor recently.

Version
: The current version of the Hello protocol being used for communication with this neighbor. If no acceptable version number has been derived, this field will be zero.

5.6.3.1.2 SVCIntegrityTimer

An SVCC-based RCC between LGNs is established by the LGN with the higher node ID.

At the calling node, the SVCIntegrityTimer is set 1) when the SVCC establishment is initiated, and 2) whenever the state machine enters the Attempt state or the OneWay state and the timer is not already running. The timer is disabled in the TwoWay and Down states. Expiration of the timer causes a return to the Down state. Upon entering the Down state the SVCC is released and normal procedures for re-establishment are followed. The SVCIntegrityTimer is started with the value SVCCallingIntegrityTime.

At the called node, the SVCIntegrityTimer is set when a setup message is received from a LGN neighbor or whenever the state machine enters the Attempt state or the OneWay state and the timer is not already running. The timer is disabled in the TwoWay state. Expiration of the timer causes a return to the Down state and the SVCC is released. The SVCIntegrityTimer is started with the value SVCCalledIntegrityTime.

5.6.3.1.3 Loss of Neighbor Relationship

If the neighbor relationship is broken, i.e. all PTSEs describing uplinks to the LGN neighbor have been deleted, then the following procedures are invoked. Note that if the neighbor relationship is truly broken, then the SVCC should get released by the lower layers. It is possible that uplinks can temporarily disappear due to a lower level peer group partition (which does not cause a partition at this level) or a new node is elected PGL.

In this event the node returns to the attempt state. Note that this will cause the SVCIntegrityTimer to be started.

5.6.3.2 LGN Horizontal Link Hello Protocol

The protocol for determining the state of horizontal links between Logical Group Nodes is also based upon the Hello protocol. The states of all horizontal links to an LGN neighbor are determined from the information in a single LGN Horizontal Link Extension information group included in the Hellos sent over the SVCC-based RCC. The LGN Horizontal Link Extension information group is present in all Hellos transmitted to the neighboring peer LGN. Each time a Hello is sent to the neighboring peer, the LGN Horizontal Link Extension information group shall contain an entry for each horizontal link to the neighboring peer node in any state other than Down. For each horizontal link the Aggregation Token, Local Port ID, and Remote Port ID are included. Each distinct aggregation token value represents a distinct horizontal link with its own independent state machine. The information group is present in all Hellos transmitted.

On receipt of Hellos received from the neighboring peer LGN, the LGN Horizontal Link Extension information group is only processed if the SVCC-based RCC Hello State Machine is in 2-Way Inside and the corresponding neighboring peer state machine is in Full state.

An LGN horizontal link is advertised in PTSEs originated by this node if and only if the LGN horizontal link hello state is 2-Way. This is done instead of tightly coupling the advertisement of horizontal links to the neighboring peer state machine as done for physical links and VPCs. This avoids unnecessarily disturbing the status of horizontal links due to temporary and recoverable SVCC failures.

When an uplink PTSE arrives including a new aggregation token value, a logical port ID is assigned and a state machine created in the Down state. The AddInducingLink event is triggered, and the state machine transitions to the Attempt state. As a result of this, subsequent instances of the LGN Horizontal Link Extension information group will contain an entry for this token, the assigned Local Port ID, and a value of zero for the Remote Port ID. The AddInducingLink event is also triggered when a new inducing uplink is seen for an existing Aggregation Token value.

When an uplink for a particular aggregation token value is removed from the uplink PTSE in which it appeared or when the uplink PTSE is deleted, either the DropInducingLink or the DropLastInducingLink event, as appropriate, is triggered. For the DropLastInducingLink event, the aggregation token value and associated port ids are removed from the information group, and the state machine remains in the down state (or is deleted) until a new PTSE with that aggregation token value arrives.

A received aggregation token value in an LGN Horizontal Link Extenstion IG for which no state machine exists is ignored.

The absence of an aggregation token in the received LGN Horizontal Link Extension information group forces the state machine associated with that aggregation token to the Attempt state. This requires that the information group must be checked for the presence of all expected Token values.

Upon the loss of an SVCC-based RCC, for reasons other than a release with cause number 53 "call cleared due to change in PGL", the status of the associated horizontal links are controlled by their own state machines. That is, each horizontal link remains up until the Horizontal Link Inactivity Timer expires, or until the last uplink containing this aggregation token is removed (i.e., the event DropLastInducingLink occurs). The Horizontal Link Inactivity Timer for horizontal links is started with the value HorizontalLinkInactivityTime. Since all horizontal links are reported in every Hello packet, a single LGN Horizontal Link Inactivity Timer is used for all the links to one neighbor.

In the event that the SVCC has been released with cause number 53 "call cleared due to change in PGL", the horizontal link FSM is triggered with the event BadNeighbor.

5.6.3.2.1 The LGN Horizontal Link Hello Data Structure

There is an LGN horizontal link hello data structure for each horizontal link to a neighboring node.

Each LGN horizontal link hello data structure consists of the following items:

Aggregation Token
: The aggregation token associated with this LGN horizontal link.

State
: The operational status of the horizontal link.

Port ID
: A number assigned by the node that identifies which horizontal link is described by this horizontal link hello data structure.

Remote Node ID
: The Node ID of the neighbor node on the other end of the horizontal link.

Remote Port ID
: The neighbor's port ID for this link. The Remote Port ID is learned when the LGN Horizontal Link Extension IG is received from the neighbor. When the Remote Port ID is not known, its value must be set to zero.

Inducing Uplinks List
: The list of inducing uplinks for this horizontal link. Each uplink is identified by the Node ID of the border node in the child peer group, and the Port ID of the inducing uplink.

5.6.3.2.2 LGN Horizontal Link States

The states that the LGN Horizontal Link Hello FSM may attain are described in this section.

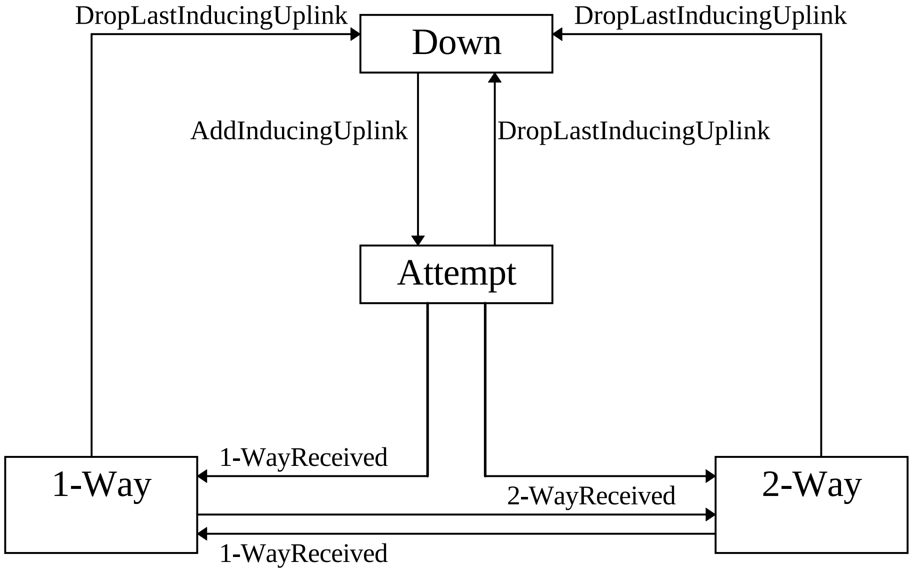


Figure 5-5: LGN Horizontal Link Hello FSM.

Down

This state indicates that no uplink PTSEs have been received including an uplink to the neighboring peer LGN with the same aggregation token value as that indicated in the LGN horizontal link hello data structure.

Attempt

This state indicates that, although at least one uplink PTSE has been received including an uplink to the neighboring peer LGN with the same aggregation token value as that indicated in the LGN horizontal link hello data structure, no appropriate confirmation from the neighboring peer exists.

1-Way

In this state, Hellos have recently been received from the neighbor including an entry for this horizontal link in the LGN Horizontal Link Extension IG, but the remote port ID was set to zero.

2-Way

In this state, Hellos have recently been received from the neighbor including an entry for this horizontal link in the LGN Horizontal Link Extension IG containing the correct remote port ID. When this state is reached, it indicates that both neighboring peer LGNs know the port IDs on both sides of the horizontal link. In this state, the horizontal link is advertised in PTSEs.

5.6.3.2.3 LGN Horizontal Link Events

AddInducingLink

Either:

(i) An uplink PTSE has arrived including a new inducing uplink to the neighboring peer node.

(ii) Connectivity has been re-established to a border node advertising an inducing uplink(s). Note that such a connectivity recovery may affect several uplinks and in turn may affect several horizontal link FSMs, or

(iii) A directly attached outside link to a descendant node of a neighboring peer LGN has reached the 2-WayOutside state carrying the same aggregation token value as in the LGN horizontal link hello data structure.

In any of these three cases, attempt to locate the LGN horizontal link hello data structure that corresponds to the aggregation token for the given inducing link. If not found, then initialize one as follows: The State is set to Down. The Aggregation Token is set to the value associated with the inducing link. A unique Port ID value is selected and its value placed into the PortID field. The Remote Node ID is set to the peer's Node ID. The Remote Port ID is set to Null. The Inducing Links List is set to empty.

1-WayReceived

A Hello has been received including an entry for the Aggregation Token in the LGN Horizontal Link Extension IG in which the remote port ID is set to zero. Additionally, if the Remote Port ID field in the LGN horizontal link hello data structure is not equal to zero, it must match the port ID in the received Hello.

2-WayReceived

A Hello has been received including an entry for the Aggregation Token in the LGN Horizontal Link Extension IG in which the remote port ID correctly identifies the Port ID in the LGN horizontal link hello data structure. Additionally, if the Remote Port ID field in the LGN horizontal link hello data structure is not equal to zero, it must match the port ID in the received Hello.

HelloMismatchReceived

A Hello has been received for which either:

(i) A HelloMismatch event is generated in the corresponding SVCC-based RCC Hello protocol, or
(ii) the Aggregation Token in the LGN horizontal link hello data structure does not appear in the LGN Horizontal Link Extension IG, or
(iii) the remote port ID in the entry for the Aggregation Token in the received LGN Horizontal Link Extension IG is different from the port ID in the LGN horizontal link hello data structure, and is not set to zero, or
(iv) the remote port ID in the LGN horizontal link hello data structure is set and is different from the received port ID in the entry for the Aggregation Token in the LGN Horizontal Link Extension IG.

HorizontalLinkInactivityTimerExpired

The Horizontal Link Inactivity Timer has expired. This means that no Hellos have been received recently from the neighbor in which the LGN Horizontal Link Extension IG could be processed.

BadNeighbor

The corresponding SVCC-based RCC has been released with cause number 53 "call cleared due to change in PGL", or a DSMismatch or BadPTSERequest has been generated in the corresponding neighboring peer FSM.

DropInducingLink

Either

(i) An inducing uplink for a particular Aggregation Token value has been removed from the uplink PTSE in which it appeared, and it was not the last inducing uplink for that particular Aggregation Token value, or
(ii) the uplink PTSE containing an inducing uplink for a particular aggregation token value has been deleted (i.e., has timed out or has been flushed), and there still remains at least one inducing uplink for that particular Aggregation Token value, or

(iii) connectivity to the border node in the child peer group that is advertising the inducing uplink(s) has been lost, and it was not the last inducing uplink for that particular Aggregation Token value. Note that such a connectivity loss may affect several uplinks and in turn may affect several horizontal link FSMs, or

(iv) A directly attached outside link to a descendant node of a neighboring peer LGN carrying the same aggregation token value as in the LGN horizontal link hello data structure has fallen out of the 2-WayOutside state, and it was not the last inducing outside link for that particular Aggregation Token value.

DropLastInducingLink

Either

(i) the last inducing uplink for a particular Aggregation Token value has been removed from the uplink PTSE in which it appeared, or

(ii) the uplink PTSE containing the last inducing uplink for a particular aggregation token value has been deleted (i.e., has timed out or has been flushed), or

(iii) connectivity to the border node in the child peer group that is advertising the last inducing uplink(s) has been lost. Note that such a connectivity loss may affect several uplinks and in turn may affect several horizontal link FSMs, or

(iv) the last directly attached outside link to a descendant node of a neighboring peer LGN carrying the same aggregation token value as in the LGN horizontal link hello data structure has fallen out of the 2-WayOutside state.

5.6.3.2.4 Description of The Horizontal Link Hello State Machine

The finite state machine is a two dimensional table with States across the top of the table and Events down the left side. Each pairing of event and state crosses at a "cell" in the table. The cell shows what state transition should occur and the action to take. For example, for the event and state pair of "AddInducingLink" and "Down" the cell reads "Hlhp1, Attempt". "Attempt" is the new state and "Hlhp1" is the Action to be taken. The actions definitions can be found following table.

Table 5-11: Horizontal Link Hello FSM

	States			
Events	Down	Attempt	1-Way	2-Way
AddInducing-Uplink	Hlhp10, Attempt	Hlhp11, Attempt	Hlhp11, 1Way	Hlhp12, 2Way
1-WayReceived	Hlhp0, Down	Hlhp1, 1Way	Hlhp0, 1Way	Hlhp6, 1Way
2-WayReceived	Hlhp0, Down	Hlhp2, 2Way	Hlhp3, 2Way	Hlhp0, 2Way
HelloMismatch Received	Hlhp0, Down	Hlhp0, Attempt	Hlhp4, Attempt	Hlhp5, Attempt
HorizontalLink-Inactivity-TimerExpired	Hlhp0, Down	Hlhp0, Attempt	Hlhp4, Attempt	Hlhp5, Attempt
BadNeighbor	Hlhp0, Down	Hlhp0, Attempt	Hlhp4, Attempt	Hlhp5, Attempt
DropInducing-Uplink	FSM_Err	Hlhp13, Attempt	Hlhp13, 1Way	Hlhp14, 2Way
DropLast-InducingUplink	FSM_Err	Hlhp13, Down	Hlhp16, Down	Hlhp15, Down

FSM_ERR Should not occur.

Hlhp0

Action: Do nothing.

Hlhp1

Action: Set the Remote Port ID in the LGN horizontal link hello data structure to the Port ID listed in the entry for the Aggregation Token in the received LGN Horizontal Link Extension IG.

Hlhp2

Action: Set the Remote Port ID in the LGN horizontal link hello data structure to the Port ID listed in the entry for the Aggregation Token in the received LGN Horizontal Link Extension IG. Trigger an advertisement of the horizontal link in a new instance of a horizontal link PTSE originated by this node, provided the corresponding Neighboring Peer State machine is in state Full.

Hlhp3

Action: Trigger an advertisement of the horizontal link in a new instance of a horizontal link PTSE originated by this node, provided the corresponding Neighboring Peer State machine is in state Full.

Hlhp4

Action: The remote Port ID in the LGN horizontal link hello data structure is cleared.

Hlhp5

Action: The remote port ID in the LGN horizontal link hello data structure is cleared. The horizontal link must be removed from the PTSE originated by this node in which it has been advertised.

Hlhp6

Action: The horizontal link must be removed from the PTSE originated by this node in which it has been advertised.

Hlhp10

Action: Add the inducing uplink (identified by the Node ID of the border node in the child peer group and the Port ID for the inducing uplink) to the Inducing Uplink List.

Hlhp11

Action: Add the inducing uplink (identified by the Node ID of the border node in the child peer group and the Port ID for the inducing uplink) to the Inducing Uplink List in the LGN horizontal link hello data structure.

Hlhp12

Action: Add the inducing uplink (identified by the Node ID of the border node in the child peer group and the Port ID for the inducing uplink) to the Inducing Uplink List in the LGN horizontal link hello data structure. If the addition of the inducing uplink causes a significant change in the topology state parameters for the aggregated horizontal link, originate a new instance of the horizontal link PTSE.

Hlhp13

Action: Delete the inducing uplink (identified by the Node ID of the border node in the child peer group and the Port ID for the inducing uplink) from the Inducing Uplink List in the LGN horizontal link hello data structure.

Hlhp14

Action: Delete the inducing uplink (identified by the Node ID of the border node in the child peer group and the Port ID for the inducing uplink) from the Inducing Uplink List in the LGN horizontal link hello data structure. If the deletion of the inducing uplink causes a significant change in the topology state parameters for the aggregated horizontal link, originate a new instance of the horizontal link PTSE.

Hlhp15

Action: The remote port ID in the LGN horizontal link hello data structure is cleared. Delete the inducing uplink (identified by the Node ID of the border node in the child peer group and the Port ID for the inducing uplink) from the Inducing Uplink List in the LGN horizontal link hello data structure. Originate a new instance of the horizontal link PTSE that does not include any entry for this horizontal link, or flush the PTSE if there is nothing else in the PTSE.

Hlhp16

Action: The remote port ID in the LGN horizontal link hello data structure is cleared. Delete the inducing uplink (identified by the Node ID of the border node in the child peer group and the Port ID for the inducing uplink) from the Inducing Uplink List in the LGN horizontal link hello data structure.

5.6.3.3 Overall Procedures

A single Hello timer exists per SVCC-based RCC to govern when Hellos are sent. Any event which requires a Hello to be sent resets this timer. Additionally, there are two inactivity timers. The Inactivity timer associated with the SVCC operates exactly as in the normal hello protocol. The Horizontal Link Inactivity Timer is reset each time an LGN Horizontal Link Extension IG is processed, since this IG describes all horizontal links to this neighbor. Note that these are only processed when the SVCC-based RCC Hello protocol is in the 2-WayInside state and the corresponding neighboring peer state machine is in Full state.

5.7 Database Synchronization

When a node first learns about the existence of a neighboring peer node (residing in the same peer group), it initiates a database exchange process in order to synchronize the topology databases of the neighboring peers. The database exchange process involves the exchange of a sequence of Database Summary packets, which contains the identifying information of all PTSEs in a node's topology database. Database Summary packets are exchanged using a lock-step mechanism, whereby one side sends a Database Summary packet and the other side responds (implicitly acknowledging the received packet) with its own Database Summary packet. At most one outstanding packet between the two neighboring peers is allowed at any one time.

When a node receives a Database Summary packet from a neighboring peer, it examines its topology database for the presence of each PTSE described in the packet. If the PTSE is not found in the topology database or if the neighboring peer has a more recent version of the PTSE, then the node must request the PTSE from this neighboring peer, or optionally from another neighboring peer whose database summary indicates that it has the most recent version of the PTSE.

For lowest-level neighboring peers, there may be multiple parallel physical links and/or VPCs between them. As described in Section 5.6, each physical link and/or VPC between the two neighboring peers will run a separate Hello state machine. However, for the purposes of database synchronization and flooding, only one conversation is held between the neighboring peers. This conversation is described by the neighboring peer state machine and the neighboring peer data structure, which includes the information required to maintain database synchronization and flooding to the neighboring peer. Whenever a link reaches the Hello state 2-WayInside, the event AddPort is triggered in the corresponding neighboring peer state machine. Similarly, when a link falls out of the Hello state 2-WayInside, the event DropPort is triggered in the corresponding neighboring peer state machine. The database exchange process commences when the event AddPort is first triggered, after the first link between the two neighboring peers comes up. When the DropPort event for the last link between the neighboring peers occurs, the neighboring peer state machine will internally generate the DropPortLast event causing all state information for the neighboring peer to be cleared.

When PTSPs, PTSE Acknowledgment packets, Database Summary packets, or PTSE Request packets are transmitted, any of the links between the neighboring peers that is in the Hello state 2-WayInside may be used. Successive packets may be sent on different links, without any harmful effects on the distribution and maintenance of PNNI routing information. Links between lowest-level neighboring peers may only be advertised in PTSEs when the neighboring peer state machine is in Full state. For the case of neighboring lowest-level peers connected by physical links and VPCs, changes into or out of the Full state will cause new instances of one or more of this node's PTSEs to be originated or flushed.

Between neighboring peer logical group nodes, only the SVCC-based RCC is used for the exchange of PNNI routing packets. Similarly to the case of lowest-level neighboring peers, the neighboring peer state machine is coupled to the Hello state machine of the RCC. Note the Hello states of the horizontal links between the LGNs do not affect the neighboring peer state. When the Hello state of the RCC reaches 2-WayInside, the event AddPort is triggered in the neighboring peer state machine and the database exchange process commences. When the Hello state of the RCC falls out of the state 2-WayInside, the event DropPort is triggered in the neighboring peer state machine, causing it to transition from Full state to NPDown state.

In the case where neighbors communicate via an SVCC-based RCC, the neighboring peer state machine does not directly affect origination of horizontal link PTSEs. Rather, it affects the origination of horizontal link PTSEs indirectly through the Horizontal Link Hello protocol (see Section 5.6.3.2). In addition, when first originating a PTSE for a given link, the associated LGN Hello machine must be in 2-WayInside and the peer data structure must be in Full state.

5.7.1 The Neighboring Peer Data Structure

Each node has a single neighboring peer data structure for each neighboring peer node regardless of the number of links between those nodes. Neighboring peer conversations in states other than NPDown are called adjacencies. The neighboring peer data structure contains all information pertinent to a forming or formed adjacency between two neighboring peers. Neighbor nodes belonging to different peer groups will not form an adjacency.

State
: The state of the neighboring peer FSM. This is described in more detail in Section 5.7.2.

Remote Node ID
: The node ID used to identify the neighboring peer node.

Port ID List
: The Port ID List is only used in the case of lowest-level neighboring peers, which are connected by physical links and/or VPCs. The Port ID List is a list of those links to the neighboring peer that are in the state 2-WayInside. When PTSPs, PTSE acknowledgment packets, database summary packets, or PTSE request packets are transmitted or retransmitted to the neighboring peer, any of the links specified in this list may be used.

Master/Slave
: When the two neighboring peers are exchanging databases, they form a master/slave relationship. This relationship is relevant only for initial topology database exchange. The master sends the first Database Summary packet and chooses the initial DS Sequence Number. Retransmissions of Database Summary packets are controlled by the master. The slave can only respond to the master's Database Summary packets. The master/slave relationship is determined in the Negotiating state.

DS Sequence Number
: An unsigned 32-bit number identifying individual database summary packets. When the Negotiating state is first entered, the DS sequence number should be set to a value not previously seen by the neighboring peer but not too large to safely avoid sequence number wrapping. One possible scheme is to use the lower 24 bits of the machine's time of day counter. The DS sequence number is then incremented by the master with each new Database Summary packet sent. The slave's DS sequence number indicates the last packet received from the master.

Peer Retransmission List
: The list of PTSEs that have been flooded but not acknowledged by the neighboring peer. These will be retransmitted periodically until they are acknowledged, or until the neighboring peer state machine is taken down.
: Associated with each entry in this list is a PTSE Retransmission Timer. This is an interval timer that fires after PTSERetransmissionInterval seconds. The timer is stopped when an acknowledgment is received that corresponds to that PTSE.

PTSERetransmissionInterval
: Each unacknowledged PTSE is retransmitted every PTSERetransmissionInterval seconds.

Peer Delayed Acks List
: The list of PTSEs for which delayed acknowledgments will be sent to a neighboring peer. Every PeerDelayedAckInterval seconds acknowledgment packets are transmitted to the neighboring peer that contain the PTSE identifying information for all entries on the Peer Delayed Acks List, and the list is cleared.

PeerDelayedAckInterval
: This is the time interval between consecutive checks of the Peer Delayed Acks List.

Peer Delayed Ack Timer
: When this timer expires, any unacknowledged PTSEs in the Peer Delayed Acks List are bundled in an acknowledgment packet and sent to the neighboring peer.

PTSE request list

The list of PTSEs that need to be requested in order to synchronize the two neighboring peers' topology databases. This list is created as Database Summary packets are received. PTSE Request packets are used to request each PTSE on this list from this neighboring peer, or optionally from any other neighboring peer known to possess the missing PTSE. The list is depleted as appropriate PTSEs are received.

DSRxmtInterval
: The amount of time, in seconds, a node waits before it sends the previous Database Summary packet again.

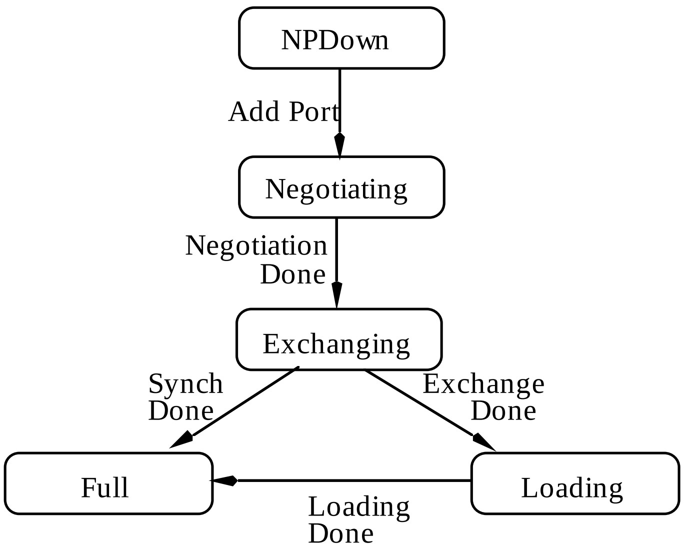
DS Rxmt Timer
: An interval timer that fires after DSRxmtInterval seconds. The timer is stopped when the node receives a correct Database Summary packet.

RequestRxmtInterval
: The amount of time, in seconds, before a node sends a new PTSE Request Packet requesting PTSEs of the last PTSE Request Packet that have not been received yet.

Request Rxmt Timer
: An interval timer that fires after RequestRxmtInterval seconds. The timer is stopped when all of the PTSEs requested in the last PTSE Request Packet have been received.

5.7.2 Neighboring Peer States

The neighboring peer state machine describes the state of database synchronization and flooding ongoing with the neighboring peer. Figure 5-6 shows a diagram of the possible state changes. The arcs of the graph are labeled with the events that cause each state change. The events are described in Section 5.7.3. For a detailed description of the neighboring peer state changes and the actions involved with each state change, see Section 5.7.4.



In addition to the state transitions pictured,

Event DSMismatch forces Negotiating state,
Event BadPTSERequest forces Negotiating state,
Event DropPort causes no state change,
Event DropPortLast forces NPDown state.

Figure 5-6: Neighboring Peer State Changes (Database Synchronization)

NPDown
: The initial state of a neighboring peer FSM. This state indicates that there are no active links (i.e., in Hello state 2-WayInside) to the neighboring peer. In this state, there are no adjacencies associated with the neighboring peer.

Negotiating
: The first step in creating an adjacency between the two neighboring peers. The goal of this step is to decide which node is the master, and to decide upon the initial DS sequence number.

Exchanging
: In this state the node describes its topology database by sending Database Summary packets to the neighboring peer. Following as a result of processing Database Summary packets, required PTSEs can be requested.

Loading
: In this state, a full sequence of Database Summary packets has been exchanged with the neighboring peer, and the required PTSEs are requested and at least one has not yet been received.

Full
In this state, this node has received all PTSEs known to be available from the neighboring peer. Links to the neighboring peer can now be advertised in PTSEs.

5.7.3 Events causing neighboring peer state changes

State changes can be brought about by a number of events. These events are triggered by procedures associated with database synchronization between the two neighboring peers, or by actions of the Hello state machines for the associated links. The events are shown in the labels of the arcs in Figure 5-6. A detailed explanation of the state changes and actions taken after an event occurs is given in Section 5.7.4. The events are defined as follows:

AddPort
A Hello state machine for a link to the neighboring peer has reached 2-WayInside state.

NegotiationDone
The Master/Slave relationship has been negotiated, and the initial DS sequence number has been agreed upon. For more information on the generation of this event, consult Section 5.7.6.

ExchangeDone
The neighboring peer's last Database Summary packet has been received, this node's last Database Summary packet has been sent, and the PTSE request list is not empty. The node now knows which PTSEs need to be requested. For more information on the generation of this event, consult Section 5.7.6.

SynchDone
The neighboring peer's last Database Summary packet has been received, this node's last Database Summary packet has been sent, and the PTSE request list is empty.

LoadingDone
The last PTSE on the PTSE Request List has been received.

DSMismatch
A Database Summary packet has been received that:
a) has an unexpected DS sequence number, or
b) unexpectedly has the Initialize bit set, or
c) has an unexpected setting of the Master bit.
Any of these conditions indicates that an error has occurred in database synchronization.

BadPTSERequest
A PTSE Request has been received for a PTSE not contained in the database, or a received PTSE is less recent than the instance on the PTSE Request List. This indicates an error in database synchronization.

DropPort
A Hello state machine for a link to the neighboring peer has exited the 2-WayInside state.

DropPortLast
In processing a DropPort event, it was determined that all ports to this neighbor have been dropped.

5.7.4 The Neighboring Peer State Machine

The finite state machine is a two dimensional table with States across the top of the table and the Events down the left side. Each pairing of event and state cross at a "cell" in the table. The cell shows what state transition should occur and the action to take. For example, for the event and state pair of "AddPort" and "NPDown" the cell reads "Ds1, Negotiating". "Negotiating" is the new state and "Ds1" is the Action to be taken. The actions can be found following table.

Table 5-12: Neighboring Peer FSM

	States				
Events	**NPDown**	**Negotiating**	**Exchanging**	**Loading**	**Full**
Add Port	Ds1 Negotiating	Ds7 Negotiating	Ds7 Exchanging	Ds7 Loading	Ds8 Full
Negotiation Done	FSM_ERR	Ds2 Exchanging	FSM_ERR	FSM_ERR	FSM_ERR
ExchangeDone	FSM_ERR	FSM_ERR	Ds3 Loading	FSM_ERR	FSM_ERR
SynchDone	FSM_ERR	FSM_ERR	Ds4 Full	FSM_ERR	FSM_ERR
Loading Done	FSM_ERR	FSM_ERR	FSM_ERR	Ds4 Full	FSM_ERR
DS Mismatch	FSM_ERR	FSM_ERR	Ds5 Negotiating	Ds5 Negotiating	Ds6 Negotiating
BadPTSE Request	FSM_ERR	FSM_ERR	Ds5 Negotiating	Ds5 Negotiating	Ds6 Negotiating
DropPort	FSM_ERR	Ds9 Negotiating	Ds9 Exchanging	Ds9 Loading	Ds9 Full
DropPort Last	FSM_ERR	Ds10 NPDown	Ds10 NPDown	Ds10 NPDown	Ds10 NPDown

FSM_ERR

Action: Protocol error, should not occur.

Ds0

Action: Do nothing

Ds1

Action: For the case of lowest-level nodes, which are connected by physical links and/or VPCs, the port ID is added to the Port ID List in the neighboring peer data structure.

Upon entering this state, the node increments the DS sequence number for this neighboring peer. If this is the first time that an adjacency has been attempted, the DS sequence number should be assigned some unique value (like the time of day clock). It then declares itself master (sets the Master bit to one), and starts sending Database Summary packets, with the Initialize, More, and Master bits set. No PTSE Summaries are included in this packet. This Database Summary packet is retransmitted at intervals of DSRxmtInterval until the next state is entered (See Section 5.7.5).

Ds2

Action: The node must begin sending a summary of the contents of its topology database to the neighboring peer in Database Summary packets. The topology database consists of the PTSEs either originated or received by this node, at the level of this node’s peer group or at a higher level. Each Database Summary packet has a DS sequence number, and is implicitly acknowledged. Only one Database Summary packet is allowed outstanding at any one time. For more detail on the sending and receiving of Database Summary packets, see Sections 5.7.5 and 5.7.6.

Ds3

Action: Stop the DS Rxmt Timer if not previously stopped. Start (or continue) sending PTSE Request packets to this neighboring peer and/or optionally to other neighboring peers (see Section 5.7.7). Each PTSE Request packet asks for some of the neighboring peer's more recent PTSEs (which were discovered but not yet received in the Exchanging state). These PTSEs are listed in the PTSE Request list in the neighboring peer data structure.

Ds4

Action: Stop the DS Rxmt Timer if not previously stopped. The databases are now synchronized. For the case of lowest-level neighbor nodes, all links to the neighbor must now be advertised in PTSEs.

Ds5

Action: The Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer are stopped if not previously stopped. The Peer Retransmission List, Peer Delayed Acks List, PTSE Request List and all related timers are cleared. The exchange of database summaries must start over again. The node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, and Master bits set. No PTSE summaries are included in this packet. The DS Rxmt Timer is started and the Database Summary packet is retransmitted each DSRxmtInterval time.

Ds6

Action: Same as Ds5, except if there are any PTSEs advertising links to that neighbor, those PTSEs must be modified to remove the links. Such PTSEs must be re-originated or if necessary, flushed.

Ds7

Action: For the case of lowest-level neighboring peers, which are connected by physical links and/or VPCs, the port ID is added to the Port ID list in the neighboring peer data structure.

Ds8

Action: Same as Ds7 with the additional requirement that this action will cause a link to the neighboring peer to be added, causing a new instance of a PTSE to be originated.

Ds9

Action: The link is removed from the Port ID list in the corresponding neighboring peer data structure. The action will cause a link to the neighboring peer to be removed. If there is a PTSE advertising that link, a new instance of the affected PTSE must be originated. If this was the last active link to this neighbor, generate the DropPortLast event.

Ds10

Action: The Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer are stopped, if not previously stopped. The Peer Retransmission List, Peer Delayed Acks List, PTSE Request List and all related timers are cleared.

5.7.5 **Sending Database Summary Packets**

This section describes how Database Summary packets are sent to a neighboring peer.

Only one Database Summary packet is allowed outstanding at any one time.

The sending of Database Summary packets depends on the neighboring peer state.

In the Negotiating state, the node sends empty Database Summary packets, with the Initialize, More and Master bits set. When sending such Database Summary packets, the DS Rxmt Timer must be restarted. These packets are retransmitted every DSRxmtInterval seconds, when the DS Rxmt Timer fires.

In the Exchanging state, including when sending the Database Summary packet in response to the event NegotiationDone, the Database Summary packets contain summaries of the topology state information contained in the node's database. In the case of Logical Group Nodes, those portions of the topology database that were originated or received at the level of the Logical Group Node or at higher levels are included in the database summary (lower level PTSEs may belong to the switching system's topology database for one or more of its incarnations as a lower level node, but do not belong to the logical group node's topology database). The PTSP and PTSE header information of each such PTSE is listed in one of the node's Database Summary packets. PTSEs for which new instances are received after the Exchanging state has been entered need not be included in any Database Summary packet, since they will be handled by the normal flooding procedures. It is recommended but not required that each PTSE be included at most once in the entire sequence of Database Summary packets sent to the neighboring peer.

In the Exchanging state, the determination of when to send a Database Summary packet depends on whether the node is master or slave. When a new Database Summary packet is to be sent, the packet's DS sequence number is set as described in Section 5.7.6 and a new set of PTSEs from the node's topology database is described. Each item is considered to have been received by the neighboring peer when the Database Summary packet in which it was included is acknowledged. Note that the More bit is set asymmetrically, with different rules used depending on whether the node is master or slave:

Master

Database Summary packets are sent when either (i) the slave acknowledges the previous Database Summary packet by echoing the DS sequence number, or (ii) DSRxmtInterval seconds elapse without an acknowledgment, in which case the previous Database Summary packet is retransmitted. The DS Rxmt Timer must be restarted whenever a Database Summary packet is transmitted. If the node has already sent its entire sequence of Database Summary packets, or if this packet includes the last portions of the database summary to be sent to the slave, then the More bit must be set to zero.

Slave

Database Summary packets are sent only in response to Database Summary packets received from the master. If the packet received from the master is new, a new Database Summary packet is sent; otherwise the previous Database Summary packet is retransmitted. If the node has already sent its entire sequence of Database Summary packets (i.e., the contents of this Database Summary packet are empty), then the More bit must be set to zero.

In states Loading and Full the slave must resend its last Database Summary packet in response to duplicate Database Summary packets received from the master. Note that in the Loading or Full state, the last packet that the slave had sent must have been empty, with the Initialize, More, and Master bits set to zero and with the same DS sequence number as in the current neighboring peer data structure.

5.7.6 Receiving Database Summary Packets

This section explains the detailed processing of a received Database Summary packet. The incoming Database Summary packet is associated with a neighboring peer by the interface over which it was received. Each Database Summary packet has a DS sequence number, and is implicitly acknowledged. The further processing of the Database Summary packet depends on the state of the neighboring peer data structure associated with the Remote Node ID. If the neighboring peer state is NPDown the packet must be ignored.

Otherwise, if the state is:

Negotiating

If the received packet matches one of the following cases, then the neighboring peer state machine must be executed with the event NegotiationDone (causing the state to transition to Exchanging) and the packet must be accepted as next in sequence and processed further. Otherwise, the packet must be ignored.

- The Initialize, More and Master bits are one, the contents of the packet are empty, and the neighboring peer's Node ID is larger than this node's own node ID. In this case this node is now a Slave. Upon generating the event NegotiationDone, the slave must take the following actions:

  - Stop the DS Rxmt Timer,
  - Set the Master bit to zero (indicating slave), set the Initialize bit to zero, set the DS sequence number to that specified by the master, and send a Database Summary packet to the master including the first portion of this node's database summary (see Section 5.7.5).

- The Initialize and Master bits are zero, the packet's DS sequence number equals the node's own DS sequence number (indicating acknowledgment), and the neighboring peer's node ID is smaller than the node's own node ID. In this case this node is Master. Upon generating the event NegotiationDone, the master must take the following actions (the last two actions need not be taken in this order):

  - Stop the DS Rxmt Timer,
  - Process the contents of the received Database Summary packet (see the last paragraph describing actions to be taken under the Exchanging state below),

    - Increment the DS sequence number by one, send a Database Summary packet to the slave including the first portion of this node's database summary (see Section 5.7.5) and restart the DS Rxmt Timer.

Exchanging

If the state of the Master bit is inconsistent with the master/slave state of the connection, generate the event DSMismatch and stop processing the packet. Otherwise:

- If the Initialize bit is set, generate the event DSMismatch and stop processing the packet.
- If the node is master and the packet's DS sequence number equals the node's own DS sequence number (this packet is the next in sequence), the packet must be accepted and processed as follows (the last two actions need not be taken in this order):
    - Stop the DS Rxmt Timer,
    - Process the contents of the received Database Summary packet (see below),
    - In the following order:
        A) Increment the DS sequence number by one,
        B) If the node has already sent its entire sequence of Database Summary packets (i.e., the previous Database Summary packet that the node sent had the More bit set to zero), and the received packet also has the More bit set to zero, generate the event ExchangeDone if the PTSE Request List is not empty, or the event SynchDone if the PTSE Request List is empty.
        C) Otherwise, send a new Database Summary packet to the slave and restart the DS Rxmt Timer (see Section 5.7.5).
- If the node is master and the packet's DS sequence number is one less than the node's DS sequence number, the packet is a duplicate. Duplicates must be discarded by the master.
- If the node is slave and the packet's DS sequence number is one more than the node's own DS sequence number (this packet is the next in sequence), the packet must be accepted and processed as follows (in no particular order):
    - Process the contents of the received Database Summary packet (see below),
    - In the following order:
        D) Set the DS sequence number to the DS sequence number appearing in the received packet,
        E) Send a Database Summary packet to the master (see Section 5.7.5),
        F) If the received packet has the More bit set to zero, and the just transmitted Database Summary packet also had its More bit set to zero (i.e., the contents of the just transmitted Database Summary packet were empty), then generate the event ExchangeDone if the PTSE Request List is not empty, or the event SynchDone if the PTSE Request List is empty.
- If the node is slave, and the packet's DS sequence number is equal to the node's DS sequence number, the packet is a duplicate. The slave must respond to duplicates by repeating the last Database Summary packet that it had sent.
- Else, generate the event DSMismatch and stop processing the packet.

Processing the contents of a received Database Summary packet:

When the node accepts a received Database Summary packet as the next in sequence, the contents of the most recently transmitted Database Summary packet are acknowledged as having been received and the contents of the received Database Summary packet are processed as follows.

For each PTSE listed, the node looks up the PTSE in its database to see whether it also has an instance of the PTSE. If it does not, or if the database copy is less recent (see Section 5.8.2.2.3), one of the following actions is taken:

- If the listed PTSE is one of this node's self-originated PTSEs, the node must either:
    - Re-originate a newer instance of the PTSE with a larger sequence number, if the node has a valid instance of the PTSE (see Section 5.8.3.7), or
    - Flush the PTSE from the routing domain after installing it in the topology database with the remaining lifetime set to ExpiredAge (see Section 5.8.3.8).
- Otherwise, if the listed PTSE has PTSE Remaining Lifetime ExpiredAge, the PTSP and PTSE header contents in the PTSE summary must be accepted as a new or updated PTSE with empty contents. Follow the procedures for receiving a PTSE in Section 5.8.3.3 to determine whether or not the PTSE must be flooded to other neighboring peers, after installing the PTSE in the topology database.
- Otherwise, the PTSE is put on the PTSE request list, so that it can be requested from a neighboring peer (immediately or at some later time) in PTSE Request packets.

Loading or Full

In either of these states, the node has sent and received an entire sequence of Database Summary packets. The only packets received should be duplicates. Any other Database Summary packets received must generate the event DSMismatch, causing the adjacency to revert to the Negotiating state and the two neighboring peers to resynchronize their databases.

The procedures followed when receiving a Database Summary packet in the Loading or Full states are the same as those followed in the Exchanging state, except that packets accepted as the next in sequence must generate the event DSMismatch instead, and further processing of such packets must be stopped. Note that receipt of packets with an inconsistent Master bit or with the Initialize bit set to one must also generate the event DSMismatch.

5.7.7 Sending PTSE Request Packets

In states Exchanging and Loading, the PTSE request list contains a list of those PTSEs that need to be obtained in order to synchronize this node's topology database with the neighboring peer's topology database. To request these PTSEs, the node sends a PTSE Request Packet containing one or more entries from the PTSE request list. PTSE request packets are only sent in states Exchanging and Loading. A node may send a PTSE Request Packet to this neighboring peer, and/or optionally to any other neighboring peers that are in states Exchanging or Loading, and whose database summaries indicate that they have the missing PTSEs.

There must be at most one PTSE Request packet outstanding at any one time for each neighboring peer. Note that a node can control the flow of replies by choosing appropriately the number of PTSEs included in each PTSE Request packet.

Whenever a PTSE Request packet is transmitted to a particular neighboring peer, the Request Rxmt Timer in the corresponding neighboring peer data structure must be restarted. When the neighboring peer responds to these requests with the proper PTSE(s), i.e., any subset of what was requested, the received PTSEs are removed from the PTSE request list. When all of the PTSEs included in the last PTSE Request packet have been received, a new PTSE Request packet may be sent to the same neighboring peer. If not all requested PTSEs are received within RequestRxmtInterval, then a new PTSE Request packet including the missing PTSEs and/or any other PTSEs from the PTSE Request List in the corresponding neighboring peer data structure may be transmitted. The missing PTSEs may instead be requested from other neighboring peers that are in states Exchanging or Loading, and whose database summaries indicate that they have the missing PTSEs, if desired. This process continues until the PTSE request list becomes empty.

When the PTSE request list becomes empty, and the neighboring peer state is Loading (i.e., a complete sequence of Database Summary packets has been both sent to and received from the neighboring peer), the LoadingDone event is generated.

5.7.8 Receiving PTSE Request Packets

Received PTSE Request packets specify a list of PTSEs that the neighboring peer wishes to receive. PTSE Request packets must be accepted when the corresponding neighboring peer state is Exchanging, Loading, or Full. In all other states PTSE Request packets must be ignored.

For each PTSE specified in the PTSE Request packet, an instance must be looked up in the node's topology database. The requested PTSEs are then bundled into PTSPs (in a fashion of this node's choosing) and are transmitted to the neighboring peer in a timely fashion. These PTSEs must not be placed on the peer retransmission list in the corresponding neighboring peer data structure. If a PTSE cannot be found in the database, something has gone wrong with the Database Exchange process and the event BadPTSERequest must be generated.

5.8 Topology Description and Distribution

5.8.1 Topology Information

Topology information includes topology state parameters and nodal information.

Topology information is specified in such a way as to allow for future extensibility. Specifically, topology information is encoded in Type/Length/Value format. This extensibility may be used, for example, to add support for preferential resource selection and policy-related functions, and QoS and traffic contract negotiation to meet phase 2 signalling requirements.

5.8.1.1 Topology State Parameters

Topology state parameter is a generic term that refers to either a link state parameter or a nodal state parameter. Topology state parameters fall into two categories: topology metric and topology attribute. A topology metric is a topology state parameter that requires the values of the state parameters of all links and nodes along a given path to be combined to determine whether the path is acceptable and/or desirable for carrying a given connection. A topology attribute is a topology state parameter that is considered individually to determine whether a given link or node is acceptable and/or desirable for carrying a given connection.

Topology attributes can further be divided into performance/resource-related topology attributes and policy-related topology attributes. A performance/resource-related topology attribute is a topology attribute that provides a measure of the cell transfer performance or a resource constraint associated with a topology component. A policy-related topology attribute is a topology attribute that characterizes the level of conformance of a topology component to a specific policy constraint. For example, Available Cell Rate is a performance/resource-related topology attribute, whereas Restricted Transit is a policy-related topology attribute.

Table 5-13: Topology States Parameters

Topology State Parameters		
Topology Metrics	**Topology Attributes**	
	Performance/Resource Related	**Policy Related**
Cell Delay Variation (CDV)	Cell Loss Ratio for CLP=0 (CLR_0)	Restricted Transit flag
Maximum Cell Transfer Delay (maxCTD)	Cell Loss Ratio for CLP=0+1 (CLR_{0+1})	
Administrative Weight (AW)	Maximum Cell Rate (maxCR)	
	Available Cell Rate (AvCR)	
	Cell Rate Margin (CRM)	
	Variance Factor (VF)	
	Restricted Branching Flag	

5.8.1.1.1 Link State Parameters

A link state parameter provides information that captures an aspect or property of a link. Since links in an ATM network are bi-directional, a link state parameter implicitly includes the direction of the link it describes. Under the PNNI hierarchical structure, a link attached to a node can either be a horizontal link or an uplink. A horizontal link is a link between two nodes in the same peer group. An Uplink represents a connectivity between a border node in a peer group and a higher level upnode that is outside of the peer group. For horizontal links, link state parameters for the outgoing direction of the link are advertised by this node as identified by the local node ID and local port ID. Link state parameters for the incoming direction of a horizontal link are advertised by the neighboring peer node as identified by the neighboring peer's node ID and port ID associated with the link. For uplinks, link state parameters for both directions of the link are advertised by this node as identified by the local node ID and local port ID.

5.8.1.1.2 Nodal State Parameters

A nodal state parameter provides information that captures an aspect or property of a node. Nodal state parameters are used to construct the PNNI complex node representation if the Nodal Representation flag in the Nodal Information is set to one. A nodal state parameter is direction specific, and the direction is identified by a pair of input and output port IDs of the logical node of interest. The nucleus or interior reference point in the complex node representation is assigned zero as its port ID. The default "radius" in the PNNI complex node representation is identified by its input and output port IDs being both zero, and is by definition symmetric. For each nodal state parameter, only the default "radius" is required in the complex node representation, and any non-default value associated with a pair of ports or a port and the nucleus is optional. The nucleus may be interpreted as the center of a peer group when the peer group is a destination peer group, or the "mid-point" between any pair of border nodes in a peer group when the peer group is a transit peer group.

5.8.1.1.3 Resource Availability Information Group (RAIG)

The Resource Availability Information Group contains information that is used to attach values of topology state parameters to nodes, links, and reachable addresses (see Section 5.14.5).

5.8.1.1.3.1 Resource Availability Information Group Flags

These flags affect the interpretation of the topology state parameters in the RAIG in PNNI protocol messages.

The upper 5 bits of the RAIG Flags indicate which service categories the topology state parameters contained in the RAIG apply to. There is 1 bit for each defined service category. This allows one set of topology state parameters to apply to one or more service categories. If the topology state parameters for different service categories are not the same, multiple RAIGs, each carrying a set of topology state parameters, are advertised; these RAIGs are all contained within a single information group describing the entity. For each service category there is at most one set of topology state parameters that is advertised. If there are no topology state parameters advertised for a service category, that service category is not supported by that entity.

The least significant bit of these flags defines the applicability of the CLP bit to the interpretation of the topology state parameters included in this RAIG. Specifically, when this bit is 0, Table 5-16 shall be used to map the PCR and SCR in signalling to the PCR and SCR used in GCAC. When the bit is 1, Table 5-17 is used.

5.8.1.1.3.2 Cell Delay Variation (CDV)

The peak-to-peak CDV is the ((1-α) quantile of the CTD) minus the fixed CTD that could be experienced by any delivered cell on a connection during the entire connection holding time, specified in microseconds. For more details refer to the Traffic Management 4.0 specification.

CDV is a required topology metric for CBR and Real Time VBR service categories. CDV is not applicable to Non-Real Time VBR, ABR and UBR service categories.

In this version of the specification, CDV accumulation is done according to the simple method specified in the Traffic Management 4.0 specification, i.e., it is additive.

5.8.1.1.3.3 Maximum Cell Transfer Delay (maxCTD)

MaxCTD is the sum of the fixed delay component across the link or node and CDV.

MaxCTD is a required topology metric for CBR, real time VBR, and non-real time VBR service categories. MaxCTD is not applicable to UBR and ABR service categories. This is specified in microseconds, and is a metric combined by addition along the path.

In this version of the specification, maxCTD accumulation is done according to the simple method specified in the Traffic Management 4.0 specification, i.e., it is additive.

5.8.1.1.3.4 Administrative Weight (AW)

The administrative weight is a value set by the network operator. It is used to indicate the relative desirability of using a link or node for whatever reason significant to the network operator. It should however not be misused to express information contained in other topology state parameters (like AvCR or maxCR) or combinations thereof.

Administrative weight is a required topology metric for all service categories. This is a dimensionless value whose default is DefaultAdminWeight. A higher value describes a link or node which is less desirable to be used than a link with a lower value.

The administrative weight of a path is defined as the sum of the administrative weights of the links and nodes contained in the path.

5.8.1.1.3.5 Cell Loss Ratio for CLP=0 (CLR_0)

CLR_0 is the maximum cell loss ratio (CLR) objective for CLP=0 traffic. CLR is defined as the ratio of the number of cells that do not make it across the link or node to the total number of cells transmitted across the link or node.

CLR_0 is a required topology attribute for CBR, real time VBR, and non-real time VBR service categories. CLR_0 is not applicable to the ABR and UBR service categories.

5.8.1.1.3.6 Cell Loss Ratio for CLP=0+1 (CLR_{0+1})

CLR_{0+1} is the maximum cell loss ratio (CLR) objective for CLP=0+1 traffic. CLR is defined as the ratio of the number of cells that do not make it across the link or node to the total number of cells transmitted across the link or node.

CLR_{0+1} is a required topology attribute for CBR, real time VBR, and non-real time VBR service categories. CLR_{0+1} is not applicable to the ABR and UBR service categories.

5.8.1.1.3.7 Maximum Cell Rate (maxCR)

MaxCR is the maximum capacity usable by connections belonging to the specified service category.

MaxCR is a required topology attribute for ABR and UBR service categories. MaxCR is an optional topology attribute for CBR, real time VBR and non-real time VBR service categories. MaxCR is expressed in units of cells per second.

MaxCR = 0 is a distinguished value used to indicate inability to accept new connections in UBR and ABR service categories.

5.8.1.1.3.8 Available Cell Rate (AvCR)

AvCR is a measure of effective available capacity for CBR, Real Time VBR and Non-Real Time VBR service categories. For ABR service category, AvCR is a measure of capacity available for minimum cell rate (MCR) reservation.

AvCR is a required topology attribute for CBR, real time VBR, non-real time VBR and ABR service categories. AvCR is not applicable to UBR service category. AvCR is expressed in units of cells per second.

5.8.1.1.3.9 Cell Rate Margin (CRM)

CRM is a measure of the difference between the effective bandwidth allocation and the allocation for sustainable cell rate in cells per second. The CRM is an indication of the safety margin allocated above the aggregate sustainable cell rate.

CRM is an optional topology attribute for real time VBR and non-real time VBR service categories. CRM is not applicable to CBR, ABR, and UBR service categories. CRM is expressed in units of cells per second.

5.8.1.1.3.10 Variance Factor (VF)

VF is a relative measure of the square of the cell rate margin normalized by the variance of the sum of the cell rates of all existing connections (see equations 1 and 2 in Appendix B).

VF is an optional topology attribute for real time VBR and non-real time VBR service categories. VF is not applicable to CBR, ABR, and UBR service categories. VF is a dimensionless ratio.

5.8.1.2 Nodal Information

Nodal Information describes the node (not including resource availability information, and not including reachability information). It specifically includes the following:

- ATM End System Address of the node
- leadership priority
- nodal information flags
- preferred peer group leader node ID
- next higher-level binding information (included if this node is Peer Group Leader)

5.8.1.2.1 ATM End System Address of the Node

This field provides the ATM address that may be used to reach the node. Note that this is distinct from the node’s Node ID.

5.8.1.2.2 Leadership Priority

This field indicates the desirability of this node to operate as Peer Group Leader. This is used in the PGL election.

5.8.1.2.3 Nodal Information Flags

Currently five flags are defined: the “I am leader” flag, the Restricted Transit flag, the Nodal Representation flag, the Restricted Branching flag, and the Non-transit for PGL Election flag. The rest of this field is reserved for use in future versions of PNNI.

The “I am leader” flag is set to one by a node in a peer group to declare that it is the PGL for the peer group. This is used in the PGL election.

In some cases a PNNI node is not permitted to be used for transit purposes except by certain restricted sets of traffic. The Restricted Transit flag is provided in the nodal state parameters to indicate whether the node is restricted. If it is restricted, then the node may not be used for transit purposes unless there is information present elsewhere in the node state parameters that explicitly explains or overrides the restriction. The Restricted Transit flag is a policy-related topology attribute.

If a node has the Restricted Transit set to one, then that node may nonetheless be used as the first or last node of a DTL (i.e., the node may originate calls, and it may be used without any restriction to reach an address that is advertised as directly reachable via an Internal ATM Address reachable via that node). For logical group nodes, any lower level peer group that is summarized into the LGN is considered part of that LGN, and thus may be reached via that node even if the node is Restricted Transit.

The Restricted Transit flag may be used on its own to indicate that a node is non-transit. It also may be used in combination with additional information to indicate that a node may be used for transit only in restricted circumstances. For example, suppose that in the future a new PNNI information group is defined, and that that new information group can be used to determine whether the node can be used for transit. In this case the node would be marked Restricted Transit. An information group would also be included that is marked “non-mandatory” (meaning that a node that does not understand the new information group can safely ignore it). In this case, an old node (the nodes that do not understand the newly defined information group) would use the node only to reach systems directly reachable via that node. A new node would interpret the information group, and would be able to use the values of the new information group to determine whether to override the Restricted Transit flag.

The Nodal Representation bit is used by a node to declare whether it is represented using the simple node representation or the complex node representation. If it is set to one to indicate the simple node representation, then no nodal state parameters PTSEs from this node will be considered in route computation; rather, the node will be represented as a single point, and all relevant topology state parameters will be given only with the corresponding horizontal links, uplinks, exterior reachable addresses and transit networks attached to the node.

The Restricted Branching flag is used to indicate if this node can branch point-to-multipoint traffic. When this bit is set to zero, then the node can support additional branching points for point-to-multipoint calls. If set to one, the node cannot currently support additional branching points for point-to-multipoint calls. The Restricted Branching flag is a resouce-related topology attribute.

The Non-transit for PGL Election flag is set by a node when it is necessary for that node to be ignored when computing connectivity in the PGL election algorithm. This flag must be set to zero in a normally operating node. It must be set to one when the node is operating in a topology database overload state (i.e., it is currently unable to store and propagate the entire set of active PTSEs).

5.8.1.2.4 Preferred Peer Group Leader Node ID

This field specifies the node ID of the node that this node considers to be its choice for PGL. This is used in the PGL election.

5.8.1.2.5 Next Higher Level Binding Information

This field consists of an information group that is sent by PGLs in order to inform all nodes in the peer group of the higher level binding information to the parent LGN representing this peer group in the parent peer group. This is used for calculation of routes across multiple levels of the hierarchy; it is also used by border nodes during the bootstrapping of the hierarchy.

5.8.1.2.5.1 Parent Logical Group Node ID

This field contains the node ID of the parent LGN.

5.8.1.2.5.2 Parent Logical Group Node's ATM End System Address

This field contains the ATM End System Address of the parent LGN.

5.8.1.2.5.3 Parent Peer Group ID

This field contains the peer group ID of the peer group in which the parent LGN is instantiated.

5.8.1.2.5.4 Node ID of PGL of Parent Peer Group

This field contains the node ID of the node elected as peer group leader of the parent peer group. If the parent LGN is not aware of a peer group leader having been elected in the parent peer group, then this field is set to zero. The parent LGN determines that it should advertise the node ID of the PGL of the parent peer group when the preferred PGL of the parent LGN advertises in that node's information group that it is PGL by setting the "I am leader" bit.

5.8.1.3 Reachability Information

Reachability Information is topology information that binds reachable addresses to nodes within the PNNI hierarchy.

Internal and exterior reachability information is logically distinguished based on its source. Exterior reachability is derived from other protocol exchanges outside this PNNI routing domain. Internal reachability represents local knowledge of reachability within the PNNI routing domain. The primary significance of this distinction is that exterior reachability information shall not be advertised to other routing protocols or routing domains (for fear of causing routing loops across routing domains). Manual configuration can be used to create internal or exterior reachability information with corresponding effects on what is advertised to other routing protocols or domains.

5.8.1.3.1 Internal Reachable Addresses

This information group describes internally reachable ATM destinations. At the lowest level, this generally represents a summary of systems which have registered with ILMI or have been manually configured. At higher levels of the hierarchy, it summarizes information provided by members of the peer group.

While internal reachable summaries are derived from configuration, they are driven by the existence of destinations which fall within those summaries. Guidelines for the proper use of configured summaries are given in Section 5.9. The reader may also wish to refer to the example in Section 3.5.

The information in an internal reachable ATM address information group is

- Port ID — A port ID of value zero indicates that the internal reachable ATM addresses are attached directly to the nucleus in the PNNI default node representation for routing purposes. If the port is specified (i.e., is non-zero), then it may optionally be used in DTLs. Also, when a complex node representation is used, inclusion of port IDs with internal reachable addresses allows for more accurate definition of the nodal state parameters associated with the node announcing the address. Any non-zero port used for an internal reachable address advertisement must not be advertised in any horizontal links or uplinks information groups in any of this node's PTSEs.
- Scope of advertisement — This is the highest level of the hierarchy to which the associated reachability is advertised.
- Address Information Length — The number of octets for each reachability summary, including the prefix length field.
- Address Information Count — The number of address prefixes in the information group.
- Pairs of prefix length and prefix — The prefix length is the number of significant bits in the prefix. The prefix itself is the summary of reachable ATM addresses. The prefix is padded with zero bits on the right (see padding algorithm specified in Section 5.14.9.1). (The Address Information Length includes the prefix length field.)
- Optional information groups for resource availability information for both the outgoing and incoming directions.

Note that there is no requirement that all prefixes in a given Internal Reachable Address Information Group have the same prefix length (though they are all encoded with the same address information length). The only requirement is the practical one that each must fit, with the individual prefix length fields, within the number of octets specified by the address information length.

Resource Availability information may optionally be associated with internal reachable addresses, and if present applies equally to all the reachable addresses in the information group. If not specified, then it is assumed that reaching the node (or node and port) which is advertising the internal address is sufficient to reach the specified destination. Resource Availability information may be specified for both the incoming and outgoing direction. If Resource Availability information is specified, then it must be specified for all service categories supported on the link in both directions, using the same set of types as is used internal to the PNNI routing domain.

5.8.1.3.2 Exterior Reachable ATM Addresses

The exterior reachability advertisement includes information about accessibility of both exterior addresses and transit networks.

Like the Internal reachable address summary, this describes reachability to a set of ATM destinations. The implication of using an exterior advertisement is that the information about the reachability came from elsewhere. This can include cases such as other complete PNNI instances, or configuration about what is reachable over a specific link.

In an exterior reachability advertisement that has transit network information, the external reachability information and RAIG(s) provided are related to the specified transit networks. Given multiple such advertisements for the same transit network, the related information may be used to select among entrances to that transit network.

The information in an exterior reachable ATM address information group is:

- Port ID — A port ID of value zero indicates that the exterior reachable ATM addresses are attached directly to the nucleus in the PNNI default node representation for routing purposes. If the port is specified (i.e., is non-zero), then it may optionally be used in DTLs. Also, when a complex node representation is used, inclusion of port IDs with exterior reachable addresses allows for more accurate definition of the nodal state parameters associated with the node announcing the address. Any non-zero port used for an exterior reachable address advertisement must not be advertised in any horizontal links or uplinks information groups in any of this node's PTSEs.
- Scope of advertisement — This is the highest level of the hierarchy to which the associated reachability is advertised.
- Address Information Length — The number of octets for each reachability summary, including the prefix length field.
- Address Information Count — The number of address prefixes in the information group.
- Pairs of prefix length and prefix — The prefix length is the number of significant bits in the prefix. The prefix itself is the summary of reachable ATM addresses. The prefix is padded with zero bits on the right (see padding algorithm specified in Section 5.14.9.1). (The Address Information Length includes the prefix length field.)
- Optional information groups for resource availability information for both the outgoing and incoming directions.
- Optional Transit Network ID information groups.

Note that there is no requirement that all prefixes in a given Exterior Reachable Address Information Group have the same prefix length (though they are all encoded with the same address information length). The only requirement is the practical one that each must fit, with the individual prefix length fields, within the number of octets specified by the address information length.

Resource Availability information may optionally be associated with exterior reachable addresses, and if present applies equally to all the reachable addresses and transit networks in the information group. If not specified, then it is assumed that reaching the node (or node and port) which is advertising the exterior address is sufficient to reach the specified destination. Resource Availability information may be specified for both the incoming and outgoing direction. If Resource Availability information is specified, then it must be specified for all service categories supported on the link in both directions, using the same set of types as is used internal to the PNNI routing domain.

5.8.2 PTSPs and PTSEs

Topology information is reliably distributed to all nodes in a peer group. This information describes the topology of the peer group, including all nodes and links in the peer group, and also the destinations that each node can reach.

For the detailed format and encodings of PTSPs and PTSEs refer to Section 5.14.

5.8.2.1 PTSPs

The PTSP is the packet used to send PNNI topology information to a neighboring peer. The PTSP contains one or more PTSEs describing individual aspects of the topology. Each PTSP carries information from a single originator for a single scope. These are carried in the following two fields of the PTSP header:

- Originating Node ID
  The node identifier of the node that originated all the PTSEs contained in this PTSP.
- Originating Node's Peer Group ID
  The peer group identifier of the node that originated all the PTSEs contained in this PTSP. This defines the span over which the PTSEs contained in this PTSP will be flooded. Note that this span logically includes all child peer groups of the identified peer group.

Note that the grouping of PTSEs within PTSPs may vary. A sender is not required to forward PTSEs to a neighboring peer grouped into PTSPs in the same manner as they were received. Similarly, a sender is not required to group PTSEs into PTSPs the same way as any particular previous transmission.

5.8.2.2 PTSEs

PTSEs are the units of flooding and retransmission. A node originates one or more PTSEs to describe its current operating characteristics (i.e., active links, their available bandwidth, etc.). PTSEs are distributed throughout a PNNI peer group via flooding.

5.8.2.2.1 PTSE Header Contents

Each PTSE header contains the following items:

- PTSE Identifier
  When a node originates multiple PTSEs, each describing different pieces of the node's environment (e.g., each may describe a different set of links), the PTSEs are distinguished by PTSE Identifier. This is an unsigned 32-bit number, whose assignment is solely up to the originating node.
- PTSE Sequence Number
  When multiple instances of the same PTSE exist simultaneously, the PTSE Sequence Number determines which instance is "more recent". PTSE Sequence Number is a 32-bit unsigned number.
- PTSE Checksum
  PTSE Checksum serves the following purposes:
    - it indicates whether a PTSE has been corrupted, either during flooding or while being held in the node's topology state database and
    - in certain exception cases, it is used to decide which of two PTSE instances is treated as more recent.
- PTSE Remaining Lifetime
  The number of seconds before the PTSE will be considered invalid. PTSE Remaining Lifetime is decremented:
    - while the PTSE is held in a node's topology database and
    - during flooding.

  If the PTSE Remaining Lifetime hits ExpiredAge, it is reflooded and removed from the peer group's topology database. The initial value for this field is PTSERefreshInterval multiplied by the PTSELifetimeFactor.

- PTSE Type
  The PTSE Type field indicates which restricted information groups are allowed to appear inside of the PTSE (see Section 5.14.9 for details).

5.8.2.2.2 PTSE Checksum algorithm

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the PTSE except the Lifetime. In addition, the Originating Node ID and Originating Node's Peer Group ID fields from the PTSP header are included. For purposes of computing the checksum, the value of the checksum field is zero.

5.8.2.2.3 PTSE identification and PTSE instance identification

To implement the flooding procedures, each PTSE, and each instance of a given PTSE, must be uniquely identified. The information used for identifying a PTSE is:

- Originating Node ID
- PTSE identifier

The additional information that distinguishes one instance of a PTSE from another instance of the same PTSE is:

- PTSE sequence number
- PTSE remaining lifetime
- PTSE checksum

Most of this identifying data is carried in the PTSE header, except for the Originating Node ID, which is carried in the header of the PTSP that contains the PTSE.

5.8.2.2.4 Comparison of PTSE instances

Two PTSEs having the same Originating Node ID and PTSE Identifier but different PTSE Sequence Numbers are different instances of the same PTSE. For example, one PTSE may be an update for the other.

When two instances of the same PTSE exist simultaneously, they must be compared to see whether they are separate instances and, if so, which instance is "more recent". A more recent PTSE always takes precedence over less recent PTSEs. This comparison consists of the following steps:

1) The PTSE instance having the larger PTSE Sequence Number, when compared as 32-bit unsigned integers, is considered more recent.
   Otherwise (i.e. both copies have the same PTSE Sequence Number),

2) If one PTSE copy has PTSE Remaining Lifetime equal to ExpiredAge, and the other does not, the copy having PTSE Remaining Lifetime of ExpiredAge is considered more recent.
   Otherwise (i.e., either both copies are expired or both are not expired),

3) The PTSE copy having the larger PTSE Checksum when compared as unsigned numbers is considered to be more recent. (Note: When the two copies have different PTSE Checksums, they have different contents, and so are different instances. It is however impossible at this point to determine which one is really "more recent". If the chosen PTSE copy is not really most recent, the PTSE's

originator will end up re-originating the PTSE.)
Otherwise, both copies have the same PTSE Checksum, and

Both copies are assumed to be the same PTSE instance.

5.8.3 Flooding

5.8.3.1 Overview

The PNNI flooding algorithm provides for reliable distribution of PTSEs throughout a peer group. It ensures that each node in the peer group has a synchronized topology database (i.e., collection of PTSEs).

All adjacencies in Exchanging, Loading, or Full states are used by the flooding procedures, and must be fully capable of transmitting and receiving all types of PNNI routing packets.

In essence, the flooding procedure is as follows. PTSEs are encapsulated within PTSPs. When a PTSP is received its component PTSEs are examined. Each PTSE is acknowledged by encapsulating its PTSE identifying information within an “Acknowledgment Packet”, which is then sent back to the neighboring peer. If in addition the PTSE is more recent than the node’s current copy, the PTSE is installed in the topology database and flooded to all other neighboring peers. The fact that the PTSEs were sent to these neighboring peers is remembered, and they will be retransmitted until acknowledged.

The flooding procedure is described in detail in the following sections.

5.8.3.2 Sending PTSPs

PNNI nodes exchange link state information in PNNI Topology State Packets (PTSPs). The format of PTSPs is given in Section 5.14.

Nodes build and send PTSPs in response to these events:

- (self) origination of a new PTSE (see Section 5.8.3.7)
- re-origination of a new instance of a (self originated) PTSE (see Section 5.8.3.7)
- installation into the database of new instances of non-self-originated PTSEs received which must be flooded onward to other peers (see Sections 5.7.6 and 5.8.3.3)
- expiration of the PTSE retransmission timer when there are unacknowledged PTSEs on a Peer Retransmission List (see Section 5.8.3.4)
- in response to a PTSE request (see Section 5.7.8)
- expiration of the PTSE remaining lifetime (see Section 5.8.3.8)

Regardless of the event that causes a node to build and send a PTSP, the actions taken are always the same. Any bundling of PTSEs is done first. The PNNI protocol requires that all PTSEs bundled into a single PTSP for transmission must be originated by a single node.

In general, it is advisable to bundle as many PTSEs as possible into a single PTSP for transmission. However, a node must not build PTSPs which are larger than the maximum packet size allowed on the link that the given PTSP will eventually be transmitted on. A node must also not violate the traffic contract for the given link it is using. This implies that the total number and size of all the PTSPs being queued up on a given link to a given neighbor needs to be monitored and controlled by the sending node. If the sending node reaches the point where it can not send a PTSP immediately to stay within the bounds of the link or the traffic contract, state must be saved so that the needed PTSP update does indeed get sent when conditions permit.

After bundling and encapsulating PTSEs in a PTSP and initializing the PTSP header, the PNNI packet is then passed to the AAL5 service interface for transmission.

No state need ever be saved regarding what PTSEs were bundled together for transmission. Any subsequent retransmission of PTSEs or transmission of new instances of PTSEs can have different PTSP bundling applied. Nodes are permitted to rearrange another node’s PTSEs into different PTSP bundles. This is in contrast to the restriction that a node may not rearrange the information inside a PTSE originated by another node.

5.8.3.3 Receiving a PTSP

When a PTSP is received from a neighboring peer, each of the contained PTSEs is examined, by executing the following steps:

(1) If the PTSE Remaining Lifetime is different from ExpiredAge, the PTSE Checksum is validated. If the checksum is determined to be invalid, processing of the PTSE is complete. Otherwise,

(2) If the receiving link has neighboring peer state Exchanging or Loading and there is an instance of the PTSE on the receiving link’s PTSE request list, compare the received PTSE to the instance on the request list:

   (a) If the received PTSE is less recent than the instance on the request list, then an error has occurred in the database exchange process. In this case, restart the database exchange process with the neighboring peer by generating the event BadPTSERequest and stop processing the PTSP.

   (b) Otherwise, delete the PTSE from the receiving link’s PTSE request list and continue processing of this PTSE.

(3) If the node has an instance of the PTSE in its topology database that is more recent than the received PTSE, and

   (a) the PTSE is found on the receiving link’s Peer Retransmission List, then receipt of the PTSE is immediately acknowledged and processing of the PTSE is complete.

   (b) the PTSE is not found on the receiving link’s Peer Retransmission List, then the database copy is encapsulated within a PTSP and flooded to the neighboring peer. The PTSE copy in the PTSP has its PTSE Remaining Lifetime decremented by 1 (unless the PTSE Remaining Lifetime is already equal to ExpiredAge); this breaks any flooding loops that may occur due to implementation errors. Also, the PTSE is added to the neighboring peer’s Peer Retransmission List so that the PTSE will be retransmitted until it is acknowledged.

Otherwise,

(4) If there is an instance in the database, and it is the same instance as the received PTSE, then:

   (a) If the PTSE is contained on the receiving link’s Peer Retransmission List, the flooded PTSE is treated as an implicit acknowledgment. In this case, the PTSE is removed from the receiving link’s Peer Retransmission List, and processing of the PTSE is complete. Otherwise,

   (b) Receipt of the PTSE is immediately acknowledged by sending a PTSE Acknowledgment packet and processing of the PTSE is complete.

Otherwise,

(5) If there is no database copy, and the received PTSE has PTSE Remaining Lifetime of ExpiredAge, then receipt of the PTSE is immediately acknowledged by sending a PTSE Acknowledgment packet and:

(a) If the receiving link has neighboring peer state Exchanging or Loading, the expired-age PTSE is installed in the receiving node's topology database and processing of the PTSE is complete. This is done to avoid a bad PTSE request event under certain race conditions. Otherwise,
(b) Processing of the PTSE is complete.

Otherwise,

(6) Either there is no database copy and the received PTSE is not at ExpiredAge, or the received PTSE instance is more recent than the database copy. In this case, the following steps must be performed:

(a) If the Originating Node ID is listed as the receiving node itself (i.e., is equal to the receiving node's Node ID) then:

(i) If the receiving node has a valid instance of the PTSE, it must reoriginate the PTSE with sequence number one higher than the sequence number in the received PTSE (see Section 5.8.3.7) and processing of the PTSE is complete. Otherwise,
(ii) The received PTSE must be flushed from the PNNI routing domain by setting the PTSE Remaining Lifetime to ExpiredAge, installing the expired PTSE in the topology database, and initiating a flood of the PTSE (see Section 5.8.3.8), after which processing of the PTSE is complete.

Otherwise,

(b) The PTSE is added to the receiving link's Peer Delayed Acks List. In addition,
(c) The PTSE is installed in the receiving node's topology database. In the process, the former database copy (if any) is deleted from all neighboring peers' Peer Retransmission List and Peer Delayed Acks Lists (including those corresponding to the receiving link). In addition,
(d) The PTSE is flooded to a subset of all neighboring peers. The following steps are carried out for each neighboring peer:

(i) If the new PTSE was received from this neighboring peer, examine the next neighboring peer. Otherwise,
(ii) If the neighboring peer is in a lesser state than Exchanging, it does not participate in flooding, and the next neighboring peer must be examined. Otherwise,
(iii) If the neighboring peer state is Exchanging or Loading, examine the PTSE request list associated with this adjacency. If there is an instance of the new PTSE on the list, compare the new PTSE to the neighboring peer's copy:

- If the new PTSE is less recent, then examine the next neighboring peer.
- If the two copies are the same instance, then delete the PTSE from the PTSE request list, and examine the next neighboring peer.
- Else, the new PTSE is more recent. Delete the PTSE from the PTSE request list and flood the PTSE to the neighboring peer, as described in step (iv).

Otherwise,

(iv) The PTSE is encapsulated within a PTSP and flooded to the neighboring peer. The PTSE copy in the PTSP has its PTSE Remaining Lifetime decremented by 1 (unless the PTSE Remaining Lifetime is already equal to ExpiredAge). Also, the PTSE is added to the neighboring peer's Peer Retransmission List so that the PTSE will be retransmitted until it is acknowledged.

5.8.3.4 Processing of Peer Retransmission List

Periodically a node must examine each of its neighboring peer's Peer Retransmission Lists for PTSEs that must be retransmitted. Those PTSEs that were last transmitted more than PTSERetransmissionInterval seconds ago are encapsulated in a PTSP and transmitted to the neighboring peer. The current version of the PTSE in the node's topology database is used for this purpose. The PTSE copy in the PTSP has its PTSE lifetime decremented by one from the current PTSE lifetime value in the database.

5.8.3.5 Sending of PTSE Acknowledgments

According to the procedures described in Section 5.8.3.3, some PTSEs need to be acknowledged immediately, while others can be delayed. The Peer Delayed Acks list holds PTSE acknowledgments that will be sent after a certain delay named PeerDelayedAckInterval. This list is scanned every PeerDelayedAckInterval seconds and if it is not empty, a PTSE Acknowledgment packet containing a number of PTSE identifying information items is generated and sent. Acknowledged PTSEs are deleted from the Peer Delayed Acks list. Note that PTSE acknowledgments on the Peer Delayed Acks list may be transmitted in conjunction with immediate Acks. Guidelines on how to select the PeerDelayedAckInterval are presented in Appendix G.

5.8.3.6 Receiving Acknowledgment Packets

An Acknowledgment Packet contains a list of PTSE identifying information items. They are sent in response to PTSPs (see Steps 3a, 4b, 5, and 6b of Section 5.8.3.3).

When an Acknowledgment Packet is received from a neighboring peer, its encapsulated list of PTSE identifying information items is examined. For each PTSE identifying information item, if the neighboring peer's Peer Retransmission List contains the exact same instance of the PTSE, the PTSE is removed from the Peer Retransmission List.

5.8.3.7 Origination of a New PTSE or a new PTSE Instance

A node initiates the flood of a PTSE when it originates or updates one of its own self-originated PTSEs.

When a node originates a PTSE, or updates a PTSE that it had previously originated, it goes through the following steps:

I. It decides on a value for the PTSE's PTSE Sequence Number. If:

   A. this is the first time that the PTSE has been originated, its PTSE Sequence Number is set to InitialSequenceNumber.

   B. this is in response to receiving a self-originated PTSE during flooding (see Sections 5.7.6 and 5.8.3.3), the next PTSE Sequence Number after that listed in the received PTSE is chosen.

   C. this is an update of an already existing PTSE, the next PTSE Sequence Number after that listed in the current database copy is chosen.

   The "next" PTSE Sequence Number means incrementing the referenced PTSE Sequence Number by 1.

To ensure that the PTSE Sequence Number space is not exhausted too quickly, a node is limited to updating any particular PTSE no more than once every MinPTSEInterval seconds. With the default value of 1 second for MinPTSEInterval, this means that, in the absence of errors, the fastest that the PTSE Sequence Number space will be exhausted is once every 136 years.

II. The node builds the PTSE contents, describing the appropriate part of the node's environment. Whenever a PTSE is re-originated, the information in all of the information groups in that PTSE must reflect the most current state of the node, whether or not a significant change has occurred in the information group since the last time that PTSE was originated.

III. The node calculates and installs the PTSE Checksum.

IV. The node calculates the remaining lifetime of the PTSE by multiplying the PTSERefreshInterval by the PTSELifetimeFactor, and installs the result as the PTSE Remaining Lifetime.

V. The PTSE is installed in the node's topology database. The former instance of PTSE (if any) is deleted from all neighboring peers' Peer Retransmision Lists and Peer Delayed Ack Lists.

VI. For each neighboring peer in state Exchanging or greater, the PTSE is encapsulated within a PTSP and flooded to the neighboring peer. The PTSE is also added to the neighboring peer's Peer Retransmission List.

5.8.3.8 Expiration of a PTSE's Lifetime

A PTSE lifetime expires in two ways:

1) A node prematurely ages one of its self-originated PTSEs. This is frequently caused by the information in the PTSE becoming invalid.
2) One of the PTSEs in the node's topology database naturally ages out (see Section 5.8.4).

Sometimes a node may want to remove one of its self-originated PTSEs from the topology database *before* it naturally ages out. The node sets the PTSE Remaining Lifetime to ExpiredAge. This is called "premature aging". A node is allowed to prematurely age only those PTSEs that the node has itself originated (i.e., those whose Originating Node ID is equal to the node's own Node Identifier).

Whenever the remaining lifetime of a PTSE becomes ExpiredAge (either due to premature or natural aging), the PTSE is flushed. To flush a PTSE, a node performs the following steps:

I. Delete the PTSE from all neighboring peers' Peer Retransmission Lists and Peer Delayed Ack Lists.

II. Set the PTSE Remaining Lifetime to ExpiredAge.

III. Flood the PTSE *without content* to each neighboring peer in state Exchanging or greater. The expired PTSE is added to the neighboring peer's Peer Retransmission List.

IV. Once the conditions described in Section 5.8.3.9 are satisfied, the expired PTSE is removed from the topology database.

5.8.3.9 Removal of PTSEs from the topology database

A PTSE must be removed from the node's topology state database when, and only when, the following conditions are met:

1) The PTSE's PTSE Remaining Lifetime is equal to ExpiredAge.

2) The PTSE is no longer contained on any of the node's Peer Retransmission Lists or on any of the node's PeerDelayedAcks lists.
3) None of the node's neighboring peers are in states Exchanging or Loading.

5.8.4 Aging PTSEs in the Topology Database

PNNI nodes must monitor the age of PTSEs collected in their topology database. The aging function adjusts the PTSE Remaining Lifetime to reflect time elapsed since the PTSE was entered in the topology database. The aging function also detects when self-originated PTSEs are to be refreshed and when non-self-originated PTSEs are to be flushed (i.e. because they have reached ExpiredAge).

PTSEs which have reached ExpiredAge must not be used during route computation.

5.8.4.1 Computing the Remaining Lifetime of a PTSE

The remaining lifetime of a PTSE is determined based on the remaining lifetime of the PTSE when it was installed into the topology database and the time elapsed since then. If the initial remaining lifetime is less than or equal to the elapsed time, then the remaining lifetime must be set to ExpiredAge and the procedures in Section 5.8.3.8 must be followed. Otherwise the remaining lifetime is computed by subtracting the elapsed time from the initial installed remaining lifetime of the PTSE.

5.8.4.2 Refreshing Self-originated PTSEs

When the remaining lifetime of a self-originated PTSE falls below the threshold

Initial Lifetime – PTSERefreshInterval

implying that the PTSE has not been refreshed for at least PTSERefreshInterval, the PTSE is re-originated (see Section 5.8.3.7).

5.8.5 Triggered Updates of Topology Information

PNNI topology information which has not changed and has not been updated must be re-originated periodically to prevent it from aging out of the network. This was discussed in Section 5.8.4.2.

The only other time a PNNI entity will re-originate one of its PTSEs is in response to a significant change event. A significant change to any component of any information group (IG) generates a significant change event for the containing PNNI Topology State Element (PTSE). Such an update made in response to a significant change event is termed a triggered update.

The period between successive re-origination of PTSEs must not be less than MinPTSEInterval. The MinPTSEInterval effectively acts as a hold down timer preventing a node from injecting new PTSEs into the network at unacceptably high rates. Re-origination of the PTSE is done using the procedures specified in Section 5.8.3.7.

5.8.5.1 Significant Change Events for PNNI Topology State Elements

PTSEs contain a variable number of IGs as described in Section 5.14. Addition of a new IG to a PTSE constitutes a significant change event for the containing PTSE. Deletion of an IG from a PTSE constitutes a significant change event. Deletion of the last IG in a PTSE constitutes a significant change event which results in premature aging of the PTSE in question. Finally, a significant change in any IG contained within a PTSE constitutes a significant change to that PTSE.

The definition of significant change to the contents of an IG depends on the type of information in the IG. There are six types of IGs:

- Nodal information
- Internal Reachable ATM Addresses
- Exterior Reachable ATM Addresses
- Nodal State Parameters
- Horizontal Links
- Uplinks

A Resource Availability information group is (sometimes optionally) contained within an IG. Any modification to the contents of the Resource Availability information group is also considered as a modification to the IG contents. Therefore any significant change to an RAIG also constitutes a significant change to the containing PTSE.

5.8.5.2 Processing Significant Changes to PTSEs

When a significant change occurs, if the PTSE was last originated more than MinPTSEInterval time ago it may be re-originated again immediately. If the PTSE in question was originated less than MinPTSEInterval time ago, it must not be re-originated immediately. The originating node must wait until MinPTSEInterval time has passed before re-originating the PTSE in question. Once MinPTSEInterval time has passed since the last origination, the PTSE in question must then be re-originated and flooded according to the flooding procedures specified in Section 5.8.3.7.

The following subsections define what constitutes a significant change for each of the types of top level information groups and RAIGs in a PTSE.

5.8.5.2.1 Changes in Nodal Information Groups

Any change in Nodal Information is a significant change.

5.8.5.2.2 Changes in Internal Reachable ATM Addresses IGs

Any change to an Internal Reachable ATM Addresses IG is a significant change. This occurs when internal addresses transition from reachable to unreachable and vice versa.

The Internal Reachable ATM Address information group optionally contains Resource Availability information groups for each direction for one or more service categories. Changes to the Resource Availability information associated with internal reachable addresses are considered significant according to the same rules as for other Resource Availability information described in Section 5.8.5.2.5.

5.8.5.2.3 Changes in Exterior Reachable ATM Addresses IGs

Any change to an Exterior Reachable ATM Addresses IG is a significant change. This occurs when exterior addresses transition from reachable to unreachable and vice versa.

The Exterior Reachable ATM Address information group optionally contains Resource Availability information groups for each direction for one or more service categories. Changes to the Resource

Availability information associated with exterior reachable addresses are considered significant according to the same rules as for other Resource Availability information described in Section 5.8.5.2.5.

5.8.5.2.4 Changes in other information groups

This section addresses changes in the remaining IGs:

- Nodal State Parameters IGs
- Horizontal Links IGs
- Uplinks IGs

Addition or deletion of any of these IGs is a significant change.

Each of these IGs also contains a Resource Availability information group. Any significant change to the Resource Availability information is considered a significant change to the IG.

These IGs contain node, port and peer group identifiers. Any changes to these fields are considered significant. The Horizontal Link and Uplink IGs also contain the Aggregation Token. A change of the Aggregation Token (due to reconfiguration) is a significant change.

The Uplink Information Attribute (ULIA) contains a sequence number which changes only when the remaining contents have undergone a significant change, as define by the originator of the ULIA. Therefore, for inclusion in an uplink advertisement, a change in the ULIA sequence number is considered significant, and no other change in the ULIA content is considered significant.

5.8.5.2.5 Changes to Resource Availability Information

The subsections below define what changes in the components of an RAIG constitute significant changes.

5.8.5.2.5.1 Administrative Weight (AW)

AW is a required metric. Any change is significant.

5.8.5.2.5.2 Cell Loss Ratio (CLR_0)

CLR_0 is a required attribute. Any change is significant.

5.8.5.2.5.3 Cell Loss Ratio (CLR_{0+1})

CLR_{0+1} is a required attribute. Any change is significant.

5.8.5.2.5.4 Available Cell Rate (AvCR)

Generally speaking, changes in AvCR are more significant as AvCR approaches zero. For example, AvCR transitioning from zero to non-zero values implies that some number of calls can now be carried where none could before; this is certainly a significant change. Likewise, as AvCR transitions from a non-zero value to zero, that implies that no more calls can be carried where some could before; this is certainly significant.

Changes in AvCR are measured in terms of a proportional difference from the last value advertised. A proportional multiplier (AvCR_PM) parameter, expressed as a percentage, provides flexible control over the definition of significant change for AvCR. There is also a minimum threshold (AvCR_mT) parameter, expressed as a percentage of maxCR, which ensures that the range of insignificance is non-zero.

Given a previous value for AvCR the algorithm establishes an upper bound and a lower bound for AvCR values which define a range of insignificance. Any new value for AvCR computed that is within the bounds is not a significant change from the previous value. Any new value for AvCR that is outside the bounds is a significant change.

The bounds of the range of insignificance are computed using the following algorithm:

```
 compute_AvCR_bounds ( PREV_AvCR, maxCR, AvCR_PM, AvCR_mT )
{
    /*
            •   PREV_AvCR = previous/currently advertised value for AvCR for
                service category in cells/sec
            •   maxCR = Maximum Cell Rate for service category in cells/sec
            •   AvCR_PM = proportional multiplier as a percentage
                ( 1 <= AvCR_PM <= 99 )
            •   AvCR_mT = minimum threshold as a percentage
                ( 1 <= AvCR_mT <= 99 )
    */

     delta = PREV_AvCR * ( AvCR_PM/100);
     min_delta = maxCR * ( AvCR_mT/100 );

    if ( delta < min_delta ) { delta = min_delta; }

    upper_AvCR_bound = PREV_AvCR + delta;
    if ( upper_AvCR_bound > maxCR )
       { upper_AvCR_bound = maxCR; }   /* set upper bound to maxCR */

    if ( delta > PREV_AvCR )
              { lower_AvCR_bound = 0; }       /* set lower bound to zero */
    else
       { lower_AvCR_bound = PREV_AvCR - delta; }

 } /* end compute_AvCR_bounds() */
```

When AvCR changes, the following algorithm is used to determine if the change is significant:

```
/* NEW_AvCR = new value for AvCR */
if (NEW_AvCR <= lower_AvCR_bound ||
                           NEW_AvCR >= upper_AvCR_bound)
   { /* change in AvCR is significant */ }
else
   { /* change AvCR is NOT significant */ }
```

5.8.5.2.5.5 Maximum Cell Transfer Delay (maxCTD)

MaxCTD is a required metric. The algorithm used to determine significant change for maxCTD is very similar to the one used above used for AvCR, except that there is no upper limit analogous to maxCR for delay.

Change in maxCTD is measured in terms of a proportional difference from the last value advertised. A proportional multiplier parameter (maxCTD_PM), expressed as a percentages, provides flexible control over the definition of significant change for maxCTD.

Given a previous value for maxCTD the algorithm establishes an upper bound and a lower bound for values which define a range of insignificance. Any new value for maxCTD computed that is within the bounds is not a significant change from the previous value. Any new value for maxCTD that is outside the bounds is a significant change.

The bounds of the range of insignificance are computed using the following algorithm:

```
compute_maxCTD_bounds ( PREV_maxCTD, maxCTD_PM )
   {
   /*
      •  PREV_maxCTD = previous/currently advertised value of
         maxCTD for service category
      •  maxCTD_PM = maxCTD proportional multiplier as a
         percentage
         ( 1 <= maxCTD_PM <= 99 )
   */

   delta = PREV_maxCTD * ( maxCTD_PM/100);
   upper_maxCTD_bound = PREV_maxCTD + delta;
   if ( delta > PREV_maxCTD )
      { lower_maxCTD_bound = 0; }/* set lower bound to zero */
   else
         { lower_maxCTD_bound = PREV_maxCTD - delta; }
   } /* end compute_maxCTD_bounds() */
```

When maxCTD changes, the following algorithm is used to determine if the change is significant:

```
   /* NEW_maxCTD = new value for maxCTD */
   if (NEW_maxCTD <= lower_maxCTD_bound ||
                 NEW_maxCTD >= upper_maxCTD_bound)
       { /* change in maxCTD is significant */ }
else
    { /* change in maxCTD is NOT significant */ }
```

5.8.5.2.5.6 Cell Delay Variation (CDV)

CDV is a required metric. The algorithm used to determine significant change for CDV is identical to the one used above used for maxCTD.

Change in CDV is measured in terms of a proportional difference from the last value advertised. A proportional multiplier parameter (CDV_PM), expressed as a percentages, provides flexible control over the definition of significant change for CDV.

Given a previous value for CDV the algorithm establishes an upper bound and a lower bound for values which define a range of insignificance. Any new value for CDV computed that is within the bounds is not a significant change from the previous value. Any new value for CDV that is outside the bounds is a significant change.

The bounds of the range of insignificance are computed using the following algorithm:

```
compute_CDV_bounds ( PREV_CDV, CDV_PM )
{
   /*
      •  PREV_CDV = previous/currently advertised value of CDV for
         service category
      •  CDV_PM = CDV proportional multiplier as a percentage
         ( 1 <= CDV_PM <= 99 )
   */
```

```
      delta = PREV_CDV * ( CDV_PM/100);
      upper_CDV_bound = PREV_CDV + delta;
   if ( delta > PREV_CDV )
      { lower_CDV_bound = 0; }       /* set lower bound to zero */
   else
      { lower_CDV_bound = PREV_CDV - delta; }
   } /* end compute_CDV_bounds() */
```

When CDV changes, the following algorithm is used to determine if the change is significant:

```
   /* NEW_CDV = new value for CDV */
   if (NEW_CDV <= lower_CDV_bound ||
                        NEW_CDV >= upper_CDV_bound )
       { /* change in CDV is significant */ }
   else
       { /* change in CDV is NOT significant */ }
```

5.8.5.2.5.7 Maximum Cell Rate (maxCR)

MaxCR is a required attribute. Any change in maxCR is significant. Note that maxCR is used to calculate the range of insignificance for AvCR.

5.8.5.2.5.8 Cell Rate Margin (CRM) and Variance Factor (VF)

CRM and VF are optional attributes. Changes in CRM or VF are not considered significant.

5.9 Advertising and Summarizing Reachable Addresses

This section describes the default behavior of a Logical Group Node in summarizing addresses of the Peer Group it represents in the next higher level of the hierarchy. The same behavior applies to a lowest level node with respect to summarizing the addresses of its attached systems.

The term “default summarization rules” implies a recognition that these rules will not serve the needs of all organizations at all times, and that other behaviors may be made available by vendors through configuration options.

5.9.1 Scope of Advertisement of Addresses

The advertisement scope of reachable addresses is specified by a level indicator, which specifies that the address will be advertised up to this level, but not into any higher level of PNNI routing. If the level indicator is set to zero, then the advertisement scope is unlimited, which means that the address may be advertised throughout the PNNI routing domain.

Addresses are eligible for advertisement or summarization into a peer group only if they have a advertisement scope that is higher than or equal to that of the peer group. If there are addresses in the summary with different advertisement scopes, the highest such advertisement scope is advertised with the summary.

The PNNI advertisement scope of an address registered with an organizational scope at the UNI is determined according to a network specific mapping, as described in Section 5.3.6.

5.9.2 Summary Address and Suppressed Summary Address

A summary address is an abbreviation of a set of addresses, represented by an address prefix that all of the summarized addresses have in common. When a node advertises a summary address, calls to any destination matching that summary address (unless the destination address matches a longer address prefix advertised elsewhere) may be routed to that node. This means that an address summary should only be used when all reachable addresses matching this summary are reachable at this node. A node can have an arbitrary number of summary addresses.

A suppressed summary address is used to suppress the advertisement of addresses which match this prefix, regardless of scope. If suppression is configured for a given reachability advertisement, advertisement of that address shall be suppressed regardless of its scope. Advertisements whose scope is sufficiently wide enough to otherwise be advertised at a higher level (i.e., advertisement scope value is smaller than level of this node), will nonetheless be suppressed by such configuration. A node can have an arbitrary number of suppressed summary addresses.

There are separate sets of summary addresses and suppressed summary addresses for internal and exterior reachable addresses. Exterior summary information does not affect advertisement of internal reachable addresses, and internal summary information does not affect advertisement of exterior reachable addresses.

By default a logical group node has one internal summary address which is identical to the Peer Group ID it represents and no suppressed summary addresses. By default, a lowest level node has one default internal summary address (the 13-octet prefix of the node’s address), unless the node is at level 104 (in which case it has no default summary address). A node by default has no internal suppressed summary addresses, exterior summary addresses, or exterior suppressed summary addresses. See Section 5.8.1.3 for a discussion of the differences between internal and exterior addresses.

When overlapping summary addresses and/or suppressed summary addresses are present, the longest matching summary address or suppressed summary address shall be used to determine whether an address is to be advertised explicitly, advertised indirectly through use of a summary address, or suppressed. Configuration of duplicate summary and suppressed summary addresses is an error.

5.9.3 Native Addresses

An address is considered native if it matches one of the node’s summary addresses. The summary address is advertised; and the address (or more detailed summary address) being summarized is suppressed.

The summary address is not advertised if no match of suitable scope has occurred.

Addresses within the range of a summary address can be used elsewhere in the network. If the address prefixes being advertised by the different parts of the network are of different length, then the longer one will be used for destinations that match it. If they are of equal length, then either may be chosen by call originators. In the latter case, unless all destinations are reachable at both places, this is likely to result in frequent crankback, which is typically undesirable.

When summarizing internal reachable addresses, aggregation of RAIGs associated with the addresses being summarized is done in an implementation specific manner.

5.9.4 Foreign Addresses

An address which does not match any of the summary addresses of a node is assumed to be foreign to that node. If the associated advertisement scope is higher than or equal to the level of the node, the address and the advertisement scope are passed along without further summarization. If the address matches one of the suppressed summary addresses, then the address is not advertised at all.

5.9.5 Group Addresses

When multiple identical group addresses are present, only one address is advertised. In this case, the highest advertisement scope is used. There are no default summary group addresses.

5.9.6 Exterior Reachable Addresses

The PNNI routing protocol allows for exterior reachable addresses to be advertised into a PNNI routing domain.

When summarizing exterior reachable addresses, aggregation of RAIGs associated with the addresses being summarized is done in an implementation specific manner.

5.10 Hierarchy

5.10.1 Peer Group Leader

The PNNI hierarchy requires that a node be selected in each peer group to perform some functions of the LGN. The node selected for this purpose is known as the Peer Group Leader. Preference for peer group leadership is established through configuration. This preference is indicated by the PGL priority advertised by each node. The node ID is used as a tie breaker among nodes with equal PGL priorities.

5.10.1.1 Peer Group Leader Election Algorithm

The PGL election algorithm is used to dynamically select an appropriate node to assume peer group leadership within a peer group or to replace an outgoing PGL. Among all nodes to which a node has connectivity, it must vote for the one with the highest non-zero PGL priority subject to tie breaking using node IDs. The leadership priority is advertised by all nodes in the peer group in PTSEs. If no node advertises a non-zero PGL priority, then no node is selected. A node will consider a 2/3 majority vote sufficient for PGL election after it determines that a unanimous vote cannot be obtained within a sufficient time, so that errant implementations in a small number of nodes in the peer group are not likely to cause a hung election. For a similar reason, all nodes in a peer group must participate in PGL election during normal operation. However, a node that has the Non-transit for PGL Election flag set in its Nodal IG does not participate in PGL election, i.e., it does not run the PGL election FSM and advertises zero for Leadership Priority and Preferred PGL. Such nodes are also not considered by other nodes in the peer group when determining connectivity in the peer group, and the PGL priority and preferred PGL advertised by such overloaded nodes are ignored by all other nodes.

The scope of any references to node, connectivity, or reachability in the following description is within a single peer group. In the case of a partitioned peer group, this means within a single partition.

5.10.1.1.1 The PGL Election Data Structure

Each node has a single Peer Group Leader election data structure, which consists of the following items:

State

The operational status of the peer group leader election FSM. This is described in more detail in Section 5.10.1.1.2.

PreferredPeerGroupLeader

The ID of the node that this node believes should be peer group leader. To select its preferred peer group leader, the node compares the leadership priorities and node IDs advertised in the PTSEs in the topology database for nodes that are reachable from this node. See Section 5.10.1.1.4, action PGLE4, for details.

PreferredPGLLeadershipPriority

The leadership priority of the preferred peer group leader.

SearchPeer Timer

An interval timer that fires after InactivityFactor times HelloInterval. When the timer fires the node considers that it has no peer and starts the election process immediately.

PGLInit Timer

An interval timer that fires after PGLInitTime. When this timer fires, the election process starts. The node selects a preferred peer group leader and advertises its selection by originating a new instance of its Nodal Information PTSE. The PGLInit Timer is used to ensure that every node casts a vote only after waiting for a sufficient amount of time for topology information to propagate across the entire peer group.

OverrideUnanimity Timer

An interval timer that fires after OverrideDelay. It is used to prevent nodes from waiting forever for unanimity.

ReElection Timer

An interval timer that is started when connectivity is lost to the PGL and fires after ReElectionInterval. After that time, the election process must restart. The node selects a new preferred peer group leader and advertises its selection in a new instance of its Nodal Information PTSE. The ReElection Timer is used to hold off each node in the peer group from voting for a PGL upon loss of connectivity until the nodes in the peer group have had a chance to receive updated topology information. This improves the stability of the system.

5.10.1.1.2 PGL Election States

The states that the Peer Group Leader election FSM may attain are described in this section. Figure 5-7 shows a diagram of the possible state changes. The arcs are labeled with the events that cause each state change. These events are described in Section 5.10.1.1.3. For a detailed description of the state changes and the actions involved with each state change, see Section 5.10.1.1.4.

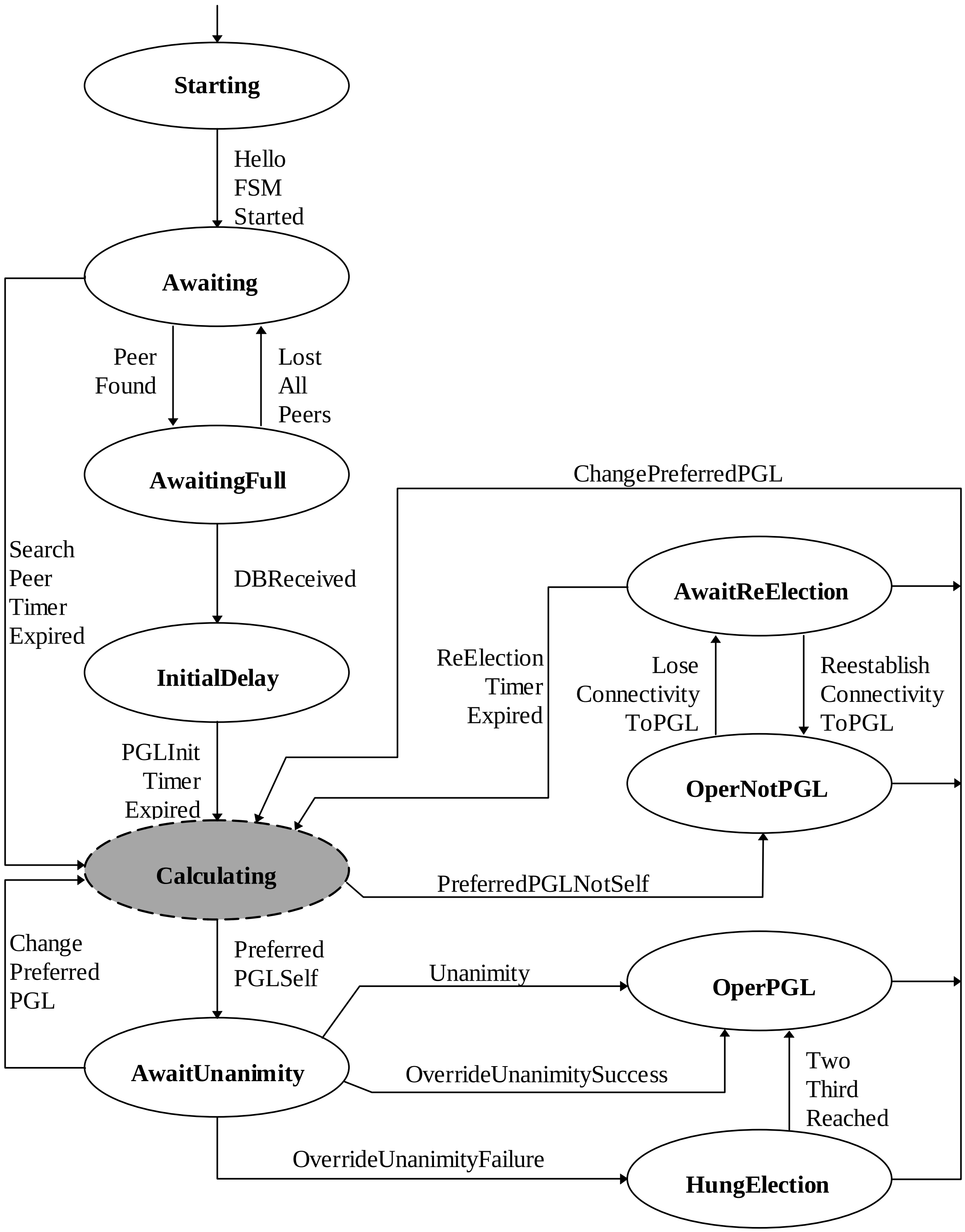


Figure 5-7: PGL Election FSM.

Starting

The initial state of the state machine.

Awaiting

The node has started the Hello FSM on at least one link. No peer has been found yet.

AwaitingFull

At least one neighboring peer has been found. No database synchroniation process has been completed yet.

InitialDelay

Database synchronisation has been completed with at least one neighboring peer and the PGLInit timer has started. The node must wait PGLInitTime before it can select and advertise its preferred PGL.

Calculating

The node is in the process of calculating what its (new) choice for PreferredPGL will be. This is a transitive state; as soon as the node selects its (new) choice for PGL, this state will be exited. The selection is made by executing the ChoosePreferredPGL algorithm detailed below.

The next state transition out of the calculating state depends on whether the node selected itself as PGL, or chose some other node. One of two events is generated to effect the transition:

i) The PreferredPglNotSelf if some other node was selected.
ii) The PreferredPglSelf event if the node selected itself.

The algorithm for choosing a preferred PGL is specified in Section 5.10.1.1.6.

OperNotPGL

This node is not Peer Group Leader. It continues examining PTSEs sent by other nodes to determine which node has the highest priority to be PGL.

OperPGL

This node is Peer Group Leader. It continues examining PTSEs sent by other nodes to see if another node has a higher priority than itself.

AwaitUnanimity

The node has chosen itself as Peer Group Leader. Upon entering this state, the node must first immediately check if it has been elected unanimously, and if so generate a Unanimity event. It waits for unanimity or the expiration of the OverrideUnanimity timer before declaring itself peer group leader.

HungElection

The node has chosen itself as Peer Group Leader, but, after the OverrideUnanimity timer has fired, less than 2/3 of the other nodes are advertising it as their preferred PGL. This may come from a change in the topology or the parameters of the network. In that case, the situation will recover by itself, i.e., either this node is going to change its choice of preferred Peer Group Leader, or the other nodes are going to accept it as PGL. This situation may also be the result of a defective switch or link.

AwaitReElection

The node has lost connectivity to the current PGL. The ReElection timer has been started. If connectivity has not been reestablished before the timer fires, the election is redone.

5.10.1.1.3 Events Causing Neighboring Peer State Changes

State changes can be brought about by several possible events. These events are brought about by procedures associated with Peer Group Leader election, especially the reception of a PTSE that contains a nodal information group. They may also be triggered by developments within the Hello state machines and the Neighboring Peer state machines. The events are shown as the labeled arcs in Figure 5-7. A detailed explanation of the state changes and actions taken after an event occurs is given in Section 5.10.1.1.4.

HelloFSMStarted
: The first Hello state machine has started on a link.

PeerFound
: A Hello state machine has reached state 2-Way Inside.

LostAllPeers
: The last remaining Hello state machine that was in state 2-Way Inside has left that state.

SearchPeerTimerExpired
: The SearchPeer timer has fired. No peer has been found. The Peer Group Leader election starts immediately.

DBReceived
: A Neighboring Peer state machine has reached state Full.

PGLInitTimerExpired
: The PGLInit timer has fired. The node can now select and advertise its preferred Peer Group Leader.

PreferredPGLNotSelf
: Having entered the Calculating state, the node determines that its preferred PGL is not itself. This happens if its leadership priority equals zero or a peer has a higher priority.

Unanimity
: The node's preferred PGL is itself, and all other nodes in the peer group now indicate in their PTSEs that their preferred PGL is this node.

OverrideUnanimitySuccess
: The OverrideUnanimity timer has fired. The node's preferred PGL is itself. 2/3 or more of other nodes in the peer group indicate in their PTSEs that their preferred PGL is this node.

OverrideUnanimityFailure
: The OverrideUnanimity timer has fired. The node's preferred PGL is itself. Less than 2/3 of other nodes in the peer group indicate in their PTSEs that their preferred PGL is this node.

TwoThirdReached
: The node's preferred PGL is itself. 2/3 of other nodes in the peer group now indicate in their PTSEs that their preferred PGL is this node.

PreferredPGLSelf
: Having entered the Calculating state, the node determines that it should be PGL.

ChangePreferredPGL

New information is received that changes this node’s choice of preferred PGL (for example, a new high priority node appears; the current PGL lowers its priority; or management raises this node’s priority to be higher than that of the current PGL).

LoseConnectivityToPGL

Connectivity to the Peer Group Leader has been lost.
Note: Determination of whether a node has connectivity to the PGL (or any other node for the purpose of running the PGL election) requires a special route computation. Specifically, it must determine whether there is connectivity to the PGL by ignoring resource and policy constraints on the links within the peer group. All links that are advertised by the nodes at both ends and all nodes advertised in the peer group must be considered for evaluating PGL connectivity, except for nodes which have the Non-transit for PGL Election flag set in their nodal flags. For example, it must use nodes and links with access controls, transit restrictions, zero remaining AvCR or unknown mandatory information groups in order to determine if connectivity exists to the current PGL.

ReestablishConnectivityToPGL

Connectivity to the Peer Group Leader has been reestablished.

ReElectionTimerExpired

The ReElection timer has fired. The election must be redone.

5.10.1.1.4 The PGL Election State Machine

The finite state machine is a two dimensional table with States across the top of the table and Events down the left side. Each pairing of event and state crosses at a "cell" in the table. The cell shows what state transition should occur and the action to take. For example, for the event and state pair of "HelloFSMStarted" and "Starting" the cell reads "PGLE1, Awaiting". "Awaiting" is the new state and "PGLE1" is the Action to be taken. The actions can be found following table.

Table 5-14: PGL Election FSM Part 1.

	States				
Events	Starting (Note 1)	Awaiting	AwaitingFull	InitialDelay	Calculating
Hello FSM Started	PGLE1 Awaiting	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Peer Found	FSM ERR	PGLE2 AwaitingFull	PGLE0 AwaitingFull	PGLE0 InitialDelay	PGLE0 Calculating
Lost All Peers	FSM ERR	FSM ERR	PGLE3 Awaiting	PGLE0 InitialDelay	PGLE0 Calculating
SearchPeer Timer Expired	FSM ERR	PGLE4 Calculating	FSM ERR	FSM ERR	FSM ERR
DB Received	FSM ERR	FSM ERR	PGLE5 InitialDelay	PGLE0 InitialDelay	PGLE0 Calculating
PGLInit Timer Expired	FSM ERR	FSM ERR	FSM ERR	PGLE4 Calculating	FSM ERR
Preferred PGL Not Self	FSM ERR	FSM ERR	FSM ERR	FSM ERR	PGLE7 OperNotPGL
Unanimity	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Override Unanim.Success	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Override Unanim. Failure	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Two Third Reached	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Preferred PGL Self	FSM ERR	FSM ERR	FSM ERR	FSM ERR	PGLE9 AwtUnanimity
Change Preferred PGL	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Lose Connectivity To PGL	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Reestablish Connectivity To PGL	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
ReElection Timer Expired	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR

Note 1: For purposes of the PGL Election FSM description, it is assumed that the PGL Election FSM is instantiated before the first Hello FSM is started.

Table 5-15: PGL Election FSM Part 2.

	States				
Events	OperNotPGL	OperPGL	AwtUnanimity	HungElection	AwtReElection
Hello FSM Started	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Peer Found	PGLE0 OperNotPGL	PGLE0 OperPGL	PGLE0 AwtUnanimity	PGLE0 HungElection	PGLE0 AwtReElection
Lost All Peers	PGLE0 OperNotPGL	PGLE0 OperPGL	PGLE0 AwtUnanimity	PGLE0 HungElection	PGLE0 AwtReElection
SearchPeer Timer Expired	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
DB Received	PGLE0 OperNotPGL	PGLE0 OperPGL	PGLE0 AwtUnanimity	PGLE0 HungElection	PGLE0 AwtReElection
PGLInit Timer Expired	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Preferred PGL Not Self	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Unanimity	FSM ERR	PGLE0 OperPGL	PGLE8 OperPGL	PGLE8 OperPGL	FSM ERR
Override Unanim. Success	FSM ERR	FSM ERR	PGLE8 OperPGL	FSM ERR	FSM ERR
Override Unanim. Failure	FSM ERR	FSM ERR	PGLE0 HungElection	FSM ERR	FSM ERR
Two Third Reached	FSM ERR	PGLE0 OperPGL	PGLE0 AwtUnanimity	PGLE8 OperPGL	FSM ERR
Preferred PGL Self	FSM ERR	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Change Preferred PGL	PGLE4 Calculating	PGLE6 Calculating	PGLE4 Calculating	PGLE4 Calculating	PGLE4 Calculating
Lose Connectivity To PGL	PGLE10 AwtReElection	FSM ERR	FSM ERR	FSM ERR	FSM ERR
Reestablish Connectivity To PGL	FSM ERR	FSM ERR	FSM ERR	FSM ERR	PGLE11 OperNotPGL
ReElection Timer Expired	FSM ERR	FSM ERR	FSM ERR	FSM ERR	PGLE4 Calculating

FSM ERR

Action: Protocol error, should not occur.

PGLE0

Action: Do nothing.

PGLE1

Action: The SearchPeer timer is started with initial value InactivityFactor times HelloInterval.

PGLE2

Action: The SearchPeer timer is stopped.

PGLE3

Action: The SearchPeer timer is restarted, with initial value InactivityFactor times HelloInterval.

PGLE4

Action: The OverrideUnanimity timer is stopped if it is running. The ReElection timer is stopped if it is running. The node re-evaluates the value of its PreferredPeerGroupLeader, as described in Section 5.10.1.1.6.

PGLE5

Action: The PGLInit timer is started, with initial value PGLInitTime. The node originates a PTSE that contains a nodal information group in order to advertise its leadership priority. The "I am leader" bit is set to 0. The Preferred peer group leader node ID is set to 0.

PGLE6

Action: The node stops behaving as PGL and deactivates its parent LGN by performing the following actions:

- A new instance of the Nodal Information PTSE must be originated. The Preferred Peer Group Leader ID field is updated, according to the procedures given in action PGLE4. The node must advertise its original leadership priority and turn off the “I am leader” bit.
- Initiate the process of deactivating the LGN. That process involves the following actions:
  - If the parent LGN is PGL of its peer group, it stops behaving as PGL by performing action PGLE6 at its level.
  - The procedures from Section 5.5.6.3 item C for a node which ceases to be PGL are performed.
  - The parent LGN stops feeding PTSEs into the child peer group.

PGLE7

Action: If the preferred Peer Group Leader is different from the previously advertised one, a new instance of the Nodal Information PTSE must be originated with the new preferred Peer Group Leader ID and the “I am leader” bit set to zero.

PGLE8

Action: The node starts to behave as PGL by taking the following actions:

- The node increases its leadership priority by GroupLeaderIncrement. This higher priority is used in future calculations regarding who is PGL.
- Stop the OverrideUnanimity timer if it is running.
- A new instance of the Nodal Information PTSE is originated with the updated leadership priority and the "I am leader" bit set to 1 to indicate that the node is claiming to be Peer Group Leader. The higher level peer group is also included in the PTSE.
- The parent LGN is instantiated, causing the following to happen in due time:
  - The parent LGN establishes SVCC-based RCCs to neighboring peer nodes in the parent peer group as described in Section 5.5.
  - Once the peer group election is complete in the higher level peer group, the results are included in the next higher level binding information in the nodal information PTSE of this node. If multiple nodes claim to be PGL in the parent peer group, information is only included about the node selected locally as higher level PGL.
  - PTSEs received from higher level peer groups are flooded into this peer group.

Note: The “I am leader” bit allows border nodes to know when the PGL is (believed to be) stable, and can be announced in Hellos sent to outside neighbors. The increase in leadership priority associated with becoming peer group leader provides some increased stability with respect to who is PGL.

PGLE9

Action: A new instance of the Nodal Information PTSE must be originated with the preferred Peer Group Leader ID set to this node's ID and the "I am leader" bit set to zero. The OverrideUnanimity timer is started, with initial value OverrideDelay.
Note: If all nodes are unanimous in their choice of this node as PGL at the time of this action, the origination of the new Nodal Information PTSE is optional as there will be an immediate transition into state OperPGL and consequently another Nodal Information PTSE will be generated.

PGLE10

Action: The ReElection timer is started, with initial value ReElectionInterval.

PGLE11

Action: The ReElection timer is stopped.

5.10.1.1.5 Sending a Nodal Information PTSE

At the beginning of the peer group leader election, the node must originate a PTSE that contains a nodal information group. This PTSE must be re-originated either when a change occurs or regularly to prevent it from aging out of the network.

In states Awaiting, AwaitingFull and InitialDelay, and when the node has the Non-transit for PGL Election flag set, the preferred peer group leader node ID field must be set to zero. Furthermore, if the node has the Non-transit for PGL Election flag set, it must set the Leadership Priority to zero. In states Calculating, AwaitUnanimity, AwaitReElection, OperNotPGL, OperPGL and HungElection the field must be set to PreferredPeerGroupLeader in the data structure (see action PGLE4 in Section 5.10.1.1.4).

The "I am leader" bit must be set only in state PGL.

5.10.1.1.6 Choosing Preferred PGL

Each time a node receives and accepts a PTSE that contains a nodal information group, it must re-evaluate the value of its PreferredPeerGroupLeader. The node also re-evaluates its PreferredPeerGroupLeader whenever connectivity to another node in the peer group is established or is lost.

The node compares the leadership priority value of all reachable nodes in the peer group (including itself). It should be noted that nodes which appear in the database, but to which there is no connectivity (by the rules for determining connectivity stated above in the description for the LoseConnectivityToPGL event), as well as nodes with peer group leadership priority zero, are not considered in this comparison. The node with the highest leadership priority is selected as preferred PGL. If there are no reachable nodes in the peer group with a leadership priority different from zero, the preferred PGL ID is set to zero. The Node ID is used as the tie-breaker. For this purpose, the 22-octet Node ID is treated as an unsigned integer with the first octet of the Node ID as the most significant octet. In the case of a tie, the node with the highest value of the Node ID from among the set with the highest leadership priority is selected as preferred PGL.

It should be noted that a node must consider only information received directly from other nodes. In particular, the preferred PGL advertised by other nodes must not affect this node's choice of preferred PGL.

5.10.1.2 Leadership Priority

The peer group leadership priority is a value between 0 and MaxLeadership. It is assigned to indicate the relative preference for this node to become PGL.

5.10.1.2.1 Group Leader Increment

To allow for flexibility and stability, when a node concludes that it will be operating as peer group leader (i.e., when it enters PGL Election FSM state OperPGL and sets the “I am leader” bit), it increments its advertised peer group leadership priority by the constant GroupLeaderIncrement.

5.10.1.2.2 Distinguished Values and Range Restrictions

The priority value 0 is reserved for a node which is either unwilling or unable to become peer group leader.

No node shall select as peer group leader a node which is advertising a priority of 0. If all nodes in a peer group are advertising priority 0, the behavior described in Section 5.10.1.4 shall be followed.

No node shall be configured with a leadership priority value greater than MaxLeadership minus GroupLeaderIncrement. Note that an active PGL advertises the configured priority plus GroupLeaderIncrement.

5.10.1.2.3 Range Recommendations

In peer groups where there is no node that must be forced to be peer group leader, peer group leadership priority for nodes which can become leader should be assigned in the range 1 to GroupLeaderIncrement-1.

In situations where certain nodes should take precedence over nodes in the first tier (even if one of them is already PGL), the nodes that are to take precedence should be assigned a leadership priority between 2*GroupLeaderIncrement and 3*GroupLeaderIncrement-1. Within this tier of nodes, precedence will still defer to the existing peer group leader to produce stability.

Finally, if it is necessary to have a strict master leader, that node should be assigned a leadership priority between 4*GroupLeaderIncrement and (MaxLeadership-GroupLeaderIncrement). This will produce a value which will take precedence over the second tier of leaders, and still be incrementable.

5.10.1.2.4 Implications of Hierarchy

There is no implicit correlation between the peer group leadership priorities of two nodes at two different levels of the hierarchy instantiated in the same switching system. There is a requirement that a switching system may only become PGL if it exists as a node within a given peer group. This does mean that it must be the leader of all lower level peer groups of which it is a member. Therefore, when a node is preempted as PGL at some level of the hierarchy, all ancestor nodes of that node instantiated in the same switching system cease to exist. If any of these ancestor nodes were PGL, this forces a new PGL election in the corresponding peer groups. Note that the PGL leadership priorities of a switching system can be different at each level of the hierarchy.

As a special case of the above, not all peer group leaders are necessarily willing to become higher level peer group leaders. If a node does not know the higher level peer group identity, it must not become leader. It may also choose, for local reasons, to be unwilling to become higher level leader. In either case, the node advertises a leadership priority of 0 at that level of the hierarchy.

5.10.1.3 Default Configuration Consistency

The default configuration for a node shall have a leadership priority of 0.

Note: This is compatible with the notion that lowest level nodes may be autoconfigured, but that higher level nodes require some (limited) amount of active configuration, since a peer group leader must know the peer group hierarchy.

5.10.1.4 Operation Without a Peer Group Leader

The behavior specified in this section also applies prior to completion of the peer group leader election. Operation without a peer group leader is a normal mode of operation for networks which are small enough to operate as a single peer group, without the need for a hierarchy. It also applies at the highest level of the hierarchy.

Internal operation of the peer group does not depend on the existence of a peer group leader. Thus, in the absence of a peer group leader, internal operation of the peer group continues normally. Nodes must transmit PNNI Hellos, and determine their local topology including identification of which of their neighboring nodes are in the same peer group. PTSPs must be transmitted and received normally within the peer group. Routes internal to the peer group are calculated, and connections may be setup in the peer group.

Because there is no peer group leader in the peer group, higher level information cannot be passed into the peer group. The peer group does not take part in the operation of higher levels of PNNI routing within the same contiguous PNNI routing domain. Nodes in a leaderless peer group are permitted to advertise exterior reachability information and such information may be used to reach destinations outside the routing domain.

Border nodes in a leaderless peer group continue to exchange Hellos on their outside links. Since there is no higher level information, an empty nodal hierarchy list is transmitted. This precludes the Hello state machine from reaching Common Outside state. Thus such links will not be used for the establishment of connections.

5.10.1.5 Exchange of Nodal Hierarchy Information on Outside Links

Once a Peer Group Leader has been elected and has advertised that fact by setting the “I am Leader” bit to 1 in its Nodal Information PTSE, border nodes include Higher Level Binding information received from the PGL in their nodal hierarchy list in Hellos exchanged with outside neighbors.

If multiple nodes claim to be peer group leader of this peer group, border nodes advertise the information from the one selected locally as peer group leader. For higher-level peer groups, if multiple nodes claim to be PGL, border nodes advertise the information from the one selected by the ancestor LGN in that peer group, as reported in the next higher level binding information in the Nodal Information group received from that LGN.

5.10.2 Advertising Uplinks

When an uplink advertisement is generated and injected into a peer group, it must contain topology state parameters for both directions so that PNNI can consider the links during path selection. These topology state parameters are acquired as follows.

5.10.2.1 Advertising Uplinks from Lowest Level Nodes

When the nodes at the ends of an outside link discover that the other node is in a different peer group, they must include Resource Availability information for the outbound direction in their Hello Packets. A ULIA is included in the Hello Packet to carry this information.

The topology state parameters to exchange here are identical to the topology state parameters that would appear in the advertisement for that link if it were internal to the peer group. The only difference is that they are carried in a Hello Packet rather than in PTSEs.

Since Hellos never propagate across more than one link, the topology state parameters can be bound by the receiver to the outside link corresponding to the port on which the Hello is received. No extra binding information is required. The topology state parameters received in the Hellos on that port are for the inbound direction of that outside link.

As the hierarchy boots, outside links become the basis for uplinks. The uplinks are advertised into the peer groups of the border nodes at each end of the outside links. The border nodes use the topology state parameters for the inbound directions on the outside links as the basis for calculating topology state parameters for the inbound (i.e., downward) direction of the uplinks. Thus, the border node which injects the uplink advertisement into a peer group specifies the topology state parameters for both directions.

Individual uplinks may be generated for each outside link. In this case, the topology state parameters received in a Hello are advertised unchanged for the corresponding uplink. Alternatively, a single uplink may be an aggregation of several outside links. The procedures for such aggregation are exactly the same as those used to aggregate links in general.

5.10.2.2 Reverse Direction Information in Uplinks

Uplink advertisements are derived from either outside links or other uplinks.

For uplinks derived from outside links, reverse direction information (including topology state parameters) is specified by the outside neighbor node. Border nodes identify the information that the outside neighbor is to include as reverse direction information in its uplink advertisements, by including an Uplink Information Attribute (ULIA) in the Hello sent over the outside link. The ULIA in Hellos encloses the complete set of information to be advertised in the uplink advertisement that the receiving outside neighbor generates.

Uplinks may also be derived from other uplinks. This case occurs when one or more border nodes in a child peer group are advertising an uplink to an upnode which is at a higher level than this LGN's peer group. In this case, the LGN advertises a resulting uplink to the same upnode. The details of how the LGN determines whether or not to aggregate multiple lower level uplinks into one higher level uplink are explained in Section 5.10.3.1.

In the case where an uplink is derived from a single outside link or inducing uplink, the Uplink Information Attribute contained in the Hello received on the outside link or advertised for the inducing uplink, respectively, must be copied into the advertisement for the derived uplink (independent of the tags on the information groups contained within the ULIA).

In the case where multiple outside links or inducing uplinks exist to the same higher level node, these multiple links may be aggregated into a common uplink from this node (see Section 5.10.3.1). In that case, the information groups contained in the ULIA being generated for the aggregate uplink are derived (aggregated) from the set of individual information groups enclosed in ULIAs contained in the Hellos received on the outside links or advertised for the inducing uplinks, respectively. Whether any unknown information groups contained in the ULIAs being aggregated are included in the advertisement for the resulting uplink is controlled by the information group tags associated with those information groups (see Section 5.14.2 for details). Since this node is constructing the topology state parameters being advertised for the aggregated uplink, it is also responsible for indicating when there is a significant change in the values it has constructed. This means that it has to generate appropriate values for the ULIA sequence number for the aggregated uplink.

Nodes can only use uplinks in their route computations and specify them in DTLs if they understand all mandatory information groups of the uplink. In addition, as an exception to the normal rules for the scope of tags, the node must also understand all mandatory information groups inside the ULIA of the uplink. For example, if there exists an information group inside the ULIA wrapper that is unrecognized and tagged as mandatory the corresponding link must not be used by route computation.

5.10.3 Topology Aggregation

Topology aggregation is the process of summarizing the topology information of a child peer group to reduce the volume of information advertised in the parent peer group. Effective aggregation is required to allow PNNI to scale to support large networks.

Link aggregation is the process of representing several parallel links as a single higher-level link. The topology state parameters for such a link are derived from those for the individual links being aggregated.

The PNNI complex node representation is a flexible scheme for describing the connectivity within a logical node. When a logical group node produces a complex node representation, it makes a tradeoff between the accuracy of that representation and its size. Alternatively, it may use the simple node representation, in which the entire LGN is treated as a point, with no resource constraints.

The algorithms used to derive the aggregated topology description are implementation specific.

5.10.3.1 Link Aggregation and Binding Information

Aggregation of a set of outside links between the same two peer groups is configured through the Aggregation Token values in the border nodes at which the outside links in question terminate. The information about link aggregation is configured into border nodes rather than logical group nodes to ensure consistent behavior when peer group leaders change. All links between a pair of logical group nodes with the same value of the Aggregation Token must be advertised by the LGNs as one logical link. The scope of an Aggregation Token is limited to links between one pair of logical group nodes. The same token value may be used for links between other pairs of nodes without confusion.

The aggregation token is communicated in the Hello protocol between two lowest level neighbor nodes connected by an outside link. The value of the token in Hellos between two lowest level neighbor nodes does not change unless it is reconfigured. This value is referred to as the Configured Aggregation Token in the subsequent text.

Once the configured aggregation tokens have been exchanged, each lowest level neighbor node runs the same algorithm on the same information to determine common values of the Derived Aggregation Token, which is the value advertised in PTSEs for the corresponding uplinks. Whereas Hellos between lowest level nodes include the value of the configured aggregation token, PTSEs and Hellos between logical group nodes always include the value of the derived aggregation token.

The derivation process was designed with minimal configuration in mind, as well as acceptable behavior in the face of misconfiguration. By default a link has the configured aggregation token value of zero, which has special significance. Any other value is established through configuration. The distinguished value zero conveys the meaning that the link does not care about aggregation behavior. In that case, the node will use as the derived aggregation token value whatever value the remote end has. If no configuration is done then all links will end up with a derived aggregation token value of zero.

The algorithm for computing the value of the derived aggregation token is:

1. If neighbors exchange identical values of the configured aggregation token, that value is the derived aggregation token value.
2. If the configured aggregation token value of one neighbor is zero, and that of the other neighbor is non-zero, the non-zero value is the derived aggregation token value.
3. If the configured aggregation token values of the neighbors are different and both non-zero, the derived aggregation token value is zero.

To effect disaggregation, at least one end of the link needs to be configured. If all links have zero configured aggregation token values, then only one end of a link need be reconfigured to cause disaggregation of that link.

The above algorithm runs continuously. When a change of the configured aggregation token takes effect, the new value is communicated immediately in the outside Hellos to the lowest level neighbor node. If this causes any change in the derived aggregation token value, new instances must be issued of all PTSEs containing links affected by the change in the derived aggregation token value. Note that this may cause an immediate change in the aggregation of links all the way up the hierarchy.

For lowest level nodes, uplinks to the same higher level neighbor node with the same derived aggregation token value may be aggregated or disaggregated in any manner desired by that node. However, uplinks with different derived aggregation token values must not be aggregated. If the lowest level node has a neighbor at the same level that is a logical group node, then all horizontal links to that logical group node with the same derived aggregation token value must be aggregated and advertised as a single link.

For logical group nodes, all links to the same neighbor node with the same derived aggregation token value must be aggregated and advertised as a single link. That is, if a peer group has multiple uplinks to the same upnode with the same derived aggregation token value, then the logical group node representing the peer group must aggregate these uplinks into a single uplink or horizontal link.

Since the default value of the configured aggregation token is zero, the default behavior is to aggregate all links between two logical group nodes into a single link with derived aggregation token value zero.

The derived aggregation token value together with the remote node ID serves as the binding information between levels of the hierarchy. These values are communicated in PTSEs that describe uplinks or horizontal links. The values advertised in uplink PTSEs are used by the parent of this node to make further aggregation decisions at the next higher level of the hierarchy. In addition, the values advertised for either uplinks or horizontal links are used by border nodes during DTL processing. When an entry border node processes a received DTL stack with a non-zero port ID in the current entry of the topmost DTL, the selected path across this peer group must end at an uplink that was aggregated into the link indicated by that port ID (i.e., the uplink exiting the peer group has the same aggregation token value as that of the port ID specified in the DTL).

For consistency of format the derived aggregation token field is also carried in the horizontal link PTSEs for physical links and VPCs. This token must be set to zero upon transmission. Since there are no inducing uplinks below this horizontal link, the aggregation token value advertised for this horizontal link has no effect.

5.10.3.2 Simple Node Representation

The end-to-end QoS and traffic parameter values of a connection are affected by each of the nodes and links traversed by that connection. For cases where traversal of a node affects the end-to-end parameter values of connections insignificantly, it is simplest to model such a node as a single point. This representation allows for relatively quick routing computation by reducing the representation of the network topology to one with nodes represented as vertices and links represented as arcs. This representation is particularly useful for modeling lowest-level nodes implemented on single switches, where the limitations of the links attached to the node are much more significant than the limitations of the switch itself. For coding of the simple node representation, see Sections 5.8.1.2.3 and 5.14.9.1.

5.10.3.3 Complex Node Representation

A complex node representation is a collection of nodal state parameters that provide detailed state information associated with a logical node. It is used to express the case where traversing into or across a node has a significant effect on the end-to-end parameter values of connections.

To accommodate traversing a logical node as well as routing to the “inside” of the node, a symmetric star topology with a uniform “radius” is used. The center of the star is the interior reference point of the logical node, and is referred to as the nucleus. The logical connectivity between the nucleus and a port of the logical node is referred to as a spoke. PNNI Routing supports a default node representation, which consists of a single value for each nodal state parameter, giving a presumed value between any entry or exit of the logical node and the nucleus, in either direction.

For each nodal state parameter associated with a logical node, a “radius” is derived from the “diameter” of the logical node. For a nodal metric, the “radius” is simply half the “diameter”. For a nodal attribute, the “radius” is the same as the “diameter”. PNNI Routing does not specify exactly how the aggregation is taken to determine the “diameter”. A conservative advertiser may take worst case values. Aggressive advertisers may consider the average case, or even the best case.

The default node representation offers the greatest reduction of advertised information (short of using the simple node representation). However, it cannot fully capture the multiple connectivity in a typical logical node or reflect asymmetric topology information.

Given that a logical node is in general not perfectly round, PNNI Routing permits the topology state parameters associated with any given spoke to be different from the default “radius”. In addition, direct port-to-port connectivities (called a “bypasses”) may also be advertised.

With this flexibility, one may advertise practically any aggregated topology ranging from a symmetric star to a full mesh. A connectivity advertisement that represents something other than the default node representation is called an exception.

The complex node representation for PNNI Routing can be constructed as described below:

1. Conceptually overlay on each logical node a star topology with a nucleus representing the “inside” of the corresponding node, and spokes connecting the ports of the logical node to the nucleus. Each port ID must be the same as the port ID used to identify the link or reachable addresses associated with the port.
2. For each nodal state parameter, advertise a “radius” to be used as the default value for the spokes.
3. Any spoke or any logical connectivity between a pair of ports may be designated as an “exception”.
4. For each such “exception”, advertise the entire set of nodal state parameters associated with it. For bypasses, nodal state parameters must be specified in both directions.
5. For each spoke advertised as an exception, the exception nodal state parameters supersede the default information in the directions in which the exceptions are specified.
6. A path through the logical node is obtained from a concatenation of any number of bypasses and at most two spokes (default or exception) in the complex node representation.

With the above complex node representation, one may choose to advertise conservatively or aggressively depending on parameter values assigned to the “radius” and “exceptions”.

PNNI Routing does not specify how spokes and bypasses are selected to be advertised as exceptions. Some guidelines are provided in Appendix C.

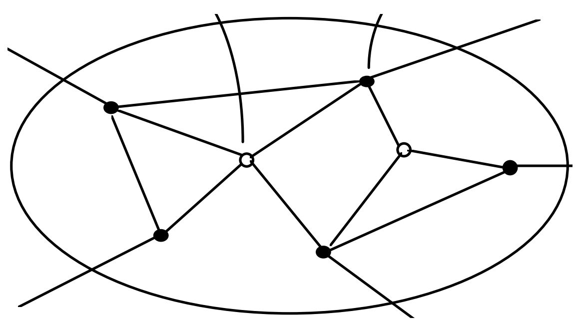


Figure 5-8: Peer Group with Five Border Nodes

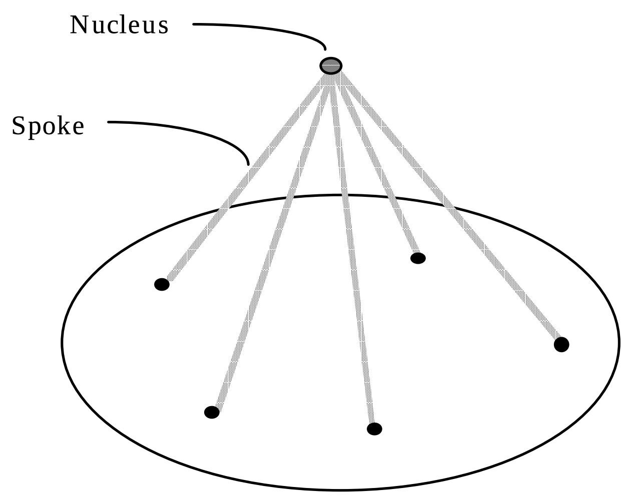


Figure 5-9: Default Node Representation

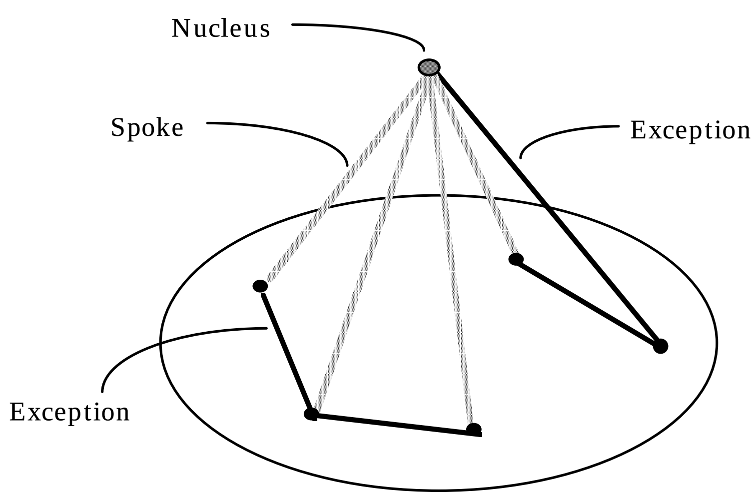


Figure 5-10: Complex Node Representation with Exceptions

5.10.4 Feeding Information Down the Hierarchy

The hierarchical summarization described above allows nodes in the highest level of the hierarchy to calculate routes to any destination represented in the highest level peer group (including systems reachable via lower levels, which are announced via summary address prefixes). However, it is necessary for all nodes in the PNNI network to be able to route calls to any destination, not just nodes actually at the highest level of the hierarchy. This implies that the topological information describing the higher levels of the hierarchy must be available to the lower level nodes.

This requires that all nodes participating in PNNI routing maintain information in their topology databases (and the capability of calculating routes) not only for their own peer group, but also for all of their ancestor peer groups.

The higher level PTSEs are flooded to all nodes of their peer group, and in addition are flooded to all nodes of all descendant peer groups, allowing all nodes to directly calculate appropriate routes (even those nodes which are not explicitly represented in the summarized higher level topology).

Flooding of PTSEs to all nodes of all descendant peer groups (i.e., to all lower-level nodes contained in the lower-level peer groups represented by the nodes in this peer group, and so on), is achieved as follows:

- When originating a new PTSE or updating a PTSE that it had previously originated, a higher-level node floods the PTSE to the PGL of the peer group that the higher-level node represents, as well as the usual process of flooding to all neighboring peers at its level. The PGL will in turn flood the PTSE in the child peer group.
- When flooding a received PTSE that is new or more recent than its topology database copy, a higher-level node floods the PTSE to the PGL of the peer group that the higher-level node represents, as well as the usual process of flooding to all neighboring peers at its level other than the one from which the PTSE was originally received. The PGL will in turn flood the PTSE in the child peer group.

PTSEs generated in a given peer group never get flooded to the next higher level peer group. Instead, the peer group leader summarizes the topology of the peer group based on the PTSEs generated within the peer group, but the summary is flooded in new PTSEs originated by the LGN at the parent peer group's level.

Under normal circumstances the peer group leader has the most current instances of PTSEs generated at higher levels. There are abnormal cases, however, in which PTSEs generated at a higher level exist in a given peer group with sequence numbers larger than the current ones. One possible scenario is when a node in a lower-level peer group is isolated from the rest of the peer group while some PTSEs known to the isolated node are prematurely aged, flushed and re-originated by a node in a higher-level peer group. When the isolated lower-level peer group gets reconnected to the rest of the lower-level peer group, it may still have old copies of the PTSEs with possibly invalid information and sequence numbers that are larger than the most current ones.

In such a case, the peer group leader will eventually get the invalid PTSEs and realize that the PTSEs are incorrect. The peer group leader detects such a PTSE when:

- The PTSE is encapsulated in a PTSP where the level of the PTSE's originator's logical node ID is higher (smaller in value) than the level of the peer group that the PGL belongs to, and
- Either the PGL has no instance whatsoever in its database of the PTSE in question or the PTSE is determined to be more recent than the peer group leader's database copy.

When such a PTSE is detected, the peer group leader flushes the PTSE in the peer group by setting the PTSELifetime to ExpiredAge and the PTSE sequence number to that in the received PTSE, and flooding it throughout the peer group. If thePGL had no copy of the PTSE in its database, the PGL need not take any further action. Otherwise processing continues as follows.

The peer group leader needs to keep both that copy and the PTSE being flushed until the flushed PTSE can be removed (see Section 5.8.3.9). Having initiated the flush the PGL determines an appropriate initial amount of time to wait before attempting to flood its valid (but still less recent) instance by multiplying the number of nodes in its peer group by PTSERetransmissionInterval, and assigns this number of seconds to a variable called TimeToFlush. It then starts a timer that will fire after TimeToFlush seconds have elapsed, and waits for it to fire. If the PGL receives the PTSE it is trying to flush again via flooding with something other than ExpiredAge before the timer fires, it again initiates flushing and restarts the timer with a new value. The new value to use is the lesser of double its previous value and 8 times PTSERetransmissionInterval times the number of nodes in the peer group. If the previous value was already 8 times PTSE RetransmissionInterval times the number of nodes in the peer group, an error should be logged to network management. The number of nodes in the peer group means the number of nodes that are connected as defined for the PGL election (see Section 5.10.1.1). If the timer expires, then the valid upper level PTSE is re-flooded into the peer group.

5.10.5 Flushing PTSEs in a Multi-Level Hierarchy

As described in Section 5.8.3.8, a PTSE is flushed either when its originator decides to prematurely age the PTSE due to the information in the PTSE becoming invalid or when the PTSE naturally ages out, i.e., its PTSE Remaining Lifetime reaches ExpiredAge.

When a node flushes one of its own PTSEs, it follows the procedures described in Section 5.8.3.8 to flood the expired PTSE to its neighboring peers at its level. In addition, if the node is not at the lowest level of the hierarchy, it floods the expired PTSE to the PGL of the peer group that this node represents. The PGL in turn floods the expired PTSE in the child peer group, and so on, as decribed in Section 5.10.4.

When a PTSE ages out (the PTSE necessarily belongs to another node), the logical node within the switching system that originally received the PTSE must flush the PTSE. This ensures that a widest scope of flushing is achieved. To flush a PTSE, a node follows the flushing procedures described in the previous paragraph.

Note: The PNNI specification does not specify how an implementation organizes PTSEs into databases. In an implementation, it may be possible for logical nodes within a switching system to age out a PTSE independently of one another. The requirement stated above does not preclude logical nodes other than the one which originally received the PTSE from flushing the expired PTSE as well.

There are cases where a PGL receives an expired PTSE that was originated at a higher level from within the peer group while its own database copy of the PTSE has not yet been aged out. These cases are handled similarly to the abnormal cases discussed in Section 5.10.4. The difference is that when the PGL receives the flush, it does not "reflush" the PTSE, but instead it forwards the flushed PTSE as usual, keeps both the expired copy and its own database copy, sets the TimeToFlush timer and follows the rest of the procedures as described in Section 5.10.4.

5.11 Peer Group Partitions

A partitioned peer group is a peer group which has been accidentally split into two or more non-contiguous pieces. This may happen due to failure of equipment (such as failure of links or switches). Each partition operates independently as a peer group. For example each partition elects a peer group leader. This implies that at the next higher level the partitioned peer group appears as multiple logical group nodes (assuming that each partition does elect a PGL). Each partition must be represented correctly and unambiguously in the hierarchy, and the routing computation must operate correctly in the presence of multiple partitions of a peer group.

5.11.1 Representation of the partition in the hierarchy

Each partition elects a peer group leader (PGL). The PGL election algorithm ensures that if a peer group partitions then each partition will elect its own PGL (assuming that there is a node capable of being PGL). Similarly, if a partitioned peer group re-attaches, then only one node will remain as PGL. The PGL is responsible for determining the identity of the parent peer group, and representing the partition in the parent peer group.

In some cases, partitions may have no peer group leader (for example, a small partition might not contain any nodes capable of acting as peer group leader). In this case the partition operates as a "leaderless peer group", as described in Section 5.10.1.4, and is effectively isolated from the rest of the PNNI routing domain.

In order for the parent peer group to operate correctly, each node in the parent peer group must have a unique node ID. This can be achieved by including the peer group leader's 48-bit ESI into the "flat ID" part of the node ID (see Section 5.3.3). This ensures that when a partition occurs, each partition is automatically represented in the parent peer group by a logical group node with a unique node ID. This does require that each node which is capable of being PGL must have a 48-bit ESI which is unique within its parent peer group. If alternative methods for creating node IDs are used they must provide for the creation of unique IDs in presence of partitions.

Note: There is a danger that the ESIs will not be unique, especially in higher-level peer groups that span countries.

5.11.2 Routing in the Presence of Partitions

Given that each partition of a peer group is treated as a separate peer group, routing within the parent peer group naturally routes calls around and across the various partitions. Similarly, calls originated within the partitioned peer group to destinations outside of the peer group are correctly routed.

There is a potential problem with routing calls to destinations within the partitioned peer group.

In many cases the set of end systems reachable in a peer group will be summarized using a single address prefix (or a small number of address prefixes). The problem with a partitioned peer group is that each partition may be advertising the ability to reach the same prefix, where in fact only some (not all) of the systems whose addresses match the prefixes are in any one particular partition. This implies that routing to a system within the partitioned peer group may end up in the wrong partition. The entry border node of a partitioned peer group will determine whether the destination address specified for the call is in fact reachable in its partition. If not, and if the same summary address is advertised elsewhere in the network (presumably by another partition) then the call is cranked back and the DTL Originator can construct a new DTL to another partition. Refer to Annex B for details.

5.11.3 What to Aggregate as Peer Group Leader

PTSEs are labeled with the peer group ID of the peer group in which they were generated. All partitions of a peer group have the same peer group ID. Thus, when a peer group has become partitioned, the topology databases of the nodes contain PTSEs from nodes in their own partition, plus (possibly out of date) PTSEs from nodes in other partitions.

When an LGN aggregates and/or summarizes the topology state information of its child peer group (partition), it must only use the topology state information that pertains to its own partition of its child peer group. Similarly, when running the Peer Group Leader election algorithms, a node is restricted to considering only those nodes in its own partition. The algorithm for determining which subset of PTSEs belongs to a node's own partition (i.e., PTSEs that were originated by nodes to which this node has connectivity) is specified in Section 5.10.1.

5.12 Operation of a Node when in a Topology Database Overload State

A node is said to be in a topology database overload state when it is unable to store the complete topology database it is required to maintain by the normal PNNI algorithms. This would happen if there is insufficient memory in the node to store all the PTSE data currently active in the peer group (including any PTSEs that describe ancestor peer group topology).

While a node in a topology database overload state is not able to operate fully as a normal PNNI node, it is nevertheless desirable for it to continue operating normally in most respects, consistent with the requirement that such operation must not drastically disrupt the correct operation of other nodes in the network. This section describes requirements and recommendations for operation while in a topology database overload state.

5.12.1 Requirement for operation in topology database overload state

5.12.1.1 Minimum state required

A node is allowed to continue operating, under the rules stated here, even when it can store only a small subset of the topology database. However, it must at all times be able to store and advertise the following:

1. All required state describing the node itself
2. All topology state describing the links to its neighbors

If it cannot meet requirement (1), the node must halt. If it cannot meet requirement (2), it must not bring up links for which it cannot store the associated state.

5.12.1.2 Disallowed Functions

When in topology database overload state, the following restrictions apply to the normal operation of the node:

1. The node shall not bring up an outside link. Instead, it shall act in the same way as a node that does not support being a border node. See Section 5.6.2.1.4, in particular Note 1 on the state machine table.
2. The node shall not act as a DTL originator (i.e., no new outgoing calls are permitted).

5.12.1.3 PGL election

When in this state, a node must advertise that fact by setting the Non-transit for PGL Election flag in its nodal flags. It must also stop its PGL election FSM, and instead advertise zero as its Leadership Priority and its Preferred PGL.

5.12.1.4 PTSP reception and transmission

In PTSP reception, PTSE acknowledgements must still be sent according to the normal rules in Section 5.8.3.3. This applies whether or not the node is able to store the PTSEs in question. This requirement ensures that the neighbor node's PTSP transmission process requires no more resources (link bandwidth, retransmit list space) than under normal operation.

The node must, as a minimum, continue to originate and flood in the normal manner PTSEs describing its own state and that of any of its active links.

5.12.1.5 Periodic resynchronization

In order to recover from transient network faults that cause topology database overload, the node must periodically attempt to resynchronize with its neighbors. It does this by restarting all its Neighbor Peer FSMs (as if a DSMismatch event had occurred). This causes database synchronization to be performed with each neighbor.

If the database synchronization process completes and the node was able to store all PTSEs, it returns to normal operation by clearing the Non-transit for PGL Election flag in its nodal flags, and returning to normal execution of the PNNI algorithms. Otherwise, the node remains in topology database overload state.

The periodic resynchronization is done at an interval given by the architectural constant OverloadRetryTime.

5.12.2 Recommendations for operation in topology database overload state

This section gives recommendations for other aspects of operation while a node is in topology database overload state. Adoption of these recommendations in an implementation will result in minimizing the adverse impact of database overload on the operation of the rest of the network.

5.12.2.1 Call processing

The discussion below assumes that memory required for the topology database is separate from that used by call and connection state. If this is not the case, then the implementation should at least reserve some minimal amount of call state memory in order to support incoming network magement access.

Calls existing at the time the node entered into topology database overload state should be preserved to the extent possible.

New calls should also be permitted to the extent possible. In particular, it is important to accept calls that terminate at this node, since remote in-band network management access needs to be able to establish SVCCs to the overloaded node.

Transit calls should be handled normally, since this can be done by obeying the DTL in the SETUP message.

DTL origination (for call origination or entry border node processing) is necessarily inaccurate since the topology database is incomplete. However, it is still useful to allow it, since routes calculated using the incomplete topology database are still valid (though they may be suboptimal).

5.12.2.2 Topology database handling

A node in topology database overload state should store whatever subset of the currently active PTSEs it has space for. While it is legal to keep only a minimal topology database, the disruptive effect on other nodes in the network is reduced by keeping as much as possible of the active PTSEs. Any PTSEs actually stored in the topology database should of course be transmitted to neighbors according to the normal rules for PTSP transmission.

A consequence of the above rules is that a node in topology database overload state may cause a peer group to become partitioned, in the sense of having two peer group leaders, even though it is still possible to set up calls from one partition to another. This situation is peculiar but desirable, since the alternate would be to treat the overloaded node as “restricted transit”. If that were done, all calls from one partition to another would have to be routed via other peer groups, as described in Section 5.12.2, but the topology that remains may not allow this to be done.

5.13 Path Selection

When selecting a route to a destination ATM address, a node shall always route to the node which has advertised the longest prefix which matches the destination. If the only nodes with the longest matching prefix are ancestors then the destination is not reachable. Only when several nodes have advertised equal length matching prefixes which are all longer than any other advertisement may the calculating node choose on a local basis which destination to use. Of the nodes advertising equal longest matching prefixes ignore any ancestors and select among the remaining ones, if any.

PNNI allows for point-to-point connections to be established to group addresses (i.e., the ATM Anycast capability). In this case, a route shall be selected to one of the members of the group (this functionally is a consequence of application of the longest match rule to group address advertisements). This group member must be reachable within the indicated connection scope.

All addresses (individual as well as group addresses) have an advertisement scope. When constructing a path a node must ensure that the destination address has an advertisement scope at least as wide as the scope of the path across the PNNI routing domain. The process of address advertisement and summarization ensures that addresses are not advertised beyond their scope; therefore this check has no affect at the DTL originator, but it does have an effect at an entry border node. This implies that scope checking must be done by the entry border node when selecting a specific destination within the final peer group of the current DTL. Specifically the scope checking takes precedence over longest prefix match checking.

When a Transit Network Selection information element is present in the SETUP message, a call is routed to a node that has reachability to the Transit Network specified in the Transit Network Selection information element. In this case, path selection is based on the specified Transit Network Selection; optionally the destination address may also be considered. Nodes that have access to a particular transit network are advertised as such by Transit Network ID information groups included in Exterior Reachable Address information groups. Given multiple advertisements for the same transit network, the RAIG(s) and/or reachable addresses associated with the transit network may be used to select among entrances to that transit network.

In case the same transit network is reachable by many nodes it might be disadvantageous (more expensive) to leave the PNNI network at the nearest node. However, as a particular geographical area may be overlaid by the different networks it may even be more disadvantageous to leave at the more remote node. No specific help can be provided by PNNI Phase 1 on this matter.

In the case of path selection for point to multipoint connections, the node shall select a path so that the resulting connection is a tree. No two branches of a point to multipoint connection may have a link in common, nor may they have a node in common other than the node at which they branch. (Note that the effect of this rule is that a node is required to keep track of the tree representation of a point to multipoint connection, rather than keeping track only of the leaves.)

Note that when parallel links exist, the above constraint prevents more than one of the links from being used for a given point-to-multipoint call because if more than one parallel link were used, then the call must have branched earlier and reconverged on this horizontal link or uplink.

PNNI Routing provides a mechanism that prevents a point-to-multipoint tree from branching at nodes which cannot support additional branching. These nodes are advertised as Restricted Branching nodes, and a point-to multipoint path selection must avoid using these nodes for branching.

Note that maximum number of elements in a DTL (currently 20) and the maximum number of DTLs in a SETUP message (currently 10) impose limits on the diameter of peer groups and the number of levels of hierarchy respectively.

5.13.1 Eligible Entities for Path Selection

When computing paths across a PNNI routing domain, paths are chosen across a concatenation of PNNI entities. These entities include nodes and ports and the horizontal links, uplinks, spokes and bypasses across complex nodes that connect them, and/or routes from nodal ports to their internal and exterior reachable addresses and transit networks. For horizontal links and for spokes and bypasses across complex nodes, each entity is eligible for inclusion in a path if it has been advertised by both endpoints, with a set of topology state parameters for the requested service category advertised by each endpoint for its outgoing direction. For internal and exterior reachable addresses and transit networks, the presence of a PTSE including an advertisement to the reachable address or transit network is sufficient to make it eligible (topology state parameters are optional for exterior reachable addresses and transit networks). If internal or exterior reachable addresses or transit nework selection information includes topology state information, it is only eligible for inclusion if it supports the requested service category. A service category is supported either if topology state parameters are absent for all categories or present in both directions for that particular category.

For an uplink to be considered when computing paths from border nodes to other nodes, the uplink must have been advertised in an uplink PTSE including one set of topology state parameters for each direction, and supporting the requested service category. For an uplink to be considered when computing paths from DTL Originators across a PNNI routing domain, there is an additional constraint: a corresponding horizontal link must have been advertised from an ancestor of the node computing the path to the upnode. This is required to ensure that routes computed across the uplink are based on current information about the topology across and beyond the upnode. Additionally, the horizontal link information is required for the computation in order to know at what port the upnode is being entered. For example, if there is no SVCC-based RCC between the two neighboring peer LGNs, it is possible that the topology across and beyond the upnode has changed significantly since the last PTSEs about this topology were received. This can result in excessive failure of call setup messages bound for destinations beyond the upnode. The requirement for a corresponding horizontal link when choosing routes from the DTL Originator is only relaxed when computing routes between LGNs, for the purpose of establishing SVCC-based RCCs between the LGNs.

5.13.2 Link Constraints, Node Constraints, and Path Constraints

PNNI routing shall determine paths that satisfy performance constraints, but are not necessarily optimized with respect to any predetermined performance criteria. Performance constraints may be implemented in one of three ways: link constraints, node constraints and path constraints. For link constraints, non-additive link-state parameters are used. For node constraints, non-additive nodal state parameters are used. For path constraints, additive link-state parameters are needed. Link constraints and node constraints are used to prune the network graph during path selection. PNNI routing supports link constraint and node constraint implementation of Cell Loss Ratio and generic CAC parameters (i.e., Available Cell Rate, Cell Rate Margin, and Variance Factor). PNNI routing supports restricted-transit and restricted-branching node constraints. PNNI routing supports path constraint implementation of Administrative Weight, Maximum Cell Transfer Delay, and Cell Delay Variation.

5.13.3 CLR Selection for CBR andVBR Service

Connection requests for the CBR, rt-VBR and nrt-VBR service categories may specify a CLR objective. Depending on the combination of bearer class, traffic parameters, and QoS parameters the objective may relate to either CLP_0 or CLP_{0+1} traffic. The rules for determining which CLR objective applies are specifed in Annex 9 of UNI Signalling 4.0. The CLR objective for a given connection request is compared to the appropriate advertised CLR from links and nodes according to these rules.

5.13.4 Generic CAC Algorithm for CBR and VBR Services

5.13.4.1 Parameter Choice for GCAC

In the generic CAC described below, a connection is characterized by two traffic parameters: PCR and SCR. These two parameters are derived from traffic parameters specified in the ATM Traffic Descriptor IE. Table 5-7 in UNI 3.1 (p. 208) defines the following six allowable combinations of traffic parameters in the ATM User Cell Rate IE for CBR and VBR service categories (PCR = peak cell rate, SCR = sustainable cell rate, MBS = maximum burst size):

1. PCR(CLP=0); PCR(CLP=0+1)
2. PCR(CLP=0); PCR(CLP=0+1); tagging requested
3. PCR(CLP=0+1); SCR(CLP=0); MBS(CLP=0)
4. PCR(CLP=0+1); SCR(CLP=0); MBS(CLP=0); tagging requested
5. PCR(CLP=0+1)
6. PCR(CLP=0+1); SCR(CLP=0+1); MBS(CLP=0+1)

Combinations 1 and 3 produce the same traffic behavior as combinations 2 and 4, respectively. So, GCAC must ignore the “Tagging” subfield.

For any topology element along a candidate path, if the RAIG CLP bit is set to 0, this indicates that it allocates resources based on CLP=0 traffic. In that case, PCR and SCR values of the connection used in GCAC are set according to the following table:

Table 5-16: PCR and SCR values for CLP=0

Parameter Combination	PCR	SCR
1, 2	PCR(CLP=0)	PCR(CLP=0)
3, 4	PCR(CLP=0+1)	SCR(CLP=0)
5	PCR(CLP=0+1)	PCR(CLP=0+1)
6	PCR(CLP=0+1)	SCR(CLP=0+1)

For any topology element along a candidate path, if the RAIG CLP bit is set to 1, this indicates that it allocates resources based on CLP=0+1 traffic. In that case, PCR and SCR values of the connection used in GCAC are set according to the following table:

Table 5-17: PCR and SCR values for CLP=1

Parameter Combination	PCR	SCR
1, 2, 3, 4	PCR(CLP=0+1)	PCR(CLP=0+1)
5	PCR(CLP=0+1)	PCR(CLP=0+1)
6	PCR(CLP=0+1)	SCR(CLP=0+1)

5.13.4.2 Complex GCAC

Having selected the values for the parameters from the call to use in evaluating whether the selected link has sufficient bandwidth, the GCAC is used to perform the compairson. When CRM and VF are advertised for a given link, this complex GCAC is the recommended algorithm to use.

5.13.4.2.1 Algorithm

Note: Refer to Appendix B for a derivation of this algorithm.

Step 1. If AvCR(i) >= PCR, include the link, stop
Step 2. If AvCR(i) < SCR, exclude the link, stop
Step 3. If [AvCR(i)-SCR]*[AvCR(i)-SCR+2*CRM(i)] >= VF(i)*SCR(PCR-SCR),
  include the link
 Else
  exclude the link.
 Endif
Step 4. Stop

5.13.4.2.2 Special Cases

Note that if PCR=SCR (i.e., if SCR is not specified in the Traffic Descriptor IE), only Steps 1 and 2 above need to be performed, i.e., Steps 1 and 2 always give a conclusive answer and so Step 3 is not needed in this case.

Note also that in the extreme case where the advertised CRM and VF are zero, Step 3 will always result in "include". In the other extreme where the advertised VF(i) = infinity, Step 3 will always result in "exclude".

5.13.4.2.3 Optional Improvement to Algorithm

In the case multiple connection requests have to be handled in between PTSEs, the values of CRM(i) and AvCR(i) can be optionally updated locally to reflect the newly established connections. When any of these connections is taken down before a new PTSE is received, CRM(i) and AvCR(i) need to be updated, too. This option, however, is to be used with extreme caution as it can result in inconsistent views of the network between different nodes.

5.13.4.3 Simple GCAC

If only AvCR is advertised, use of the simple GCAC is recommended. the S-GCAC algorithm is:

Step 1. If AvCR >= C, include the link, stop
Else exclude the link, stop

where C is given by (let x = PCR/SCR)
if (x > 39) C = SCR * (0.0145 * x + 4.22);
else if (x > 5) C = SCR * (0.042 * x + 3.14);
else C = SCR * (0.48 * x + 0.52).

5.13.5 Generic CAC Algorithm for Best-Effort Service

For UBR connections, a link/node is included if and only if the UBR service category is supported and Maximum Cell Rate is not equal to zero.

For ABR connections, a link/node is included if and only if the ABR service category is supported, Maximum Cell Rate is not equal to zero, and the advertised Available Cell Rate for the ABR service category is greater than or equal to the Minimum Cell Rate specified by the connection.

5.13.5.1 Minimum Acceptable ATM Traffic Descriptor

If there is a Minimum Acceptable ATM Traffic Descriptor with a peak cell rate in the call, then that peak cell rate is compared with the advertised Maximum Cell Rate for determining admissibility of the call.

5.14 Packet Formats

The packet formats specified in this section have been designed to provide a flexible and expandable encoding for PNNI routing packets. In particular, they rely on the use of nested type-length-value (TLV) encodings. Each TLV entity is referred to as an information group. The type and length fields are each two octets long, in all information groups. The four most significant bits of the type field are used for encoding information group tags. Type values need only be unique among all possible type values that may occur at the same positions. Note that for PNNI Phase 1 the type values have been assigned in a globally unique manner, in order to make it easier for humans to determine the identity of each TLV information group. The value of the length field includes the lengths of the type and length fields (two octets each) as well as that of the value (the remainder of the information group).

Note that each TLV information group is arbitrarily extensible for future versions of PNNI routing. In particular, after the fixed fields at the beginning of the value part of each information group, several child TLV information groups may appear. Type values are only understood within the context of the containing information group. Specifically, even with the current globally unique assignment of type values, a type that is not defined to occur within a containing IG shall be treated as unknown rather than assuming the global definition. If any of these child information groups is not understood, its effect upon interpretation of the parent information group is determined by the values of the information group tags encoded in the type field of the child information group.

When necessary, all TLVs will be padded so that their lengths will always be a multiple of four octets. If padding is required, the TLV format in question will specify the formula for padding those octets.

All reserved fields and any padding fields must be set to zero by the originator and must be ignored upon reception.

Unless otherwise specified, all fields are unsigned integers. Multi-octet fields are transmitted in “network order” (big endian octet order), that is with the most significant octet occuring first in the octet stream.

5.14.1 Notation

Bit positions within an octet are specified by numbers 1 through 8, where 1 represents the least significant bit, and 8 represents the most significant bit of the octet. Bit positions within n-octet fields are specified by numbers 1 through 8*n, where 1 represents the least significant bit of the last octet, and 8*n represents the most significant bit of the first octet.

5.14.2 Information Group Tags

As described above, PNNI packets are constructed using information groups. Many of these information groups are defined in this specification. However, it is not possible to define all possible information groups. To provide for future extensibility, it is important that information groups be identified in such a way that systems which do not recognize them will process them properly. This will allow for the incremental addition of new features.

Other routing protocols have found that a viable technique to deal with the future growth of information groups is to tag them. For example, BGP and IDRP have a Mandatory vs. Optional tag, and a transitive vs. intransitive tag. While we cannot use exactly the same tags, we can use their experience.

In order to provide for consistent processing, all information groups have tags. These tags are only used for processing unrecognised information groups.

5.14.2.1 Mandatory Tag

The mandatory tag prevents systems which do not understand the information group from using the immediately containing information group. For example if the unknown mandatory information group occurs immediately within a horizontal link information group, then the described link must not be used in route computation.

If a system receives a PTSE with a top level mandatory tagged information group that is otherwise unknown, it must accept the PTSE, check the checksum and sequence number in the normal manner, and acknowledge, store, and forward the PTSE as appropriate in the normal manner. However, the network entity (node or link) described by that PTSE must not be used for any route computation, nor included in DTLs which are created resulting from a route computation.

However, a node receiving a call request containing a specified DTL must follow the DTL as specified independent of whether any element contained in the DTL may have been described using unknown mandatory information groups. The node receiving the call request must process the call request, and assume that the system which prepared the associated DTLs knew what it was doing (i.e., understood any mandatory information groups) and therefore was able to choose a correct DTL for the call. Similarly, if a higher level DTL specifies use of a specific link, which maps to a particular lower level uplink or uplinks, then a node selecting a corresponding lower level DTL must choose a corresponding lower level uplink in its DTL, regardless of any unknown mandatory information groups on the uplink. This is described in more detail in Section 6.

For phase 1 all information groups defined in this specification must be recognized.

5.14.2.2 Summarizing Tag

The PGL prepares summaries of the peer group information for use at the next level up. If some of the information contains unknown information groups, the PGL must decide what to do with the information. Note that this decision is independent of the Mandatory tag.

An unknown information group may be tagged to indicate that the PGL must not summarize the containing entity into the parent peer group. Internal links and nodes containing unknown non-summarizable information groups would not be considered in calculating the default node representation based on peer group transit characteristics. Reachability information containing unknown non-summarizable information groups would not be advertised from the higher level logical group node. Uplinks containing unknown non-summarizable information groups are ignored in forming higher layer uplinks and horizontal links. One effect of this rule is that if all the uplinks with a given aggregation token have unknown non-summarizable information groups, then the induced uplink or horizontal link will not be created.

5.14.2.3 Transitive Tag

Assuming that an item with an unknown information group may be summarized, there is still the question of what to do with the individual information group. If the information group is tagged non-transitive, then the information group must be removed prior to summarization. If it is tagged transitive, then the information group must be preserved.

With items which are exposed individually, such as a single outside link, preservation of a transitive information group is not a problem. With aggregated entities there is more question.

If an information group is tagged as transitive, and is aggregated into an advertisement in the parent peer group, and the information group is not known by the peer group leader, then the resulting advertisement must contain the information group (still tagged as transitive and summarizable). Note that this means that if a single node in a peer group has a summarizable transitive unknown metric, then that information group will be applied to the entire peer group.

If two advertisements which are being aggregated both carry an information group of the same transitive type, but with different values, then the PGL will preserve both information groups, implying that the same information group may occur multiple times.

5.14.2.4 Mandatory Non-Transitive Information Groups

There is a slight complication involving information groups which are mandatory, but non-transitive. In this case the fact that the information group is mandatory implies that the associated node or link cannot be advertised at a higher level unless the mandatory information group is included. The fact that the information group is non-transitive implies that if the associated network entity is aggregated at a higher level, then the information group cannot be included in the summary entity advertisement.

The solution here is that a mandatory non-transitive information group attached to the advertisement of a network entity implies that the entity cannot be summarized for advertisement at higher levels. Thus, if an uplink is marked as mandatory and non-transitive, then that outside link cannot be aggregated with other links in the corresponding uplink advertisement. If a node or internal link within a peer group is marked with a mandatory non-transitive information group, then that node or internal link is not used for purposes of calculating the characteristics of the corresponding logical group node.

5.14.2.5 Examples

In order to strengthen the case for these information group tags, a few examples information groups that might be added later, and their tags, are given.

5.14.2.5.1 Advisory Metrics and Attributes

As the technology evolves, we may create new metrics, attributes, or service related characteristics. While these may be needed for some calls, clearly calls which do not need them may still use the links, even if the nodes have no understanding of these capabilities.

Thus, these would be Optional, Summarizable information groups. Whether or not such information groups are transitive probably depends on the details of their aggregation behavior, and so will vary from information group to information group.

5.14.2.5.2 Local Information Groups

A particular customer may persuade one or more providers of equipment to provide new information. Such information is probably ignorable, but because it is experimental it must not be summarized outside the domain of the experiment. The tagged elements are still usable externally. As such, these information groups will usually be Optional, Summarizable, Non-Transitive information groups.

On the other hand, one might have a special purpose link whose usage was limited to a certain class of user. In order to ensure that a sloppy peer group leader did not let the advertisement out, this link might have an information group identifying the class of permitted users. This advertisement would be Mandatory and Non-Summarizable.

5.14.2.5.3 User lists

We may at some time define information groups for carrying lists of which users are permitted or not permitted to use a resource (link or node). We would then define these information groups as Mandatory and Non-transitive. Making this a Non-Transitive information group would ensure that during deployment, the resources were made externally visible explicitly rather than through an aggregation method which would misrepresent the restriction.

5.14.2.5.4 Security Labels

Without delving into all the aspects of security, one can envision at some future time creating a security label. These would probably be Mandatory, Summarizable, Transitive information groups.

5.14.2.6 Information Group Tag Bit Definitions

All PNNI types are encoded in 16 bit fields. The most significant 4 bits of this field are reserved for the information group tags. Specifically:

Mandat	D-Sum	Trans	Reserv

are the four most significant bits.

- Mandat = The Mandatory bit (bit 16). This bit is set to 1 for information groups where, if this information group is not understood the contain IG or PTSE must not be used in path calculation.
- D-Sum = the Don't Summarize bit (bit 15). This bit is set to 1 for information groups where, if this information group is not understood the containing IG or PTSE must not be summarized into the containing peer group.
- Trans = the Transitive bit (bit 14). This bit is set to 1 if this information group is to be preserved when the containing IG or PTSE is aggregated and the information group is not understood by the peer group leader.
- Reserv = a reserved bit (bit 13). As with all reserved fields, this must be 0 on transmission and ignored on reception.

All PNNI 1.0 information groups shall be originated with their information group tags set to optional, summarizable, non-transitive, with one exception; the Transit network ID information group shall have information group tag values optional, summarizable, transitive. The reason for choosing these default values is that most of the currently defined information groups are self-contained, in that loss of the information does not change the meaning of parent or sibling information groups. So, although the unrecognized information must be dropped since it is not understood how to summarize it, the containing information group can be summarized.

5.14.3 Information Group Summary

Table 5-18: Information Group Summary

Type	IG Name	Contains IGs one level down
32	Aggregation token	
33	Nodal hierarchy list	
34	Uplink information attribute	Outgoing resource availability (128)
35	LGN horizontal link extension	
64	PTSE	Nodal state parameters (96), Nodal information group (97), Internal reachable ATM addresses (224), Exterior reachable ATM addressees (256), Horizontal links (288), Uplinks (289), System Capabilities (640)
96	Nodal state parameters	Outgoing resource availability (128)
97	Nodal information group	Next higher level binding information (192)
128	Outgoing resource availability	Optional GCAC parameters (160)
129	Incoming resource availability	Optional GCAC parameters (160)
160	Optional GCAC parameters	
192	Next higher level binding information	
224	Internal reachable ATM addresses	Outgoing resource availability (128), Incoming resource availability (129)
256	Exterior reachable ATM addresses	Outgoing resource availability (128), Incoming resource availabillity (129), Transit network ID (304)
288	Horizontal links	Outgoing resource availability (128)
289	Uplinks	Uplink information attribute (34), Outgoing resource availability (128)
304	Transit network ID	
384	Nodal PTSE ack	
512	Nodal PTSE summaries	
513	Requested PTSE header	
640	System capabilities	

Table 5-18: Information Group Summary continued

Type	IG Name	Contained in IGs one level up	Contained in packets
32	Aggregation token		Hello (1)
33	Nodal hierarchy list		Hello (1)
34	Uplink information attribute	Uplinks (289)	Hello (1)
35	LGN horizontal link extension		Hello (1) for LGN horizontal Hello
64	PTSE		PTSP (2)
96	Nodal state parameters	PTSE — restricted IG	PTSP (2)
97	Nodal information group	PTSE — restricted IG	PTSP (2)
128	Outgoing resource availability	Uplink information attribute (34), Nodal state parameters (96), Internal Reachable ATM Address (224), Exterior reachable ATM addresses (256), Horizontal links (288), uplinks (289)	Hello (1), PTSP (2)
129	Incoming resource availability	Internal Reachable ATM Address (224), Exterior reachable ATM addresses (256)	PTSP (2)
160	Optional GCAC parameters	Outgoing resource availability (128), Incoming resource availability (129)	Hello (1), PTSP (2)
192	Next higher level binding information	Nodal information group (97)	PTSP (2)
224	Internal reachable ATM addresses	PTSE — restricted IG	PTSP (2)
256	Exterior reachable ATM addresses	PTSE — restricted IG	PTSP (2)
288	Horizontal links	PTSE — restricted IG	PTSP (2)
289	Uplinks	PTSE — restricted IG	PTSP (2)
304	Transit network ID	Exterior reachable ATM addresses (256)	PTSP (2)
384	Nodal PTSE ack		PTSE Ack (3)
512	Nodal PTSE summaries		DBSummary (4)
513	Requested PTSE header		PTSE Request (5)
640	System capabilities	PTSE	all packets

Table 5-19: Information Groups in PNNI Packets

Type	Packet Name	Contains IGs
1	Hello	Aggregation token (32), Nodal hierarchy list (33), Uplink information attribute (34), LGN horizontal link extension (35), Outgoing resource availability (128), Optional GCAC parameters (160), System capabilities (640)
2	PTSP	PTSE (64), Nodal state parameters (96), Nodal information group (97), Outgoing resource availability (128), Incoming resource availability (129), Next higher level binding (192), Optional GCAC parameters (160), Internal reachable ATM addresses (224), Exterior reachable ATM addresses (256), Horizontal links (288), Uplinks (289), Transit network ID (304) , System capabilities (640)
3	PTSE ACK	Nodal PTSE Ack (384) , System capabilities (640)
4	DBSummary	Nodal PTSE summaries (512) , System capabilities (640)
5	PTSE Request	Requested PTSE header (513) , System capabilities (640)

5.14.4 PNNI Packet Header

All PNNI routing packets begin with a common PNNI packet header, shown in the following table.

Table 5-20: The PNNI Packet Header

Offset	Size (Octets)	Name	Function/Description
0	2	Packet Type	See Table 5-21 PNNI Packet Types.
2	2	Packet length	
4	1	Protocol version	Specifies the version according to which this packet was formatted. This specification defines version one of the PNNI routing protocol packet formats.
5	1	Newest version supported	The newest version supported and oldest version supported fields are included in order for nodes to negotiate the most recent protocol version that can be understood by both nodes exchanging a particular type of packet.
6	1	Oldest version supported	See Above.
7	1	*Reserved*	

Table 5-21: PNNI Packet Types

Packet Type	**Packet Name**
1	Hello
2	PTSP
3	PTSE Acknowledgement
4	Database Summary
5	PTSE Request

5.14.5 The Resource Availability Information Group

Before proceeding with specification of the various types of PNNI packets, specification is first given of two TLV information groups that occur at several places in the PNNI packet formats. These are the outgoing and incoming resource availability information groups, which are used to attach values of routing metrics and attributes to nodes, links, and reachable addresses. For nodes, only the outgoing resource availability information group is used. For links and reachable addresses, the outgoing resource availability information group (type = 128) is used to specify routing metrics and attributes from this node to the neighbor node or reachable address. The incoming resource availability information group (type = 129) is used to specify routing metrics and attributes from the neighbor node or reachable address to this node. Except for the initial type code, the two resource availability information groups are identical.

Each resource availability information group applies to a set of service categories, which are described using a bit-mask encoding. This encoding allows for one information group to be used to specify routing metrics and attributes for several service categories, when the values of all shared routing metrics and attributes are identical for all specified service categories. Metrics or attributes that are not applicable to certain service categories must be ignored when considered for use with those service categories. In order to advertise different routing metrics and attributes for different sets of service categories, separate resource availability information groups must be used. A resource must not be used to support a service category unless the category is specified in the bit-mask of a Resource Availability Information Group advertised for the resource.

Table 5-22: The Resource Availability Information Group

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 128 for outgoing resource availability information Type = 129 for incoming resource availability information
2	2	Length	
4	2	RAIG Flags	For Bit definitions see Table 5-23 RAIG Flags.
6	2	*Reserved*	
8	4	Administrative Weight	default value = DefaultAdminWeight, additive
12	4	Maximum Cell Rate	Units : cells/second
16	4	Available Cell Rate	Units : cells/second
20	4	Cell Transfer Delay	Units : microseconds
24	4	Cell Delay Variation	Units : microseconds
28	2	Cell Loss Ratio (CLP=0)	Encoded as the negative logarithm of the value, i.e., the value n in a message indicates a CLR of 10^{-n}
30	2	Cell Loss Ratio (CLP=0+1)	Encoded as the negative logarithm of the value, i.e., the value n in a message indicates a CLR of 10^{-n}
Optional GCAC related information:			
32	2	Type	Type = 160 (optional GCAC parameters)
34	2	Length	
36	4	Cell Rate Margin	Units : cells/seconds
40	4	Variance Factor	units of 2^{-8}. Note : the value of 0xFFFFFFFF for Variance Factor is used to indicate infinity

Table 5-23: RAIG Flags

Bit ID:	Bit 16 (MSB)	Bit 15	Bit 14	Bit 13	Bit 12	Bits 11..2	Bit 1 (LSB)
Meaning:	CBR	rt-VBR	nrt-VBR	ABR	UBR	*Reserved*	GCAC CLP Attribute

5.14.6 Uplink Information Attribute

Uplink advertisements made by border nodes and PGLs as well as Hellos sent on outside links, must include an Uplink Information Attribute (ULIA). The ULIA is basically a TLV encoded container or wrapper used to enclose and identify other information groups that apply to the reverse direction on uplinks.

A Resource Availability Information Group must be included for each service category supported. Optionally any other information group needed to describe any other aspect of the uplink, will also be included here.

Table 5-24: The Uplink Information Attribute

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 34 (uplink information attribute)
2	2	Length	
4	4	Sequence number	
• All outgoing Resource Availability Information Groups (type = 128) to describe the reverse direction of the uplink.			
• Any additional optional IGs needed to describe the reverse direction of the uplink.			

5.14.7 Transit Network ID

Transit network identification information is needed in PNNI. This information group is used to carry such information.

This information group may appear in the exterior reachable address IG.

Table 5-25: Transit Network ID

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 304 (transit network ID)
2	2	Length	
4	2	Length of TNS	Number of octets in the Network Identification string + Network Identification Data.
6	1	Network identification data	See Table 5-26.
7	*N*	Network identification	IRA characters. 1 octet per character. See ITU-T T.50 (1992) International Reference Alphabet (IRA) (Formerly International Alphabet No. 5 or IA5).
7+N		Padding	0-3 octets Padding is calculated using the formula: (4 **-** [(3 + N) modulus 4]) modulus 4

Table 5-26: Network Identification Data

Note: The values of the Network identification data fields are as specified by ITU.

Bit ID:	Bit 8 (MSB)	Bits 7..5	Bits 4..1
Description:	*Reserved*	Type of network identification	Network identification plan

5.14.8 PNNI Hellos

PNNI Hellos are transmitted by each node over (a) all physical links to immediate neighbor nodes, (b) all virtual path connections for which this node is an endpoint, and (c) all SVCCs established for the purpose of exchanging PNNI routing information for which this node is an endpoint. The purpose of exchanging Hellos between neighbor nodes is in order to establish the state of the link(s) between them.

Table 5-27: The PNNI Hello

Offset	Size (Octets)	Name	Function/Description
0	8	PNNI Header	A PNNI packet header structure with Packet type = 1 (Hello). See Table 5-20.
8	2	Flags	*Reserved*
10	22	Node ID	Node ID of the originating node.
32	20	ATM End System Address	A system which implements PNNI routing must have a unique ATM End System Address for each node which it represents, and must be able to accept calls to that ATM End System Address (for example, this may be used for network management). For split systems and for peer group leaders, up to 256 unique ATM End System Addresses may be realized by using the selector octet to differentiate between the ATM End System Addresses corresponding to different nodes.
52	14	Peer Group ID	Peer group ID of the originating node.
66	22	Remote node ID	set based on value of node ID in received Hellos on same link, set to all zeros if not yet known
88	4	Port ID	• A valid port ID is not allowed to have the value zero. • For cases where either one or both ends are acting as peer group leaders (Hellos are being exchanged over SVCCs rather than over VCI 18), the port ID takes the value 0xFFFFFFFF.
92	4	Remote Port ID	Set based on value of port ID in received Hellos on same link. Set to zero if not yet known.
96	2	Hello Interval	Hello Interval indicates how frequently the Hello packet is sent. If no Hello is received within a time period of (InactivityFactor times Hello Interval) then the link is assumed to have failed.
98	2	*Reserved*	

Hellos sent over links destined for outside of this node's peer group include the **Aggregation Token**, **Nodal Hierarchy List**, and **Uplink Information Attribute** information groups (see Section 5.6.2.2). The **Uplink Information Attribute** is described in Table 5-24.

Hellos sent between LGNs as part of the LGN Horizontal Link Hello Protocol (see Section 5.6.3.2) include the **LGN Horizontal Link Extension** information group.

5.14.8.1 The Aggregation Token IG

The Aggregation Token is used to control how outside links are aggregated at higher levels of the hierarchy. See Section 5.10.3.1 for a complete description of link aggregation.

Table 5-28: The Aggregation Token IG

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 32 (aggregation token)
2	2	Length	
4	4	Aggregation token	

5.14.8.2 The Nodal Hierarchy List IG

The nodal hierarchy list is included in order to allow lower-level neighbor nodes to determine their lowest-level common peer group, and also to exchange the node ID and ATM End System Addresses of the corresponding higher-level neighbor nodes within the common peer group. The details of operation in synchronizing the behavior between neighboring peer groups and achieving correct operation of higher level routing are described in Section 5.6.

Table 5-29: The Nodal Hierarchy List

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 33 (nodal hierarchy list)
2	2	Length	
4	4	Sequence number	
8	2	*Reserved*	
10	2	Level Count	The number of known higher level peer groups.
• Repeat the following structure Level Count times, beginning with the parent level and proceeding in order until the node's identity at the highest known level has been listed:			
	22	Next higher level logical node ID	
	20	Next higher level ATM End System Address	ATM End System Address of the next higher level logical node.
	14	Next higher level peer group ID	Next higher level logical node's peer group ID

5.14.8.3 The LGN Horizontal Link Extension IG

The following TLV is known as the LGN Horizontal Link Extension. It carries the state information about all of the horizontal links between two Logical Group Nodes. It is carried as an additional TLV in the same message which is used to maintain the state of the SVCC between the LGNs.

Table 5-30: The LGN Horizontal Link Extension IG

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 35 (LGN Horizontal Link Extenstion)
2	2	Length	
4	2	*Reserved*	
6	2	HLink Count	The number of horizontal links being described.
Repeat the following structure HLinkCount times:			
	4	Aggregation token	
	4	Local LGN port	
	4	Remote LGN port	

5.14.9 PNNI Topology State Packets (PTSP)

PNNI Topology State Packets (PTSPs) are used to distribute information throughout a peer group.

All PTSPs include the PTSP header described in the following table.

Table 5-31: The PTSP Header

Offset	**Size (Octets)**	**Name**	**Function/Description**
0	8	PNNI Header	A PNNI packet header structure with Packet type = 2 (PNNI Topology State Packet). See Table 5-20.
8	22	Originating Node ID	Identifies the node that originated the PTSE(s) in this PTSP.
30	14	Originating Node's peer group ID	

Each PTSP consists of multiple PNNI Topology State Elements (PTSEs), all from the same originating node. Each PTSE includes its own checksum and authentication field (null in PNNI Phase 1), allowing for PTSEs from the same originating node to be bundled in different combinations upon origination and as they are propagated across a peer group. A PTSE provides the information unit that is used for purposes of retransmission, application of sequence numbers and checksums, and authentication.

Each PTSE includes the PTSE header described in the following table:

Table 5-32: The PTSE Header

Offset	**Size (Octets)**	**Name**	**Function/Description**
0	2	Type	Type = 64 (PTSE)
2	2	Length	
4	2	PTSEType	Indicates which restricted information groups are allowed to appear inside of the PTSE. Only those restricted information groups with a matching TLV type may be included. Note that this is not aiming to influence the types of TLVs embedded inside of the restricted information groups. Only the 'top-level' restricted information groups in the PTSE have to conform to this rule.
6	2	*Reserved*	
8	4	PTSE Identifier	Identifies one of multiple different PTSEs from a node. PTSEs with the PTSE Identifier value of '1' are reserved for the purpose of distributing the nodal information group. Its PTSEType field has the value matching the nodal information group type value.
12	4	PTSE Sequence Number	
16	2	PTSE Checksum	NOTE: The PTSE checksum includes the logical node ID and the Originating Node's Peer Group ID from the PTSP header as well as the entire contents of the PTSE (including the PTSE header), and excludes the PTSE Remaining Lifetime. The checksum algorithm used for computation of the PTSE checksum is described in Section 5.8.2.2.2.
18	2	PTSE Remaining Lifetime	

In addition to restricted information groups, any number of unrestricted information groups, described below, can appear in all PTSE types.

5.14.9.1 Restricted Information Groups

5.14.9.1.1 The Nodal state parameters IG

Nodal state parameter information groups are used to advertise elements of this node's complex node representation. All metrics and attributes for each input-output port pair must be included in the same nodal state parameters information group. This information group can appear multiple times and in multiple different PTSEs, with different pairs of input and output ports each time.

Table 5-33: The Nodal State Parameters IG

Offset	**Size (Octets)**	**Name**	**Function/Description**
0	2	Type	Type = 96 (nodal state parameters)
2	2	Length	
4	2	Flags	See Table 5-34.
6	2	*Reserved*	
8	4	Input Port ID	
12	4	Output Port ID	
Repeat for each (set of) service category(ies):			
Outgoing resource availability information group (type = 128)			

Table 5-34: Flags

Bit ID:	bit 16 (MSB)	bits 15..1
Bit Name:	VP Capability Flag	*Reserved*

The VP Capability Flag is set to one if the corresponding entity is capable of carrying VPCs. If VPCs are not supported, then the VP Capability Flag is set to zero. For example, if a horizontal link or uplink is made up entirely of VPCs or aggregate VPCs, with no components that are entire physical links, then it will be incapable of carrying VPCs within it and the VP Capability Flag must be set to zero.

5.14.9.1.2 The Nodal Information Group

The nodal information group is used to convey general information about the node such as its peer group ID and its ATM End System Address, and also for all PGL election information. This information group appears once among all PTSEs advertised by this node.

Table 5-35: The Nodal IG

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 97 (nodal information group)
2	2	Length	
4	20	ATM Address	ATM End System Address of originating node.
24	1	Leadership priority	The value of 0 is reserved for a node which is either unwilling or unable to become peer group leader.
25	1	Nodal Flags	See Table 5-36.
26	22	Preferred peer group leader node ID	
Next higher level binding information: (sent by peer group leaders if there is a higher level):			
48	2	Type	Type = 192 (next higher level binding information
50	2	Length	
52	22	Parent logical group node ID	Parent logical group node ID (in parent peer group)
74	20	Parent LGN’s ATM End System Address	
94	14	Parent Peer Group ID	
108	22	Node ID of PGL of parent peer group	NULL if unknown
130	2	*Reserved*	

Table 5-36: Nodal Flags

Bit ID:	**bit 8 (MSB)**	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bits 3..1**
Bit Name:	“I am Leader” bit	“Restricted Transit” bit	“Nodal Representation” bit	”Restricted Branching” bit	Non-transit for PGL election	Reserved
Description:	value 0: “I am not PGL” value 1: “I am PGL	value 0: “I am a transit node” value 1: “I am a restricted transit node”	value 0: “simple node representation” value 1: “complex node representation”	value 0: “can support additional branch points” value 1: “cannot support additional branch points”	value 0: “normal operation” value 1: “no connectivity through this node for PGL election”	

5.14.9.1.3 The Internal Reachable ATM Addresses Information Group

The internal reachable ATM addresses information group is used to advertise the addresses of directly attached ATM endpoints. This information group can appear multiple times and in multiple different PTSEs, listing different sets of internal reachable ATM addresses each time.

Table 5-37: The Internal Reachable ATM Address IG

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 224 (internal reachable ATM addresses)
2	2	Length	
4	2	Flags	See Table 5-34.
6	2	*Reserved*	
8	4	Port ID	
12	1	Scope of advertisement	A PNNI routing level. The scope of advertisement applies to all address prefixes included in the information group, whether they are individual or group addresses or a mixture of the two.
13	1	Address information length	*ail*, in octets
14	2	Address information count	*aic*
Repeat (*aic*) times:			
	1	prefix length	units: bits
	ail-1	reachable address prefix	padded with [8*(*ail*-1) - prefix length] bits of zeros at the right most tail (lowest bits) of the last octet(s) of the Reachable Address Prefix
		Padding	0-3 octets. Note : The size of the Padding field is calculated using the following formula: (4 - ((*aic* * *ail*) modulus 4)) modulus 4
Optional TLV groups for resource availability information, repeat for each (set of) service category(ies): • Outgoing resource availability information group (type = 128) • Incoming resource availability information group (type = 129) If present, the resource availability information groups apply to all present reachable address prefixes, and are to be combined directly with PNNI internal service category parameters.			

5.14.9.1.4 The Exterior Reachable ATM addresses Information Group

The exterior reachable ATM addresses information group is used to advertise the addresses of ATM endpoints that can be reached by passing through exterior networks, that do not participate in PNNI routing. This information group can appear multiple times and in multiple different PTSEs, listing different sets of exterior reachable ATM addresses each time. All metrics and attributes associated with each exterior reachable ATM address must be included in the same exterior reachable ATM address information group in which the reachable address appears.

Table 5-38: The Exterior Reachable ATM Address IG

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 256 (exterior reachable ATM addresses)
2	2	Length	
4	2	Flags	See Table 5-34.
6	2	*Reserved*	
8	4	Port ID	
12	1	Scope of advertisement	A PNNI routing level. The scope of advertisement applies to all transit networks. It also applies to all address prefixes included in the information group, whether they are individual or group addresses or a mixture of the two.
13	1	Address information length	*ail*, in octets
14	2	Address information count	*aic*
Repeat (*aic*) times:			
	1	prefix length	units : bits
	ail-1	reachable address prefix	padded with [8*(*ail*-1) - prefix length] bits of zeros at the right most tail (lowest bits) of the last octet(s) of the Reachable Address Prefix
	0..3	Padding	0-3 octets. Note : The size of the Padding field is calculated using the following formula: (4 - ((*aic* * *ail*) modulus 4)) modulus 4

• Optional TLV groups for resource availability information, repeat for each (set of) service category(ies). If present, the resource availability information groups apply to all present reachable address prefixes, and are to be combined directly with PNNI internal service category parameters:
  • Outgoing resource availability information group (type = 128)
  • Incoming resource availability information group (type = 129)
• Additional optional TLV groups
  • Transit network ID (type = 304).			

5.14.9.1.5 The Horizontal Links Information Group

The horizontal links information group is used to advertise links to other nodes within the same peer group. This information group can appear multiple times and in multiple different PTSEs, listing different horizontal links each time. All metrics and attributes associated with each horizontal link must be included in the same horizontal link information group.

Table 5-39: The Horizontal Links IG

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 288 (horizontal links)
2	2	Length	
4	2	Flags	See Table 5-34.
6	22	Remote Node ID	The combination of local node ID and port ID unambiguously identify the link. The combination of remote node ID and remote port ID unambiguously identify the reverse link—this is needed to allow metrics in both directions to be related.
28	4	Remote Port ID	See above
32	4	Local Port ID	
36	4	Aggregation Token	

- Repeat for each (set of) service category(ies):
    - Outgoing resource availability information group (type = 128)

5.14.9.1.6 The Uplinks Information Group

The uplinks information group is used to advertise links to nodes that are outside of this peer group. This information group can appear multiple times and in multiple different PTSEs, listing different uplinks each time. All metrics and attributes associated with each uplink must be included in the same uplink information group.

Table 5-40: The Uplinks IG

Offset	Size (Octets)	Name	Function/Description
0	2	Type	Type = 289 (uplinks)
2	2	Length	
4	2	Flags	See Table 5-34.
6	2	*Reserved*	
8	22	Remote Higher Level Node ID	
30	14	Common Peer Group ID	
44	4	Local Port ID	
48	4	Aggregation Token	
52	20	ATM End System Address of Upnode	

- Repeat for each (set of) service category(ies):
    - Outgoing resource availability information group (type = 128)
- Uplink Information Attribute (type = 34)

5.14.9.2 Unrestricted Information Groups

The Systems Capabilities information group is the only unrestricted information group used in PNNI Phase 1. Other unrestricted information groups may be defined in later versions of PNNI. Those foreseen are for authentication and access control.

5.14.9.3 Unknown Information Groups

Unknown information groups are those which appear in a PTSE of a certain type, but which do not match the type indicated and are not known to be unrestricted TLVs.

Such information groups might be encountered when PTSEs issued by a node running a newer version of PNNI are received by a node running an older version.

5.14.9.4 Processing PTSEs Violating Restrictions

Due to the described restrictions in the content of PTSEs, it cannot be excluded that certain PTSEs will be flooded which violate these rules. The possible error cases and the actions to be taken are:

1) If a PTSE includes a restricted information group whose type does not match the PTSEType, the information contained in such a group is ignored for the sake of route computation.
2) Unknown information groups are treated like unrestricted information groups. They are processed according to the specified information group tags.
3) If a mandatory restricted information group is missing in a PTSE (an example would be an empty PTSE with type equal to “nodal information” and PTSE Identifier = 1), the PTSE must be ignored during route computation.
4) If a PTSE with an unknown PTSEType is received, all restricted information groups inside are treated according to rules 1) and 2).

A TLV or IG should not only be ignored for the sake of path computation but also ignored for any other kind of processing on the topology database mirroring into the content of other PTSEs. Those kinds of computation are called state-significant computations because they influence the internal state of the protocol and are as specified today:

i) link aggregation
ii) uplink generation
iii) computation of complex node representation of the peer group
iv) peer group leader election except counting the vote for the election if present
v) computation of hierarchy to be advertised in outside hellos
vi) reachability computation

Path computation is not considered a state-significant computation due to the fact that if two nodes compute different routes from the same set of information, it may adversely affect the throughput of the network and call rejection rates but does not prevent the protocol state machines themselves from operating correctly. Therefore, path computation is free to choose different strategies when dealing with the cases described below varying from one implementation to another although it is recommended that the rules specified are used. Information which is ignored according to the following sections must be re-evaluated when the condition which caused it to be ignored changed.

5.14.9.5 PTSE Error Scenarios

The TLV formatting supported in the PTSEs does not syntactically prevent ambiguities in packet semantics such as:

a) a TLV is supposed not to appear, appears at least once
b) a TLV is supposed to appear exactly once inside an embedding TLV, appears multiple times
c) a TLV is supposed to appear at least once, does not appear at all
d) a TLV of this type is not supposed to appear embedded in the embedding type

e) the same TLV appears multiple times in different PTSEs (PTSEs with different IDs)

Case d) is treated uniformly. An unexpected TLV appearing inside of embedding TLV (e.g. nodal info TLV appearing inside of a horizontal link TLV) is treated according to Section 5.14.9.4. (where a RIG appears inside a PTSE indicating a different type).

Note: There is a subtle difference between a) and d) and between a) and the unknown TLV case (the latter is handled by attribute tags).

5.14.9.5.1 Nodal Info Group in Nodal Info PTSE

a) not applicable
b) if a nodal info TLV appears multiple times in the designated PTSE, two implementations interpreting different of those TLVs as valid could lead to scenarios where PGL election does not converge. The solution proposed is to only consider the first appearing nodal info TLV as valid and significant for state-significant computation purposes and ignore all following.
c) if a nodal info TLV does not appear in the designated PTSE at all, it is proposed that the PTSE and all further PTSEs issued by this node should be ignored for the state-significant computations except vi). So a node with an invalid nodal info can be used to compute reachability to other nodes in the network for the sake of e.g. peer group leader election but its PTSEs like e.g. horizontal links cannot be used for the sake of path computation.
d) n/a.
e) is not possible due to the fact that this IG can appear only in a PTSE with ID 1, otherwise it is ignored.

5.14.9.5.2 Next higher Level Binding TLV in Nodal Info TLV

a) if such a TLV appears once or multiple times inside of a nodal info TLV with cleared `I'm leader'-Bit, such a TLV must be ignored.

   The following cases only apply to TLVs where the `I'm leader'-Bit is set and additionally the node has been determined as peer group leader:

b) if the binding appears multiple times, only the first thereof must be used for state-significant computation and all following ignored.
c) normal situation.
d) n/a.
e) equivalent to 5.14.8.5.1 e).

5.14.9.5.3 Nodal State Parameters TLV

The following cases only apply to TLVs where the `Complex Repr'-Bit is set.

a) not applicable
b) if a nodal state TLV appears mutliple times inside of a single PTSE only the first one is used for any state-significant computations (even if the service category sets defined do not intersect). This includes the default spoke.
c) default spoke does not appear. Any PTSE of the node is ignored for the sake of state-significant computations except for vi). Such a node can be used to compute reachability to other nodes in the network for the sake of e.g. peer group leader election but its PTSEs like e.g. horizontal links cannot be used for the sake of path computation.

d) n/a.
e) if a nodal state appears multiple times in different PTSEs (even if the service category sets defined do not intersect), only the one appearing in the PTSE with the lowest ID is used for state-significant computations.

If the node does not have the `Complex Repr'-Bit set.

a) if nodal state TLVs appear they are ignored for the sake of any state-significant computations
b) not applicable.
c) not applicable.
d) n/a.
e) analog a).

5.14.9.5.4 Horizontal Link TLV

a) not applicable
b) if such TLV appears mutliple times inside of a single PTSE only the first one is used for any state-significant computations (even if the service category sets defined do not intersect).
c) not applicable
d) n/a.
e) if a TLV appears multiple times in different PTSEs (even if the service category sets defined do not intersect), only the one appearing in the PTSE with the lowest ID is used for state-significant computations.

5.14.9.5.5 Uplink TLV

Equivalent to 5.14.8.5.4.

5.14.9.5.6 ULIA

a) not applicable
b) if multiple ULIAs appear in an uplink TLV, only the first one is used for state-significant computations.
c) not applicable
d) n/a.
e) not applicable

5.14.9.5.7 RAIGs

If a service category appears in multiple RAIGs within a horizontal link, uplink, nodal state parameters, or ULIA IG, then only the one appearing in the first RAIG is used for state-significant computations.

5.14.9.6 Nodal Info PTSE and its connection to other PTSEs issued by the node

Since nodal info carries several important flags and specifications of the node issuing, it is important to specify the semantics of a set of PTSEs that has been received without the appropriate nodal info PTSE being present. To improve the stability of the protocol all state-significant computation except vi) should ignore PTSEs for which a valid nodal info PTSE is not present. Path computation can omit this restriction, however it should not be forgotten that e.g. paths computed through a node's complex representation are not valid after nodal info PTSE is received that specifies that the node does not have a complex representation.

5.14.10 PTSE Acknowledgment Packets

PTSE Acknowledgment Packets are used to acknowledge the receipt of PTSEs from a neighbor node. Each PTSE Acknowledgment Packet consists of multiple nodal PTSE acknowledgment information groups. Each information group is used to acknowledge the receipt of multiple PTSEs with the same originating node and transmitted by the same neighbor node, by including the PTSE identifying information for each acknowledged PTSE.

Table 5-41: The PTSE Acknowledgement Packet

Offset	**Size (Octets)**	**Name**	**Function/Description**
0	8	PNNI Header	A PNNI packet header structure with Packet type = 3 (PTSE acknowledgment packet). See Table 5-20.
Repeat for each set of PTSE acknowledgments about one node:			
	2	Type	Type = 384 (Nodal PTSE acknowledgment)
	2	Length	
	22	Node ID	
	2	AckCount	The number of acknowledgments for this node.
Repeat the following structure AckCount times:			
	4	PTSE Identifier	
	4	PTSE Sequence Number	
	2	PTSE Checksum	
	2	PTSE Remaining Lifetime	

5.14.11 Database Summary Packets

Database summary packets are used during the initial database exchange process between two neighboring peers. During the database exchange process, database summary packets are sent containing the PTSP and PTSE header information of all PTSEs in a node’s topology database. Database summary packets also contain a sequence number and flags used to negotiate the master/slave relationship necessary to ensure proper functioning of the lock-step protocol.

Table 5-42: The Database Summary Packet

Offset	**Size (Octets)**	**Name**	**Function/Description**
0	8	PNNI Header	A PNNI packet header structure with Packet type = 4 (Database summary packet). See Table 5-20.
8	2	Flags	See Table 5-43.
10	2	*Reserved*	
12	4	DS sequence number	
Repeat for each set of PTSEs in the topology database:			
	2	Type	Type = 512 (Nodal PTSE summaries)
	2	Length	
	22	Originating node ID	
	14	Originating node's Peer Group ID	
	2	*Reserved*	
	2	PTSE Summary Count	The number of PTSE summaries for this originating node ID.
Repeat the following structure PTSE Summary Count times:			
	2	PTSEType	
	2	*Reserved*	
	4	PTSE Identifier	
	4	PTSE Sequence Number	
	2	PTSE Checksum	
	2	PTSE Remaining Lifetime	

Table 5-43: Database Summary Packet Flags

Bit ID:	bit 16 (MSB)	bit 15	bit 14	bits 13..1 (LSB)
Bit Name:	'Initialize' (I) bit	'More'(M) bit	'Master' (MS) bit	*Reserved*
Description:	Set to one during initialization of the database synchronization process. Otherwise set to zero.	Set to one if the transmitting node has additional PTSEs to summarize, or zero if all PTSEs have been summarized (see Section 5.7.5).	Set to one initially by both nodes. Later set to zero by the node that assumes the slave role in the database synchronization process.	

5.14.12 PTSE Request Packets

PTSE request packets are used during database synchronization to request from a neighboring peer those PTSEs that have been newly discovered or that have been found to be obsolete.

Table 5-44: The PTSE Request Packet

Offset	Size (Octets)	Name	Function/Description
0	8	PNNI Header	A PNNI packet header structure with Packet type = 5 (PTSE request packet). See Table 5-20.
Repeat for each set of PTSEs requested:			
	2	Type	Type = 513 (Nodal PTSE request list)
	2	Length	
	22	Originating Node ID	
	2	PTSE Request Count	The number of PTSEs requested for this originating node ID.
Repeat PTSE Request Count times:			
	4	PTSE Identifier	

5.14.13 System Capabilities

Systems may indicate support for optional PNNI capabilities by including the “Systems Capabilities” IG in PNNI packets. This may be used to indicate support for standard capabilities (specified by the ATM Forum), and proprietary capabilities (which may be specified by individual equipment manufacturers).

The interpretation of the system capabilities IG depends on where it occurs. If it occurs at the top level of a packet it describes system capabilities of the sender of that packet, i.e., the neighbor node. If it occurs within a PTSE, it describes the system capabilities of the PTSE originator.

The system capabilities IG MAY optionally be included at the top level in any PNNI packets (specifically including Hellos, Database Summary Packets, and PTSPs) and also in a PTSE, but not within an information group inside of a PTSE. Also, this field may be occur multiple times (for example, it may be desirable to identify both extended standard capabilities as well as proprietary capabilities).

Table 5-45: The Systems Capabilities IG

Offset	**Size (Octets)**	**Name**	**Function/Description**
0	2	Type	Type=640 (system capabilities)
2	2	Length	
4	2	Length of system capabilities contents	Length of IEEE OUI + System Capabilities Information.
6	3	IEEE OUI	IEEE Organizationally Unique Identifier, reference IEEE Standard 802-1990.
9	n	System capabilities information	The semantics of this field are administered by the organization identified by the OUI.
9 + n	0..3	Padding	The size of the Padding field is calculated using the following formula: (4 - [(5+n) modulus 4]) modulus 4

Standard capabilities may be specified by the ATM Forum, using the OUI assigned to the Forum. Currently, there are no defined system capabilities which may be announced using the ATM Forum OUI.

The system capabilities advertised by an LGN represent the capabilities of its child peer group and/or of the LGN implementation.

6. PNNI Signalling Specification

This specification contains the procedures to dynamically establish, maintain and clear ATM connections at the Private network-to-network interface or network node interface (PNNI) between two ATM networks or two ATM network nodes. The PNNI signalling protocol is based on the ATM Forum UNI specification and for the finite state machine definition and the specification of symmetrical interactions, it is derived from the Frame Relay NNI signalling defined in ITU-T draft Recommendation Q.934. The procedures are defined in terms of messages and information elements used to characterize the ATM connection.

6.1 PNNI model

A network configuration is shown in Figure 6-1 to help clarify terminology.

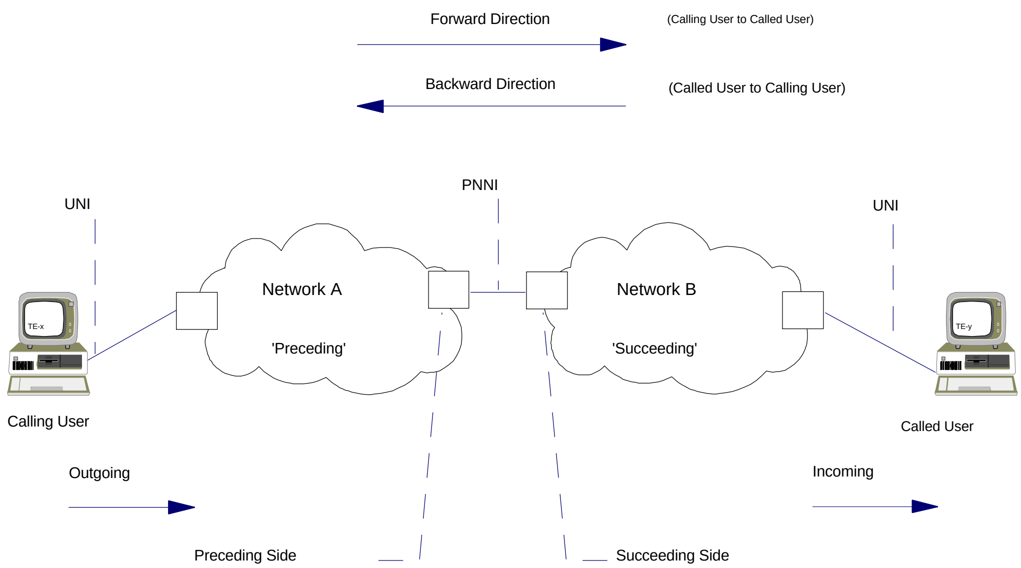


Figure 6-1 PNNI interface

6.1.1 Definitions

Forward/Backward direction: The forward direction is the direction from the calling user to the called user. The backward direction is the direction from the called user to the calling user.

Preceding/Succeeding side: A preceding side is a network node that comes before another network node in the call/connection setup path. A succeeding side is a network node that comes after another network node in the call/connection setup path.

6.1.2 Protocol model

The figure below shows, within the control plane, the relationship between the signalling procedures and the underlying services.

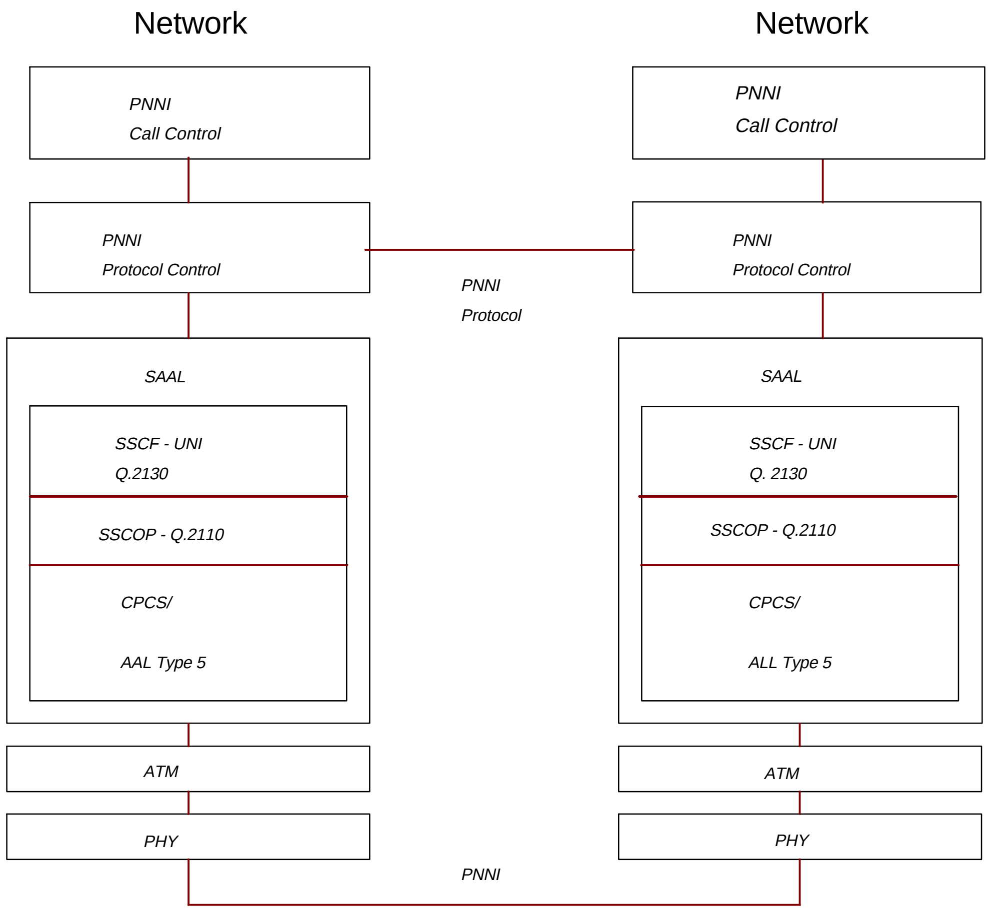


Figure 6-2 PNNI Control Plane

6.1.2.1 Signalling layer

The signalling layer contains two distinct entities: PNNI call control and PNNI protocol control. The PNNI call control services the upper layers for functions such as resource allocation and routing information. The PNNI protocol control entity provides services to the PNNI call control. Primitives exchanged across the boundary between call control and protocol control correspond to the information flows exchanged between the call control functional entities.

The PNNI protocol control processes the actual signalling finite state machines (incoming and outgoing calls) using symmetric procedures.

6.1.2.2 Signalling adaptation layer

The PNNI signalling symmetric procedures use the services of the UNI Signalling ATM Adaptation Layer (SAAL). The SAAL layer comprises:

a) A Service Specific Coordination Function (SSCF) as specified in ITU-T Recommendation Q.2130. This function maps the service of the Service Specific Connection Oriented Protocol (SSCOP) to the needs of the signalling procedures.
b) The service specific connection oriented protocol as specified in ITU-T Recommendation Q.2110. This assured service provides a peer-to-peer protocol to transfer signalling information between any pair of SSCOP entities.
c) The ATM Adaptation Layer Type 5 (AAL-5) CPCS and SAR sublayers as specified in ITU-T Recommendation I.363. This service provides segmentation and reassembly of signalling data units per ATM layer cell structure requirements.

6.1.2.3 ATM layer

The ATM layer as specified in ITU-T Recommendation I.150, provides for the transparent transfer of fixed size ATM layer service data units (ATM-SDUs) between communicating AAL entities. The structure of the cell header used in the PNNI is the cell header format and encoding at NNI (see Figure 3/I.361). Any bits of the VPI or VCI subfield that are not allocated are set to ‘0’.

6.2 Overview of Call/connection control

This section defines the call control states that individual calls may have. These definitions do not apply to the state of the interface itself, any attached equipment, or the signalling virtual channel. Because several calls may exist simultaneously at a PNNI and each call may be in a different state, the state of the interface cannot be unambiguously defined.

6.2.1 ATM Point-to-point call states

This section defines the point-to-point call control states for ATM calls at both sides of a PNNI between preceding and succeeding sides.

6.2.1.1 Null (NN0)

No call exists.

6.2.1.2 Call Initiated (NN1)

This state exists in a succeeding side after it has received a call establishment request from the preceding network node but has not yet responded.

6.2.1.3 Call Proceeding Sent (NN3)

This state exists when a succeeding side has acknowledged the receipt of the information necessary to establish a call.

6.2.1.4 Alerting Delivered (NN4)

This state exists when a succeeding side has sent an ALERTING message to the preceding side.

6.2.1.5 Call Present (NN6)

This state exists at a preceding side after it has sent a call establishment request to the succeeding side but has not yet received a response.

6.2.1.6 Alerting Received (NN7)

This state exists when a preceding side has received an ALERTING message from the succeeding side of the PNNI interface.

6.2.1.7 Call Proceeding Received (NN9)

This state exists when a preceding side has received acknowledgment that the succeeding side has received the call establishment request.

6.2.1.8 Active (NN10)

This state exists when the ATM connection has been established.

6.2.1.9 Release Request (NN11)

This state exists when a network node has sent a request to the network node at the other side of the PNNI interface to release the ATM connection and is waiting for the response.

6.2.1.10 Release Indication (NN12)

This state exists when a network node has received a request from the network node at the other side of the PNNI interface to release the ATM connection and has not responded yet.

6.2.2 States associated with global call reference

For clarity, the section refers to ITU-T Recommendation Q.2931 section numbers because UNI 4.0 is a pointer to Q.2931 with a list of clarifications. It should be understood that modifications described in UNI 4.0 signalling specification shall apply.

See section 2.3 of Recommendation Q.2931.

6.3 Message functional definitions and contents

This section provides an overview of the message structure, which highlights the functional definitions and information content (i.e., semantics) of each message. Each definition includes:

1. A brief description of the message direction and use, including whether the message has:
   - a) Local significance, i.e., relevant only at the local PNNI, or
   - b) Access significance, i.e., relevant at the originating and terminating access, but not in the network, or
   - c) Global significance, i.e., relevant at the local PNNI, other PNNIs and UNIs related to the call.
2. A table listing the codeset 0 (ITU-T [CCITT] standardized) information elements. For each information element the table indicates:

a) the section of this Implementation Agreement describing the information element;

b) whether inclusion is mandatory ('M') or optional ('O'), with a reference to notes explaining the circumstances under which the information element shall be included; and

c) the length of the information element (or permissible range of lengths), in octets.

3. Further explanatory notes, as necessary.

In view of the symmetrical nature of the protocol at the PNNI, the reference model used in this section considers a PNNI link connecting two network nodes which are identified as a Preceding side and a Succeeding side, where the direction of flow is that of the call. In the context of a call, the Preceding side routes the call over the PNNI link and the Succeeding side receives the call.

From the point of view of the Preceding side, a call is an outgoing call.
From the point of view of the Succeeding side, a call is an incoming call.

6.3.1 Messages for ATM point-to-point call and connection control

Table 6-1 summarizes the messages for ATM point-to-point call and connection control.

Table 6-1 Messages for ATM Call and Connection Control

Message	Reference
Call establishment messages:	
ALERTING	6.3.1.1
CALL PROCEEDING	6.3.1.2
CONNECT	6.3.1.3
SETUP	6.3.1.6
Call clearing messages:	
RELEASE	6.3.1.4
RELEASE COMPLETE	6.3.1.5
Miscellaneous messages:	
NOTIFY	6.3.1.9
STATUS	6.3.1.7
STATUS ENQUIRY	6.3.1.8

6.3.1.1 ALERTING

This message is sent by the succeeding side to indicate that called user alerting has been initiated.

Message Type: ALERTING
Direction: Succeeding to Preceding
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2

Endpoint reference	6.4.8.1	O(1)	4-7
Notification indicator	6.4.5.27	O(2)	4-*
Generic identifier transport	6.4.5.31	O(2,3)	4-33

Note 1 - Mandatory if an Endpoint reference was included in the SETUP message.
Note 2 - Included if the received alerting indication contains this information.
Note 3 - This information element may be present up to 3 times.

Figure 6-3 ALERTING Message Contents

6.3.1.2 CALL PROCEEDING

This message is sent by the Succeeding side to indicate that the requested call/connection establishment has been initiated and no more call establishment information will be accepted.

Message Type: CALL PROCEEDING
Direction: Succeeding to Preceding
Significance: Local

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Connection identifier	6.4.5.22	M	9
Endpoint reference	6.4.8.1	O(1)	4-7

Note 1 - Mandatory if an Endpoint reference was included in the SETUP message.

Figure 6-4 CALL PROCEEDING Message Contents

6.3.1.3 CONNECT

This message is sent by the Succeeding side and delivered to the Preceding side to indicate call/connection acceptance by the called user.

Message Type: CONNECT
Direction: Succeeding to Preceding
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2

ABR additional parameters	6.4.5.5	O(1)	4-14
ABR setup parameters	6.4.5.6	O(1)	4-36
AAL parameters	6.4.5.8	O(1)	4-11
ATM traffic descriptor	6.4.5.9	O(4)	4-30
Broadband low layer information	6.4.5.12	O(1)	4-17
Called party soft PVPC or PVCC	6.4.6.2	O(2)	4-11
Connected number	6.4.5.20	O(1)	4-26
Connected sub-address	6.4.5.21	O(1)	4-25
Endpoint reference	6.4.8.1	O(3)	4-7
End-to-end transit delay	6.4.5.24	O(1)	4-7
Extended QoS parameters	6.4.5.25	O(1)	4-13
Generic identifier transport	6.4.5.31	O(1,5)	4-33
Notification indicator	6.4.5.27	O(1)	4-*

Note 1 - Included if the received connect indication contains this information.
Note 2 - Included in case of soft PVPC or PVCC setup.
Note 3 - Mandatory if the Endpoint reference was included in the SETUP message.
Note 4 - Mandatory if the calling user requested an ABR traffic category connection.
Note 5 - May be present up to three times.

Figure 6-5 CONNECT Message Contents

6.3.1.4 RELEASE

This message is sent by a network node to an adjacent network node to indicate that it has cleared the connection and is waiting to release the call reference.

Message Type: RELEASE
Direction: Both
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Cause	6.4.5.19	M(1)	6-34
Crankback	6.4.6.3	O(2)	4-72
Notification indicator	6.4.5.27	O(3)	4-*
Generic identifier transport	6.4.5.31	O(3,4)	4-33

Note 1 - This information element may appear twice in the message.

Note 2 - Included to indicate crankback.
Note 3 - Included if the received release indication contains this information.
Note 4 - This information element may be present up to 3 times.

Figure 6-6 RELEASE message content

6.3.1.5 RELEASE COMPLETE

This message is sent by a network node to an adjacent network node to indicate that it has cleared internally the connection (if any) and released the call reference.

Message Type: RELEASE COMPLETE
Direction: Both
Significance: Local (Note 1)

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Cause	6.4.5.19	O(2)	4-34
Crankback	6.4.6.3	O(3)	4-72

Note 1 - This message has local significance; however, it may carry information of global significance when used as the first call clearing message.
Note 2 - Mandatory in the first call clearing message; including when the RELEASE COMPLETE message is sent as a result of an error condition. This information element may appear twice in the message.
Note 3 - Included to indicate crankback.

Figure 6-7 RELEASE COMPLETE message content

6.3.1.6 SETUP

This message is sent by Preceding side to Succeeding side to initiate call/connection establishment.

Message Type: SETUP
Direction: Preceding to Succeeding
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
AAL parameters	6.4.5.8	O(1)	4-21

ABR additional parameters	6.4.5.5	O(1)	4-14
ABR setup parameters	6.4.5.6	O(11)	4-36
Alternative ATM traffic descriptor	6.4.5.7	O(12)	4-30
ATM traffic descriptor	6.4.5.9	M	12-30
Broadband bearer capability	6.4.5.10	M	6-7
Broadband high layer information	6.4.5.11	O(1)	4-13
Broadband repeat indicator	6.4.5.13	O(6)	4-5
Broadband low layer information	6.4.5.12	O(1)	4-17
Called party number	6.4.5.15	M	(2)
Called party soft PVPC or PVCC	6.4.6.2	O(4)	4-11
Called party subaddress	6.4.5.16	O(1)	4-25
Calling party number	6.4.5.17	O(1)	4-26
Calling party soft PVPC or PVCC	6.4.6.1	O(3)	4-10
Calling party subaddress	6.4.5.18	O(1)	4-25
Connection identifier	6.4.5.22	O(5)	4-9
Connection scope selection	6.4.5.23	O(1)	4-6
Designated transit list	6.4.6.4	M(7)	33-546
Endpoint reference	6.4.8.1	O(1)	4-7
End-to-end transit delay	6.4.5.24	O(8)	4-13
Extended QoS parameters	6.4.5.25	O(10)	4-25
Generic identifier transport	6.4.5.31	O(1,9)	4-33
Minimum acceptable ATM traffic descriptor	6.4.5.26	O(12)	4-20
Notification indicator	6.4.5.27	O(1)	4-*
QoS parameter	6.4.5.28	O(1)	4-6
Transit network selection	6.4.5.30	O(1)	4-9

Note 1 - Included if the received setup indication contains this information.

Note 2 - Minimum length depends on the numbering plan. Maximum length is 25 octets.

Note 3 - May be included in case of soft PVPC or PVCC setup, when the calling endpoint wants to inform the destination network interface of the values used for the PVPC or PVCC segment at the calling end.

Note 4 - Included in case of soft PVPC or PVCC setup.

Note 5 - Included when Preceding side wants to indicate a specific virtual path or virtual channel. If not included, its absence is interpreted as any virtual path or virtual channel is acceptable. This information element may only be absent when using the non-associated signalling procedures.

Note 6 - When the Broadband repeat indicator information element immediately precedes the DTL information element, it indicates the order of Designated transit list information elements in the DTL stack. This information element is mandatory, even when there is only one Designated transit list information element. When the Broadband repeat indicator information element immediately precedes any other information element, it is included if the received setup indication contains this information

Note 7 - Included by the source node to indicate the hierarchical source route for the call. Included by the node at the entry to a hierarchical level to indicate the path through that hierarchical level. This information element may be repeated up to 10 times.

Note 8 - Included to specify an end-to-end transit delay requirement.

Note 9 - This information element may be present up to three times.

Note 10 - Included to specify individual QoS parameter requirements for the call.

Note 11 - Mandatory if the calling user requested an ABR traffic category connection.

Note 12 - May only be present when this information is present in the received setup indication.

Figure 6-8 SETUP message content

6.3.1.7 STATUS

This message is sent by either side in response to a STATUS ENQUIRY message or at any time to report certain error conditions.

Message type: STATUS
Direction: Both
Significance: Local

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Call state	6.4.5.14	M	5
Cause	6.4.5.19	M	6-34
Endpoint reference	6.4.8.1	O(1)	4-7
Endpoint state	6.4.8.2	O(2)	4-5

Note 1 - Included when responding to a status enquiry about a party state or at any time to report certain error conditions in the point-to-multipoint procedures.

Note 2 - Included when the Endpoint reference information element is included.

Figure 6-9 STATUS message content

6.3.1.8 STATUS ENQUIRY

The STATUS ENQUIRY message may be sent by either side at any time to solicit a STATUS message from the peer entity. Sending a STATUS message in response to a STATUS ENQUIRY message is mandatory.

Message type: STATUS ENQUIRY
Direction: Both
Significance: Local

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Endpoint reference	6.4.8.1	O(1)	4-7

Note 1 - Included when enquiring about a party state in the point-to-multipoint procedures.

Figure 6-10 STATUS ENQUIRY message content

6.3.1.9 NOTIFY

This message is sent by either side to indicate information pertaining to a call/connection.

Message type: NOTIFY
Direction: Both
Significance: Access

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Endpoint reference	6.4.8.1	O(1)	4-7
Notification indicator	6.4.5.27	M	5-*

Note 1 - Included if the received notify indication contains this information.

Figure 6-11 NOTIFY message content

6.3.2 Additional or modified messages for the support of 64kbit/s based ISDN circuit mode services

Table 6-2 summarizes the messages for ATM call and connection control for the support of 64 kbit/s based ISDN circuit-mode services.

Except where explicitly identified in the procedures, the information elements added in this section are transported without change across the PNNI.

Table 6-2 Messages for ATM call/ connection control

for the support of 64 kbit/s based ISDN circuit code services

Message	Reference
Call establishment messages:	
ALERTING	6.3.2.1
CONNECT	6.3.2.2
PROGRESS	6.3.2.3
SETUP	6.3.2.5
Call clearing messages	
RELEASE	6.3.2.4

6.3.2.1 ALERTING

This message is sent by the succeeding side to indicate that the called user alerting has been initiated, if ALERTING is received.

Message Type: ALERTING
Direction: Succeeding to Preceding
Significance: Global

Information Element	Reference	Type	Length
Narrow-band bearer capability	6.4.7.1	O(1)	4-14
Narrow-band high layer compatibility	6.4.7.2	O(1)	4-7
Progress indicator	6.4.7.4	O(1,2)	4-6
Other information elements as described in subclause 6.3.1.1.			

Note 1 - Included if the received alerting indication contains this information.
Note 2 - May appear twice in the message.

Figure 6-12 ALERTING message contents

6.3.2.2 CONNECT

This message is sent by the Succeeding side and delivered to the Preceding side to indicate call/connection acceptance by the called user.

Message Type: CONNECT
Direction: Succeeding to Preceding
Significance: Global

Information Element	Reference	Type	Length
Narrow-band bearer capability	6.4.7.1	O(1)	4-14
Narrow-band high layer compatibility	6.4.7.2	O(1)	4-7
Narrow-band low layer compatibility	6.4.7.3	O(1)	4-20
Progress indicator	6.4.7.4	O(1,2)	4-6
Other information elements as described in subclause 6.3.1.3.			

Note 1 - Included if the received connect indication contains this information.
Note 2 - May appear twice in the message.

Figure 6-13 CONNECT message contents

6.3.2.3 PROGRESS

This message is passed without change by PNNI to indicate the progress of a call in the event of interworking.

Message Type: PROGRESS
Direction: Succeeding to Preceding
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Narrow-band bearer capability	6.4.7.1	O(1)	4-14
Narrow-band high layer compatibility	6.4.7.2	O(1)	4-7
Notification indicator	6.4.5.27	O(1,2)	4-*
Progress indicator	6.4.7.4	O(1)	4-6

Note 1 - Included if the received progress indication contains this information.
Note 2 - May appear twice in the message.

Figure 6-14 PROGRESS message contents

6.3.2.4 RELEASE

This message is sent by a network node to an adjacent node to indicate that it has cleared the connection and is waiting to release the call reference.

Message Type: RELEASE
Direction: Both
Significance: Global

Information Element	Reference	Type	Length

Progress indicator	6.4.7.4	O(1,2)	4-6
Other information elements as described in subclause 6.3.1.4.			

Note 1 - Included if the received release indication contains this information.
Note 2 - May appear twice in the message.

Figure 6-15 RELEASE message content

6.3.2.5 SETUP

This message is sent by Preceding side to Succeeding side to initiate call/connection establishment.

Message Type: SETUP
Direction: Preceding to Succeeding
Significance: Global

Information Element	Reference	Type	Length
Narrow-band bearer capability	6.4.7.1	O(1,2)	4-14
Narrow-band high layer compatibility	6.4.7.2	O(1,3)	4-7
Broadband repeat indicator	6.4.5.13	O(1,4)	4-5
Narrow-band low layer compatibility	6.4.7.3	O(1,4)	4-20
Progress indicator	6.4.7.4	O(1,3)	4-6
Other information elements as described in subclause 6.3.1.6.			

Note 1 - Included if the received setup indication contains this information.
Note 2 - May be present up to three times in the message.
Note 3 - May be repeated twice in the message.
Note 4 - The Broadband repeat indicator is included when two Narrow-band low layer compatibility information elements are included. Refer to Q.2931, section 3.2.7, table 3-19, for details.

Figure 6-16 SETUP message content

6.3.3 Messages used with the global call reference

Table 6-3 Messages Used with the Global Call Reference

Message	Reference
RESTART	6.3.3.1
RESTART ACKNOWLEDGE	6.3.3.2
STATUS	6.3.1.7

6.3.3.1 RESTART

This message is sent by either side to request the recipient to restart (i.e., release all resources associated with) the indicated virtual channel/path or all virtual channels/paths controlled by the signalling virtual channel.

Message type: RESTART
Direction: Both
Significance: Local

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M(1)	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Connection identifier	6.4.5.22	O(2)	4-9
Restart indicator	6.4.5.29	M	5

Note 1 - This message is sent with the global call reference.
Note 2 - Included when necessary to indicate the particular virtual channel or virtual path to be restarted.

Figure 6-17 RESTART message content

6.3.3.2 RESTART ACKNOWLEDGE

This message is sent to acknowledge the receipt of a RESTART message and to indicate that the requested restart is complete.

Message type: RESTART ACKNOWLEDGE
Direction: Both
Significance: Local

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M(1)	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Connection identifier	6.4.5.22	O(2)	4-9
Restart indicator	6.4.5.29	M	5

Note 1 - This message is sent with the global call reference.
Note 2 - Included when necessary to indicate the particular virtual channel or virtual path which has been restarted.

Figure 6-18 RESTART ACKNOWLEDGE message content

6.3.4 Messages for Point-to-multipoint call and connection control

Table 6-4 Messages used with ATM Point-to-multipoint call/connection control

Message	Reference
ADD PARTY	6.3.4.1
ADD PARTY ACKNOWLEDGE	6.3.4.2
PARTY ALERTING	6.3.4.3
ADD PARTY REJECT	6.3.4.4
DROP PARTY	6.3.4.5
DROP PARTY ACKNOWLEDGE	6.3.4.6

6.3.4.1 ADD PARTY

This message is sent to add a party to an existing connection.

Message type: ADD PARTY
Direction: Preceding to Succeeding
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
AAL parameters	6.4.5.8	O(1)	4-21
Broadband high layer information	6.4.5.11	O(1)	4-13
Broadband low layer information	6.4.5.12	O(1)	4-17
Called party number	6.4.5.15	M	(2)
Calling party soft PVPC or PVCC	6.4.6.1	O(3)	4-10
Called party soft PVPC or PVCC	6.4.6.2	O(4)	4-11
Called party subaddress	6.4.5.16	O(1)	4-25
Calling party number	6.4.5.17	O(1)	4-26
Calling party subaddress	6.4.5.18	O(1)	4-25
Broadband repeat indicator	6.4.5.13	M(5)	5
Designated transit list	6.4.6.4	M(6)	33-546
Endpoint reference	6.4.8.1	M(7)	7
End-to-end transit delay	6.4.5.24	O(1)	4-12
Generic identifier transport	6.4.5.31	O(1,8)	4-33
Notification indicator	6.4.5.27	O(1)	4-*
Transit network selection	6.4.5.30	O(1)	4-8

Note 1 - Included if the received add party indication contains this information.
Note 2 - Minimum length depends on the numbering plan. Maximum length is 25 octets.
Note 3 - Included in case of soft PVCC setup, when the calling endpoint wants to inform the values used for the PVC segment at the calling end.

Note 4 - Included in case of soft PVCC setup.
Note 5 - Indicates the order of Designated transit list information elements in the DTL stack. This information element is present even when there is only one Designated transit list information element.
Note 6 - Included by the source node to indicate the hierarchical source route for the party connection. Included by the node at the entry to a hierarchical level to indicate the path through that hierarchical level. This information element may be repeated up to 10 times.
Note 7 - The endpoint reference must be unique within a given call reference on a given link.
Note 8 - This information element may be present up to 3 times.

Figure 6-19 ADD PARTY message content

6.3.4.2 ADD PARTY ACKNOWLEDGE

This message is sent to acknowledge that the ADD PARTY request was successful.

Message type: ADD PARTY ACKNOWLEDGE
Direction: Succeeding to Preceding
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Broadband low layer information	6.4.5.12	O(1)	4-17
AAL parameters	6.4.5.8	O(1)	4-11
Called party soft PVPC or PVCC	6.4.6.2	O(1)	4-11
Connected number	6.4.5.20	O(1)	4-26
Connected subaddress	6.4.5.21	O(1)	4-25
Endpoint reference	6.4.8.1	M(2)	7
End-to-end transit delay	6.4.5.24	O(1)	4-7
Generic identifier transport	6.4.5.31	O(1,3)	4-33
Notification indicator	6.4.5.27	O(1)	4-*

Note 1 - Included if the received add party acknowledge indication contains this information.
Note 2 - The endpoint reference must be the same value as in the ADD PARTY message being responded to.
Note 3 - This information element may be present up to 3 times.

Figure 6-20 ADD PARTY ACKNOWLEDGE message content

6.3.4.3 PARTY ALERTING

This message is sent by the succeeding side to indicate that called user alerting has been initiated, if ALERTING is received.

Message Type: PARTY ALERTING
Direction: Succeeding to Preceding
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Endpoint reference	6.4.8.1	M	7
Notification indicator	6.4.5.27	O(1)	4-*
Generic identifier transport	6.4.5.31	O(1,2)	4-33

Note 1 - Included if the received party alerting indication contains this information.
Note 2 - This information element may be present up to 3 times.

Figure 6-21 PARTY ALERTING message content

6.3.4.4 ADD PARTY REJECT

This message is sent to acknowledge that the ADD PARTY request was not successful.

Message type: ADD PARTY REJECT
Direction: Succeeding to Preceding
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Cause	6.4.5.19	M	6-34
Crankback	6.4.6.3	O(1)	4-72
Endpoint reference	6.4.8.1	M(2)	7
Generic identifier transport	6.4.5.31	O(3,4)	4-33

Note 1 - Included to indicate crankback.
Note 2 - The endpoint reference must be the same value as in the ADD PARTY message being responded to.
Note 3 - Included if the received add party reject indication contains this information.
Note 4 - This information element may be present up to 3 times.

Figure 6-22 ADD PARTY REJECT message content

6.3.4.5 DROP PARTY

This message is sent to drop (clear) a party from an existing point-to-multipoint connection.

Message type: DROP PARTY
Direction: Both
Significance: Global

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Cause	6.4.5.19	M	6-34
Endpoint reference	6.4.8.1	M	7
Notification indicator	6.4.5.27	O(1)	4-*
Generic identifier transport	6.4.5.31	O(1,2)	4-33

Note 1 - Included if the received drop party indication contains this information.
Note 2 - This information element may be present up to 3 times.

Figure 6-23 DROP PARTY message content

6.3.4.6 DROP PARTY ACKNOWLEDGE

This message is sent in response to a DROP PARTY message to indicate that the party was dropped from the connection.

Message type: DROP PARTY ACKNOWLEDGE
Direction: Both
Significance: Local

Information Element	Reference	Type	Length
Protocol discriminator	6.4.2	M	1
Call reference	6.4.3	M	4
Message type	6.4.4.1	M	2
Message length	6.4.4.2	M	2
Cause	6.4.5.19	O(1)	4-34
Endpoint reference	6.4.8.1	M	7

Note 1 - Mandatory when DROP PARTY ACKNOWLEDGE is sent as a result of an error condition. This information element may appear twice in the message.

Figure 6-24 DROP PARTY ACKNOWLEDGE message content

6.4 General message format and information element coding

The figures and text in this section describe message contents.

6.4.1 Overview

Section 4.1 of Recommendation Q.2931 applies with the following change:
The last sentence of the paragraph immediately below Figure 4-1/Q.2931 does not apply.

6.4.2 Protocol discriminator

Section 4.2 of Recommendation Q.2931 applies with the following changes:

In the first sentence of the first paragraph, replace 'user-network' by 'PNNI'.

The Protocol discriminator is coded as shown in the figure below:

Bits 8	7	6	5	4	3	2	1	Octets
PNNI signalling messages								
1	1	1	1	0	0	0	0	1
protocol discriminator								

6.4.3 Call reference

Section 4.3 of Recommendation Q.2931 applies with the following change:

In the first sentence of the first paragraph, replace 'user-network interface' by 'PNNI'.

6.4.4 Message type and message length

6.4.4.1 Message type

Section 4.4.1 of Recommendation Q.2931 applies with the following changes:

In Figure 4-6/Q.2931 - message type format Octet 2 bit 4 is changed from “Spare” to “Pass along request”. The coding in Table 4-2/Q.2931 (part 2 of 2) is modified as follows:

- Pass along request (Octet 2)
Bit
4
1 Pass along request
0 No pass along request

The following message types are not supported:
CONNECT ACKNOWLEDGE
SETUP ACKNOWLEDGE
INFORMATION

Escape to national specific message types is not supported.

6.4.4.2 Message length

See section 4.4.2 of Recommendation Q.2931.

6.4.5 Variable length information elements

6.4.5.1 Coding rules

Section 2 §4.5.1/Q.2931 of UNI 4.0 signalling specification applies with the following change:

In Figure 4-8/Q.2931 - General information element format Octet 2 bit 4 is changed from “Reserved” to “Pass along request”. The coding in Table 4-3/Q.2931 (part 2 of 2) Octet 2 bit 4 is modified as follows:

- Pass along request (Octet 2)

Bit
4
1 Pass along request
0 No pass along request

Maximum length and maximum number of occurrences are specified for each supported information element, are given as follows (note: this additional information is presented for clarification only; for ITU defined information elements, the values are consistent with the specifications of Q.2931, Q.2961 and Q.2971):

Table 6-5 Information elements used in PNNI

Bits 8 7 6 5	4 3 2 1	Information Element	Max Length	Max no. of Occurrences
0 0 0 0	0 1 0 0	Narrow-band bearer capability [1,2]	14	3
0 0 0 0	1 0 0 0	Cause [1]	34	2
0 0 0 1	0 1 0 0	Call state	5	1
0 0 0 1	1 1 1 0	Progress indicator [1]	6	2
0 0 1 0	0 1 1 1	Notification indicator	[3]	[3]
0 1 0 0	0 0 1 0	End-to-end transit delay	13	1
0 1 0 0	1 1 0 0	Connected number	26	1
0 1 0 0	1 1 0 1	Connected subaddress	25	1
0 1 0 1	0 1 0 0	Endpoint reference	7	1
0 1 0 1	0 1 0 1	Endpoint state	5	1
0 1 0 1	1 0 0 0	ATM adaptation layer parameters	21	1
0 1 0 1	1 0 0 1	ATM traffic descriptor	30	1
0 1 0 1	1 0 1 0	Connection identifier	9	1
0 1 0 1	1 1 0 0	Quality of service parameter	6	1
0 1 0 1	1 1 0 1	Broadband high layer information	13	1
0 1 0 1	1 1 1 0	Broadband bearer capability	7	1
0 1 0 1	1 1 1 1	Broadband low-layer information [2]	17	3
0 1 1 0	0 0 0 0	Broadband locking shift	5	[4]
0 1 1 0	0 0 0 1	Broadband non-locking shift	5	[4]

0 1 1 0	0 0 1 1	Broadband repeat indicator[(1)]	5	3
0 1 1 0	1 1 0 0	Calling party number	26	1
0 1 1 0	1 1 0 1	Calling party subaddress[(1)]	25	2
0 1 1 1	0 0 0 0	Called party number	25	1
0 1 1 1	0 0 0 1	Called party subaddress[(1)]	25	2
0 1 1 1	1 0 0 0	Transit network selection	9	1
0 1 1 1	1 0 0 1	Restart indicator	5	1
0 1 1 1	1 1 0 0	Narrow-band low layer compatibility[(2)]	20	2
0 1 1 1	1 1 0 1	Narrow-band high layer compatibility[(1)]	7	2
0 1 1 1	1 1 1 1	Generic identifier transport[(1)]	33	3
1 0 0 0	0 0 0 1	Minimum acceptable ATM traffic descriptor	20	1
1 0 0 0	0 0 1 0	Alternate ATM traffic descriptor	30	1
1 0 0 0	0 1 0 0	ABR setup parameters	36	1
1 1 1 0	0 0 0 0	Called party soft PVPC or PVCC	11	1
1 1 1 0	0 0 0 1	Crankback	72	1
1 1 1 0	0 0 1 0	Designated transit list[(2)]	546	10
1 1 1 0	0 0 1 1	Calling party soft PVPC or PVCC	10	1
1 1 1 0	0 1 0 0	ABR additional parameters	14	1
1 1 1 0	1 0 1 1	Connection scope selection	6	1
1 1 1 0	1 1 0 0	Extended QoS parameters	25	1

Note 1 - This information element may be repeated without the Broadband repeat indicator information element.

Note 2 - This information element may be repeated in conjunction with the Broadband repeat indicator information element.

Note 3 - The maximum length and the number of repetitions of this information element are network dependent.

Note 4 - See section 6.5.6.8.1 for details.

Coding standard for ISO/IEC standard and national standard not supported.
Reservation of the value "1111 1111" of the information element identifier is not supported.

6.4.5.2 Extension of codesets

See section 2 §4.5.2/Q.2931 of UNI 4.0 signalling specification.

6.4.5.3 Broadband locking shift procedures

See section 2 §4.5.3/Q.2931 of UNI 4.0 signalling specification.

6.4.5.4 Broadband non-locking shift procedures

See section 2 §4.5.4/Q.2931 of UNI 4.0 signalling specification.

6.4.5.5 ABR additional parameters

See section 10.1.2.1 of UNI 4.0 signalling specification.

6.4.5.6 ABR setup parameters

See section 10.1.2.2 of UNI 4.0 signalling specification, with the exception that all octet groups are required.

6.4.5.7 Alternative ATM traffic descriptor

See section 8.1.2.1 of UNI 4.0 signalling specification.

6.4.5.8 ATM adaptation layer parameters

See section 2 §4.5.5/Q.2931 of UNI 4.0 signalling specification.

6.4.5.9 ATM traffic descriptor

Section 2 §4.5.6/Q.2931 of UNI 4.0 signalling specification applies with the additions in Section 10.1.2.3 of the UNI 4.0 signalling specification and with the following changes:

Tb (tagging backward) (octet 17.1)

Bit 2	in CONNECT message (Note 1)	in SETUP message (Note 2)
0	tagging not allowed	tagging not supported
1	tagging requested	tagging supported.

Note 1 - If octet 17.1 is omitted, a default of “tagging not allowed” shall apply.
Note 2 - If octet 17.1 is omitted, a default of “tagging not supported” shall apply.

Tf (tagging forward) (octet 17.1)

Bit 1	in SETUP message (Note 3)	in CONNECT message (Note 4)
0	tagging not allowed	tagging not applied
1	tagging requested.	tagging applied

Note 3 - If octet 17.1 is omitted, a default of “tagging not allowed” shall apply.
Note 4 - If octet 17.1 is omitted, a default of “tagging not applied” shall apply.

Coding	Meaning
Tb = "Tagging not allowed"	Backward traffic from the succeeding exchange will already be "tagged"
Tb = "Tagging requested"	Backward traffic from the succeeding exchange might not be "tagged" and need only conform to the CLP=0+1 traffic parameters
Tb = "Tagging not supported"	Backward traffic from the succeeding exchange must already be "tagged"
Tb = "Tagging supported"	Preceding exchange will support backward traffic from the succeeding exchange that is not "tagged" and that need only conform to the CLP=0+1 traffic parameters
Tf = "Tagging not allowed"	Forward traffic from the preceding exchange will already be "tagged"
Tf = "Tagging requested"	Preceding exchange requesting permission to send forward traffic to the succeeding exchange that is not "tagged" and that need only conform to the CLP=0+1 traffic parameters
Tf = "Tagging not applied"	Forward traffic from the preceding exchange must already be "tagged"
Tf = "Tagging applied"	Succeeding exchange will allow forward traffic to the succeeding exchange that is not "tagged" and that need only conform to the CLP=0+1 traffic parameters

6.4.5.10 Broadband bearer capability

See section 2 §4.5.7/Q.2931 of UNI 4.0 Signalling specification.

6.4.5.11 Broadband high layer information

See section 2 §4.5.8/Q.2931 of UNI 4.0 signalling specification.

6.4.5.12 Broadband low layer information

See section 2 §4.5.9/Q.2931 of UNI 4.0 signalling specification.

6.4.5.13 Broadband repeat indicator

See section 4.5.19 of Recommendation Q.2931 with the following changes in octet 5.

Broadband repeat indication (octet 5)

Bits 4 3 2 1	Meaning
0 0 1 0	Prioritized list for selecting one possibility
1 0 1 0	Last-in, First out stack (Note)

Note - The last appearance of the repeated information element is interpreted as the top (last entered) element of the stack. The second-to-last, third-to-last etc. occurrences of the repeated information element shall be interpreted as the second, third etc. entries on the stack.

6.4.5.14 Call state

The purpose of the Call state information element is to describe the current status of a call at the PNNI or a global interface state.

8	7	6	5	4	3	2	1	Octet
Call state 0 0 0 1 0 1 0 0 information element identifier								1
1 ext	Coding standard		IE Instruction field					2
Length of the call state contents								3
Length of the call state contents (continued)								4
0 0 Spare		PNNI call state value / global interface state value						5

Figure 6-25 Call state information element

PNNI call state value (octet 5) (Note)

Bits 6	5	4	3	2	1	*Meaning*
0	0	0	0	0	0	NN0 - Null
0	0	0	0	0	1	NN1 - Call initiated
0	0	0	0	1	1	NN3 - Call proceeding sent
0	0	0	1	0	0	NN4 - Alerting delivered
0	0	0	1	1	0	NN6 - Call present
0	0	0	1	1	1	NN7 - Alerting received
0	0	1	0	0	1	NN9 - Call proceeding received
0	0	1	0	1	0	NN10 - Active
0	0	1	0	1	1	NN11 - Release request
0	0	1	1	0	0	NN12 - Release indication

Note - PNNI call state values are coded using coding standard “1 1” ATM Forum specific.

Global interface state value (octet 5)

Bits 6	5	4	3	2	1	Meaning
0	0	0	0	0	0	REST 0 - Null
1	1	1	1	0	1	REST 1 - Restart request
1	1	1	1	1	0	REST 2 - Restart

6.4.5.15 Called party number

See section 2 §4.5.11/Q.2931 of UNI 4.0 signalling specification.

6.4.5.16 Called party subaddress

See section 2 §4.5.12/Q.2931 of UNI 4.0 signalling specification.

6.4.5.17 Calling party number

See section 2 §4.5.13/Q.2931 of UNI 4.0 signalling specification.

6.4.5.18 Calling party subaddress

See section 2 §4.5.14/Q.2931 of UNI 4.0 signalling specification.

6.4.5.19 Cause

See section 2 §4.5.15/Q.2931 of UNI 4.0 signalling specification. In addition the following PNNI specific cause values, using coding standard 1 1, are applicable:

Bits 7 6 5 4 3 2 1	Number	Meaning	Diagnostics
0 1 1 0 1 0 1	53	Call cleared due to change in PGL	
0 1 0 0 0 1 0	34	Requested called party soft PVPC or PVCC not available	

6.4.5.20 Connected number

See Annex 4 section A4.5 of the UNI 4.0 signalling specification.

6.4.5.21 Connected subaddress

See Annex 4 section A4.5 of the UNI 4.0 signalling specification.

6.4.5.22 Connection identifier

Section 2 §4.5.16/Q.2931 of UNI 4.0 signalling specification applies with the following change:

associated signalling is supported.

6.4.5.23 Connection scope selection

See section 7.1.3.1 of UNI 4.0 signalling specification.

6.4.5.24 End-to-end transit delay

The purpose of the End-to-end transit delay information element is to indicate the forward maximum cell transfer delay acceptable on a per call basis and to indicate the cumulative forward maximum cell transfer delay to be expected for a connection. Note that the cumulative backward maximum cell transfer delay can be derived from the cumulative forward maximum cell transfer delay and the cumulative forward and backward cell delay variation.

The maximum cell transfer delay of user data transferred during the data transfer phase on the user plane is defined in Section 3.6.1 of the Traffic Management 4.0 Specification.

The End-to-end transit delay is coded as shown in Figure 6-26 of this specification.

The maximum length of this information element is 13 octets.

8	7	6	5	4	3	2	1	Octet
End-to-end transit delay								
0	1	0	0	0	0	1	0	1
information element identifier								
1 ext	Coding standard		IE Instruction field					2
Length of the End-to-end transit delay contents								3
Length of the End-to-end transit delay contents (continued)								4
0	0	0	0	0	0	0	1	5*
Cumulative Forward Maximum Cell Transfer Delay Identifier								(Note 2)
Cumulative Forward Maximum Cell Transfer Delay								5.1*
Cumulative Forward Maximum Cell Transfer Delay (continued)								5.2*
0	0	0	0	1	0	1	1	6*
PNNI Acceptable Forward Maximum Cell Transfer Delay Identifier								
PNNI Acceptable Forward Maximum Cell Transfer Delay								6.1*
PNNI Acceptable Forward Maximum Cell Transfer Delay (continued)								6.2*
PNNI Acceptable Forward Maximum Cell Transfer Delay (continued)								6.3*
0	0	0	1	0	0	0	1	7*
PNNI Cumulative Forward Maximum Cell Transfer Delay Identifier								(Note 2)
PNNI Cumulative Forward Maximum Cell Transfer Delay								7.1*
PNNI Cumulative Forward Maximum Cell Transfer Delay (continued)								7.2*
PNNI Cumulative Forward Maximum Cell Transfer Delay (continued)								7.3*
0	0	0	0	1	0	1	0	8* (Note 1)
Network generated indicator								

Note 1 - Included if and only if the origin of this information element is other than the originating user.

Note 2 - Octet groups 5 and 7 are mutually exclusive. Octet group 5 is used in the CONNECT and ADD PARTY ACKNOWLEDGE messages and octet group 7 is used in the SETUP and ADD PARTY messages.

Figure 6-26 End-to-end transit delay information element

Coding standard (octet 2)

Bits 7 6	Meaning
1 1	ATM Forum specific

Cumulative Forward Maximum Cell Transfer Delay (octets 5.1-5.2)

The cumulative forward maximum cell transfer delay value is expressed in units of milliseconds. It is coded as a 16-bit binary integer, with Bit 8 of the first octet being the most significant bit and Bit 1 of the second octet being the least significant bit.

PNNI Acceptable Forward Maximum Cell Transfer Delay (octets 6.1-6.3)

The PNNI acceptable forward maximum cell transfer delay parameter indicates the calling user's highest acceptable (least desired) maximum cell transfer delay value, expressed in units of microseconds. It is coded as a 24-bit binary integer, with Bit 8 of the first octet being the most significant bit and Bit 1 of the third octet being the least significant bit. The value "1111 1111 1111 1111 1111 1111", however, is not to be interpreted as an acceptable maximum cell transfer delay value. This codepoint indicates: 'any forward maximum cell transfer delay value acceptable'.

PNNI Cumulative Forward Maximum Cell Transfer Delay (octets 7.1-7.3)

The PNNI cumulative forward maximum cell transfer delay value is expressed in units of microseconds. It is coded as a 24-bit binary integer, with Bit 8 of the first octet being the most significant bit and Bit 1 of the third octet being the least significant bit.

Network generated indicator (octet 8)

If this subfield is not present, then the origin of this information element is the originating user (so the called party can assume that the received cumulative values are end-to-end values). Otherwise, the presence of this subfield indicates the origin of this information element is other than the originating user

6.4.5.25 Extended QoS parameters

See section 9.1.2.2 of UNI 4.0 signalling specification.

6.4.5.26 Minimum acceptable ATM traffic descriptor

See section 8.1.2.2 of UNI 4.0 signalling specification.

6.4.5.27 Notification indicator

See Annex 7, section A7.1.2.1 of the UNI 4.0 signalling specification, and section 4.5.23 of Recommendation Q.2931.

6.4.5.28 Quality of service parameter

See section 2 §4.5.18/Q.2931 of UNI 4.0 signalling specification.

6.4.5.29 Restart indicator

See section 2 §4.5.20/Q.2931 of UNI 4.0 signalling specification.

6.4.5.30 Transit network selection

See section 2 §4.5.22/Q.2931 of UNI 4.0 signalling specification.

6.4.5.31 Generic identifier transport

See section 2.1.1 of the UNI 4.0 signalling specification.

6.4.6 Information elements specific to PNNI

6.4.6.1 Calling party soft PVPC or PVCC

The purpose of the Calling party soft PVPC or PVCC information element is to indicate the VPI or VPI/VCI values used for the PVC segment by the calling connecting point. These values are conveyed to the called connecting point transparently.

8	7	6	5	4	3	2	1	Octet
Calling party soft PVPC or PVCC Information element identifier								
1	1	1	0	0	0	1	1	1
1 ext	Coding standard		IE Instruction Field					2
Length of calling party soft PVPC or PVCC contents								3
Length of calling party soft PVPC or PVCC contents (continued)								4
1	0	0	0	0	0	0	1	5*
VPI identifier								
VPI value								5.1*
								5.2*
1	0	0	0	0	0	1	0	6* (Note 1)
VCI identifier								
VCI value								6.1*
								6.2*

Note 1 - This octet group is only present in case of a soft PVCC.

Figure 6-27 Calling party soft PVPC or PVCC Information element

Coding standard (octet 2)

Bits 7 6	Meaning
1 1	ATM Forum specific

VPI value (octets 5.1 and 5.2)

A two octet binary number assigned to the ATM connection representing the identifier of the Virtual Path Identifier. The VPI value is coded in the low order 12 bits and the high order 4 bits (octet 5.1 bits 8 to 5) are coded to all zero.

VCI value (octet 6.1 and 6.2)

A two octet binary number assigned to the ATM connection representing the identifier of the Virtual Channel Connection.

6.4.6.2 Called party soft PVPC or PVCC

The purpose of the Called party soft PVPC or PVCC information element is to indicate the VPI or VPI/VCI values of a PVC segment between the called connecting point and the user of a PVPC or PVCC respectively. These values are conveyed to the called connecting point transparently.

8	7	6	5	4	3	2	1	Octet
Called party soft PVPC or PVCC								
1	1	1	0	0	0	0	0	1
Information element identifier								
1 ext	Coding standard		IE Instruction Field					2
Length of called party soft PVPC or PVCC contents								3
Length of called party soft PVPC or PVCC contents (continued)								4
Selection type								5
1	0	0	0	0	0	0	1	6* (Note 1)
VPI identifier								
VPI value								6.1*
								6.2*
1	0	0	0	0	0	1	0	7* (Note 1,2)
VCI identifier								
VCI value								7.1*
								7.2*

Note 1 - This octet group is not included when the selection type indicates “any value”. If present, it shall be ignored.
Note 2 - This octet group is only present in case of a soft PVCC.

Figure 6-28 Called party soft PVPC or PVCC Information element

Coding standard (octet 2)

Bits 7 6	Meaning
1 1	ATM Forum specific

Selection type (octet 5)

Bits 8 7 6 5 4 3 2 1	Meaning
0 0 0 0 0 0 0 0	Any value
0 0 0 0 0 0 1 0	Required value
0 0 0 0 0 1 0 0	Assigned value

VPI value (octets 6.1 and 6.2)

A two octet binary number assigned to the ATM connection representing the identifier of the Virtual Path Identifier. The VPI value is coded in the low order 12 bits and the high order 4 bits (octet 6.1 bits 8 to 5) are coded to all zero.

VCI value (octet 7.1 and 7.2)

A two octet binary number assigned to the ATM connection representing the identifier of the Virtual Channel Connection.

6.4.6.3 Crankback

The purpose of the Crankback information element is to indicate that crankback procedures have been initiated. It also indicates the node or link where the call/connection or party cannot be accepted, and the level of PNNI hierarchy at which crankback is being carried out.

8	7	6	5	4	3	2	1	Octet
Crankback								
1	1	1	0	0	0	0	1	1
Information element identifier								
1 ext	Coding standard		IE Instruction Field					2
Length of crankback contents								3
Length of crankback contents (continued)								4
Crankback level								5
Blocked transit type								6
Blocked transit identifier (format and length dependent on value of blocked transit type)								6.1 etc
Crankback cause								7 (note)
Crankback cause diagnostics (if any)								7.1* etc.

Note - The crankback cause in addition to the Cause information element identifies the cause for crankback.

Figure 6-29 Crankback Information element

Coding standard (octet 2)

Bits 7 6	Meaning
1 1	ATM Forum specific

Crankback level (octet 5)

The Crankback level indicates the level of PNNI hierarchy at which the call/connection or party is being cranked back. The crankback level coded in binary, with a length of 1 octet.

Blocked transit type (octet 6)

Bits 8 7 6 5 4 3 2 1	Meaning	Length of blocked transit identifier
0 0 0 0 0 0 1 0	call or party has been blocked at the succeeding end of this interface	0
0 0 0 0 0 0 1 1	blocked node	22

0 0 0 0 0 1 0 0	blocked link	48

Blocked transit identifier (octet 6.1 etc)

The blocked transit identifier format depends on the blocked transit type.

For Blocked transit type = "blocked node identifier"

Blocked node identifier	6.1 to 6.22

Blocked node identifier (octets 6.1 to 6.22)

The Blocked node identifier identifies the logical node at which the call/connection or party has been blocked. It is coded in binary, with a length of 22 octets. Section 5.3.3 describes the abstract syntax of logical node identifiers.

For Blocked transit type = "blocked link identifier"

Blocked link's preceding node identifier	6.1 to 6.22
Blocked link's port identifier	6.23 to 6.26
Blocked link's succeeding node identifier	6.27 to 6.48

Blocked link's preceding node identifier (octets 6.1 to 6.22)

The Blocked link's preceding node identifier identifies the logical node preceding a link at which the call/connection or party has been blocked. It is coded in binary, with a length of 22 octets. Section 5.3.3 describes the syntax of logical node identifiers.

Blocked link's port identifier (octet 6.23 to 6.26)

The Blocked link's port identifier identifies a logical port of the blocked link's preceding node identifier. The combination of the blocked link's preceding node identifier and the blocked link's port identifier unambiguously (but not uniquely) identifies the link at which the call/connection or party has been blocked. A port identifier of 0 indicates that all links from the blocked link's preceding node to the blocked link's succeeding node should be considered blocked. The logical port identifier is coded in binary, with a length of 4 octets. Section 5.3.4 describes the syntax of logical port identifiers.

Blocked link's succeeding node identifier (octets 6.27 to 6.48)

The Blocked link's succeeding node identifier identifies the logical node succeeding a link at which the call/connection or party has been blocked. It is coded in binary, with a length of 22 octets. Section 5.3.3 describes the syntax of logical node identifiers.

Crankback cause (octet 7)

Bits 8 7 6 5 4 3 2 1	Number	Meaning	Diagnostics
0 0 0 0 0 0 1 0	2	transit network unreachable	

0 0 0 0 0 0 1 1	3	destination unreachable	
0 0 1 0 0 0 0 0	32	too many pending add party requests	
0 0 1 0 0 0 1 1	35	requested VPCI/VCI not available	
0 0 1 0 0 1 0 1	37	user cell rate not available	Note 1
0 0 1 0 0 1 1 0	38	network out of order	
0 0 1 0 1 0 0 1	41	temporary failure	
0 0 1 0 1 1 0 1	45	no VPCI/VCI available	
0 0 1 0 1 1 1 1	47	resource unavailable, unspecified	
0 0 1 1 0 0 0 1	49	Quality of Service unavailable	Note 2
0 0 1 1 1 0 0 1	57	bearer capability not authorized	
0 0 1 1 1 0 1 0	58	bearer capability not presently available	
0 0 1 1 1 1 1 1	63	service or option not available, unspecified	
0 1 0 0 0 0 0 1	65	bearer service not implemented	
0 1 0 0 1 0 0 1	73	unsupported combination of traffic parameters	
1 0 0 0 0 0 0 0	128	next node unreachable	
1 0 1 0 0 0 0 0	160	DTL Transit not my node ID	

Crankback cause diagnostics (Octet 7.1 etc.)

The use of crankback cause diagnostics and the coding format of the diagnostics varies among the different cause values. The diagnostics column of the table above indicates whether diagnostics are applicable and the coding format of the diagnostics field.

Note 1 Updated topology state parameters for generic connection admission control can optionally be included in the crankback cause diagnostics. Refer to Annex B Section 8.4 for further details. These topology state parameters can be used instead of the most recent GCAC parameters received for the blocked node or link. They are superseded when the next set of topology state parameters are received, either in PTSEs or in other call clearing messages. If AvCR, CRM, and VF are included, then complex GCAC applies (see Section 5.13.4.2). If only AvCR is specified, then simple GCAC applies (see Section 5.13.4.3). Only updated topology state parameters for one direction can be included using the following coding:

8 7 6 5 4 3 2 1	Octet
Direction	7.1*
Port Identifier	7.2* to 7.5*
AvCR	7.6* to 7.9*
CRM	7.10* to 7.13*
VF	7.14* to 7.17*

Direction (octet 7.1)

Bits 8 7 6 5	Bits 4 3 2 1	Meaning
0 0 0 0	0 0 0 0	Forward
0 0 0 0	0 0 0 1	Backward

Port Identifier (octets 7.2 to 7.5)

The port identifier at the preceding node of the link to which the updated topology state parameters apply. Note that this port ID is determined by the preceding node of a blocked link. If the blocked transit is a node or when sent by the succeeding side of a blocked link, the port identifier is set to zero.

If the Direction (octet 7.1) is Forward, then the updated topology state parameters apply to the outgoing direction of this port from the blocked link's preceding node. If the direction is backward and the blocked link is a horizontal link, then the updated topology state parameters apply to the outgoing direction of the corresponding port on the blocked link's succeeding node. If the direction is backward and the blocked link is an uplink, then the updated topology state parameters apply to the incoming direction of this port on the blocked link's preceding node.

Note 2 - The following coding is used:

8	7	6	5	4	3	2	1	Octet
Spare				CTD	CDV	CLR	Other QoS	7.1

CTD (octet 7.1)

Bits 4	Meaning
0	CTD available
1	CTD unavailable

CDV (octet 7.1)

Bits 3	Meaning
0	CDV available
1	CDV unavailable

CLR (octet 7.1)

Bits 2	Meaning
0	CLR available
1	CLR unavailable

Other QoS parameters (octet 7.1)

Bits 1	Meaning
0	Other QoS parameters available
1	Other QoS parameters unavailable

6.4.6.4 Designated transit list

The purpose of the designated transit list is to indicate the logical nodes and logical links that a connection is to traverse through a peer group at some level of hierarchy. Annex A describes the procedures for constructing designated transit list.

8	7	6	5	4	3	2	1	Octet
Designated transit list								
1	1	1	0	0	0	1	0	1
Information element identifier								
1 ext	Coding standard		IE Instruction Field					2
Length of designated transit list contents								3
Length of designated transit list contents (continued)								4
Current transit pointer								5
Current transit pointer (continued)								6
0	0	0	0	0	0	0	1	7 (Note 3)
Logical node / logical port indicator								
Logical node identifier								7.1 to 7.22
Logical port identifier								7.23 to 7.26

Note 1 - The designated transit list is an ordered list. Interpretation of octet groups within the Designated transit list information element is position dependent.

Note 2 - The designated transit list information element must be structured in a way such that each transit is capable of advancing the current transit pointer from itself to the next node in the list.

Note 3 - Octet group 7 (in its entirety) may appear as many as 20 times.

Figure 6-30 Designated transit list information element

Coding standard (octet 2)

Bits 7 6	Meaning
1 1	ATM Forum specific

Current transit pointer (octets 5 and 6)

The current transit pointer is encoded as an octet offset pointer to the transit node/logical port which indicates the current call/connection’s progress along the designated transit list. When a node receives a DTL, this pointer points to the ID of this node or of an ancestor of this node. It is encoded as a 16 bit unsigned integer. The value 0 indicates the first transit that appears in the information element (i.e., the node that generated the information element), the value 27 indicates the second transit (i.e., the nearest neighbor of the node that generated the information element), the value 54 indicates the third transit, and so on.

Logical node identifier (octets 7.1 to 7.22)

The logical node identifier uniquely identifies a logical node (i.e., a real switching system or a peer group at some level of hierarchy) which the call/connection is to transit. It is coded in binary, with a length of 22 octets. Section 5.3.3 describes the abstract syntax and semantics of logical node identifiers.

Logical Port Identifier (octet 7.23 to 7.26)

The logical port identifier uniquely identifies a logical port of the logical node which the call/connection is to transit. The combination of Logical Node Identifier and Logical Port Identifier unambiguously (but not uniquely) identifies a logical link. The logical port identifier is coded in binary, with a length of 4 octets. The value 00000000 (hexadecimal) is reserved to indicate no logical port identified (.i.e., this transit only identifies a logical node). Section 5.3.4 describes the abstract syntax and semantics of logical port identifiers.

6.4.7 Information Elements for the support of 64 kbit/s based circuit mode services

The information elements described in this section are transported transparently through the PNNI except where explicitly addressed by procedures.

6.4.7.1 Narrow-band bearer capability

See section 4.6.2 of Recommendation Q.2931.

6.4.7.2 Narrow-band high layer compatibility

See section 4.6.3 of Recommendation Q.2931.

6.4.7.3 Narrow-band low layer compatibility

See section 4.6.4 of Recommendation Q.2931.

6.4.7.4 Progress indicator

See section 4.6.5 of Recommendation Q.2931.

6.4.8 Information Elements for Point-to-Multipoint Call/connection Control

6.4.8.1 Endpoint reference

See section 8.2.1 of ITU-T Recommendation Q.2971.

6.4.8.2 Endpoint state

See section 8.2.2 of ITU-T Recommendation Q.2971.

6.5 Call/connection control procedures for ATM point-to-point calls

This section describes procedures for point-to-point calls only. Section 6.6 contains additional procedures for point-to-multipoint calls as well as changes to the point-to-point procedures needed to support point-to-multipoint calls.

Procedures based on ATM Forum User-to-Network Interface (UNI) and ITU-T Recommendation Q.2931 are used to establish ATM switched virtual connections. The signalling virtual channel uses VPI=0,VCI=5 in case of non-associated signalling. In case of associated signalling, VCI=5 is used as the signalling channel in the VPC.

This specification allows for optional support of point-to-point switched virtual paths (SVPs). Except where noted, the references to switched virtual channels (SVCs) apply equally to SVPs.

6.5.1 Establishment of a signalling AAL

Before these procedures are invoked, an assured mode signalling AAL connection must be established between the two private network nodes. All layer 3 messages shall be sent to the signalling AAL using an AAL-DATA-REQUEST primitive. For more information on the Signalling ATM Adaptation layer (SAAL) Specification, see section 4.0 of UNI 4.0 Signalling specification.

Note: The service primitives used by Q.2931 entities to request and accept services from the SAAL are described in Annex A/Q.2931.

6.5.2 Call/connection establishment

6.5.2.1 Call/connection request

Call establishment shall be initiated by the preceding side by sending a SETUP message. The preceding side shall start timer T303 and enter the Call Present state. The message shall always contain a call reference, selected according to the procedures given in §6.4.3. This message is sent only if resources for the call are available; otherwise, the call is cleared towards the calling user.

The SETUP message shall contain all the information required to process the call. In particular, the called party address information is contained in the Called party number information element possibly supplemented by the Called party subaddress information element. The ATM traffic descriptor, Broadband bearer capability, and Quality of service parameter information elements are mandatory in the SETUP message.

When the SETUP message contains an ATM group address in the called party number information element, this indicates that the call shall be progressed to one of the members of the group within the connection scope indicated in the Connection scope selection information element (i.e., this call uses the ATM Anycast capability). In this case, if no Connection scope selection information element is included in the SETUP message, the node shall assume a default Connection scope selection of localNetwork(1).

If no response to the SETUP message is received by the preceding side before the first expiration of timer T303, the SETUP message may be retransmitted and timer T303 restarted.

If the preceding side does not receive any response to the SETUP after the final expiration of timer T303, the preceding side shall enter the Null state and send a RELEASE COMPLETE message to the succeeding side with cause #102, "recovery on timer expiry" and initiate clearing (without crankback) towards the calling party with cause #102, “recovery on timer expiry”. Call control shall be notified of the failure of the call.

On receipt of the SETUP message the succeeding side shall enter the Call Initiated state.

6.5.2.2 Connection identifier allocation/selection

Two cases exist:

i) Associated Signalling

The layer 3 signalling entity exclusively controls the VCs in the VPC which carries its signalling VC.

ii) Non-Associated Signalling

The layer 3 signalling entity controls the VCs in the VPC which carries its signalling VC and may control VCs in other VPCs.

In PNNI, the virtual channel with VPI=0/VCI=5 is the only virtual channel used for non-associated signalling. This signalling channel does not control virtual channels within VPCs configured for use with associated signalling, but does control all the remaining virtual channels and virtual paths on the physical link.

The PNNI interface shall support the non-associated signalling procedures and may as an option support the associated signalling procedures. The associated signalling procedures are used only when two PNNI network nodes are connected by a virtual path connection used as a logical link.

When a network node receives a connection identifier information element with the VP-associated signalling field (see §6.4.5.22) coded with a value not supported by this network node, the call shall be rejected with cause #36, "VPCI/VCI assignment failure".

6.5.2.2.1 Associated signalling

For associated signalling, the preceding side requests a virtual channel in the VPC carrying the signalling VC. The VPC carrying the signalling VC is implicitly indicated.

In the Connection identifier information element, the VP associated signalling field is coded as “VP associated signalling” in the Connection identifier information element and one of the following values is indicated in the Preferred/exclusive field:

a) Exclusive VPCI; any VCI; or,

b) Exclusive VPCI; exclusive VCI.

In case a), the succeeding side selects any available VCI within the VPC carrying the signalling VC.

In case b), if the indicated VCI within the VPC carrying the signalling VC is available, the succeeding side selects it for the call.

The selected VCI value is indicated in the Connection identifier information element in the first message returned by the succeeding side in response to the SETUP message (i.e., CALL PROCEEDING message). The VP associated signalling field is coded as “VP associated signalling”. The Preferred/exclusive field is coded as “Exclusive VPCI; exclusive VCI”.

In case a), if no VCI is available, a RELEASE COMPLETE message with cause #45, "no VPCI/VCI available", is sent by the succeeding side. In addition, a Crankback information element is included with crankback cause #45.

In case b), if the indicated VCI is not available, a RELEASE COMPLETE message with the cause #35, "requested VPCI/VCI not available", is sent by the succeeding side. In addition, a Crankback information element is included with crankback cause #35.

Call collision can occur when both sides of an interface simultaneously transfer SETUP messages indicating the same exclusive VPCI and VCI. For PNNI interfaces, in order to avoid call collision, the side which has the higher node identifier (see Section 5.3.3) shall allocate the connection identifier (VPCI, VCI) values. A preceding side which has a higher node identifier shall include a Connection identifier information element in the SETUP message with option (b) (exclusive VPCI and exclusive VCI). A SETUP message from a preceding side which has a lower node identifier shall use option (a).

6.5.2.2.2 Non-associated signalling

6.5.2.2.2.1 Allocation for switched virtual channels

When the preceding side requests a virtual channel in the SETUP message, the preceding side shall indicate one of the following:

a) Exclusive VPCI; any VCI;

b) Exclusive VPCI; exclusive VCI; or,

c) No indication is included (i.e., the Connection identifier information element is not included in the SETUP message).

In cases a) and b), the VP associated signalling field is coded as "explicit indication of VPCI" in the Connection identifier information element.

In cases a) and b), if the indicated VPCI is available, the succeeding side selects it for the call. In case a), the succeeding side selects any available VCI in the VPCI. In case b), if the indicated VCI is available within the VPCI, the succeeding side selects it for the call.

In case c), the succeeding side selects any available VPCI and VCI.

The selected VPCI/VCI value is indicated in the Connection identifier information element in the first message returned by the succeeding side in response to the SETUP message (i.e., CALL PROCEEDING message). The VP associated signalling field is coded as "explicit indication of VPCI". The Preferred/exclusive field is coded as "exclusive VPCI; exclusive VCI".

In cases a) and b), if the specified VPCI is not available, a RELEASE COMPLETE message with cause #35, "requested VPCI/VCI not available", is sent by the succeeding side. In addition, a Crankback information element is included with crankback cause #35.

In case a), if no VCI is available, a RELEASE COMPLETE message with cause #45, "no VPCI/VCI available", is sent by the succeeding side. In addition, a Crankback information element is included with crankback cause #45.

In case b), if the VCI in the indicated VPCI is not available, a RELEASE COMPLETE message with cause #35, "requested VPCI/VCI not available", is sent by the succeeding side. In addition, a Crankback information element is included with crankback cause #35.

In case c), if the succeeding side is not able to allocate a VCI in any VPCI, a RELEASE COMPLETE message with cause #45, "no VPCI/VCI available", is sent by the succeeding side. In addition, a Crankback information element is included with crankback cause #45.

Call collision can occur when both sides of an interface simultaneously transfer SETUP messages indicating the same exclusive VPCI and VCI. For PNNI interfaces, in order to avoid call collision, the side which has the higher node identifier (see Section 5.3.3) shall allocate the connection identifier (VPCI, VCI) values. A preceding side which has a higher node identifier shall include a Connection identifier information element in the SETUP message with option (b) (exclusive VPCI and exclusive VCI). A SETUP message from a preceding side which has a lower node identifier shall use options (a) or (c).

6.5.2.2.2.2 Allocation for switched virtual paths

When the establishment of a switched virtual path (SVP) is requested (i.e., the Bearer class field of the Broadband bearer capability information element in the SETUP message indicates "VP service"), the preceding side shall indicate one of the following:

c) No indication is included (i.e., the connection identifier IE is not included in the SETUP message)
d) Exclusive VPCI; no VCI

In case (c), the succeeding side selects any available VPCI.

In case (d), if the indicated VPCI is available, the succeeding side selects it for the call. The selected VPCI is indicated in the Connection identifier information element in the first message returned by the succeeding side in response to the SETUP message (i.e., CALL PROCEEDING). The VP associated signalling field is coded as "explicit indication of VPCI". The preferred/exclusive field is coded as "exclusive VPCI; no VCI".

In case (c), if the succeeding side is not able to allocate a VPCI, a RELEASE COMPLETE message with cause #45 "*No VPCI/VCI available*" is sent by the succeeding side. In addition, a Crankback information element is included with crankback cause #45.

In case (d), if the indicated VPCI is not available, a RELEASE COMPLETE message with cause #35 "*Requested VPCI/VCI not available*" is sent by the succeeding side. In addition, a Crankback information element is included with crankback cause #35.

Call collision can occur when both sides of an interface simultaneously transfer SETUP message indicating the same exclusive VPCI. For PNNI interfaces, in order to avoid call collision, the side which has the higher node identifier (see Section 5.3.3) shall allocate the connection identifier (VPCI) values. A preceding side which has a higher node identifier shall include a Connection identifier information element in the SETUP message with option (d) (exclusive VPCI and no VCI). A SETUP message from a preceding side which has a lower node identifier shall use option (c).

6.5.2.2.3 Use of VPCIs

The Connection identifier information element is used in signalling messages to identify the corresponding user information flow. The Connection identifier information element contains the Virtual Path Connection Identifier (VPCI) and the Virtual Channel Identifier (VCI). The VPCI is used instead of the Virtual Path Identifier (VPI) since Virtual Path Cross Connects may be used and multiple interfaces could be controlled by the signalling virtual channel.

Both the preceding side and the succeeding side must understand the relationship between the VPCI used in the signalling protocol and the actual VPI used for the user information flow. VPCIs only have significance with regard to a given signalling virtual channel.

For this Implementation Agreement, each non-associated signalling virtual channel only controls a single interface at the PNNI, and the VPCI and VPI have the same numerical value.

6.5.2.2.4 VPCI and VCI ranges

The range of valid VCI values is indicated below:

0-31	Not used for on-demand user plane connections
32-65535	Identifier of the Virtual Channel (Note)

Note - Some values in the range may not be available for use (e.g., some values may be used for permanent connections). The upper bound on the VCI range (i.e., 65535) may be further restricted by the number of active VCI bits. In addition, a need might arise to reserve more than 32 VCI values.

The range of valid VPCI values is as indicated below:

0-4095	Identifier of the Virtual Path (Note)

Note - Some values in the range may not be available for use (e.g., some values may be used for permanent virtual path connections). The upper bound on the VPCI range (i.e., 4095) may be further restricted by the number of active VPI bits.

6.5.2.3 Service category, traffic parameter, and QoS selection procedures

6.5.2.3.1 Determination of service category

See Annex 9 §A9.2 of UNI 4.0 signalling specification.

In addition, if the network node determines that the requested service category is not available, it shall initiate crankback in accordance with §6.5.3.3 and Annex B, with one of the following cause and crankback cause codes.

#57 bearer capability not authorized;
#58 bearer capability not presently available;
#65 bearer service not implemented.

6.5.2.3.2 Allowed combination of bearer capabilities, traffic parameters, and QoS

Annex 9 §A9.3 of UNI 4.0 signalling specification applies with the following addition:

If the succeeding side detects that the Broadband bearer capability, ATM traffic descriptor, End-to-end transit delay, and Extended QoS parameters information elements contain a non-supported set of parameters, the succeeding side shall return a RELEASE COMPLETE message with cause #73 "Unsupported combination of traffic parameters". In addition, a Crankback information element is included with crankback cause #73.

6.5.2.3.3 Traffic parameter selection procedures

If the succeeding side is not able to provide the indicated ATM traffic parameters, the succeeding side shall crankback the call, returning a RELEASE COMPLETE message with cause and crankback cause #37, "user cell rate unavailable".

If the received setup indication specifies a Tf field coded to “Tagging requested” then the preceding side:

- may set the Tf field coded as “Tagging requested” in the transmitted SETUP message, if this node does not perform tagging for this connection

- shall specify a Tf indication of “Tagging not allowed” in the transmitted SETUP message, if this node does perform tagging for this connection

If the received setup indication specified a Tf indication of “Tagging not allowed”, the preceding side shall specify a Tf indication of “Tagging not allowed” in the transmitted SETUP message.

If the received setup indication specifies a Tb indication of “Tagging not supported” then the preceding side:

- shall specify a Tb indication of “Tagging not supported” in the transmitted SETUP message, if this node will not perform tagging for this connection
- may set the Tb field coded as “Tagging supported” in the transmitted SETUP message, if this node can perform tagging for this connection

If the received setup indication specified a Tb field coded “Tagging supported”, the preceding side may send a Tb field coded “Tagging supported” in the transmitted SETUP message. (This indicates to the succeeding side that the capability of supporting tagging exists for backward traffic, and that the preceding side is willing to accept untagged traffic.)

The succeeding side may include the Tf field coded to “Tagging requested” in the SETUP message when the incoming call request contained such an indication. The succeeding side may include the Tb field coded to “Tagging supported” in the SETUP message when there are CLP=0 traffic parameters in the ATM traffic descriptor information element.

When a SETUP message with an indication of frame discard is received, the PNNI node follows the procedures described in section 2.2.1 of UNI 4.0 signalling specification.

6.5.2.3.4 Procedures for negotiation of traffic parameters during call/connection setup

When both the Minimum acceptable ATM traffic descriptor and the Alternative ATM traffic descriptor information elements are present in a SETUP message, a network node shall reject the connection establishment request as specified in 6.5.3 with cause #73, "Unsupported combination of traffic parameters" (without crankback). If the parameters of the Alternative ATM traffic descriptor information element or Minimum acceptable ATM traffic descriptor information element are not according to the allowed combinations as specified in Sections 6.4.5.7 and 6.4.5.2.6 respectively, the network shall handle these information elements as if they were non-mandatory information elements with content error as specified in 5.6.8/Q.2931.

When the Minimum acceptable ATM traffic descriptor is included in the SETUP message and a network node is able to provide the traffic parameter values specified in the ATM traffic descriptor information element, the network node shall progress the connection establishment request with both the ATM traffic descriptor information element and the Minimum acceptable ATM traffic descriptor element.

When the Minimum acceptable ATM traffic descriptor is included in the SETUP message and a network node is not able to provide some of the cell rates indicated in the ATM traffic descriptor information element, but is able to provide at least their corresponding cell rates in the Minimum acceptable ATM traffic descriptor information element, the network node shall progress the connection establishment request after adjusting the cell rates with reduced values at or above those specified in the Minimum acceptable ATM traffic descriptor information element. If some of the parameters in the Minimum acceptable ATM traffic descriptor information element are still less than the corresponding parameters in the modified ATM traffic descriptor information element then the call shall be progressed with the Minimum acceptable ATM traffic descriptor information element containing only such parameters in addition to the modified ATM traffic descriptor information element. Otherwise, the call shall progress with the modified ATM traffic descriptor information element and without the Minimum acceptable ATM traffic descriptor information element.

When the Alternative ATM traffic descriptor information element is included in the SETUP message and the network node is able to provide the traffic parameter values specified in the ATM traffic descriptor information element and the network node is able to provide the traffic parameter values specified in the Alternative ATM traffic descriptor information element, the network node shall progress the connection establishment request with both the ATM traffic descriptor information element and the Alternative ATM traffic descriptor information element.

When the Alternative ATM traffic descriptor information element is included in the SETUP message and a network node is able to provide the traffic parameter values specified in the ATM traffic descriptor information element, but is not able to provide the traffic parameter values specified in the Alternative ATM traffic descriptor information element, the network node shall progress the connection establishment request with the ATM traffic descriptor information element and without the Alternative ATM traffic descriptor information element.

When the Alternative ATM traffic descriptor information element is included in the SETUP message and a network node is not able to provide the traffic parameter values specified in the ATM traffic descriptor information element, but is able to provide the traffic parameter values specified in the Alternative ATM traffic descriptor information element, the network node shall progress the connection establishment request by using the contents of the Alternative ATM traffic descriptor information element as the ATM traffic descriptor and no Alternative ATM traffic descriptor information element shall be forwarded.

If a network node cannot support the traffic parameter values specified in the ATM traffic descriptor information element and cannot support the traffic parameter values in the Alternative ATM traffic descriptor information element or Minimum acceptable ATM traffic descriptor information element as appropriate, the network node shall reject the connection establishment request as specified in 6.5.3 with cause #37 "User cell rate unavailable". In addition, a Crankback information element with the corresponding crankback cause code shall be included.

6.5.2.3.5 QoS parameter selection procedures

QoS requirements are expressed inside PNNI networks through the values of individual quality of service parameters, which are included in the Extended QoS parameters information element and/or the End-to-end transit delay information element. The allowed set of individual QoS parameters in the SETUP message is determined by the ATM service category of the call.

For each individual QoS parameter, if an acceptable value of a parameter is included and the end-to-end value of that parameter is determined by accumulation, then the corresponding cumulative value of the parameter shall also be included. The cumulative forward and backward values of individual QoS parameters sent in the SETUP message are updated sequentially along the route of the call to determine the values to be expected for this call. The procedures specified in this section support the simple method for accumulation (i.e., additive) of delay parameters described in the ATM Forum Traffic Management 4.0 specification.

When the preceding side receives a setup indication that includes a QoS parameter information element, the preceding side shall include a QoS parameters information element in the corresponding SETUP message.

When the preceding side receives a setup indication from a previous interface that is not a PNNI interface, the ATM service category of the call is CBR, real-time VBR, or non-real-time VBR, and no Extended QoS parameters information element is contained in the received SETUP message, the preceding side shall generate an Extended QoS parameters information element, using a local mapping from the service category and the forward and backward QoS class fields in the QoS parameter information element. In addition, an End-to-end transit delay information element may be generated as part of the above mapping, if it was not contained in the received setup indication. When such a mapping is used, all individual QoS parameters for which values are implied (from the QoS classes included in the QoS parameters information element and the ATM service category of the call) must be specified, and the origin of each information element including one or more of the newly generated individual QoS parameters must be marked as an ‘intermediate network’ (i.e., in the Extended QoS parameters information element the Origin field is set to ‘intermediate network’, and in the End-to-end transit delay information element a ‘network-generated indicator’ is included). All cumulative parameter values generated from the mapping must be initialized to zero before beginning processing of the individual QoS parameters.

For each parameter contained in the Extended QoS parameters information element and/or the End-to-end transit delay information element, the preceding side shall take the following actions (whether the information element was contained in the received SETUP message or whether the information element was generated using a local mapping from the QoS class):

1. If the previous interface was not a PNNI interface and the parameter is cumulative:

   a) the preceding side shall increment the cumulative forward value of the parameter to account for expected increases due to user data transfer over the previous link, from the network boundary to this switching system.

   b) In addition, the preceding side shall increment the cumulative backward value of the parameter to account for expected increases due to user data transfer over the previous link, from this switching system to the network boundary, and also to account for expected increases due to user data transfer within this switching system.

   Note that the procedures described in this step are not necessarily associated with this PNNI interface; they may also be considered as procedures undertaken at the succeeding side of the previous (non-PNNI) interface. They are provided here in order to complete specification of the steps required to support the network side procedures in the UNI 4.0 Signalling specification.

2. The preceding side shall increment the forward cumulative value of the parameter, if the parameter is cumulative, to account for the expected increases due to user data transfer within this switching system and over the link from this switching system to the switching system at the succeeding side.

3. The preceding side shall determine if the highest/lowest acceptable values of that parameter can be supported. If no values {less than/greater than} or equal to the highest/lowest acceptable value can be supported, then the preceding side shall follow the crankback procedures specified in Annex B §8.3.1.2,using cause and crankback cause No. 49, "Quality of service unavailable".

If no acceptable forward value of an allowed individual QoS parameter for the corresponding ATM service category is specified (in the Extended QoS Parameters or End-to-end transit delay information elements), then the default is that any value of the individual QoS parameter is acceptable and the preceding side shall continue to process the call.

If the preceding side is able to provide the acceptable values of all specified individual QoS parameters, the preceding side shall progress the call to the succeeding side.

The succeeding side does not make use of the QoS parameter information element, but passes it on if it is present and if the call is progressed.

For each parameter contained in the Extended QoS parameters information element and/or the End-to-end transit delay information element, the succeeding side shall:

1. Increment the backward cumulative value of the parameter, if the parameter is cumulative, to account for the expected increases due to user data transfer within this switching system and over the link from this switching system to the switching system at the preceding side.

2. If the next interface over which the call is to be routed is not a PNNI interface and the parameter is cumulative

   a) the succeeding network node shall increment the cumulative forward value of that parameter to account for the expected increases due to user data transfer within this switching system and over the following link, between this switching system and the network boundary.

   b) In addition, increment the cumulative backward value to account for the expected increases due to user data transfer over the following link, from the network boundary to this switching system.

   Note that the procedures described in this step are not necessarily associated with this PNNI interface; they may also be considered as procedures undertaken at the preceding side of the next (non-PNNI) interface. They are provided here in order to complete specification of the steps required to support the network side procedures in the UNI 4.0 Signalling specification.

   The result of performing the accumulation procedures described above is that the network determines the cumulative forward and backward parameters measured from network boundary to network boundary. Note that this applies even in the case where the path for the call does not traverse any PNNI interfaces at all.

3. Determine if the highest/lowest acceptable values of that parameter can be supported. If no values {less than/greater than} or equal to the highest/lowest acceptable value can be supported, then the succeeding side shall follow the crankback procedures specified in Annex B §8.3, returning a RELEASE or RELEASE COMPLETE message (depending on whether or not a CALL PROCEEDING message has been sent yet) with cause and, if applicable, crankback cause #49, "Quality of service unavailable".

If no acceptable forward value of an allowed individual QoS parameter for the corresponding ATM service category is specified (in the Extended QoS Parameters or End-to-end transit delay information elements), then the default is that any value of the individual QoS parameter is acceptable and the succeeding side shall continue to process the call.

If the succeeding side is able to provide the acceptable values of all specified individual QoS parameters, the succeeding side shall continue to progress the call.

6.5.2.3.6 Traffic parameter selection procedures for ABR connections

In the case of ABR connections, the following additional and modified procedures apply for the processing of traffic parameters.

Parameter defaulting for ABR parameters as defined in the ATM Forum UNI 4.0 signalling specification shall be performed at the network side of the calling user UNI. The ABR setup parameters are all mandatory in the SETUP message. In the case of soft PVPC or PVCC setup, the originating switching system is responsible for sending a SETUP message with all these parameters present.

The negotiation procedures allow each switching system to adjust the parameters of a call as needed to protect the resources in that switching system and the quality of service commitments made to other connections. The resources in question are not necessarily associated with a particular PNNI; they may relate to a UNI, or be common to several interfaces.

Since the algorithms in this chapter describe the PNNI, negotiation is modeled as occurring at the PNNI (at both the preceding and succeeding sides). It should be understood that this is not meant to constrain the switching system. In particular, for the first and last switching system in the call path, the resources related to the calling user UNI and called user UNI, respectively, are considered in the negotiation process.

- Negotiation of parameters in the ATM traffic descriptor and ABR setup parameters information elements:

  Parameter values for a given direction for PCR, ICR, TBE, RIF, and RDF can be negotiated by either side. MCR can be negotiated using the procedures in Section 6.5.2.3.4 if the corresponding MCR parameter is included in the Minimum acceptable ATM traffic descriptor information element in the SETUP message.

  If able to provide the indicated PCR and ABR setup parameter values, the PNNI network shall progress the call towards the called user, with the original parameters.

  If unable to provide the indicated PCR, but able to provide at least the MCR value as negotiated, the PNNI network shall progress the call towards the called user, after adjusting the PCR value. The adjusted PCR value will be greater than or equal to MCR value.

  When progressing the call, the network may, if necessary, also adjust either or both “forward” and “backward” parameters: ABR setup parameters: ABR initial cell rate, ABR transient buffer exposure, Rate increase factor and rate decrease factor.

  The following table summarizes the modifications that may be made:

Parameter for a given direction	**Modification by the network**
PCR	Decrease only
ICR	Decrease only
TBE	Decrease only
RIF	Decrease only
RDF	Increase only

  Parameter negotiation maintains the following invariant:

  MCR <= ICR <= PCR

  If the network is not able to provide the peak cell rates which are equal to MCR, then the crankback procedures in Annex B, §8.3 shall be followed, with cause and crankback cause #37, *“User cell rate not available”.*

- Negotiation of parameters in the ATM additional parameters information element:

  Parameter values for ATM additional parameter can be negotiated by either side, but only when the parameter is present in the SETUP message (i.e., was supplied by the calling user). If the parameter is absent, the default value applies, and no negotiation is possible for the parameter in this case. If the ATM additional parameters information element is not included in the SETUP message, then the default values apply to all the above parameters and none are negotiable. The default values for additional parameters are specified in the Traffic Management specification 4.0.

The ATM Forum Traffic Management specification, 4.0, provides further detail on considerations relating to negotiation. The following table summarizes the modifications that may be made:

Parameter for a given direction	Modification by the network
Nrm	No negotiation (Note)
Trm	No negotiation (Note)
CDF	Increase only
ADTF	Decrease only

Note - If the network does not support the indicated value, it can change to default by changing the indicator present bit.

The parameter is negotiable up to its fixed bound specified for its encoding.

6.5.2.3.7 Processing of Cumulative RM fixed round-trip time parameter for ABR connections

The succeeding side shall adjust the cumulative RM fixed round-trip time parameter in the ABR setup parameters information element. The amount of the adjustment is the sum of the forward and reverse direction fixed portion of the RM cell delay on the PNNI, including the forward and reverse link propagation delays and any fixed processing delays at the PNNI and within the switching system. The adjustment value, expressed in microseconds encoded as an integer, is added to the cumulative RM fixed round-trip time parameter.

In the first and last switching systems of the call path, the network side of the UNI shall adjust the cumulative RM fixed round-trip time parameter in the ABR setup parameters information element. The amount of the adjustment is the sum of the forward and reverse direction fixed portion of the RM cell delay on the UNI, including the forward and reverse link propagation delays upto network boundary and any fixed processing delays at the UNI and within the switching system.

The effect of the above rules is that at each switching system the Cumulative ABR RM Fixed Round Trip Time parameter is adjusted by the round trip time contribution of the previous hop, whether that hop was a UNI or PNNI. The final switching system is an exception: it includes both the previous hop and the next hop (to the called user) in its adjustment.

Note that the Cumulative ABR RM Fixed Round Trip Time parameter is adjusted only in SETUP message processing, not in CONNECT message processing.

6.5.2.4 Call/connection proceeding

The succeeding side shall send a CALL PROCEEDING message to the preceding side to acknowledge the SETUP message and to indicate that the call is being processed; and enter the Call Proceeding Sent state. When the preceding side receives the CALL PROCEEDING message, the preceding side shall stop timer T303, start timer T310, and enter the Call Proceeding Received state.

Note: Sending of the CALL PROCEEDING message is mandatory.

If, following the receipt of a SETUP message, the succeeding side determines that for some reason the call cannot be supported, then the succeeding side shall initiate call clearing as defined in §6.5.3. Some of the causes that may be used are given in §6.5.2.10.

If the preceding side has received a CALL PROCEEDING message, but does not receive a CONNECT, ALERTING, or RELEASE message prior to the expiration of timer T310, then the preceding side shall: initiate clearing procedures (without crankback) towards the originating interface with cause #102, "recovery on timer expiry" and clears towards the called party with cause #102, "recovery on timer expiry".

6.5.2.5 Call/connection alerting

Upon receiving an indication that the called party is alerting, the succeeding side shall send an ALERTING message to the preceding side and enter the Alerting Delivered state.

When the preceding side receives an ALERTING message from the succeeding side, it shall stop timer T310; start T301; enter the Alerting Received state and send an alerting indication towards the calling user.

6.5.2.6 Call/connection connected

Upon receiving an indication from Call Control that the call has been accepted, the succeeding side shall: send a CONNECT message to the preceding side and enter the Active state.

This message indicates to the preceding side that a connection has been established from this interface to the called party.

On receipt of the CONNECT message, the preceding side shall: stop timer T310 or T301 and enter the Active state.

6.5.2.6.1 Procedures for traffic parameter selection during call/connection acceptance

If the received CONNECT message contained an ATM traffic descriptor information element, the succeeding side shall forward the ATM traffic descriptor information element as it received it, with the possible exception of the Tf and Tb fields.

If the SETUP message contained an ATM traffic descriptor information element with the Tf parameter coded to "Tagging requested" then the succeeding exchange shall indicate in the CONNECT message:

- "Tagging applied", if this node performs tagging for this connection, or
- the same tagging value as the incoming connect indication, if this node does not perform tagging for this connection.

If the SETUP message contained an ATM traffic descriptor information element with the Tf parameter coded to "Tagging not allowed" then the succeeding exchange shall indicate in the CONNECT message "Tagging not applied".

If:

- the preceding side sent a SETUP message containing an ATM traffic descriptor information element with the Tf field coded to "Tagging requested", and
- the preceding side receives a CONNECT message containing an ATM traffic descriptor information element with the Tf field indicating "Tagging not applied",

then the resources allocated to the connection for forward traffic may be modified, since forward traffic must conform to CLP=0 traffic parameters as well as CLP=0+1 traffic parameters.

If the SETUP message contained an ATM traffic descriptor information element with the Tb parameter coded to "Tagging supported" then the succeeding exchange shall indicate in the CONNECT message:

- "Tagging not allowed" if this node performs tagging for this connection, or
- the same tagging value as the incoming connect indication, if this node does not perform tagging for this connection.

If:

- the preceding side sent a SETUP message containing an ATM traffic descriptor information element with the Tb field coded to “Tagging supported”, and
- the preceding side receives a CONNECT message containing an ATM traffic descriptor information element with the Tb field indicating “Tagging not allowed”,

then the resources allocated to the connection for backward traffic may be modified, since backward traffic must conform to CLP=0 traffic parameters as well as CLP=0+1 traffic parameters.

When a CONNECT message with an indication of frame discard is received, the PNNI node follows the procedures described in section 2.2.1 of UNI 4.0 signalling specification.

6.5.2.6.2 Procedures for negotiation of traffic parameters during call/connection acceptance

If a peer returns the ATM traffic descriptor information element in the CONNECT message, the network node shall forward the ATM traffic descriptor information element as it received it. If the traffic parameter values in the CONNECT message differ from those that the node has forwarded in the SETUP message, the resources that the node has allocated to the connection shall be modified accordingly.

If a peer returns no ATM traffic descriptor information element in the CONNECT message and the network node has negotiated the traffic parameters when it processed the corresponding SETUP message, the node shall include the ATM traffic descriptor information element in the CONNECT message before forwarding it. The contents of the ATM traffic descriptor shall be the same as in the forwarded SETUP message, with the exception of the Traffic Management Options which shall be coded as specified in Section 6.5.2.6.2.

If a peer returns no ATM traffic descriptor in the CONNECT message and the network node has not negotiated the traffic parameters when it processed the corresponding SETUP message, the node can optionally include the ATM traffic descriptor information element in the CONNECT message. The contents of the ATM traffic descriptor shall be the same as in the forwarded SETUP message, with the exception of the Traffic Management Options which shall be coded as specified in Section 6.5.2.6.2.

6.5.2.7 Failure of call establishment

In addition to the procedures described in Section 6.5.2.3, the following shall apply. If the succeeding side determines that the requested service is not available or is not able to progress the call, the succeeding side shall initiate crankback in accordance with §6.5.3.3 and Annex B, with one of the following cause and crankback cause codes.

#38 network out of order;
#41 temporary failure;
#58 bearer capability not presently available;
#63 service or option not available, unspecified; or,
#65 bearer service not implemented.

This list is not exhaustive.

6.5.2.8 Generic identifier transport procedures

See section 2.2.2 of the UNI 4.0 signalling specification.

6.5.3 Call/connection clearing

In case of crankback, additional procedures to these Call/connection clearing procedures are described in Annex B.

6.5.3.1 Terminology

See section 5.4.1 of ITU-T Recommendation Q.2931.

6.5.3.2 Exception conditions

Under normal conditions, call clearing shall be initiated at the UNI. At the PNNI, call clearing may be initiated by either side of PNNI as a response to a call clearing request initiated at a UNI or due to a failure, administrative action, or other exception condition. The clearing procedures are defined in §6.5.3.3. The exception to the above rule is as follows:

> The succeeding side can reject a call/connection after receiving a SETUP message by: responding with a RELEASE COMPLETE message provided no other response has previously been sent; releasing the call reference, and entering the Null state.

6.5.3.3 Clearing

Apart from the exceptions identified in §6.5.3.2, the clearing procedures are symmetrical and may be initiated by either preceding or succeeding side of the PNNI. In the interest of clarity, the following procedures describe only the case where preceding side initiates clearing.

The preceding side shall initiate clearing by: sending a RELEASE message, starting timer T308; release the virtual channel and entering the Release Request state.

The succeeding side of the PNNI interface shall enter the Release Indication state upon receipt of a RELEASE message. On receipt of this message, the succeeding side shall release the virtual channel, and initiate procedures for clearing the network connection by informing the PNNI Call control entity. Once the virtual channel used for the call has been released, the succeeding side shall: send a RELEASE COMPLETE message to the preceding side; release both the call reference and virtual channel (i.e. Connection identifier); and enter the Null state.

> Note - The RELEASE COMPLETE message has only local significance and does not imply an acknowledgement of end-to-end clearing.

On receipt of the RELEASE COMPLETE message the preceding side shall: stop timer T308; release the virtual channel; release the call reference and return to the Null state.

If timer T308 expires for the first time, the preceding side shall: retransmit a RELEASE message to the succeeding side with the cause number originally contained in the first RELEASE message; restart timer T308 and remain in the Release Request state. In addition, the preceding side may indicate a second Cause information element with cause #102, "recovery on timer expiry". If no RELEASE COMPLETE message is received from the succeeding side before timer T308 expires a second time, the preceding side shall: release the call reference; and return to the Null state. Additional recovery procedures, such as initiating restart, are implementation dependent.

6.5.3.4 Clear collision

Clear collision can occur when both sides simultaneously transfer a RELEASE message related to the same call reference value. If the preceding or succeeding side receives a RELEASE message while in the Release Request state, the receiving entity shall: stop timer T308; release the call reference and virtual channel; and enter the Null state (without sending or receiving a RELEASE COMPLETE message).

6.5.4 Call/connection collisions

Call/connection collision can occur when both sides of the PNNI interface simultaneously transfer SETUP messages indicating traffic parameters which together exceed the remaining resources on the interface. Both sides of the PNNI interface shall clear the call/connection according to the procedures of 6.5.3 with a Crankback information element. The cause and crankback cause used shall be:

#47 resources unavailable, unspecified;
#49 quality of service unavailable; or
#51 user cell rate not available.

6.5.5 Restart Procedures

Section 2 §5.5/Q.2931 of UNI 4.0 signalling specification shall apply with the following changes:

Replace the first paragraph with the following text:

> Both sides of the PNNI interface shall implement these procedures.

6.5.5.1 Sending RESTART

Section 2 §5.5.1/Q.2931 of UNI 4.0 signalling specification shall apply with the following changes:

Replace the first sentence of first paragraph with the following text:

> A RESTART message is sent by either side of PNNI in order to return virtual channels to the idle condition.

6.5.5.2 Receipt RESTART

See section 2 §5.5.2/Q.2931 of UNI 4.0 signalling specification.

6.5.5.3 Restart collision

A restart collision occurs at PNNI when signalling entities on both sides of the interface simultaneously transmit a RESTART message. The call reference flag of the global call reference applies to restart procedures. In the case when both sides of the interface initiate simultaneous restart requests, they shall be handled independently. In the case when the same virtual channel(s) are specified, they shall not be considered free for reuse until all the relevant restart procedures are completed.

6.5.6 Handling of error conditions

Section 5.6 of the Recommendation Q.2931 shall apply with the following changes:

> In the first paragraph, ‘Q.2931 user-network call control message’ is replaced by ‘PNNI signalling message’.

> The references to sections '5.6.1' and '5.6.8', in the first and third paragraphs, are replaced by '6.5.6.1' and '6.5.6.8' respectively.

6.5.6.1 Protocol discrimination error

When a message is received with a protocol discriminator coded other than "PNNI signalling message", that message shall be ignored. "Ignore" means to do nothing, as if the message had never been received.

6.5.6.2 Message too short

See section 5.6.2 of the Recommendation Q.2931.

6.5.6.3 Call reference error

6.5.6.3.1 Invalid call reference format

See section 5.6.3.1 of the Recommendation Q.2931.

6.5.6.3.2 Call reference procedural errors

Section 5.6.3.2 of the Recommendation Q.2931 applies with the following change:

> The references to sections '5.6.12' and '5.6.11' in list items 'f' and 'g' are replaced by '6.5.6.12' and '6.5.6.11' respectively.

6.5.6.4 Message type or message sequence errors

Section 5.6.4 of the Recommendation Q.2931 is replaced by the following:

> The error procedures in this section apply only if the flag in the message compatibility instruction indicator is set to "message instruction field not significant". If it is set to "follow explicit instructions", the procedures in section 6.5.7 take precedence.
>
> Whenever an unexpected message (including messages that are standardized by ITU-T but which are not included in this specification), except RELEASE, RELEASE COMPLETE, or an unrecognized message is received in any state other than Null state, a STATUS message shall be returned with one of the following causes:
>
> > #97 *" message type non-existent or not implemented"*; or,
> > #101 *"message not compatible with call state"*.
>
> However, two exceptions to this procedure exist. The first exception is when the preceding or succeeding side of the PNNI receives an unexpected RELEASE message in response to a SETUP message. In this case no STATUS or STATUS ENQUIRY message is sent. Whenever the preceding or succeeding side receives an unexpected RELEASE message, the receiving entity shall: release the virtual channel and clear the connection; return a RELEASE COMPLETE message to the other side; release the call reference; stop all timers; enter the Null state and inform the PNNI Call Control entity.
>
> The second exception is when the preceding or succeeding side receives an unexpected RELEASE COMPLETE message. In this case, the receiving entity shall: release the virtual channel, clear the

connection; release the call reference; stop all timers; enter the Null state; and inform the PNNI call control entity.

6.5.6.5 Message length error

Section 5.6.5 of the Recommendation Q.2931 applies with the following change:

The reference to section ‘5.6.6’ is replaced by ‘6.5.6.6’.

6.5.6.6 General information element errors

6.5.6.6.1 Information element sequence

Section 5.6.6.1 of the Recommendation Q.2931 applies with the following change:

The references to section ‘4.5.1’ are replaced by ‘6.4.5.1’.

6.5.6.6.2 Duplicated information elements

See Section 5.6.6.2 of the Recommendation Q.2931.

6.5.6.6.3 Coding standard error

Section 5.6.6.3 of the Recommendation Q.2931 applies with the following change:

The reference to section ‘5.6.7.2’ is replaced by ‘6.5.6.7.2’.
The reference to section ‘5.6.8.2’ is replaced by ‘6.5.6.8.2’.

6.5.6.7 Mandatory information element error

6.5.6.7.1 Mandatory information element missing

Section 5.6.7.1 of the Recommendation Q.2931 applies with the following change:

The reference to section ‘5.4’ in the third paragraph is replaced by ‘6.5.3’.

6.5.6.7.2 Mandatory information element content error

Section 5.6.7.2 of the Recommendation Q.2931 applies with the following changes:

1. The reference to section ‘5.7’ in the first paragraph is replaced by ‘6.5.7’.
2. The reference to section ‘5.4’ in the fourth paragraph is replaced by ‘6.5.3’.
3. The note placed at the end of this section is not applicable.

6.5.6.8 Non-mandatory information element errors

The error procedures in this section apply only if the flag (bit 5) in the instruction field is set to “IE instruction field not significant”. If it is set to “follow explicit instructions”, the procedures in section 6.5.7 take precedence.

6.5.6.8.1 Unrecognized information element

Section 5.6.8.1 of the Recommendation Q.2931 shall apply with the following changes:

The first sentence of the third paragraph is replaced by the following:

> If a clearing message contains one or more unrecognized information elements, the following procedures are used:

Add the following at the end of the section:

> When a Broadband-locking shift information element occurs in a message, the Broadband-locking shift information element and all subsequent information elements may be treated as unrecognized information elements. However, when a Cause information element is generated to report errors of these shifted information elements the diagnostic field of the Cause information element (if present) should only include the information element identifier for the Broadband-locking shift information element and of any non-shifted information element in error.
>
> When a Broadband-non-locking shift information element occurs in a message, the Broadband-non-locking shift information element and the subsequent information element may be treated as unrecognized information elements. However, when a Cause information element is generated to report errors of these shifted information elements the diagnostic field of the Cause information element (if present) should only include the information element identifier for the Broadband-non-locking shift information element and of any non-shifted information element in error.

6.5.6.8.2 Non-mandatory information element content error

Section 5.6.8.2 of the Recommendation Q.2931 applies with the following changes:

a) The reference to section ‘3’ in the second paragraph is replaced by ‘6.3’.

6.5.6.8.3 Unexpected recognized information element

Section 5.6.8.3 of the Recommendation Q.2931 applies with the following change:

The reference to section ‘ 5.6.8.1’ is replaced by ‘6.5.6.8.1’.

6.5.6.9 Signalling AAL reset

Section 5.6.9 of the Recommendation Q.2931 is replaced by the following text:

> Whenever indication of a Signalling AAL reset is received from the SAAL layer by means of the AAL-ESTABLISH-INDICATION primitive, the following procedures apply:
>
> a) For calls in the clearing phase (states NN11 and NN12) no action shall be taken.
>
> b) Calls in the establishment phase (states NN1, NN3, NN6 and NN9) shall be maintained. Optionally, the status enquiry procedure may be invoked.
>
> c) Calls in the active state shall be maintained according to the procedures in other parts of section 6.5.

6.5.6.10 Signalling AAL failure

Section 5.6.10 of the Recommendation Q.2931 shall apply with the following changes:

1. The reference to section ‘5.6.11’ in the second item of the second list is replaced by ‘6.5.6.11’.

2. The text of the fourth paragraph is replaced by the following text:

If timer T309 expires prior to Signalling AAL re-establishment, the affected entity shall: release the virtual channel; clear the connection; release the call reference; enter the Null state and inform the PNNI Call Control of the Signalling AAL re-establishment failure.

3. The last paragraph of the section is not applicable.

6.5.6.11 Status enquiry procedure

Section 5.6.11 of the Recommendation Q.2931 shall apply with the following changes:

1. The first paragraph is to be replaced by the following:

   To check the correctness of a call state at the Preceding or Succeeding side of the PNNI, a STATUS ENQUIRY message may be sent requesting the call state. This may, in particular, apply to procedural error conditions described in 6.5.6.9 and 6.5.6.10.

2. The last paragraph of the section is replaced by the following:

   If timer T322 expires, and no STATUS message was received, the STATUS ENQUIRY message may be retransmitted one or more times until a response is received. The number of times the STATUS ENQUIRY message is retransmitted is an implementation dependent value. If the number of retransmission of the STATUS ENQUIRY message exceed the limit, the call shall be cleared. The cause that should be used when clearing the call is cause #41 *"temporary failure"*. The PNNI Call Control entity shall be notified of the failure of the call.

6.5.6.12 Receiving a STATUS message

Section 5.6.12 of the Recommendation Q.2931 shall apply with the following changes:

The following text is added at the end of item 'c)' of the second list:

The PNNI Call Control entity shall be notified of the failure of the call.

6.5.7 Error procedures with explicit action indicator

6.5.7.1 Unexpected or unrecognized message type

Section 5.7.1 of Recommendation Q.2931 shall apply with the following changes:

The reference to section '5.4.3 or 5.4.4' in the second paragraph of section is replaced by '6.5.3'.

Add a new paragraph at the beginning of the section as follows:

The error procedures in this section apply to unrecognized messages only if the pass along request (bit 4) in the instruction field is set to "no pass along request". If it is set to "pass along request" in an unrecognized message, the message shall be passed on without error checking, provided the next interface is a PNNI.

6.5.7.2 Information element error

Section 5.7.2 of Recommendation Q.2931 shall apply with the following changes:

Add a new paragraph at the beginning of the section as follows:

> The error procedures in this section apply to unrecognized information elements only if the pass along request (bit 4) in the instruction field is set to “no pass along request”. If it is set to “pass along request” in an unrecognized information element, the information element shall be passed on in the message without error checking, provided the next interface is a PNNI.

6.5.8 Handling of messages with insufficient information

Section 5.8 of Recommendation Q.2931 shall apply with the following changes:

The reference to sections ‘5.6.7.1’ and ‘5.7.1’ are replaced by ‘6.5.6.7.1’ and ‘6.5.7’ respectively.

6.5.9 Network node call control requirements

Any node that might take part in alternate routing must keep copies of the originally received SETUP message contents, including QoS and other parameters used to determine routing and call admission control, and also of any DTLs that it generated. Copies must also be kept of any blocked node or blocked link subfield contents received by the node during crankback procedures. The copies of the original SETUP message contents, additional DTLs, and possibly blocked transits should be kept until a CONNECT or ALERTING message is received from the following node, or until the call or party is cleared.

6.5.10 Notification procedures

See section 5.9 of the Recommendation Q.2931.

6.5.11 Notification of interworking

If the Progress indicator information element is included in a call control message, the procedures of 6.5 apply. If the Progress indicator information element is included in the PROGRESS message, no state change will occur but any supervisory timers except T301 and T322 shall be stopped if the progress description is No.1, No. 2, or optionally No. 4.

6.5.12 List of Timers

The description of the timers for basic call will follow the timers defined in Table 7-1/Q.2931 of Recommendation Q.2931. The values in this table shall apply with the exception of the value of T310 which shall have the value of 30-120 seconds. The precise details are found in section 6.5. Retransmission of SETUP messages is optional.

6.6 Call/connection Control Procedures for Point-to-Multipoint Calls

This section describes procedures for point-to-multipoint calls and uses ITU-T Recommendation Q.2971. It supports point-to-multipoint calls where user plane information is multicasted unidirectionally from one calling user to a set of called users. The network node will indicate the arrival of an add party request at the PNNI interface by transferring a SETUP (see section 6.6.1) or ADD PARTY (see section

6.6.2) message across the interface. The procedures for point-to-multipoint are based on ITU-T Q.2971 section 10.

6.6.1 Setup of the initial party

The procedures of the section 10.2.1 of ITU-T Recommendation Q.2971 shall apply with the following changes:

1. Since the procedures apply between preceding and succeeding sides of the PNNI interface, replace the following items throughout the text:
    - "user-network interface" to "PNNI interface".
    - "network" to "succeeding network node".
    - "user" to "preceding network node".
2. Bullet two is not applicable. The bullet is shown below:

    If a CALL PROCEEDING, ALERTING, or CONNECT message ....the procedures of subclause 9.2 (instead of the procedures of subclause 10.2.2).
3. Changes to section 9.2 of Q.2971
    a) The first paragraph is to be replaced by the following:
    The preceding and succeeding sides of the PNNI interface shall follow the procedures of section 6.5 with the following additions.
    b) In the second paragraph delete sentence
    The instruction indicator in the Endpoint reference information element shall be coded to "discard information element and proceed".
    c) Paragraph four is not applicable. The paragraph is shown below:
    If a CALL PROCEEDING, ALERTING, or CONNECT message is the first message .... the procedures of subclause 9.2.1 shall be followed.
    d) Last two paragraphs of section 9.2 are not applicable. The paragraphs are shown below:
    At the terminating interface, the ADD PARTY, ... and DROP PARTY messages shall not be used.
    At the terminating interface, the receipt of an ADD PARTY, ...message shall be treated as an unrecognized or unexpected message.
    e) Section 9.2.1 is not applicable.

6.6.2 Adding a party

The procedures of the section 10.2.2 of ITU-T Recommendation Q.2971 shall apply with the following changes:

1. Since the procedures apply between preceding and succeeding sides of the PNNI interface, replace the following items throughout the text:
    - "user-network interface" to "PNNI interface".
    - "network" to "preceding side".
    - "user" to "succeeding network node".
2. For sections 10.2.2.1 through 10.2.2.5, when an ADD PARTY REJECT message is sent it shall include a Crankback information element with the crankback cause set to the same value as the specified cause value.
3. In Section 10.2.2.5, add the following paragraph:

    If the received ADD PARTY REJECT message contains a Crankback information element, the procedures for crankback processing contained in Annex B apply.

Note that when an add party request is in the add party queue, it has no party state.

6.6.3 Party dropping

The procedures of the section 10.3 of ITU-T Recommendation Q.2971 shall apply with the following changes:

1. Section 10.3.2 list item a):

   Change “the call clearing procedures of subclause 5.4.2/Q.2931 shall apply” to “the call clearing procedures of section 6.5.3.3 shall apply”.
2. Change “clearing procedures of subclause 5.4/Q.2931” to “clearing procedures of section 6.5.3” in sections 10.3.3, 10.3.4, 10.3.5 and 10.3.6.
3. In Section 10.3.3, after the fourth bullet in the paragraph starting with “When the network receives a RELEASE message” add another bullet item:
   - If the received RELEASE message contains a Crankback information element, the procedures for crankback processing contained in Annex B apply.

6.6.4 Restart procedures

The procedures of the section 10.4 of ITU-T Recommendation Q.2971 shall apply.

6.6.5 Handling of error conditions

The procedures of the section 10.5 of ITU-T Recommendation Q.2971 shall apply with the following changes:

1. Change “clearing procedures of subclause 5.4/Q.2931” to “clearing procedures of section 6.5.3” in section 9.5.7.2.

6.6.6 List of timers for point-to-multipoint

The list of timers specified in section 13.2 of Q.2971 shall apply except for the default value of T399 which shall be in the range of 34-124 seconds.

7. Annex A: Designated Transit List

This Annex describes Designated transit lists. Designated Transit Lists (DTLs) indicate the transits (i.e., logical nodes and possibly logical links) that a connection is to traverse through a peer group at some level of hierarchy. The hierarchical nature of the representation of network topology leads to the representation of a hierarchically complete source route as a stack (last-in, first out list) of DTLs. The stack of DTLs is represented in the SETUP and ADD PARTY messages as a sequence of Designated transit list information elements.

Pseudocode for the procedures specified in this section are contained in Section 7.5. When there is an ambiguity in the narrative text, the pseudocode should be used to resolve the conflict. Where the text and the pseudocode are in disagreement, the text should be used as the prime source.

The DTL processing procedures carried out at each node along the path of a call are composed of two parts: First, the node determines whether any DTLs need to be added to the DTL stack and takes appropriate measures; second, the node determines whether any DTLs need to be removed from the stack and processes the remaining DTL stack to prepare it for transmission to the next node. Typically, a node either adds to the DTL stack or removes from the DTL stack. If a node both adds and removes DTLs from the stack, it must first remove the DTLs it added.

Each PNNI node along the path of a call can be categorized based on its role in DTL processing. The categories of a node in terms of generating DTLs are:

a) DTL originator (see Section 7.2.1)
b) Entry border node (see Section 7.2.2)
c) Neither DTL originator nor entry border node (see Section 7.2.3)

The categories of a node in terms of removing DTLs are:

d) DTL terminator (see Section 7.3)
e) Exit border node (see Section 7.3)
f) Neither DTL terminator nor exit border node (see Section 7.3)

Note that the DTL processing procedures for the latter three categories are described as one common set of procedures.

7.1 Terminology

Call entering a peer group: A call is understood to be entering a peer group at a level of the hierarchy if it has been progressed to a border node to that peer group, across a logical link which is connected to another peer group.

Call exiting a peer group: A call is understood to be exiting a peer group at a level of the hierarchy if it is to be progressed by a border node to that peer group, across a logical link which is connected to another peer group.

7.2 Initial DTL processing

7.2.1 Procedures at the DTL originator

The DTL originator shall calculate a path across the routing domain towards the called party, as indicated in the SETUP or ADD PARTY message for the call/connection. If a Transit network selection information element is present, the transit network and optionally called party number shall be used to determine the path. The node at the end of the selected path must have connectivity to the called party number or transit network with advertised membership scope (see Section 5.8.1.3) at least as high as the path scope (defined as the highest level of PNNI hierarchy used by the path). In addition, if a Connection scope selection information element is present in the SETUP message, then the path scope must lie within the connection scope of the call (i.e., connection scope <= level indicator of any node in the path >= scope of advertisement of reachable address or transit network).

When the called party number is a group address (i.e., this call uses the ATM Anycast capability), the target of the path selection must be a member of the ATM group identified by the called party number. In this case, if no Connection scope selection information element is present in the SETUP message, a default connection scope of “localNetwork” shall be used for path selection purposes, (it is not included in the SETUP message)

The path is represented by a stack of DTLs. If no suitable path is found, the connection shall be cleared with the appropriate cause. If a Transit network selection information element is present, then cause #2 “no route to specified transit network” shall be used. Otherwise, cause #3 “no route to destination” shall be used.

The information about the path comprises all logical nodes (and optionally logical links) along the path which are known to the originating node, including:

1. the complete list of logical nodes (and optionally logical links) necessary to traverse the originating node’s own peer group;
2. a list of logical nodes (and optionally logical links) which determines a path at higher levels of the hierarchy, ending with a logical node containing the DTL terminator.

The DTL originator shall append to the SETUP or ADD PARTY message a Broadband repeat indicator information element coded to indicate Last-in, First Out Stack, followed by one or more Designated transit list information elements. Designated transit list information elements shall be pushed onto the stack, thus, they appear in the reverse order in which they are to be traversed, i.e., the first Designated transit list information element to appear shall be that which includes the logical node that contains the DTL terminator, and the last Designated transit list information element to appear shall be that which contains the DTL originator.

Each Designated transit list information element shall contain, in the order which they are to be traversed, a list of transits at a single level of the hierarchy. This list shall start with the first logical node to be traversed at that level, and end at the last logical node or logical link to be traversed at that level. Whenever the path changes from one level of hierarchy to another, a Designated transit list information element shall be pushed onto the stack. The current transit pointer shall be set to 0, indicating the first logical node (and optionally logical link) in the designated transit list.

The top Designated transit list information element in the stack shall contain, in the order which they are to be traversed, a list of all of the logical nodes (and optionally logical links) within the DTL originator’s own lowest level peer group, starting with the DTL originator, followed by the nearest neighbor logical node in the list and ending with either the DTL terminator (if in the DTL originator’s own peer group) or the peer group exit border logical node (if the DTL terminator is not in the DTL originator’s own peer group). The transit pointer shall be coded as 0, indicating the originating node.

7.2.2 Procedures at an entry border node

When a call enters a peer group, the border node shall examine the top designated transit list information element in the SETUP or ADD PARTY message. If the logical node identifier indicated by the current transit pointer is not the same as the node ID of the border node’s ancestor at the level of the common peer group for the receiving link, then the call shall be cleared with cause #41 "temporary failure" and optionally cranked back with either (i) a succeeding end blocked indicator and crankback cause #128 "next node unreachable", or (ii) crankback cause #160 "DTL transit not my node ID" and the DTL transit listed as the blocked node. If the logical port identifier is not set to 0, it indicates the logical port to be used to progress the call when it exits the peer group.

The border node shall calculate a route across the peer group as follows:

a) It shall examine the designated transit list at the top of the stack. If the current transit pointer indicates the end of the designated transit list, it shall perform the procedures of A.2.2 (b). Otherwise, it shall proceed with A.2.2 (c), using, as the target of the path calculation, the transit immediately following the transit indicated by the current transit pointer .
b) Upon reaching the end of a designated transit list, it shall examine the next lower designated transit list on the stack, if there is one. If the logical node identifier indicated by the current transit pointer of the next designated transit list is not a node ID of one of the border node's ancestors, then the call shall be cleared with cause #41 "temporary failure" and optionally cranked back with crankback cause #160 "DTL transit not my node ID". If the logical port identifier is not set to 0 and no logical port has been specified yet, then the port identifier indicates the logical port to be used to progress the call when it exits the peer group. If the logical port identifier is not set to 0 and does not contain all logical ports found during previous recursions of these procedures (i.e., specified in DTLs at lower levels of the hierarchy), such that there is conflicting information as to which port to use to progress the call, then the call shall be cleared with cause #41 "temporary failure".

   - If the current transit pointer indicates the end of that designated transit list, and there is a designated transit list lower on the stack, it shall recursively repeat the procedures of A.2.2 (b).
   - If the current transit pointer does not indicate the end of the designated transit list, it shall proceed with A.2.2 (c), using the transit immediately following the transit indicated by the current transit pointer as the target of the path calculation.
   - If there is not a designated transit list lower on the stack, it shall save the level of this DTL (at the bottom of the DTL stack) as the path scope before determining whether the node is the DTL terminator; in this case, it shall follow procedures as described in Section 7.3.
   - Otherwise, it shall proceed with A.2.2 (c), using as target of the path selection:
     i) If a Transit Network Selection information element is present, the designated transit network and optionally the called party number.
     ii) If no Transit Network Selection information element is present, the called party number.

     In this case, any port ID indicating a logical port to be used to progress the call shall be ignored.
c) A path shall be calculated from the entry border node at the lowest level in which the entry border node appears towards the target. The node at the end of the selected path must have connectivity to the target with advertised membership scope (see Section 5.8.1.3) at least as high as the path scope (i.e., path scope >= scope of advertisement of reachable address or transit network). When the target is a called party number that is a group address (for ATM anycast calls), the path must lead to a member of the ATM group with a sufficiently high membership scope.
   DTLs describing the selected path shall be pushed onto the DTL stack. These DTLs are pushed onto the stack in the reverse order in which they are to be traversed. Each DTL shall contain, in the order in which they are to be traversed, a list of transits at a single level of the hierarchy. The current transit pointers in these DTLs shall be set to 0, indicating the first transit. If no suitable path exists to the target of the path selection, then the call shall be cleared or cranked back with the appropriate cause, as determined according to the procedures of Section 8.2.1.

Note: No current transit pointer shall be incremented, and no designated transit list shall be popped from the stack in the procedures of Section 7.2.2.

Follow the procedures of Section 7.3, unless the call/connection has been cleared or cranked back.

7.2.3 Procedures at a node which is not the DTL originator or an entry border node

The top designated transit list information element in the SETUP or ADD PARTY message shall be examined. If the Node identifier indicated by the current transit pointer is not the same as that node's own node identifier, then the call shall be cleared with cause #41 "temporary failure" and optionally cranked back with either (i) a succeeding end blocked indicator and crankback cause #128 "next node unreachable", or (ii) crankback cause #160 "DTL transit not my node ID" and the DTL transit listed as the blocked node.

Follow the procedures of Section 7.3, unless the call/connection has been cleared or cranked back.

7.3 Next step in DTL processing

If the current transit pointer in the top designated transit list information element on the DTL stack (after the procedures of Section 7.2 have been completed) does not point to the last transit in the DTL information element, the current transit pointer shall be advanced. If it:

a) Indicates a transit which is a neighbor at the same level of hierarchy, the connection shall be progressed to that transit, using the specified logical port, if any. However, if the specified logical port does not correspond to a logical link to the next transit, then the call shall be cranked back with crankback cause #128 "next node unreachable" and cause #2 "no route to specified transit network" or cause #3 "no route to destination".
b) Indicates a transit which is not a neighbor, the connection shall be cranked back with crankback cause #128 "next node unreachable" and cause #2 "no route to specified transit network or cause #3 "no route to destination".

When the call is progressed, the SETUP or ADD PARTY message shall contain the stack of designated transit lists, as modified.

If the logical port identifier is not set to 0, it indicates a specific logical port to be used to progress the call.

If the current transit pointer indicates the end of the DTL, the DTL shall be popped from the stack, saving the level of the DTL as the path scope if this is the last DTL on the stack. If there are any DTLs remaining on the stack, the node shall examine the next designated transit list. If the logical node identifier indicated by the current transit pointer of the next designated transit list is not a node ID of one of the border node's ancestors, then the call shall be cleared with cause #41 "temporary failure" and optionally cranked back with crankback cause #160 "DTL transit not my node ID". If the logical port identifier is not set to 0 and does not contain all logical ports found during previous recursions of these procedures (i.e., specified in DTLs at lower levels of the hierarchy), such that there is conflicting information as to which port to use to progress the call, then the call shall be cleared with cause #41 "temporary failure". If the logical port identifier is not set to 0 and no logical port has been specified yet, then the port identifier indicates the logical port to be used to progress the call. The first specific logical port which is found during these procedures (i.e., the one at the lowest level of hierarchy of those specified) is to be used to progress the call.

The procedures of Section 7.3 shall then be recursively repeated.

If the DTL stack becomes empty, then this node is the DTL Terminator. With the exception of the port ID specified by the current transit pointer in the top DTL of the originally received DTL stack, any port ID indicating a logical port to be used to progress the call shall be ignored. If a Transit Network Selection information element is present and the indicated transit network is served by this node, then the node shall progress the call to the transit network using UNI procedures. If there is no Transit Network Selection information element and the called party is served by this node, then the node shall progress the call to the called party using UNI procedures. The connectivity from the DTL terminator to the transit network or called party number must have advertised membership scope higher than or equal to the path scope (i.e., scope of advertisement <= path scope). When the called party number is a group address (i.e., this call uses the ATM anycast capability), the call may only be delivered to a member of the ATM group with a scope higher than or equal to the path scope.

If the DTL stack becomes empty but there is no connectivity with sufficient membership scope to the called party or transit network, then the call must be cleared or cranked back according to the procedures of Section 8.2.1.

7.4 Recommendations on Inclusion of Port IDs in DTLs

Specification of port IDs in DTLs is optional. In particular, port IDs may be set to zero to indicate that any port to the next node in the DTL stack can be used.

It is recommended, but not required, that port IDs be left unspecified whenever the next node in a DTL stack is represented in the topology database using the simple node representation or the default node representation. This allows for an entry border node in a logical group node to choose the exit border node leading outside of the logical group node, using knowledge of the topology of the child peer group represented by the logical group node. This will often result in better selection of the exit border node than can be achieved at the originating node, which has no knowledge of the internal topology of the child peer group. However, when other issues arise such as differentiation in policy between links to the next node, this recommendation need not be considered.
When the next node in a DTL stack is represented using a complex node representation with one or more exceptions, it is recommended that the port ID be specified. In this case, the originating node takes advantage of its knowledge of the complex topology to cross the next node when computing a path. The originating node can choose links leading into ports that are appropriate for traversing the complex node. This is in contrast to the route computation carried out at entry border nodes, which normally do not consider the implications of routes that they select beyond the entry port into the target node.

7.5 Pseudocode of DTL procedures

The operations required to process the received stack of DTLs and to generate the new stack of DTLs are as follows:

1) receive a PNNI SETUP or ADD PARTY message.
2) extract a DTL stack from the SETUP or ADD PARTY message.
3) does the current transit in the topmost DTL on the stack correspond to me at any level of the routing hierarchy?
    a. yes: set the current node and current port (local variables) to the current transit in the topmost DTL on the stack.
    b. no: abort, an error has occurred.
4) does the current node (from 3) correspond to me at the lowest possible level of the routing hierarchy?
    a. yes: goto 6.
    b. no: is the current transit pointer referencing the last element in the DTL?
        b.1 yes: Are there any other DTLs lower down on the stack?
            b.1.1 yes: check the next DTL down on the stack.
                (Note: do *not* pop any DTLs off the stack at this stage.)
                Does the current transit in the DTL correspond to me at any level of the routing hierarchy?
                b.1.1.1 yes: is the current port set to zero (unspecified)?
                    b.1.1.1.1 yes: reset the current port to that of the current transit in this DTL. goto b.

b.1.1.1.2 no: is the current port included within the current transit's port in this DTL?

b.1.1.1.2.1 yes: goto b.

b.1.1.1.2.2 no: abort, error has occurred.

b.1.1.2 no: abort, an error has occurred.

b.1.2 no: you have arrived at a logical group node that has reachability to the called party number or specified transit network. set the path scope (a local variable) to the level of this DTL (at the bottom of the DTL stack), and set the target (a local variable) to the called party number and/or specified transit network in the SETUP or ADD PARTY message.

b.2 no: set the target (a local variable) to the next entry in the DTL. (note: do *not* advance the current transit pointer at this stage.)

5) determine a route across the current node (from 3) and through the current port (from 3 or 4) to the target (from 4). When the target is the called party number and/or specified transit network, the path must lead to a node with connectivity to the target of scope higher than or equal to the path scope. Convert the resulting path into a set of DTL information elements and push the resulting DTL information elements onto the current DTL stack. **

6) set the output port (a local variable) to that of the current transit in the topmost DTL on the stack. begin step 7 using the topmost DTL on the stack.

7) is the current transit pointer referencing the last element in the DTL?

a. yes: is this the last DTL on the stack?

a.1 yes: you have arrived at a node that has direct reachability to the called party number or specified transit network. Set the path scope (a local variable) to the level of this DTL, then pop the DTL off the stack. The connectivity to the called party number or specified transit network must have scope higher than or equal to the path scope. Send a UNI SETUP message directly to the called party or specified transit network.

a.2 no: pop the DTL off the stack. does the current transit in the next DTL on the stack correspond to me at any level of the routing hierarchy?

a.2.1 yes: is the output port set to zero (unspecified)?

a.2.1.1 yes: reset the output port to that of the current transit in this DTL. goto 7.

a.2.1.2 no: is the output port included within the current transit's port in this DTL?

a.2.1.2.1 yes: goto 7.

a.2.1.2.2 No: abort, an error has occurred.

a.2.2 no: abort, an error has occurred.

b. no: continue.

8) advance the current transit pointer in the topmost DTL on the stack.

9) put the resulting DTL stack back into the SETUP or ADD PARTY message.

10) send the PNNI SETUP or ADD PARTY message directly to the current transit listed in the topmost DTL on the stack, through the output port.

** The way to specify a path and convert the resulting path to a DTL stack can be described in many ways. These possible descriptions vary depending on where to draw the line between the routing algorithm and the specification of DTLs.

To summarize the algorithm, the main steps are as follows:

- determine current node (including current level of hierarchy) (step 3)
- determine target of path selection (step 4)
- determine route and push path onto stack of DTLs (step 5)
- determine output port (step 6)
- pop completed DTLs from stack (step 7)
- advance current transit pointer (step 8)

7.6 Exterior Routes and DTLs

Exterior routes are by definition routes which leave the PNNI routing domain. In many or most cases the organizations which are managing the exterior networks will not want to expose any internal topology within their network. Thus even if a DTL were used, it would provide little details of the exterior route. There is no guarantee that all PNNI routing domains worldwide will use a consistent definition of levels. Thus, if a DTL were used for an exterior route, it might not be possible for the DTL to make use of consistent level indicators (the first octet of each node ID) along the path.

Therefore, where a path makes use of an exterior route, the DTL will terminate at the node (or node and specified port) which is advertising the exterior route into the PNNI Routing Domain.

8. Annex B: Crankback Procedures

This annex describes the additional clearing procedures and behavior of PNNI nodes during crankback. Crankback is indicated by including a Crankback information element in the first call clearing message (RELEASE, RELEASE COMPLETE, or ADD PARTY REJECT). These procedures are additional procedures to the clearing procedures described in Section 6.5.3.

8.1 Scope of Crankback Procedures

A call that cannot be progressed within a PNNI domain may be subject to crankback. This specification indicates those cases where crankback within a PNNI domain is required. Whenever the specification indicates call clearing with specific cause codes, crankback shall not be performed unless stated explicitly.

Any call that progresses all the way to the called user and gets rejected by the called user will not be cranked back. This includes rejections by the called user that happen when a problem is discovered at the called user's end of the UNI, as well as rejections from within the endpoint or user system. In such cases a RELEASE or RELEASE COMPLETE message with a Cause information element will be returned. This will result in the call being cleared all the way back to the calling user. These are not crankback situations, and no alternate routing is to be attempted in these cases.

Calls that get rejected when the DTL Terminator determines that the UNI to the called user cannot carry the call may be cranked back, similarly to cases where call rejection occurs at PNNI interfaces, but in these cases crankback is not required.

The cases where crankback is used in PNNI fall into three categories:
- Reachability errors
- Resource errors
- DTL processing errors

The DTL processing errors can be considered as a subset of reachability errors, as they are usually the result of changes in connectivity.

In addition to these categories, it is possible for a call to be cranked back because the path selection at some node determines that there are no paths that meet the policy constraints. Specification of policy constraints in general has been deferred to later releases of PNNI, so definition of policy violations and crankback cause codes are not included in this specification.

8.2 Crankback Cause

In case of call clearing, a Cause information element is mandatory in the first call clearing message. In case of crankback, a Crankback information element must be included as well as the Cause information element. The Crankback information element must include a crankback cause code, which is used instead of the Cause information element within the PNNI domain.

Note that the cause code and diagnostics in the Crankback information element may be updated whenever the identity of the blocked node or link is changed. Specifically, this is done whenever cranking back beyond an entry border node that had originally specified a DTL including a blocked node or link.

8.2.1 Reachability Errors

Generally, reachability errors indicate that a path to a destination could not be found. This applies to both intermediate destinations in the DTL(s) in the SETUP/ADD PARTY message, and to the final destination (transit network or called party). This type of failure indicates that no path within the scope of the received DTL stack exists to the destination. It is different from finding a path that exists but does not satisfy the requested QoS; such failures are discussed in Section 8.2.2 as Resource Errors.

Whenever crankback occurs due to reachability errors, the blocked transit specified in the Crankback information element must be a blocked link. If the next node, transit network, or called party address cannot be reached from this node through any logical link, then the Blocked link’s port ID shall be set to zero. If the next node, transit network, or called party address cannot be reached through a port specified in the DTL stack, then the Blocked link’s port ID shall be set to that port ID. For unreachable transit networks and called party addresses, the Blocked link’s succeeding node ID shall be set to all zeros.

8.2.1.1 Destination unreachable and transit network unreachable

These cause codes are returned when a node receives a SETUP/ADD PARTY message with a DTL stack indicating that it is the last node in the path, but from which there is no connectivity of scope at least as high as the path scope to the called party or transit network. In the case where no reachability information whatsoever exists in the node's link state database for the called party or transit network, the call shall be cleared with cause #2 "no route to specified transit network" or cause #3 "no route to destination". Similarly, if the only best match address prefixes are summary addresses advertised by this node or one of its ancestors, the call shall be cleared with cause #2 "no route to specified transit network" or cause #3 "no route to destination".

If the node's link state database includes reachability information for the called party or transit network, but only at one or more different nodes (that are not ancestors of this node), the call shall be cranked back with cause and crankback cause #2 "no route to specified transit network" or cause and crankback cause #3 "destination unreachable". This case may occur when a call enters a certain partition of a peer group that has become partitioned. This crankback cause code indicates that the node doing the crankback believes that although the destination is not within its partition of the peer group, the destination is possibly still reachable via some other partition.

In the case of partitioned peer groups it is possible that more than one logical node will be advertising reachability to a given destination. The advertisement in question may be a summary which is being advertised by two logical nodes. Any one specific destination within the summary will only be reachable via one of the partitions, otherwise the partition would not exist. When crankback cause #3 "destination unreachable" is specified, the node that generated the DTL which resulted in the SETUP or ADD PARTY message being sent to the wrong partition shall either a) also be able to see the other partition(s) or b) not be able to see the other partition(s). If a), then alternate routing can be attempted for the call; If b) the call can be cranked back further or cleared.

A node only attempts alternate routing within the scope of the DTL it receives. If all possible alternate paths or partitions have been tried, the call must be cleared or cranked back further, as determined by applying these procedures iteratively.

8.2.1.2 Next node unreachable

This cause code is returned when the next transit in the DTL stack is not directly reachable from this node. Whenever this case occurs, the call shall be cranked back with crankback cause #128 "next node unreachable". The cause code used in the Cause information element shall be set to cause #2 "no route to specified transit network", if a Transit network selection information element is present, or cause #3 "no route to destination", otherwise.

8.2.2 Resource Errors

8.2.2.1 Resource Errors Due To Service Category

Crankback can occur due to unsupported service category or bearer class. See Sections 6.5.2.3.1 and 6.5.2.7 for further details.

8.2.2.2 Resource Errors Due to Traffice and QoS Parameters

The resources needed to support a call are calculated from the traffic parameters and/or QoS parameters included in the SETUP or ADD PARTY message. Resource Errors are used to signal that a path could not be found for the call to satisfy the requested traffic and QoS parameters.

Inability to satisfy the requested traffic and QoS parameters can either be detected during GCAC/path selection (see Section 5.13), or during actual CAC (see Section 6.5.2.3). Calls that are rejected in PNNI domains due to insufficient resources are always cranked back.

If the requested user cell rate(s) from the ATM traffic descriptor information element cannot be satisfied, the call will be cranked back with crankback cause #37, "user cell rate not available". Updated value(s) of the relevant generic CAC parameters of the blocked node/link may also be included (see Section 8.4).

If no path can be found to satisfy the requested maximum CTD, peak-to-peak CDV, and/or CLR (in one and/or the other direction for a call), the call will be cranked back with cause and crankback cause #49, "QoS unavailable". The specific QoS parameter(s) that caused the call rejection are indicated in the diagnostics by setting the appropriate bits to "CTD unavailable", "CDV unavailable", and/or "CLR unavailable", respectively.

When blocking due to insufficient resources occurs, it must be determined whether blocking was due to insufficient resources at the succeeding end of the previous link (calls requesting similar resource requirements on other ports might be accepted), insufficient resources at the preceding end of the following link (calls requesting similar resource requirements on other ports might be accepted), or insufficient resources within the node itself (all calls requesting similar resource requirement from this node are likely to be blocked). Depending on the answer, the procedures in Section 8.3.1.3, 8.3.1.2, or 8.3.1.1, respectively, shall apply.

8.2.2.3 Resource Errors Due to VPCI/VCI Allocation

When the preceding side is unable to allocate a VPCI (for SVPs) or a VPCI/VCI pair (for SVCs), the procedures of Section 8.3.1.2 shall be followed. If no alternate routing is attempted or if alternate routing fails, crankback cause #45 “No VPCI/VCI available” shall be used.

Resource errors due to VPCI/VCI allocation at the succeeding side result in crankback with cause #35 “Requested VPCI/VCI not available” or cause #45 “No VPCI/VCI available”, as discussed in Sections 6.5.2.2.1 and 6.5.2.2.2. Whenever VPCI/VCI resource errors occur at the succeeding side of a PNNI interface, the blocked transit type in the Crankback information element must be set to “call or party has been blocked at the succeeding end of this interface”.

8.2.3 DTL Processing Errors

When a node receives a Designated transit list information element in which the logical node identifier indicated by the current transit pointer does not identify the node or one of its ancestor nodes, crankback may occur. These types of DTL processing errors may occur temporarily due to changes in connectivity, in particular when a change in peer group leader occurs and a new parent logical group node is created. The Crankback information element generated in this case will include either (i) a succeeding end blocked indicator and crankback cause #128 “next node unreachable”, or (ii) crankback cause #160 “DTL transit not my node ID” and the DTL transit listed as the blocked node, as discussed in Annex A.

8.3 Procedures for Crankback Level and Blocked Transit

This section describes the procedures used for generating, interpreting, and modifying the crankback level, the blocked transit type, and the blocked transit identifier. These procedures provide the mechanisms required to manage crankback from the point of blocking, through intermediate nodes, to one or more nodes that are allowed to choose alternate routes for the call.

The crankback level and blocked transit type are used during crankback to determine which nodes are allowed to choose alternate routes or alternate links, respectively. In order to indicate the node(s) or link(s) at which problems occurred, the Crankback information element includes a blocked transit identifier subfield in addition to the blocked transit type. The blocked node or link must be avoided in all alternate routes attempted for this call.

Pseudocode for the procedures specified in this section is contained in Section 8.3.3. When there is an ambiguity in the narrative text, the pseudocode should be used to resolve the conflict. Where the text and the pseudocode are in disagreement, the text should be used as the prime source.

8.3.1 Procedures at the Point of Blocking

The procedures carried out at the point of blocking vary depending on whether crankback occurs due to problems at the input port, at the output port, or within the node itself. These procedures are discussed in Sections 8.3.1.3, 8.3.1.2, and 8.3.1.1, respectively.

8.3.1.1 Blocking at a node

Upon blocking at a node, crankback procedures are initiated by sending an appropriate call/connection clearing message (RELEASE, RELEASE COMPLETE, or ADD PARTY REJECT) including a Crankback information element. The Crankback information element must include a crankback level subfield whose value is set to the level of the first node ID indicated in the top DTL of the stack. The blocked transit type must be set to “blocked node” and the blocked transit identifier must indicate the node’s own node ID at the corresponding level of hierarchy, as indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message. The Crankback information element also contains the crankback cause subfield.

8.3.1.2 Blocking at the preceding end of a link

Link blocking can be determined at the preceding end of a link when connection admission control (CAC) within the node at the preceding end of the link determines that insufficient resources are available. If the node at the preceding end of the link is an entry border node for the call, then the procedures of Section 8.3.2.2 shall apply. Otherwise, if other links exist that still satisfy the DTLs in the SETUP or ADD PARTY message received by this node, then alternate routing may be attempted.

If no alternate routing is attempted or if alternate routing fails, then the node at the preceding end of the link must crankback the call or party by sending an appropriate clearing message (RELEASE, RELEASE COMPLETE, or ADD PARTY REJECT) including a Crankback information element. The Crankback information element must include a crankback level subfield whose value is set to the level of the first node ID indicated in the top DTL of the stack. The blocked transit type subfield must be set to "blocked link", and the blocked transit identifier must indicate the identity of the blocked link. The Blocked link's preceding node identifier and port identifier are set to the node and port IDs indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message. The Blocked link's succeeding node ID is set to the next node ID in the top DTL on the stack, if the current transit is not the last node in the DTL, or to the node ID of the next node determined from the received DTL stack (see Section 7.3), if this node is an exit border node for the call. However, if this node is the DTL Terminator, then the blocked link's succeeding node ID is set to all zeros.

8.3.1.3 Blocking at the succeeding end of a link

Link blocking can be determined at the succeeding end of a link when CAC within the node at the succeeding end of the link determines that insufficient resources are available. In this case, the node must crankback the call or party by sending an appropriate clearing message (RELEASE, RELEASE COMPLETE, or ADD PARTY REJECT) including a Crankback information element. The Crankback information element must include a crankback level subfield whose value is set to the level of the first node ID indicated in the top DTL of the stack. The blocked transit type must be set to "call or party has been blocked at the succeeding end of this interface". The Crankback information element also contains the crankback cause subfield.

8.3.2 Receiving a clearing message with a Crankback information element

Upon receiving a clearing message (RELEASE, RELEASE COMPLETE, or ADD PARTY REJECT) including a Crankback information element, a node first checks whether the blocked transit type indicates that the "call or party has been blocked at the succeeding end of this interface". In this case, the procedures of Section 8.3.2.1 must be followed. Otherwise, if the node generated any DTLs for this call of equal or higher level than the crankback level, then the procedures of Section 8.3.2.2 must be followed. Otherwise, the node must crankback the call or party by sending an appropriate clearing message (RELEASE or ADD PARTY REJECT) including an unchanged Crankback information element over its previous interface (towards the calling party).

8.3.2.1 Receiving a clearing message indicating blocking at this interface

When a clearing message (RELEASE, RELEASE COMPLETE, or ADD PARTY REJECT) is received which includes a Crankback information element with blocked transit type indicating "call or party has been blocked at the succeeding end of this interface", the following procedures are carried out. If the node at the preceding end of the link is an entry border node for the call, the procedures of Section 8.3.2.2 shall apply. Otherwise, if other links exist that still satisfy the DTLs in the SETUP or ADD PARTY message received by this node, then alternate routing may be attempted. In addition, a SETUP message may only be resent on the blocked link with a different VPCI (for SVPs) or VPCI/VCI pair (for SVCs) if crankback cause #35 "Requested VPCI/VCI not available" is present.

If no alternate routing is attempted or if alternate routing fails, then the node must continue to crankback the call or party. The blocked transit type subfield must be changed to indicate "blocked link", and a blocked transit identifier must be inserted into the Crankback information element. The Blocked link's preceding node identifier and port identifier are set to the node and port IDs indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message. The Blocked link's succeeding node ID is set to the next node ID in the top DTL on the stack, if the current transit is not the last node in the DTL, or to the node ID of the next node determined from the received DTL stack (see Section 7.3), if this node is an exit border node for the call. However, if this node is the DTL Terminator, then the blocked link's succeeding node ID is set to all zeros. The Crankback information element must include a crankback level subfield whose value is set to the level of the first node ID indicated in the top DTL of the stack. After the above changes to Crankback information element have been made, the node shall send an appropriate clearing message (RELEASE or ADD PARTY REJECT) over its previous interface (towards the calling party).

8.3.2.2 Receiving a clearing message at the entry border node at the crankback level

The node that generated the DTL at the crankback level has been reached. This node must determine whether to try alternate routing or to continue cranking back the call or party. If crankback continues, then the call shall be progressed according to the procedures of Section 8.3.2.2.2. If alternate routing is attempted, then the call shall be progressed according to the procedures of Section 8.3.2.2.1.

8.3.2.2.1 Alternate routing

If alternate routing is attempted, the routing computation must produce a path consistent with the DTLs in the originally received SETUP or ADD PARTY message, that does not contain any blocked nodes and/or links received in the Crankback information element of any clearing messages (RELEASE, or ADD PARTY REJECT). Except as a result of those procedures normally undertaken by entry border nodes, the SETUP or ADD PARTY message progressed should be similar to the originally received SETUP or ADD PARTY message.

8.3.2.2.2 Crankback to next higher level

If alternate routing is not attempted or if alternate routing fails to find a suitable path across the peer group, then crankback must proceed to the entry border node of the next higher-level peer group. The crankback level must be set to the level of the first node in the top DTL on the stack in the received SETUP or ADD PARTY message. The identity of the blocked node or link in the crankback information element must be changed, to reflect a node or link known to the parent peer group as opposed to a node or link known to the child peer group.

If only nodes and/or links internal to the peer group were returned as blocked nodes or links in the Crankback information element of RELEASE, or ADD PARTY REJECT messages, then the logical node corresponding to the peer group should be listed as blocked. In this case, the blocked node ID should match the node ID indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message.

If at least one of the routing attempts was blocked at a link exiting the peer group, then the logical link exiting the logical group node representing the peer group should be listed as blocked. Specifically, the blocked link's preceding node ID and port ID should be set to the values indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message, and the blocked link's succeeding node ID should be set to the node ID of the target of the path calculation, as determined from the received DTL stack in the procedures of Section 7.2.2. However, if the target of the path calculation is not a node, then the blocked link's succeeding node ID is set to all zeros.

8.3.2.3 Additional procedures for point-to-multipoint calls/connections

The procedures for processing clearing messages with crankback for point-to-multipoint calls/connectons are the same as those for point-to-point calls/connections, except when a RELEASE or RELEASE COMPLETE message is received and there are queued ADD PARTY requests on the ADD PARTY queue.

If the network node does not attempt alternate routing for the party for which the RELEASE or RELEASE COMPLETE message was received, and the blocked transit in the crankback information element is the succeeding end of this interface, then the network node shall send an ADD PARTY REJECT message for each queued ADD PARTY request towards the preceding network node, with the crankback indication according to the previous sub-sections.

If the network node does not attempt alternate routing for the party for which the RELEASE or RELEASE COMPLETE message was received, and the blocked transit in the crankback information element is not the succeeding end of this interface, then the network node shall crank back the party for which the RELEASE or RELEASE COMPLETE message was received, and either:

1. Progress one of the ADD PARTY requests on the ADD PARTY queue by sending a SETUP message, leaving the remaining ADD PARTY requests pending, or
2. Crank back all ADD PARTY requests on the ADD PARTY queue whose DTLs contain the blocked transit, and progress one of the ADD PARTY requests remaining on the ADD PARTY queue (if any) by sending a SETUP message, leaving the remaining ADD PARTY requests pending.

If alternate routing is attempted for the party for which the RELEASE or RELEASE COMPLETE message was received, the node shall determine if and how the queued ADD PARTY requests can be satisfied. If the request can be satisfied by adding it to a branch which is in the Active state, then it shall send an ADD PARTY message to the corresponding succeeding node. If a new branch is required and the branch is in the Null state, it shall send a SETUP message to the corresponding succeeding node. If a new branch is required, and the branch is in the Call Initiated, Outgoing Call Proceeding, or Call Delivered state, then it shall queue the party. More than one branch may be needed to satisfy all the queued ADD PARTY requests. If an ADD PARTY request cannot be satisfied, then the network node shall send an ADD PARTY REJECT message for that ADD PARTY request toward the preceding network node, with the crankback indication according to the previous sub-sections. The manner in which the node determines whether an ADD PARTY request can be satisfied, and how it will do so, is implementation specific.

8.3.3 Pseudocode of Crankback Procedures

The operations required when blocking is detected at a node:

1) Blocking was due to insufficient resources at the:
   a. whole node:
      call or party clearing message is returned including:
      crankback level = level of node ID in top DTL on stack
      blocked transit type = blocked node
      blocked node ID = node ID of current transit in top DTL on stack.
   b. preceding end of the following link:

this node added any DTLs? (was an entry border node for this call?)
b.1 yes: goto 2.b.1.1.
b.2 no: are other links satisfying received DTLs available?
b.2.1 yes: try alternate links.
b.2.2 no: call or party clearing message is returned including:
crankback level = level of node ID in top DTL on stack
blocked transit type = blocked link
blocked link's preceding node ID = node ID of current transit in top DTL on stack
blocked link's port ID = port ID of current transit in top DTL on stack
blocked links succeeding node ID =
(if not an exit border node or DTL Terminator)
next node ID (after current transit) in top DTL on stack
(if exit border node)
next node determined from received DTL stack (in steps 7 and8 of DTL processing pseudocode).
(if DTL Terminator)
all zeros.
c. succeeding end of the previous link:
call or party clearing message is returned including:
crankback level = level of node ID in top DTL on stack
blocked transit type = call or party blocked at succeeding end of interface.

The operations required when receiving a clearing message with a Crankback information element:
2) does blocked transit type = call or party blocked at succeeding end of interface?
a. yes: is crankback cause "requested VPCI/VCI not available"?
a.1 yes: retry SETUP on same link using different VPCI/VCI values
a.2 no: this node added any DTLs? (was an entry border node for this call?)
a.2.1 yes: goto 2.b.1
a.2.2 no: are other links satisfying received DTLs available?
a.2.2.1 yes: try alternate links.
a.2.2.2 no:
call or party clearing message is returned including:
crankback level = level of node ID in top DTL on stack
blocked transit type = blocked link
blocked link's preceding node ID = node ID of current transit in top DTL on stack
blocked link's port ID = port ID of current transit in top DTL on stack
blocked links succeeding node ID =
(if not an exit border node or DTL Terminator)
next node ID (after current transit) in top DTL on stack
(if exit border node)
next node determined from received DTL stack (in steps 7 and 8 of DTL processing pseudocode).
(if DTL Terminator)
all zeros.

b. no: This node generated any DTLs of equal or higher level than the crankback level?

b.1 Yes: The node that generated the DTL at the crankback level has been reached (the node is an entry border node, or the DTL originator)
Try alternate routing?
b.1.1 Yes: (copy of SETUP or ADD PARTY was saved)
is there an alternate path that does not violate received DTLs and does not include any blocked transits received in RELEASE messages?

b.1.1.1 Yes: send SETUP or ADD PARTY on new path
(new path necessarily results in new DTLs at least up to the one being cranked back)
b.1.1.2 No:
crankback to next higher level in DTL stack **
crankback level = level of node ID in top DTL on received DTL stack
If link previous to logical group node was cause of blocking:
blocked transit type = call or party blocked at succeeding end of interface
If cause of blocking was within logical group node:
blocked transit type = blocked node
blocked node ID = node ID of current transit in top DTL on receivedDTL stack
If link following logical group node was cause of blocking:
blocked transit type = blocked link
blocked link's preceding node ID = node ID of current transit in top DTL on received DTL stack
blocked link's port ID = port ID of current transit in top DTL on received DTL stack
blocked link's succeeding node ID = target determined from received DTL stack (in step 4.b of DTL processing pseudocode), or all zeros if target is not a node

b.1.2 No:
crankback to next higher level in DTL stack **
crankback level = level of node ID in top DTL on received DTL stack
If link previous to logical group node was cause of blocking:
blocked transit type = call or party blocked at succeeding end of interface
If cause of blocking was within logical group node:
blocked transit type = blocked node
blocked node = node ID of current transit in top DTL on received DTL stack
If link following logical group node was cause of blocking:
blocked transit type = blocked node
blocked link's preceding node ID = node ID of current transit in top DTL on received DTL stack
blocked link's port ID = port ID of current transit in top DTL on received DTL stack
blocked link's succeeding node ID = next destination determined from received DTL stack (in step 4.b of DTL processing pseudocode), or all zeros if next destination is not a node

b.2 No: Progress crankback to previous interface

** Did at least one of the routing attempts get blocked at a link exiting the peer group (for which this node is an entry border node)?
b.1.2.1/b.1.1.2.1 yes: link exiting the peer group should be listed as blocked.
b.1.2.2/b.1.1.2.2 no: logical group node representing the peer group should be listed as blocked

8.4 Carrying Updated Topology State Parameters in Crankback information element

In many cases, blocking due to resource errors occurs due to the use of out-of-date or inaccurate values of routing parameters. In order to add up-to-date parameter values to the information contained in the originating or entry border node's topology database, the GCAC routing parameters specified in Sections

5.8.1.1.3.7, 5.8.1.1.3.8, and 5.8.1.1.3.9 can be optionally included as diagnostics for crankback cause #37, "user cell rate not available". This will allow for more effective routing of future call setups.

A node initiating crankback of a call may include updated GCAC parameter values as a diagnostic. Only updated parameter values for one direction are included, since each node is responsible for originating routing parameter values only for the outgoing direction on any given link. The updated GCAC parameters apply to the same service category as in the connection request. If AvCR, CRM, and VF are included, then complex GCAC applies (see Section 5.13.4.2). If only AvCR is specified, then simple GCAC applies (see Section 5.13.4.3).

For blocked links, the values to be included are the same values that would be advertised in a PTSE (or in a ULIA in Hellos for the upnode side of uplinks) if one were to be emitted at that instant in time. For blocked nodes, the values to be included are the same values that would be advertised in a PTSE as the default values for spokes in the complex node representation. For sake of discussion these values are referred to as the current values. The local node always has the best knowledge of its topology state parameters. The possibly out-of-date values in a given (remote) node’s link state database will be referred to as the last advertised values.

When forming the Crankback information element with a Resource Error, the node initiating crankback of the call includes the relevant current values for the appropriate parameters. The last advertised values may or may not match the current values. The current values could be new/more accurate values which will be advertised in a PTSE to be emitted momentarily, after any holddown finishes. The current values may match the last advertised values in the most recently emitted PTSE which possibly "crossed in the mail" as the call progressed towards this node. The current values could be ones that will not be advertised until either some other more significant event occurs or the refresh timer fires.

In addition to the updated parameter values, a direction and port ID must be specified. For blocked nodes, the direction subfield is coded as "forward" and the port ID is coded as zero. For blocked links, the "forward" direction indicates that the call was blocked at the preceding side of a link, and the "backward" direction indicates that the call was blocked at the succeeding side of the link. If the direction is set to "forward", then the updated topology state parameters apply to the outgoing direction of the port specified by the port ID, from the blocked link’s preceding node.

If the direction is set to "backward", then the port ID is set to zero by the node initiating crankback of the call. The updated parameter values are those values that apply to the physical link or VPC over which the SETUP or ADD PARTY message was received. When the preceding node receives the Crankback information element with the succeeding end blocked indicator, it must insert a port ID that identifies its port for the PNNI interface. For uplinks, the updated parameter values apply to the incoming direction of the specified port on the blocked link’s preceding node. For horizontal links, the updated parameter values apply to the outgoing direction on the corresponding port on the blocked link’s succeeding node.

The port ID included with the updated topology state parameter values must be updated at each entry border node, whenever the call is cranked back to a new level of hierarchy. However, whenever the port is aggregated into another logical port (either at an entry border node or, for the case of blocking at the succeeding end of an uplink, possibly at the preceding node), then the updated parameter values and the port ID must either be replaced with values for the aggregate link or be discarded.

9. Annex C: Soft Permanent Virtual Connection Procedures

This Annex describes the procedures for establishment of soft Permanent Virtual Connections which are established by management action. Both permanent virtual channel connection (PVCC) and permanent virtual path connections (PVPC) are supported at the PNNI corresponding to PVCC and PVPC endpoints at the UNI.

A soft PVPC/PVCC is established and released between the two network interfaces (NIs) serving the permanent virtual connection as a management action. The endpoints of a soft PVPC/PVCC are identified by assigning unique ATM addresses including the SEL octet in the case of a private ATM address (see Section 3.1 of UNI 4.0 signalling specification) to the corresponding NIs. The soft PVPC/PVCC is established between the two network interfaces. The NIs thus becomes the endpoints of the soft PVPC/PVCC.

One of the two endpoints of a PVPC/PVCC “owns” the PVPC/PVCC and is responsible for establishing and releasing the connection. This network interface will be referred to as the calling endpoint. If the switched portion of the PVPC/PVCC gets disconnected because of switching system or link failure, it is also the responsibility of the calling endpoint to try to re-establish the connection. Frequency of re-establishment is an implementation option.

Before a soft PVPC/PVCC can be established, there must be a means to uniquely identify the endpoints of the PVPC/PVCC. The identity of the calling endpoint is encoded in the Calling Party number information element. The Called party of the soft PVPC/PVCC identifies the network interface at the destination network node. The network management system provides the ATM addresses of the endpoints of a soft PVPC/PVCC as well as information about the VPI/VCI values to be used at the two endpoints.

This Annex describes the signalling procedures for the establishment of point-to-point soft PVPC/PVCCs and point-to-multipoint soft PVCCs. The procedures are the same as point-to-point and point-to-multipoint procedures for switched virtual connections with the following exceptions:

9.1 Messages

The information elements AAL parameters, Broadband high layer information, Broadband low layer information, Called party subaddress, and Calling party subaddress information elements, which are used to carry end-to-end information are not included in the SETUP, CONNECT, ADD PARTY, and ADD PARTY ACK messages.

9.2 Procedures for point-to-point soft PVPC/PVCCs

The procedures of this Annex utilize the basic call/connection control procedures of Section 6.5. Additional procedures are described below.

Since the parameters of the PVPC or PVCC are established administratively, separate from the process that establishes the soft PVPC/PVCC, negotiation of end to end parameters is not possible. The specific parameters that are not negotiable are: traffic parameters and QoS parameters. If any of these parameters as specified in the SETUP message cannot be supported by the switching system, the call shall be cranked back with cause and crankback cause #47, *“Resources not available, unspecified”*.

9.2.1 Initiating PVPC/PVCC establishment

The “owner” of the PVPC/PVCC connecting point shall initiate PVPC/PVCC establishment. A SETUP message shall be sent when the PVPC/PVCC is initially configured, or when the switching node which is the owner of the PVPC/PVCC becomes operational (e.g., power up), or during recovery from an outage.

A Bearer Class of VP or X is included in the Broadband bearer capability which identifies establishment of a VPC or a VCC, respectively, between the connecting points. To identify the VPI/VCI of the PVPC/PVCC at the destination network node, the Called party soft PVPC/PVCC information element is included in the SETUP message. The Called party number information element shall contain the configured peer PVPC/PVCC connecting point identifier, and the calling party number information element shall contain the PVPC/PVCC connecting point’s own identifier.

9.2.2 Receiving a CONNECT message at the calling NI

When the originating node receives a CONNECT message, it puts the PVPC/PVCC in an operational state. If the CONNECT message contains the Called party soft PVPC/PVCC information element, the VPI or VPI/VCI values of the PVPC/PVCC segment between the called connecting point and the user will be passed to the management entity.

9.2.3 Receiving a SETUP message at the called NI

When a SETUP message is received at the called connecting point, the procedures of Sections 6.5.2.4 and 6.5.2.6 shall apply. The called party number identifies the network interface serving as the called connecting point of the soft PVPC/PVCC.

9.2.3.1 Last PVPC segment VPI allocation/selection

The calling connecting point shall indicate one of the following for the called endpoint of soft PVPCs:

a) Any VPI;

b) Required VPI.

In case b), if the indicated VPI is available, the called connecting point selects it for the call. In case a), the called connecting point selects any available VPI.

The selected VPI value is indicated in the Called party soft PVPC/PVCC information element in the CONNECT message. The selection type field is coded as "assigned value".

In case b), if the VPI is not available, a RELEASE COMPLETE message with cause #34, "requested called party soft PVPC/PVCC not available", is sent.

9.2.3.2 Last PVCC segment VPI/VCI allocation/selection

The calling connecting point shall indicate one of the following for the called endpoint of soft PVCCs:

a) Any VPI; any VCI;

b) Required VPI; required VCI.

In case b), if the indicated VPI/VCI is available, the called connecting point selects it for the call. In case a), the called connecting point selects any available VPI/VCI.

The selected VPI/VCI value is indicated in the Called party soft PVPC/PVCC information element in the CONNECT message. The selection field is coded as "assigned value".

In case b), if the VPI/VCI is not available, a RELEASE COMPLETE message with cause #34, "requested called party soft PVPC/PVCC not available", is sent.

9.3 Procedures for point-to-multipoint PVCCs

The procedures of this Annex utilize the Point-to-Multipoint call/connection control procedures of Section 6.6.

9.3.1 Adding a party at the originating interface

Connection establishment for point-to-multipoint PVCCs shall be initiated by the root connecting point. When the PVCC is initially configured, or a new party is added by network management, or when the switching node which is the root connecting point becomes operational (e.g., power up), or during recovery from an outage, a SETUP/ADD PARTY message shall be sent to one of the leaf connecting points.

9.3.2 Set up of the first party

The set up of the first party of the point-to-multipoint PVCC is always initiated by sending a SETUP message. A Bearer Class of X is included in the Broadband bearer capability information element which identifies the establishment of a VCC between the connecting points. To identify the VPI/VCI value of the PVCC at the destination network node, the Called party soft PVPC/PVCC information element is included in the SETUP message. The Called party number information element shall contain the configured peer PVCC connecting point identifier, and the calling party number information element shall contain the PVCC connecting point’s own identifier.

9.3.2.1 Receiving a CONNECT message at the calling NI

The procedures of Section 9.2.2 apply.

9.3.2.2 Receiving a SETUP message at the called NI

The procedures of Section 9.2.3 apply.

9.3.3 Adding a party

After the connection is established to the first leaf, connections shall be established to additional leaves. This shall be done by sending an ADD PARTY message for each leaf. To identify the VPI/VCI of the PVCC at the destination network node to the user, the Called party soft PVPC/PVCC information element is included in the ADD PARTY message. The Called party number information element shall contain the configured leaf PVCC connecting point identifier, and the calling party number information element shall contain the root PVCC connecting point's own identifier.

9.3.3.1 Receiving an ADD PARTY ACKNOWLEDGE message at the calling NI

If the ADD PARTY ACKNOWLEDGE message contains the Called party soft PVPC/PVCC information element, the VPI/ VCI values of the PVCC segment between the called connecting point and the user will be passed to the management entity.

9.3.3.2 Receiving an ADD PARTY message at the called NI

When an ADD PARTY message is received, the procedures of Section 6.6.2 shall apply. The called party number identifies the network interface serving as the called connecting point of the PVCC.

9.3.3.2.1 Last PVCC segment VPI/VCI and Endpoint reference allocation/selection

The calling connecting point shall indicate one of the following for the called endpoint of soft PVCCs:

a) Any VPI; any VCI;
b) Required VPI; required VCI.

In case b), if the indicated VPI/VCI is available, the called connecting point selects it for the call. In case a), the called connecting point selects any available VPI/VCI and endpoint reference.

The selected VPI/VCI value is indicated in the Called party soft PVPC/PVCC information element in the ADD PARTY ACK message. The selection field is coded as "assigned value".

In case b), if the VPI/VCI is not available, an ADD PARTY REJECT message with PNNI specific cause #34, "requested called party soft PVPC/PVCC not available", is sent.

10. Annex D: Architectural Constants

These are the architectural constants defined by the PNNI specification.

DefaultAdminWeight = 5040.

This value was chosen for the following reasons:

- The number 5040 has the nice property of being divisible by 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 18, 20, etc. This results in round numbers for administrative weights that are smaller than the default.
- This default value allows for increased numbers of multiple equal weight routes from a source to a destination when a range of administrative weights are used throughout a network.
- This number allows administrators flexibility in configuring administrative weights that are smaller than the default.

ExpiredAge = zero (0).

In a PTSE header, the value of PTSELifetime indicating that the PTSE has expired and should be removed from the topology database.

GroupLeaderIncrement = 50.

To allow for flexibility and stability, when a system concludes that it will be operating as peer group leader, it increments its advertised peer group leadership priority by GroupLeaderIncrement.

InitialSequenceNumber = one (1).

The first instance of all PTSEs is originated with this value as PTSESequenceNumber in the header.

MaxLeadership = 255.

The highest leadership priority for the PGL election protocol.

MaxSequenceNumber = $2^{32} - 1$.

In a PTSE header, the maximum value of PTSESequenceNumber.

OverloadRetryTime = 300 seconds.

The time interval between periodic database resynchronization attempts when a node is in topology database overload state.

RCCQoSClass = 0 (Unspecified QoS).

The QoS class for an RCC.

11. Annex E: Architectural Variables

These are the architectural variables used in the PNNI specification.

AvCR_PM: default value 50, allowed range 1 through 99.

The percentage of the last advertised Available Cell Rate such that a change within the range AvCR ± (AvCR * AvCR_PM / 100) is not considered significant.

AvCR_mT: default value 3, allowed range 1 through 99.

The percentage of maxCR such that changes in AvCR of less than that amount from the last advertised value are never considered significant, even when the rule for AvCR_PM would indicate that the change was significant.

CDV_PM: default value 25, allowed range 1 through 99.

The percentage of the last advertised Cell Delay Variation such that a change within the range CDV ± (CDV * CDV_PM / 100) is not considered significant.

ConfiguredLinkAggregationToken: default value 0.
The link aggregation token value that will be advertised in the Hello protocol.

DSRxmtInterval: default value 5 seconds.
The amount of time, in seconds, a node waits before it sends the previous Database Summary packet again.

HelloInterval: default value 15 seconds.
The amount of time in seconds between Hellos that a node sends on a link, in the absence of event-triggered Hellos.

HorizontalLinkInactivityTime: default value 120 seconds.
The amount of time in seconds a node will continue to advertise a horizontal link for which it has not received and processed the LGN horizontal link IG.

InactivityFactor: default value 5.
The number of HelloIntervals allowed to pass without receiving a Hello, before the Hellos FSM declares that a link is down.

InitialLGNSVCTimeout: default value 4 seconds.
The time interval to defer establishing an RCC SVC so as to minimize the synchronization of multiple SETUPs to a new LGN.

maxCTD_PM: default value 50, allowed range 1 through 99.

The percentage of the last advertised Maximum Cell Transfer Delay such that a change within the range maxCTD ± (maxCTD * maxCTD_PM / 100) is not considered significant.

MinHelloInterval: default value 1 second; minimum value 0.1 seconds.
The minimum interval between successive Hello transmissions.

MinPTSEInterval: default value 1 second; minimum value 0.1 seconds.
The minimum interval between updates of any given PTSE. In other words, new instances of a PTSE can be issued no more often than every MinPTSEInterval seconds.

OverrideDelay: default value 30 seconds.
In the PGL election, the amount of time that a node will wait for all nodes to agree on which node should be elected PGL.

PeerDelayedAckInterval: default value 1 second.
The minimum number of seconds between transmissions of delayed PTSE acknowledgment packets.

PGLInitTime: default value 15 seconds.
The initial value for the PGLInitTimer in the peer group leader election. The node delays specifying its choice for peer group leader until this timer has expired.

PTSELifetimeFactor: default value 200%.
This is used to calculate the initial lifetime of self originated PTSEs. The initial lifetime is set to the product of the PTSERefreshInterval and the PTSELifetimeFactor.

PTSERefreshInterval: default value 1800seconds.
This is the time in seconds between reoriginations of a self-originated PTSE in the absence of triggered updates. A node will reoriginate its PTSEs at this rate in order to prevent flushing of these PTSEs by other nodes.

PTSERetransmissionInterval: default value 5 seconds.
The interval at which unacknowledged PTSEs will be retransmitted. A PTSE will be retransmitted every PTSERetransmissionInterval seconds unless explicitly acknowledged through receipt of either a) an Acknowledgment Packet specifying the PTSE instance or b) the same instance or a more recent instance of the PTSE by flooding.

RCCMaximumBurstSize: default value 171 cells.
The maximum burst size requested for CLP=0+1 for both directions of an RCC.

RCCPeakCellRate: default value 906 cells per second.
The peak cell rate requested for CLP=0+1 for both directions of an RCC.

RCCSustainableCellRate: default value 453 cells per second.
The sustainable cell rate requested for CLP=0+1 for both directions of an RCC.

ReElectionInterval: default value 15 seconds.
The time interval a node waits before restarting the Peer Group Leader election process, after it discovers that it has no connectivity to the current PGL.

RequestRxmtInterval: default value 5 seconds.
The amount of time, in seconds, before a node sends a new PTSE Request Packet requesting PTSEs of the last PTSE Request Packet that have not been received yet.

RetryLGNSVCTimeout: default value 30 seconds.
The time interval to delay between a failed attempt to establish an RCC SVC and a new SETUP for the same RCC SVC.

SVCCalledIntegrityTime: default value 50 seconds.
The default value that is used to initialize the SVCIntegrityTimer at the node that accepts an LGN-LGN SVC originated by a neighbor node.

SVCCallingIntegrityTime: default value 35 seconds.
The default value that is used to initialize the SVCIntegrityTimer at the node that initiates an LGN-LGN SVC.

12. Annex F: Configuration of the PNNI Hierarchy

This Annex presents one preferred way of configuring the PNNI hierarchy. The PNNI MIB (see Annex H) allows one to configure the hierarchy in this manner. Additional methods of configuration are not precluded.

Each lowest level node in the PNNI hierarchy must know its own address, zero or more address summaries to advertise as reachable, its node ID, and the peer group ID.

The default level for a node is 96. The PNNI hierarchy is not necessarily of the same “depth” throughout any particular PNNI routing domain. Each lowest level peer group may therefore be configured, where appropriate, to have any level between 0 and 104.

Either the level or the peer group ID may be configured. If the level is configured, the peer group ID is a prefix of *level* bits on the node’s address padded with trailing zero bits.

Node IDs can also be defaulted based on level; see Section 5.3.3 for examples of how to do this.

Configuration at the lowest level can therefore be simplified if all nodes in a peer group have been assigned addresses that start with the same 12 octet prefix, such that the 12 octet prefix is unique to that peer group. In this case the peer group ID and node ID do not need to be individually configured in each node. However it must be possible to configure lowest-level nodes to use other alternate peer group IDs and node IDs.

Each node must advertise reachability to those end systems that are reachable via that node. In some cases this may consists of a list of host addresses of every reachable host. However, in order to allow PNNI routing to scale, it is highly preferable to summarize host reachability by using a small set of address prefixes to represent reachability to multiple hosts. If a lowest level node has a level other than 104, it has a default summary address which is the 13 octet prefix of its node address. (This implies a presumption that these 13 octet prefixes are unique.) It must be possible to configure nodes to advertise other summary address prefixes either in addition to or instead of the default summary address.

Each node also needs to know the topology state parameters describing the characteristics of each link between that node and neighboring nodes, and the characteristics of the node itself. At the lowest level, it is permissible for these topology state parameters to default based on the physical characteristics of the links and node. At higher levels, these topology state parameters may be calculated based on the lower level links and nodes being aggregated.

Nodes that are not capable of being peer group leader (even at the lowest level of the hierarchy) do not need any additional configured information. Thus, the absolute minimum configuration information needed for such nodes is simply the address.

The peer group leader of each peer group must know the peer group ID of its parent peer group. Each node capable of being peer group leader must be configured to know its leadership priority, as well as its choice for the peer group ID of its parent peer group. Configuration of the higher level parent peer group ID may be done either by manual configuration of the entire ID or by manual configuration of a level. When only a level is configured the ID is a prefix of the lower-level peer group ID.

At a given level of the hierarchy, the PGL is instantiated in a switching system that also instantiates nodes acting as PGL of each descendant peer group in which that switching system participates. As a consequence of this, if such a node fails, it affects a large number of peer groups. This can be ameliorated by having a lowest-level node that exists at a higher level of the hierarchy, and has high PGL priority. Such a node can take over the duties of PGL from that level upwards.

Further detail of the configuration of each node is provided in the PNNI MIB specified in Annex H.

13. Annex G: PNNI Minimum Subsets

This annex defines the minimum functional requirements for implementation of different subsets of a complete PNNI switching system. The four PNNI switching system subsets considered are as follows:

- Minimum function switching system - a switching system that is capable of supporting inside links in a multi-level PNNI hierarchy.
- PGL/LGN switching system - a switching system that is capable of instantiating LGNs.
- Border node capable switching system - a switching system that is capable of supporting outside links.
- Border node capable switching system with LGN peer support - a border node capable switching system whose lowest level nodes support being the peer of an LGN.

The border node and PGL/LGN subsets are extensions of the minimum function switching system subset. The border node with LGN peer support subset is an extension of the border node subset. This relationship is shown in Figure 13-1 below.

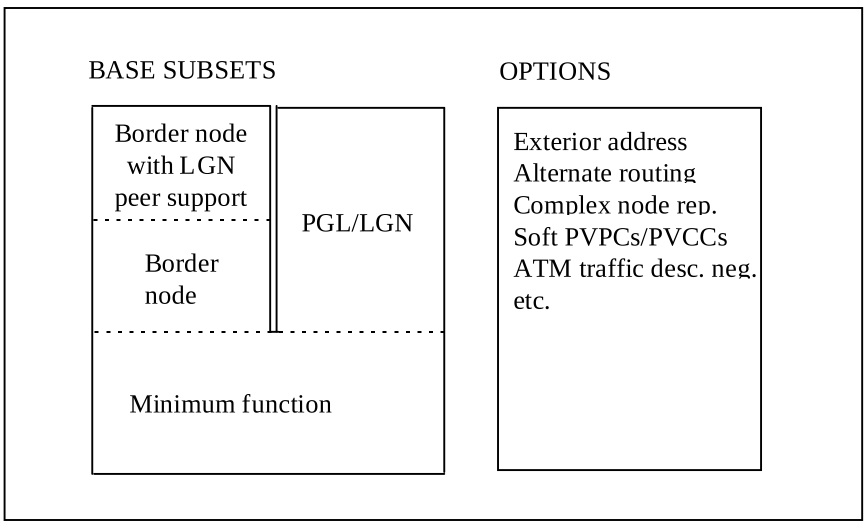


Figure 13-1: PNNI Minimum Subsets

All PNNI switching systems shall support the capabilities required of a minimum function switching system. All other capabilities are optional.

Table 13-1 below shows the the mandatory and optional capabilities corresponding to each of the different PNNI minimum subsets defined above.

Table 13-1: Mandatory and Optional Capabilities

No.	Capabilities	Switching System
1	Supports inside links	M
2	Version negotiation	M
3	Information group tags	M
4	Hello protocol over inside physical links	M
5	Database synchronization	M

6	Flooding	M
7	Understand all defined PTSE types	M
8	Origination of Nodal Information PTSEs	M
9	Origination of Horizontal Link PTSEs	M
10	Origination of Internal Reachable Address PTSEs	M
11	Vote in PGL elections	M
12	Perform default internal address summarization	M
13	Point-to-point Calls	M
14	Point-to-multipoint Calls	M
15	Signalling of Individual QoS Parameters	M
16	Notification of End-to-end Connection Completion	M
17	ATM Anycast	M
18	Generate a Multi-level Designated Transit List	M
19	Follow a Multi-level DTL	M
20	Crankback	M(Note 1)
21	Outside link support	B
22	Hello protocol over outside links	B
23	Originate uplink PTSEs	B, P
24	Entry and exit border node DTL handling	B
25	Entry border node crankback handling	B
26	SVC-based RCC	N, P
27	SVC-based RCC Hello protocol	N, P
28	LGN horizontal link Hello protocol	N, P
29	PGL capable	P
30	Link aggregation	N, P
31	Nodal aggregation	P
32	Internal and Exterior address summarization	P
33	Exterior reachable address	O
34	Alternate routing as a result of Crankback	O
35	Hello protocol over VPCs	O
36	Associated Signalling	O
37	Negotiation of ATM traffic descriptors	O
38	Switched Virtual Path (VP) service	O
39	Soft PVPC and PVCC support	O
40	ABR Signalling for Point-to-point Calls	O
41	Generic Identifier Transport	O
42	Frame Discard	O(Note 2)
43	ILMI over PNNI links	O

M: Mandatory for all categories of switching systems (i.e. minimum function subset)
B: Required for border node capable switching systems; otherwise optional
N: Required for border node capable switching systems with peer LGN support; otherwise optional
P: Required for PGL/LGN capable switching systems; otherwise optional
O: Optional for all categories of switching systems

Note 1 - Support of alternate routing is optional.

Note 2 - Transport of the Frame Discard indication is mandatory.

14. Annex H: The PNNI Management Information Base

```
PNNI-MIB DEFINITIONS ::= BEGIN

        IMPORTS
                MODULE-IDENTITY, OBJECT-TYPE, OBJECT-IDENTITY,
                Counter32, Gauge32, Integer32, enterprises
                        FROM SNMPv2-SMI
                TEXTUAL-CONVENTION, RowStatus, DisplayString,
                TimeStamp, TruthValue
                        FROM SNMPv2-TC
                InterfaceIndex, ifIndex
                        FROM IF-MIB
                AtmTrafficDescrParamIndex
                        FROM ATM-MIB
                MODULE-COMPLIANCE, OBJECT-GROUP
                        FROM SNMPv2-CONF;


pnniMIB MODULE-IDENTITY
        LAST-UPDATED    "9602270000Z"
        ORGANIZATION    "The ATM Forum"
        CONTACT-INFO
            "The ATM Forum
             2570 West El Camino Real, Suite 304
             Mountain View, CA 94040-1313 USA
             Phone: +1 415-949-6700
             Fax:   +1 415-949-6705
             info@atmforum.com"
        DESCRIPTION
            "The MIB module for managing ATM Forum PNNI routing."
        REVISION        "9602270000Z"
        DESCRIPTION
            "Initial version of the MIB for monitoring and controlling
             PNNI routing."
        ::= { atmfPnni 1 }

-- The object identifier subtree for ATM Forum PNNI MIBs

atmForum        OBJECT IDENTIFIER ::= { enterprises 353 }
atmForumNetworkManagement       OBJECT IDENTIFIER ::= { atmForum 5 }
atmfPnni        OBJECT IDENTIFIER ::= { atmForumNetworkManagement 4 }


pnniMIBObjects OBJECT IDENTIFIER ::= { pnniMIB 1 }


    Unsigned32 ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "This definition, which is in compliance with RFC 1902, is a
             temporary inclusion in the PNNI MIB until such time as MIB
             compilers are upgraded and thereby can accept references to
             the new definitions in RFC 1902."
        REFERENCE
            "RFC 1902"
        SYNTAX        Gauge32


    PnniAtmAddr ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
```

```
        "The ATM address used by the network entity.  The address
         types are: no address (0 octets), and NSAP (20 octets)."
    REFERENCE
        "ATM Forum PNNI 1.0 Section 5.2"
    SYNTAX        OCTET STRING (SIZE(0|20))


PnniNodeIndex ::= TEXTUAL-CONVENTION
    STATUS        current
    DESCRIPTION
        "An index that identifies a logical PNNI entity within the
         managed system.

         The distinguished value zero indicates the null instance or
         no instance in the PnniNodeCfgParentNodeIndex.  In all
         other cases, the distinguished value zero indicates a
         logical entity within the switching system that manages
         routes only over non-PNNI interfaces.

         By default, only the node identified by node index one is
         created, and all PNNI interfaces are associated with that
         node."
    SYNTAX        Integer32 (0..65535)


PnniNodeId ::= TEXTUAL-CONVENTION
    STATUS        current
    DESCRIPTION
        "A PNNI node ID - this is used to identify the logical PNNI
         node."
    REFERENCE
        "ATM Forum PNNI 1.0 Section 5.3.3"
    SYNTAX        OCTET STRING (SIZE(22))


PnniPortId ::= TEXTUAL-CONVENTION
    STATUS        current
    DESCRIPTION
        "A PNNI port ID - this is used to identify a point of
         attachment of a logical link to a given logical node.

         The values 0 and 0xffffffff have special meanings in
         certain contexts and do not identify a specific port.

         The distinguished value 0 indicates that no port is
         specified."
    REFERENCE
        "ATM Forum PNNI 1.0 Section 5.3.4"
    SYNTAX        Unsigned32


PnniAggrToken ::= TEXTUAL-CONVENTION
    STATUS        current
    DESCRIPTION
        "A PNNI aggregation token - this is used to determine which
         links to a given neighbor node are to be aggregated and
         advertised as a single logical link."
    REFERENCE
        "ATM Forum PNNI 1.0 Section 5.3.5"
    SYNTAX        Unsigned32


PnniPeerGroupId ::= TEXTUAL-CONVENTION
    STATUS        current
```

```
        DESCRIPTION
            "A PNNI peer group ID."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.2"
        SYNTAX        OCTET STRING (SIZE(14))


    PnniLevel ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "A PNNI routing level indicator."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.1"
        SYNTAX        Integer32 (0..104)


    PnniSvccRccIndex ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "The value of this object identifies the SVCC-based RCC for
             which the entry contains management information."
        SYNTAX        Integer32


    AtmAddrPrefix ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "A prefix of one or more ATM End System Addresses.  The
             significant portion of a prefix is padded with zeros on the
             right to fill 19 octets."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.2"
        SYNTAX        OCTET STRING (SIZE(19))


    PnniPrefixLength ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "The number of bits that are significant in an ATM address
             prefix used by PNNI."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.2"
        SYNTAX        Integer32 (0..152)


    PnniMetricsTag ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "An index into the pnniMetricsTable.  The suffix tag is used
             to indicate that there may be many related entries in the
             table further discriminated by other index terms."
        SYNTAX        Integer32 (0..MAX)


    ServiceCategory ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "Indicates the service category."
        REFERENCE
            "ATM Forum Traffic Management 4.0 Section 2"
        SYNTAX        INTEGER { other(1),
                                cbr(2),
                                rtVbr(3),
                                nrtVbr(4),
```

```
                            abr(5),
                            ubr(6) }


    ClpType ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "Indicates the CLP type of a traffic stream."
        SYNTAX        INTEGER { clpEqual0(1), clpEqual0Or1(2) }


    TnsType ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "Indicates the type of network identification of a
             specified transit network."
        REFERENCE
            "ATM Forum UNI Signalling 4.0 Section 2 4.5.22/Q.2931"
        SYNTAX        INTEGER { nationalNetworkIdentification(2),
                                other(8) }


    TnsPlan ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "Indicates the network identification plan of a
             specified transit network."
        REFERENCE
            "ATM Forum UNI Signalling 4.0 Section 2 4.5.22/Q.2931"
        SYNTAX        INTEGER { carrierIdentificationCode(1),
                                other(16) }

    PnniVersion ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "Indicates a version of the PNNI protocol."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.6.1"
        SYNTAX        INTEGER { unsupported(1), version1point0(2) }


    PnniHelloState ::= TEXTUAL-CONVENTION
        STATUS        current
        DESCRIPTION
            "The state of an instance of the PNNI Hello State machine."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.6.2.1.2"
        SYNTAX        INTEGER {
                                notApplicable(1),
                                down(2),
                                attempt(3),
                                oneWayInside(4),
                                twoWayInside(5),
                                oneWayOutside(6),
                                twoWayOutside(7),
                                commonOutside(8)
                                }


    zeroDotZero   OBJECT-IDENTITY
        STATUS        current
        DESCRIPTION
            "A value used for null identifiers.
```

```
             This definition, which is in compliance with RFC 1902, is a
             temporary inclusion in the PNNI MIB until such time as MIB
             compilers are upgraded and thereby can accept references to
             the new definitions in RFC 1902."
        REFERENCE
            "RFC 1902"
        ::= { 0 0 }


-- the base group

pnniBaseGroup OBJECT IDENTIFIER ::= { pnniMIBObjects 1 }

pnniHighestVersion OBJECT-TYPE
        SYNTAX          PnniVersion
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The highest version of the PNNI protocol that the
             software in this switching system is capable of executing."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.6.1"
        ::= { pnniBaseGroup 1 }

pnniLowestVersion OBJECT-TYPE
        SYNTAX          PnniVersion
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The lowest version of the PNNI Protocol that the
             software in this switching system is capable of executing."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.6.1"
        ::= { pnniBaseGroup 2 }

pnniDtlCountOriginator OBJECT-TYPE
        SYNTAX          Counter32
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The total number of DTL stacks that this switching system
             has originated as the DTLOriginator and placed into
             signalling messages. This includes the initial DTL stacks
             computed by this system as well as any alternate route
             (second, third choice etc.) DTL stacks computed by this
             switching system in response to crankbacks."
        ::= { pnniBaseGroup 3 }

pnniDtlCountBorder OBJECT-TYPE
        SYNTAX          Counter32
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The number of partial DTL stacks that this switching system
             has added into signalling messages as an entry border node.
             This includes the initial partial DTL stacks computed by
             this system as well as any alternate route (second, third
             choice etc.) partial DTL stacks computed by this switching
             system in response to crankbacks."
        ::= { pnniBaseGroup 4 }

pnniCrankbackCountOriginator OBJECT-TYPE
        SYNTAX          Counter32
        MAX-ACCESS      read-only
```

```
        STATUS        current
        DESCRIPTION
            "The count of the total number of connection setup messages
             including DTL stacks originated by this switching system
             that have cranked back to this switching system at all
             levels of the hierarchy."
        ::= { pnniBaseGroup 5 }

pnniCrankbackCountBorder OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The count of the total number of connection setup messages
             including DTLs added by this switching system as an entry
             border node that have cranked back to this switching system
             at all levels of the hierarchy. This count does not include
             Crankbacks for which this switching system was not the
             crankback destination, only those crankbacks that were
             directed to this switching system are counted here."
        ::= { pnniBaseGroup 6 }

pnniAltRouteCountOriginator OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The total number of alternate DTL stacks that this
             switching system has computed and placed into
             signalling messages as the DTLOriginator."
        ::= { pnniBaseGroup 7 }

pnniAltRouteCountBorder OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The total number of alternate partial DTL stacks that this
             switching system has computed and placed into signalling
             messages as an entry border node."
        ::= { pnniBaseGroup 8 }

pnniRouteFailCountOriginator OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The total number of times where the switching system failed
             to compute a viable DTL stack as the DTLOriginator for some
             call.  It indicates the number of times a call was cleared
             from this switching system due to originator routing
             failure."
        ::= { pnniBaseGroup 9 }

pnniRouteFailCountBorder OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The total number of times where the switching system failed
             to compute a viable partial DTL stack as an entry border
             node for some call.  It indicates the number of times a
             call was either cleared or cranked back from this switching
             system due to border routing failure."
```

```
        ::= { pnniBaseGroup 10 }

pnniRouteFailUnreachableOriginator OBJECT-TYPE
        SYNTAX          Counter32
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The total number of times where the switching system failed
             to compute a viable DTL stack as the DTLOriginator because
             the destination was unreachable, i.e., those calls that are
             cleared with cause #2 `specified transit network
             unreachable' or cause #3 `destination unreachable' in the
             cause IE."
        ::= { pnniBaseGroup 11 }

pnniRouteFailUnreachableBorder OBJECT-TYPE
        SYNTAX          Counter32
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The total number of times where the switching system failed
             to compute a viable partial DTL stack as an entry border
             node because the target of the path calculation was
             unreachable, i.e., those calls that are cleared or cranked
             back with cause #2 `specified transit network unreachable'
             or cause #3 `destination unreachable' in the cause IE."
        ::= { pnniBaseGroup 12 }


-- node table

pnniNodeTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF PnniNodeEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "The pnniNodeTable collects attributes that affect the
             operation of a PNNI logical node.

             There is a single row in this table for each PNNI peer
             group that the managed system is expected or eligible
             to become a member of."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex F"
        ::= { pnniMIBObjects 2 }

pnniNodeEntry OBJECT-TYPE
        SYNTAX          PnniNodeEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "An entry in the table, containing information about a PNNI
             logical node in this switching system."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex F"
        INDEX          { pnniNodeIndex }
        ::= { pnniNodeTable 1 }

PnniNodeEntry ::=
        SEQUENCE {
                pnniNodeIndex           PnniNodeIndex,
                pnniNodeLevel           PnniLevel,
                pnniNodeId              PnniNodeId,
                pnniNodeLowest          TruthValue,
```

```
                pnniNodeAdminStatus     INTEGER,
                pnniNodeOperStatus      INTEGER,
                pnniNodeDomainName      DisplayString,
                pnniNodeAtmAddress      PnniAtmAddr,
                pnniNodePeerGroupId     PnniPeerGroupId,
                pnniNodeRestrictedTransit       TruthValue,
                pnniNodeComplexRep      TruthValue,
                pnniNodeRestrictedBranching     TruthValue,
                pnniNodeDatabaseOverload        TruthValue,
                pnniNodePtses           Gauge32,
                pnniNodeRowStatus       RowStatus
                }

pnniNodeIndex OBJECT-TYPE
        SYNTAX        PnniNodeIndex
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A value assigned to a node in this switching system that
             uniquely identifies it in the MIB."
        ::= { pnniNodeEntry 1 }

pnniNodeLevel OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The level of PNNI hierarchy at which this node exists. This
             attribute is used to determine the default node ID and the
             default peer group ID for this node. This object may only
             be written when pnniNodeAdminStatus has the value down."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.1, Annex F"
        DEFVAL { 96 }
        ::= { pnniNodeEntry 2 }

pnniNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The value the switching system is using to represent
             itself as this node. This object may only be written when
             pnniNodeAdminStatus has the value down.

             If pnniNodeLowest is true, then the default node ID takes
             the form defined in Section 5.3.3 for lowest level nodes,
             with the first octet equal to pnniNodeLevel, the second
             octet equal to 160, and the last 20 octets equal to
             pnniNodeAtmAddress.

             If pnniNodeLowest is false, then the default
             node ID takes the form defined in Section 5.3.3 for logical
             group nodes, with the first octet equal to pnniNodeLevel,
             the next fourteen octets equal to the value of
             pnniNodePeerGroupId for the child node whose election as
             PGL causes this LGN to be instantiated, the next six octets
             equal to the ESI of pnniNodeAtmAddress, and the last octet
             equal to zero."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.3, Annex F"
        ::= { pnniNodeEntry 3 }

pnniNodeLowest OBJECT-TYPE
```

```
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Indicates whether this node acts as a lowest level node or
             whether this node is a logical group node that becomes
             active when one of the other nodes in this switching system
             becomes a peer group leader.  The value 'false' must not be
             used with nodes that are not PGL/LGN capable.

             This object may only be
             written when pnniNodeAdminStatus has the value down."
        DEFVAL { true }
        ::= { pnniNodeEntry 4 }

pnniNodeAdminStatus OBJECT-TYPE
        SYNTAX        INTEGER { up(1), down(2) }
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Indicates whether the administrative status of the node is
             up (the node is allowed to become active) or down (the node
             is forced to be inactive).

             When pnniNodeAdminStatus is down, then pnniNodeOperStatus
             must also be down."
        DEFVAL { up }
        ::= { pnniNodeEntry 5 }

pnniNodeOperStatus OBJECT-TYPE
        SYNTAX        INTEGER { up(1), down(2) }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the node is active or whether the node
             has yet to become operational.  When the value is down, all
             state has been cleared from the node and the node is not
             communicating with any of its neighbor nodes."
        ::= { pnniNodeEntry 6 }

pnniNodeDomainName OBJECT-TYPE
        SYNTAX        DisplayString
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The name of the PNNI routing
             domain in which this node participates.  All lowest-level
             PNNI nodes with the same pnniNodeDomainName are presumed to
             be connected."
        DEFVAL { "" }
        ::= { pnniNodeEntry 7 }

pnniNodeAtmAddress OBJECT-TYPE
        SYNTAX        PnniAtmAddr
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "This node's ATM End System Address.  Remote systems wishing
             to exchange PNNI protocol packets with this node should
             direct packets or calls to this address.

             This attribute may only be written when pnniNodeAdminStatus
             has the value down."
        REFERENCE
```

```
            "ATM Forum PNNI 1.0 Section 5.2.2"
        ::= { pnniNodeEntry 8 }

pnniNodePeerGroupId OBJECT-TYPE
        SYNTAX        PnniPeerGroupId
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The Peer Group Identifier of the peer group that the given
             node is to become a member of.

             The default value of this attribute has the first octet
             equal to pnniNodeLevel, the next pnniNodeLevel bits equal
             to the pnniNodeLevel bits starting from the third octet of
             pnniNodeId, and the remainder padded with zeros.

             This object may only be written when pnniNodeAdminStatus
             has the value down."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.2, Annex F"
        ::= { pnniNodeEntry 9 }

pnniNodeRestrictedTransit OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Specifies whether the node is restricted to not allowing
             support of SVCs transiting this node.  This attribute
             determines the setting of the restricted transit bit in the
             nodal information group originated by this node."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.2.3"
        DEFVAL { false }
        ::= { pnniNodeEntry 10 }

pnniNodeComplexRep OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Specifies whether this node uses the complex node
             representation.  A value of `true' indicates that the
             complex node representation is used, whereas a value of
             `false' indicates that the simple node representation is
             used.  This attribute determines the setting of the nodal
             representation bit in the nodal information group
             originated by this node."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.2.3"
        ::= { pnniNodeEntry 11 }

pnniNodeRestrictedBranching  OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the node is able to support additional
             point-to-multipoint branches.  A value of 'false' indicates
             that additional branches can be supported, and a value of
             'true' indicates that additional branches cannot be
             supported.  This attribute reflects the setting of the
             restricted branching bit in the nodal information group
             originated by this node."
```

```
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.2.3"
        ::= { pnniNodeEntry 12}

pnniNodeDatabaseOverload OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Specifies whether the node is currently operating in
             topology database overload state.  This attribute has the
             same value as the Non-transit for PGL Election bit in the
             nodal information group originated by this node."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.2.3"
        ::= { pnniNodeEntry 13 }

pnniNodePtses OBJECT-TYPE
        SYNTAX        Gauge32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Gauges the total number of PTSEs currently in this
             node's topology database(s)."
        ::= { pnniNodeEntry 14 }

pnniNodeRowStatus OBJECT-TYPE
        SYNTAX        RowStatus
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "To create, delete, activate and de-activate a Node."
        ::= { pnniNodeEntry 15 }


-- PGL election table

pnniNodePglTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniNodePglEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "Peer group leader election information for a PNNI node in
             this switching system."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.10.1"
        ::= { pnniMIBObjects 3 }

pnniNodePglEntry OBJECT-TYPE
        SYNTAX        PnniNodePglEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing PGL election information
             of a PNNI logical node in this switching system."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.10.1"
        AUGMENTS      { pnniNodeEntry }
        ::= { pnniNodePglTable 1 }

PnniNodePglEntry ::=
        SEQUENCE {
                pnniNodePglLeadershipPriority   INTEGER,
                pnniNodeCfgParentNodeIndex      PnniNodeIndex,
```

```
                pnniNodePglInitTime             Integer32,
                pnniNodePglOverrideDelay        Integer32,
                pnniNodePglReelectTime          Integer32,
                pnniNodePglState                INTEGER,
                pnniNodePreferredPgl            PnniNodeId,
                pnniNodePeerGroupLeader         PnniNodeId,
                pnniNodePglTimeStamp            TimeStamp,
                pnniNodeActiveParentNodeId      PnniNodeId
                }

pnniNodePglLeadershipPriority OBJECT-TYPE
        SYNTAX        INTEGER (0..205)
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The Leadership priority value this node should advertise in
             its nodal information group for the given peer group.  Only
             the value zero can be used with nodes that are not PGL/LGN
             capable.  If there is no configured parent node index or no
             corresponding entry in the pnniNodeTable, then the
             advertised leadership priority is zero regardless of this
             value."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.10.1.2"
        DEFVAL { 0 }
        ::= { pnniNodePglEntry 1 }

pnniNodeCfgParentNodeIndex OBJECT-TYPE
        SYNTAX        PnniNodeIndex
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The local node index used to identify the node that will
             represent this peer group at the next higher level of
             hierarchy, if this node becomes peer group leader.  The
             value 0 indicates that there is no parent node."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex F"
        DEFVAL { 0 }
        ::= { pnniNodePglEntry 2 }

pnniNodePglInitTime OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The amount of time in seconds this node will delay
             advertising its choice of preferred PGL after having
             initialized operation and reached the full state with at
             least one neighbor in the peer group."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G PGLInitTime"
        DEFVAL  { 15 }
        ::= { pnniNodePglEntry 3 }

pnniNodePglOverrideDelay OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The amount of time in seconds a node will wait for itself
             to be declared the preferred PGL by unanimous agreement
```

```
              among its peers.  In the absence of unanimous agreement
              this will be the amount of time that will pass before this
              node considers a two thirds majority as sufficient
              agreement to declare itself peer group leader, abandoning
              the attempt to get unanimous agreement."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G OverrideDelay"
        DEFVAL  { 30 }
        ::= { pnniNodePglEntry 4 }

pnniNodePglReelectTime OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The amount of time in seconds after losing connectivity to
             the current peer group leader, that this node will wait
             before re-starting the process of electing a new peer group
             leader."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G ReElectionInterval"
        DEFVAL  { 15 }
        ::= { pnniNodePglEntry 5 }

pnniNodePglState OBJECT-TYPE
        SYNTAX        INTEGER {
                                starting(1),
                                awaiting(2),
                                awaitingFull(3),
                                initialDelay(4),
                                calculating(5),
                                awaitUnanimity(6),
                                operPgl(7),
                                operNotPgl(8),
                                hungElection(9),
                                awaitReElection(10)
                                }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates the state that this node is in with respect to
             the Peer Group Leader election that takes place in the
             node's peer group. The values are enumerated in the Peer
             Group Leader State Machine."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.10.1.1.2"
        ::= { pnniNodePglEntry 6 }

pnniNodePreferredPgl OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The Node ID of
             the node which the local node believes should be or become
             the peer group leader.  This is also the value the local
             node is currently advertising in the `Preferred Peer Group
             Leader Node ID' field of its nodal information group within
             the given peer group.  If a Preferred PGL has not been
             chosen, this attribute's value is set to (all) zero(s)."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.10.1.1.6"
        ::= { pnniNodePglEntry 7 }
```

```
pnniNodePeerGroupLeader OBJECT-TYPE
        SYNTAX          PnniNodeId
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The Node Identifier of the node which is currently
             operating as peer group leader of the peer group this node
             belongs to. If a PGL has not been elected, this attribute's
             value is set to (all) zero(s)."
        ::= { pnniNodePglEntry 8 }

pnniNodePglTimeStamp OBJECT-TYPE
        SYNTAX          TimeStamp
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The time at which the current Peer Group Leader established
             itself."
        ::= { pnniNodePglEntry 9 }

pnniNodeActiveParentNodeId OBJECT-TYPE
        SYNTAX          PnniNodeId
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The Node Identifier value being used by the Peer Group
             Leader to represent this peer group at the next higher
             level of the hierarchy. If this node is at the highest
             level of the hierarchy or if no PGL has yet been elected
             the PNNI Protocol Entity sets the value of this attribute
             to (all) zero(s)."
        ::= { pnniNodePglEntry 10 }


-- initial timer values table

pnniNodeTimerTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF PnniNodeTimerEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "A table of initial PNNI timer values and significant
             change thresholds."
        ::= { pnniMIBObjects 4 }

pnniNodeTimerEntry OBJECT-TYPE
        SYNTAX          PnniNodeTimerEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "An entry in the table, containing initial PNNI timer values
             and significant change thresholds of a PNNI logical node in
             this switching system."
        AUGMENTS        { pnniNodeEntry }
        ::= { pnniNodeTimerTable 1 }

PnniNodeTimerEntry ::=
        SEQUENCE {
                pnniNodePtseHolddown            Integer32,
                pnniNodeHelloHolddown           Integer32,
                pnniNodeHelloInterval           Integer32,
                pnniNodeHelloInactivityFactor   Integer32,
                pnniNodeHlinkInact              Integer32,
```

```
                pnniNodePtseRefreshInterval     Integer32,
                pnniNodePtseLifetimeFactor      INTEGER,
                pnniNodeRxmtInterval            Integer32,
                pnniNodePeerDelayedAckInterval  Integer32,
                pnniNodeAvcrPm                  INTEGER,
                pnniNodeAvcrMt                  INTEGER,
                pnniNodeCdvPm                   INTEGER,
                pnniNodeCtdPm                   INTEGER
                }

pnniNodePtseHolddown OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "100 milliseconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The initial value for the PTSE hold down timer that will be
             used by the given node to limit the rate at which it can
             re-originate PTSEs. It must be a positive non-zero number."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G MinPTSEInterval"
        DEFVAL { 10 }
        ::= { pnniNodeTimerEntry 1 }

pnniNodeHelloHolddown OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "100 milliseconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The initial value for the Hello hold down timer that will
             be used by the given node to limit the rate at which it
             sends Hellos. It must be a positive non-zero number."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G MinHelloInterval"
        DEFVAL { 10 }
        ::= { pnniNodeTimerEntry 2 }

pnniNodeHelloInterval OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The initial value for the Hello Timer.
             In the absence of triggered Hellos, this node will send one
             Hello packet on each of its ports on this interval."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G HelloInterval"
        DEFVAL { 15 }
        ::= { pnniNodeTimerEntry 3 }

pnniNodeHelloInactivityFactor OBJECT-TYPE
        SYNTAX        Integer32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The value for the Hello Inactivity factor that this
             node will use to determine when a neighbor has gone down."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G InactivityFactor"
        DEFVAL { 5 }
        ::= { pnniNodeTimerEntry 4 }
```

```
pnniNodeHlinkInact OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The amount of time a node will continue to
             advertise a horizontal (logical) link for which it has
             not received and processed a LGN Horizontal Link
             information group."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G HorizontalLinkInactivityTime"
        DEFVAL { 120 }
        ::= { pnniNodeTimerEntry 5 }

pnniNodePtseRefreshInterval OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The initial value for the Refresh timer that this node will
             use to drive (re-)origination of PTSEs in the absence of
             triggered updates."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G PTSERefreshInterval"
        DEFVAL  { 1800 }
        ::= { pnniNodeTimerEntry 6 }

pnniNodePtseLifetimeFactor OBJECT-TYPE
        SYNTAX        INTEGER (101..1000)
        UNITS         "percent"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The value for the lifetime multiplier, expressed as a
             percentage.  The result of multiplying the
             pnniNodePtseRefreshInterval attribute value by this
             attribute value is used as the initial lifetime that this
             node places into self-originated PTSEs."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G PTSELifetimeFactor"
        DEFVAL  { 200 }
        ::= { pnniNodeTimerEntry 7 }

pnniNodeRxmtInterval OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The period between retransmissions of unacknowledged
             Database Summary packets, PTSE Request packets, and PTSPs."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G DSRxmtInterval,
             RequestRxmtInterval, PTSERetransmissionInterval"
        DEFVAL { 5 }
        ::= { pnniNodeTimerEntry 8 }

pnniNodePeerDelayedAckInterval OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "100 milliseconds"
        MAX-ACCESS    read-create
        STATUS        current
```

```
        DESCRIPTION
            "The minimum amount of time between transmissions of
             delayed PTSE acknowledgement packets."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G PeerDelayedAckInterval,
             Appendix G"
        DEFVAL { 10 }
        ::= { pnniNodeTimerEntry 9 }

pnniNodeAvcrPm OBJECT-TYPE
        SYNTAX        INTEGER (1..99)
        UNITS         "percent"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The proportional multiplier used in the algorithms that
             determine significant change for AvCR parameters, expressed
             as a percentage."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.5.2.5.4, Annex G AvCR_PM"
        DEFVAL  { 50 }
        ::= { pnniNodeTimerEntry 10 }

pnniNodeAvcrMt OBJECT-TYPE
        SYNTAX        INTEGER (1..99)
        UNITS         "percent"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The minimum threshold used in the algorithms that determine
             significant change for AvCR parameters, expressed as a
             percentage."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.5.2.5.4, Annex G AvCR_mT"
        DEFVAL  { 3 }
        ::= { pnniNodeTimerEntry 11 }

pnniNodeCdvPm OBJECT-TYPE
        SYNTAX        INTEGER (1..99)
        UNITS         "percent"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The proportional multiplier used in the algorithms that
             determine significant change for CDV metrics, expressed as
             a percentage."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.5.2.5.6, Annex G CDV_PM"
        DEFVAL  { 25 }
        ::= { pnniNodeTimerEntry 12 }

pnniNodeCtdPm OBJECT-TYPE
        SYNTAX        INTEGER (1..99)
        UNITS         "percent"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The proportional multiplier used in the algorithms that
             determine significant change for CTD metrics, expressed as
             a percentage."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.5.2.5.5, Annex G maxCTD_PM"
        DEFVAL  { 50 }
        ::= { pnniNodeTimerEntry 13 }
```

```
-- nodal SVCC-based RCC variables table

pnniNodeSvccTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniNodeSvccEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A table of variables related to SVCC-based routing control
             channels.."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.5"
        ::= { pnniMIBObjects 5 }

pnniNodeSvccEntry OBJECT-TYPE
        SYNTAX        PnniNodeSvccEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing SVCC-based RCC variables
             of a PNNI logical node in this switching system."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.5"
        AUGMENTS      { pnniNodeEntry }
        ::= { pnniNodeSvccTable 1 }

PnniNodeSvccEntry ::=
        SEQUENCE {
                pnniNodeSvccInitTime                    Integer32,
                pnniNodeSvccRetryTime                   Integer32,
                pnniNodeSvccCallingIntegrityTime        Integer32,
                pnniNodeSvccCalledIntegrityTime         Integer32,
           pnniNodeSvccTrafficDescriptorIndex  AtmTrafficDescrParamIndex
                }

pnniNodeSvccInitTime OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The amount of time this node will delay initiating
             establishment of an SVCC to a neighbor with a numerically
             lower ATM address, after determining that such an SVCC
             should be established."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G InitialLGNSVCTimeout"
        DEFVAL { 4 }
        ::= { pnniNodeSvccEntry 1 }

pnniNodeSvccRetryTime OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The amount of time this node will delay after an apparently
             still necessary and viable SVCC-based RCC is unexpectedly
             torn down, before attempting to re-establish it."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G RetryLGNSVCTimeout"
        DEFVAL { 30 }
        ::= { pnniNodeSvccEntry 2 }
```

```
pnniNodeSvccCallingIntegrityTime OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The amount of time this node will wait for an SVCC, which
             it has initiated establishment of as the calling party, to
             become fully established before giving up and tearing it
             down."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G SVCCallingIntegrityTime"
        DEFVAL { 35 }
        ::= { pnniNodeSvccEntry 3 }

pnniNodeSvccCalledIntegrityTime OBJECT-TYPE
        SYNTAX        Integer32
        UNITS         "seconds"
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The amount of time this node will wait for an SVCC, which
             it has decided to accept as the called party, to become
             fully established before giving up and tearing it down."
        REFERENCE
            "ATM Forum PNNI 1.0 Annex G SVCCalledIntegrityTime"
        DEFVAL { 50 }
        ::= { pnniNodeSvccEntry 4 }

pnniNodeSvccTrafficDescriptorIndex OBJECT-TYPE
        SYNTAX        AtmTrafficDescrParamIndex
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An index into the atmTrafficDescrParamTable defined in
             RFC 1695.  This traffic descriptor is used when
             establishing switched virtual channels for use as
             SVCC-based RCCs to/from PNNI logical group nodes."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.5.2, Annex G
             RCCMaximumBurstSize, RCCPeakCellRate,
             RCCSustainableCellRate"
        ::= { pnniNodeSvccEntry 5 }


-- scope mapping table

pnniScopeMappingTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniScopeMappingEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The pnniScopeTable contains the mappings of membership and
             connection scope from organizational scope values (used at
             UNI interfaces) to PNNI scope (i.e. in terms of PNNI
             routing level indicators)."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.6"
        ::= { pnniMIBObjects 6 }

pnniScopeMappingEntry OBJECT-TYPE
        SYNTAX        PnniScopeMappingEntry
        MAX-ACCESS    not-accessible
```

```
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing scope mapping information
             for a PNNI logical node in this switching system."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.6"
        AUGMENTS      { pnniNodeEntry }
        ::= { pnniScopeMappingTable 1 }

PnniScopeMappingEntry ::=
        SEQUENCE {
                pnniScopeLocalNetwork              PnniLevel,
                pnniScopeLocalNetworkPlusOne       PnniLevel,
                pnniScopeLocalNetworkPlusTwo       PnniLevel,
                pnniScopeSiteMinusOne              PnniLevel,
                pnniScopeIntraSite                 PnniLevel,
                pnniScopeSitePlusOne               PnniLevel,
                pnniScopeOrganizationMinusOne      PnniLevel,
                pnniScopeIntraOrganization         PnniLevel,
                pnniScopeOrganizationPlusOne       PnniLevel,
                pnniScopeCommunityMinusOne         PnniLevel,
                pnniScopeIntraCommunity            PnniLevel,
                pnniScopeCommunityPlusOne          PnniLevel,
                pnniScopeRegional                  PnniLevel,
                pnniScopeInterRegional             PnniLevel,
                pnniScopeGlobal                    PnniLevel
                }

pnniScopeLocalNetwork OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value localNetwork(1)."
        DEFVAL { 96 }
        ::= { pnniScopeMappingEntry 1 }

pnniScopeLocalNetworkPlusOne OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value localNetworkPlusOne(2)."
        DEFVAL { 96 }
        ::= { pnniScopeMappingEntry 2 }

pnniScopeLocalNetworkPlusTwo OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value localNetworkPlusTwo(3)."
        DEFVAL { 96 }
        ::= { pnniScopeMappingEntry 3 }

pnniScopeSiteMinusOne OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
```

```
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy  (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value siteMinusOne(4)."
        DEFVAL { 80 }
        ::= { pnniScopeMappingEntry 4 }

pnniScopeIntraSite OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value intraSite(5)."
        DEFVAL { 80 }
        ::= { pnniScopeMappingEntry 5 }

pnniScopeSitePlusOne OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value sitePlusOne(6)."
        DEFVAL { 72 }
        ::= { pnniScopeMappingEntry 6 }

pnniScopeOrganizationMinusOne OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value organizationMinusOne(7)."
        DEFVAL { 72 }
        ::= { pnniScopeMappingEntry 7 }

pnniScopeIntraOrganization OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value intraOrganization(8)."
        DEFVAL { 64 }
        ::= { pnniScopeMappingEntry 8 }

pnniScopeOrganizationPlusOne OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value organizationPlusOne(9)."
        DEFVAL { 64 }
        ::= { pnniScopeMappingEntry 9 }

pnniScopeCommunityMinusOne OBJECT-TYPE
```

```
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value communityMinusOne(10)."
        DEFVAL { 64 }
        ::= { pnniScopeMappingEntry 10 }

pnniScopeIntraCommunity OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value intraCommunity(11)."
        DEFVAL { 48 }
        ::= { pnniScopeMappingEntry 11 }

pnniScopeCommunityPlusOne OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value communityPlusOne(12)."
        DEFVAL { 48 }
        ::= { pnniScopeMappingEntry 12 }

pnniScopeRegional OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value regional(13)."
        DEFVAL { 32 }
        ::= { pnniScopeMappingEntry 13 }

pnniScopeInterRegional OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value interRegional(14)."
        DEFVAL { 32 }
        ::= { pnniScopeMappingEntry 14 }

pnniScopeGlobal OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The highest level of PNNI hierarchy (i.e. smallest PNNI
             routing level) that lies within the organizational scope
             value global(15)."
        DEFVAL { 0 }
        ::= { pnniScopeMappingEntry 15 }
```

```
-- Summary advertising table

pnniSummaryTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniSummaryEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A list of the summary address prefixes that may be
             advertised by the specified logical PNNI entity."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.9.2"
        ::= { pnniMIBObjects 7 }
pnniSummaryEntry OBJECT-TYPE
        SYNTAX        PnniSummaryEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing summary address prefix
             information in this switching system."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.9.2"
        INDEX         { pnniNodeIndex,
                        pnniSummaryAddress,
                        pnniSummaryPrefixLength }
        ::= { pnniSummaryTable 1 }

PnniSummaryEntry ::=
        SEQUENCE {
                pnniSummaryAddress              AtmAddrPrefix,
                pnniSummaryPrefixLength         PnniPrefixLength,
                pnniSummaryType                 INTEGER,
                pnniSummarySuppress             TruthValue,
                pnniSummaryState                INTEGER,
                pnniSummaryRowStatus            RowStatus
        }

pnniSummaryAddress OBJECT-TYPE
        SYNTAX        AtmAddrPrefix
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The ATM End System Address prefix for the summary."
        ::= { pnniSummaryEntry 1 }

pnniSummaryPrefixLength OBJECT-TYPE
        SYNTAX        PnniPrefixLength
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The prefix length for the summary."
        ::= { pnniSummaryEntry 2 }

pnniSummaryType OBJECT-TYPE
        SYNTAX        INTEGER { internal(1), exterior(2) }
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The type (e.g. internal or exterior) of summary being
                        described."
        DEFVAL { internal }
        ::= { pnniSummaryEntry 3 }
```

```
pnniSummarySuppress OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Determines what is done with addresses that are being
             summarized by the instance. The default value (e.g. false)
             will indicate that the summary should propagate into the
             peer group. Network Management will be able to set the
             value of this attribute to `suppress' (e.g. true), which
             suppresses the summary and any reachable addresses it
             summarizes from being advertised into the peer group."
        DEFVAL { false }
        ::= { pnniSummaryEntry 4 }

pnniSummaryState OBJECT-TYPE
        SYNTAX        INTEGER {
                                advertising(1),
                                suppressing(2),
                                inactive(3)
                                }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the summary is currently being advertised
             by the node within the local switching system into its peer
             group."
        ::= { pnniSummaryEntry 5 }

pnniSummaryRowStatus OBJECT-TYPE
        SYNTAX        RowStatus
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "To create, delete, activate and de-activate a summary."
        ::= { pnniSummaryEntry 6 }


-- Interface table

pnniIfTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniIfEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The pnniIfTable contains the attributes necessary to
             configure a physical interface on a switching system which
             is capable of being used for PNNI routing.  Interfaces may
             represent physical connection points (i.e. copper/fiber
             connection points) or VPCs which have been configured for
             PNNI's use.  Each interface is attached to a specific
             lowest-level node within the switching system.

             An ifIndex is used as the instance ID to uniquely identify
             the interface on the local switching system. This index has
             the same value as the ifIndex object defined in RFC 1573
             for the same interface, since this table correlates with
             the ifTable in RFC 1573.

             One row in this table is created by the managed system for
             each row in the ifTable that has an ifType of atm(37) or
             atmLogical(80)."
        ::= { pnniMIBObjects 8 }
```

```
pnniIfEntry OBJECT-TYPE
        SYNTAX         PnniIfEntry
        MAX-ACCESS     not-accessible
        STATUS         current
        DESCRIPTION
            "An entry in the table, containing PNNI specific interface
             information in this switching system."
        INDEX          { ifIndex }
        ::= { pnniIfTable 1 }

PnniIfEntry ::=
        SEQUENCE {
                pnniIfNodeIndex         PnniNodeIndex,
                pnniIfPortId            PnniPortId,
                pnniIfAggrToken         PnniAggrToken,
                pnniIfVPCapability      TruthValue,
                pnniIfAdmWeightCbr      Unsigned32,
                pnniIfAdmWeightRtVbr    Unsigned32,
                pnniIfAdmWeightNrtVbr   Unsigned32,
                pnniIfAdmWeightAbr      Unsigned32,
                pnniIfAdmWeightUbr      Unsigned32,
                pnniIfRccServiceCategory    ServiceCategory,
                pnniIfRccTrafficDescrIndex  AtmTrafficDescrParamIndex
                }

pnniIfNodeIndex OBJECT-TYPE
        SYNTAX         PnniNodeIndex
        MAX-ACCESS     read-write
        STATUS         current
        DESCRIPTION
            "Identifies the node within the switching system that the
             interface is directly attached to."
        DEFVAL { 1 }
        ::= { pnniIfEntry 1 }

pnniIfPortId OBJECT-TYPE
        SYNTAX         PnniPortId
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
            "The Port Identifier of the port as selected by the PNNI
             protocol entity for the given interface.  This value has
             meaning only within the context of the node to which the
             port is attached."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.4"
        ::= { pnniIfEntry 2 }

pnniIfAggrToken OBJECT-TYPE
        SYNTAX         PnniAggrToken
        MAX-ACCESS     read-write
        STATUS         current
        DESCRIPTION
            "The configured aggregation token for this interface.  The
             aggregation token controls what other links the link
             associated with this interface will be aggregated together
             with."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.3.5, 5.10.3.1"
        DEFVAL { 0 }
        ::= { pnniIfEntry 3 }

pnniIfVPCapability OBJECT-TYPE
        SYNTAX         TruthValue
```

```
        MAX-ACCESS    read-write
        STATUS        current
        DESCRIPTION
            "Indicates whether the interface is capable of having VPCs
             established within it or not.

             This object may only have the value `true' for physical ATM
             interfaces, i.e. those with an ifType of atm(37)."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.14.9.1 Table 5-34"
        ::= { pnniIfEntry 4 }

pnniIfAdmWeightCbr OBJECT-TYPE
        SYNTAX        Unsigned32 (1..MAX)
        MAX-ACCESS    read-write
        STATUS        current
        DESCRIPTION
            "The administrative weight of this interface for the
             constant bit rate service category."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.4"
        DEFVAL { 5040 }
        ::= { pnniIfEntry 5 }

pnniIfAdmWeightRtVbr OBJECT-TYPE
        SYNTAX        Unsigned32 (1..MAX)
        MAX-ACCESS    read-write
        STATUS        current
        DESCRIPTION
            "The administrative weight of this interface for the
             real-time variable bit rate service category."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.4"
        DEFVAL { 5040 }
        ::= { pnniIfEntry 6 }

pnniIfAdmWeightNrtVbr OBJECT-TYPE
        SYNTAX        Unsigned32 (1..MAX)
        MAX-ACCESS    read-write
        STATUS        current
        DESCRIPTION
            "The administrative weight of this interface for the
             non-real-time variable bit rate service category."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.4"
        DEFVAL { 5040 }
        ::= { pnniIfEntry 7 }

pnniIfAdmWeightAbr OBJECT-TYPE
        SYNTAX        Unsigned32 (1..MAX)
        MAX-ACCESS    read-write
        STATUS        current
        DESCRIPTION
            "The administrative weight of this interface for the
             available bit rate service category."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.4"
        DEFVAL { 5040 }
        ::= { pnniIfEntry 8 }

pnniIfAdmWeightUbr OBJECT-TYPE
        SYNTAX        Unsigned32 (1..MAX)
        MAX-ACCESS    read-write
        STATUS        current
```

```
        DESCRIPTION
            "The administrative weight of this interface for the
             unspecified bit rate service category."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.4"
        DEFVAL { 5040 }
        ::= { pnniIfEntry 9 }

pnniIfRccServiceCategory OBJECT-TYPE
        SYNTAX        ServiceCategory
        MAX-ACCESS    read-write
        STATUS        current
        DESCRIPTION
            "The service category used for the PNNI routing control
             channel (VCI=18) on this interface."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.5.2, 5.5.3"
        ::= { pnniIfEntry 10 }

pnniIfRccTrafficDescrIndex OBJECT-TYPE
        SYNTAX        AtmTrafficDescrParamIndex
        MAX-ACCESS    read-write
        STATUS        current
        DESCRIPTION
            "The traffic descriptor index referring to the entry in the
             atmTrafficDescrParamTable defined in RFC 1695 that
             specifies the traffic allocation for the PNNI routing
             control channel (VCI=18) on this interface."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.5.2, 5.5.3, Annex G
             RCCMaximumBurstSize, RCCPeakCellRate,
             RCCSustainableCellRate"
        ::= { pnniIfEntry 11 }


-- link table

pnniLinkTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniLinkEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "This table contains the attributes necessary to describe
             the operation of logical links attached to the local
             switching system and the relationship with the neighbor
             nodes on the other end of the links.  Links are attached to
             a specific node within the switching system.  A
             concatenation of the Node Index of the node within the
             local switching system and the port ID are used as the
             instance ID to uniquely identify the link.  Links may
             represent horizontal links between lowest level neighboring
             peers, outside links, uplinks, or horizontal links to/from
             LGNs.

             The entire pnniLink object is read-only, reflecting the
             fact that this information is discovered dynamically by the
             PNNI protocol rather than configured."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.6"
        ::= { pnniMIBObjects 9 }

pnniLinkEntry OBJECT-TYPE
        SYNTAX        PnniLinkEntry
        MAX-ACCESS    not-accessible
```

```
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about a link
             attached to a PNNI logical node in this switching system."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.6"
        INDEX         { pnniNodeIndex,
                        pnniLinkPortId }
        ::= { pnniLinkTable 1 }

PnniLinkEntry ::=
        SEQUENCE {
                pnniLinkPortId          PnniPortId,
                pnniLinkType            INTEGER,
                pnniLinkVersion         PnniVersion,
                pnniLinkHelloState      PnniHelloState,
                pnniLinkRemoteNodeId    PnniNodeId,
                pnniLinkRemotePortId    PnniPortId,
                pnniLinkDerivedAggrToken        PnniAggrToken,
                pnniLinkUpnodeId        PnniNodeId,
                pnniLinkUpnodeAtmAddress        PnniAtmAddr,
                pnniLinkCommonPeerGroupId       PnniPeerGroupId,
                pnniLinkIfIndex         InterfaceIndex,
                pnniLinkSvccRccIndex    PnniSvccRccIndex,
                pnniLinkRcvHellos       Counter32,
                pnniLinkXmtHellos       Counter32
                }

pnniLinkPortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The Port Identifier of the link as selected by the local
             node.  This value has meaning only within the context of
             the node to which the port is attached."
        ::= { pnniLinkEntry 1 }

pnniLinkType OBJECT-TYPE
        SYNTAX        INTEGER {
                          unknown(1),
                          lowestLevelHorizontalLink(2),
                          horizontalLinkToFromLgn(3),
                          lowestLevelOutsideLink(4),
                          uplink(5),
                          outsideLinkAndUplink(6)
                          }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates the type of link being described."
        ::= { pnniLinkEntry 2 }

pnniLinkVersion OBJECT-TYPE
        SYNTAX        PnniVersion
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For horizontal and outside links between lowest-level nodes
             and for links of unknown type, this attribute indicates the
             version of PNNI routing protocol used to exchange
             information over this link.  If communication with the
             neighbor node has not yet been established, then the
             Version is set to `unknown'.  For uplinks (where the
```

```
             port ID is not also used for the underlying outside link)
             or links to/from LGNs, the Version is set to `unknown'."
        ::= { pnniLinkEntry 3 }

pnniLinkHelloState OBJECT-TYPE
        SYNTAX        PnniHelloState
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For horizontal and outside links between lowest-level nodes
             and for links of unknown type, this attribute indicates the
             state of the Hello protocol exchange over this link.  For
             links to/from LGNs, this attribute indicates the state of
             the corresponding LGN Horizontal Link Hello State Machine.
             For uplinks (where the port ID is not also used for the
             underlying outside link), this attribute is set to
             notApplicable."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.6.2.1.2"
        ::= { pnniLinkEntry 4 }

pnniLinkRemoteNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates the node identifier of the remote (neighboring)
             node on the other end of the link.  If the pnniLinkType is
             `outside link and uplink', this is the node identifier of
             the lowest-level neighbor node on the other end of the
             outside link.  If the remote node ID is unknown or if the
             pnniLinkType is `uplink', this attribute is set to all
             zeros."
        ::= { pnniLinkEntry 5 }

pnniLinkRemotePortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates the port identifier of the port at the remote end
             of the link as assigned by the remote node.  If the
             pnniLinkType is `outside link and uplink', this is the port
             identifier assigned by the lowest-level neighbor node to
             identify the outside link.  If the remote port ID is
             unknown or if the pnniLinkType is `uplink', this attribute
             is set to zero."
        ::= { pnniLinkEntry 6 }

pnniLinkDerivedAggrToken OBJECT-TYPE
        SYNTAX        PnniAggrToken
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates the derived aggregation token value used on this
             link.  For horizontal links between lowest-level nodes and
             when the link type is not yet known, this attribute takes
             the value of zero."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.10.3.1"
        ::= { pnniLinkEntry 7 }

pnniLinkUpnodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
```

```
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For outside links and uplinks, this attribute contains the
             Node Identifier of the upnode (the neighbor node's identity
             at the level of the common peer group).  When the upnode
             has not yet been identified, this attribute is set to zero.
             For horizontal links or when the link type is not yet
             known, this attribute is set to zero."
        ::= { pnniLinkEntry 8 }

pnniLinkUpnodeAtmAddress OBJECT-TYPE
        SYNTAX        PnniAtmAddr
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For outside links and uplinks, this attribute contains the
             ATM End System Address used to establish connections to the
             upnode.  When the upnode has not yet been identified, this
             attribute is set to zero.  For horizontal links or when the
             link type is not yet known, this attribute is set to zero."
        ::= { pnniLinkEntry 9 }

pnniLinkCommonPeerGroupId OBJECT-TYPE
        SYNTAX        PnniPeerGroupId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For outside links and uplinks, this attribute contains the
             peer group identifier of the lowest level common Peer Group
             in the ancestry of the neighboring node and the node within
             the local switching system.  The value of this attribute
             takes on a value determined by the Hello exchange of
             hierarchical information that occurs between the two
             lowest-level border nodes.  When the common peer group has
             not yet been identified, this attribute is set to zero.
             For horizontal links or when the link type is not yet
             known, this attribute is set to all zeros."
        ::= { pnniLinkEntry 10 }

pnniLinkIfIndex OBJECT-TYPE
        SYNTAX        InterfaceIndex
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For horizontal and outside links between lowest-level nodes
             and for links of unknown type, this attribute identifies
             the interface to which the logical link corresponds.

             For all other cases, the value of this object is zero."
        ::= { pnniLinkEntry 11 }

pnniLinkSvccRccIndex OBJECT-TYPE
        SYNTAX        PnniSvccRccIndex
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For horizontal links to/from LGNs, this attribute
             identifies the SVCC-based RCC used to exchange information
             with the neighboring peer logical group node.  If the
             pnniLinkType is not `horizontal link to/from LGN', this
             attribute shall take the value of zero."
        ::= { pnniLinkEntry 12 }
```

```
pnniLinkRcvHellos OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For horizontal and outside links between lowest-level nodes
             and for links of unknown type, this attribute contains a
             count of the number of Hello Packets received over this
             link.  If the pnniLinkType is `horizontal link to/from LGN'
             or `uplink', this attribute is set to zero."
        ::= { pnniLinkEntry 13 }

pnniLinkXmtHellos OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For horizontal and outside links between lowest-level nodes
             and for links of unknown type, this attribute contains a
             count of the number of Hello Packets transmitted over this
             link.  If the pnniLinkType is `horizontal link to/from LGN'
             or `uplink', this attribute is set to zero."
        ::= { pnniLinkEntry 14 }


-- neighboring peer table

pnniNbrPeerTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniNbrPeerEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The pnniNbrPeer Object contains all the attributes
             necessary to describe the relationship a node in this
             switching system has with a neighboring node within the
             same peer group.  A concatenation of the Node Identifier of
             the node within the local switching system and the
             neighboring peer's Node Identifier is used to form the
             instance ID for this object.

             The entire pnniNbrPeer object is read-only, reflecting the
             fact that neighboring peers are discovered dynamically by
             the PNNI protocol rather than configured."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.7, 5.8"
        ::= { pnniMIBObjects 10 }

pnniNbrPeerEntry OBJECT-TYPE
        SYNTAX        PnniNbrPeerEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about this
             node's relationship with a neighboring peer node."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.7, 5.8"
        INDEX         { pnniNodeIndex,
                        pnniNbrPeerRemoteNodeId }
        ::= { pnniNbrPeerTable 1 }

PnniNbrPeerEntry ::=
        SEQUENCE {
                pnniNbrPeerRemoteNodeId          PnniNodeId,
                pnniNbrPeerState                 INTEGER,
```

```
                pnniNbrPeerSvccRccIndex         PnniSvccRccIndex,
                pnniNbrPeerPortCount            Gauge32,
                pnniNbrPeerRcvDbSums            Counter32,
                pnniNbrPeerXmtDbSums            Counter32,
                pnniNbrPeerRcvPtsps             Counter32,
                pnniNbrPeerXmtPtsps             Counter32,
                pnniNbrPeerRcvPtseReqs          Counter32,
                pnniNbrPeerXmtPtseReqs          Counter32,
                pnniNbrPeerRcvPtseAcks          Counter32,
                pnniNbrPeerXmtPtseAcks          Counter32
                }

pnniNbrPeerRemoteNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The Node Identifier of the neighboring peer node."
        ::= { pnniNbrPeerEntry 1 }

pnniNbrPeerState OBJECT-TYPE
        SYNTAX        INTEGER {
                                npdown(1),
                                negotiating(2),
                                exchanging(3),
                                loading(4),
                                full(5)
                                }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates the state of this node's Neighboring Peer State
             Machine associated with pnniNbrPeerRemoteNodeId."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.7.2"
        ::= { pnniNbrPeerEntry 2 }

pnniNbrPeerSvccRccIndex OBJECT-TYPE
        SYNTAX        PnniSvccRccIndex
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Identifies the SVCC-based RCC being used to communicate
             with the neighboring peer if one exists.  If both the local
             node and the neighboring peer node are lowest-level nodes,
             this attribute is set to zero."
        ::= { pnniNbrPeerEntry 3 }

pnniNbrPeerPortCount OBJECT-TYPE
        SYNTAX        Gauge32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the total number of ports that connect to the
             neighboring peer.  If the neighboring peer only
             communicates via an SVCC-based RCC, the value of this
             attribute is set to zero.  Otherwise it is set to the total
             number of ports to the neighboring peer in the Hello state
             2-WayInside."
        ::= { pnniNbrPeerEntry 4 }

pnniNbrPeerRcvDbSums OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
```

```
        STATUS        current
        DESCRIPTION
            "A count of the number of Database Summary Packets received
             from the neighboring peer."
        ::= { pnniNbrPeerEntry 5 }

pnniNbrPeerXmtDbSums OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the number of Database Summary Packets
             transmitted to the neighboring peer."
        ::= { pnniNbrPeerEntry 6 }

pnniNbrPeerRcvPtsps OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the number of PTSPs received from the
             neighboring peer."
        ::= { pnniNbrPeerEntry 7 }

pnniNbrPeerXmtPtsps OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the number of PTSPs (re)transmitted to the
             neighboring peer."
        ::= { pnniNbrPeerEntry 8 }

pnniNbrPeerRcvPtseReqs OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the number of PTSE Request packets received from
             the neighboring peer."
        ::= { pnniNbrPeerEntry 9 }

pnniNbrPeerXmtPtseReqs OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the number of PTSE Request packets transmitted
             to the neighboring peer."
        ::= { pnniNbrPeerEntry 10 }

pnniNbrPeerRcvPtseAcks OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the number of PTSE Ack packets received from the
             neighboring peer."
        ::= { pnniNbrPeerEntry 11 }

pnniNbrPeerXmtPtseAcks OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
```

```
        DESCRIPTION
            "A count of the number of PTSE Ack packets transmitted to
             the neighboring peer."
        ::= { pnniNbrPeerEntry 12 }


-- neighboring peer port table

pnniNbrPeerPortTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniNbrPeerPortEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A table of all ports in Hello state 2-Way Inside to a given
             neighboring peer node.  A concatenation of the Node Index
             of the node within the local switching system, the
             neighbor's Node Identifier and the Interface Index of the
             port being described forms the instance ID for this object.
             This object is only used for lowest-level nodes."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.7.1 Port ID List"
        ::= { pnniMIBObjects 11 }

pnniNbrPeerPortEntry OBJECT-TYPE
        SYNTAX        PnniNbrPeerPortEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about a port
             in the Hello state 2-Way Inside from a PNNI logical node in
             this switching system to a neighboring peer node."
        INDEX         { pnniNodeIndex,
                        pnniNbrPeerRemoteNodeId,
                        pnniNbrPeerPortId
                         }
        ::= { pnniNbrPeerPortTable 1 }

PnniNbrPeerPortEntry ::=
        SEQUENCE {
                pnniNbrPeerPortId               PnniPortId,
                pnniNbrPeerPortFloodStatus      TruthValue
               }

pnniNbrPeerPortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The port ID of a port to the neighboring peer that is in
             the Hello state 2-Way Inside."
        ::= { pnniNbrPeerPortEntry 1 }

pnniNbrPeerPortFloodStatus OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the port is being used for transmission
             of flooding and database synchronization information to the
             neighboring peer."
        ::= { pnniNbrPeerPortEntry 2 }


-- pnni SVCC-based routing control channel table
```

```
pnniSvccRccTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniSvccRccEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A table containing the attributes necessary to analyze the
             operation of the PNNI protocol on SVCC-based Routing
             Control Channels. This entire object is read-only,
             reflecting the fact that SVCC-based RCCs are established
             dynamically during operation of the PNNI protocol rather
             than configured."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.5.6, 5.6.3.1"
        ::= { pnniMIBObjects 12 }

pnniSvccRccEntry OBJECT-TYPE
        SYNTAX        PnniSvccRccEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about an
             SVCC-based RCC from a PNNI logical node in this switching
             system."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.5.6, 5.6.3.1"
        INDEX         { pnniNodeIndex,
                        pnniSvccRccIndex }
        ::= { pnniSvccRccTable 1 }

PnniSvccRccEntry ::=
        SEQUENCE {
                pnniSvccRccIndex            PnniSvccRccIndex,
                pnniSvccRccVersion          PnniVersion,
                pnniSvccRccHelloState       PnniHelloState,
                pnniSvccRccRemoteNodeId     PnniNodeId,
                pnniSvccRccRemoteAtmAddress     PnniAtmAddr,
                pnniSvccRccRcvHellos        Counter32,
                pnniSvccRccXmtHellos        Counter32,
                pnniSvccRccIfIndex          InterfaceIndex,
                pnniSvccRccVpi              INTEGER,
                pnniSvccRccVci              INTEGER
                }

pnniSvccRccIndex OBJECT-TYPE
        SYNTAX        PnniSvccRccIndex
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An index into the node's tables of SVCC-based RCCs."
        ::= { pnniSvccRccEntry 1 }

pnniSvccRccVersion OBJECT-TYPE
        SYNTAX        PnniVersion
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The version of the PNNI routing protocol used to exchange
             information with the neighbor node."
        ::= { pnniSvccRccEntry 2 }

pnniSvccRccHelloState OBJECT-TYPE
        SYNTAX        PnniHelloState
        MAX-ACCESS    read-only
```

```
        STATUS        current
        DESCRIPTION
            "The state of the Hello protocol exchange over the
             SVCC-based RCC.

             Note: the Down state indicates that the SVCC
             establishment is in progress."
        ::= { pnniSvccRccEntry 3 }

pnniSvccRccRemoteNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The remote node at which the SVCC-based RCC terminates."
        ::= { pnniSvccRccEntry 4 }

pnniSvccRccRemoteAtmAddress OBJECT-TYPE
        SYNTAX        PnniAtmAddr
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The ATM End System Address to which SVCC establishment is
             attempted."
        ::= { pnniSvccRccEntry 5 }

pnniSvccRccRcvHellos OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the number of Hello Packets received over this
             SVCC-based RCC."
        ::= { pnniSvccRccEntry 6 }

pnniSvccRccXmtHellos OBJECT-TYPE
        SYNTAX        Counter32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "A count of the number of Hello Packets transmitted over
             this SVCC-based RCC."
        ::= { pnniSvccRccEntry 7 }

pnniSvccRccIfIndex OBJECT-TYPE
        SYNTAX        InterfaceIndex
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The interface from which the SVCC-based RCC leaves the
             switching system.  If the SVCC-based RCC has not yet been
             established, then this attribute takes the value of zero."
        ::= { pnniSvccRccEntry 8 }

pnniSvccRccVpi OBJECT-TYPE
        SYNTAX        INTEGER (0..4095)
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The VPI used at the interface from which the SVCC-based RCC
             leaves the switching system.  If the SVCC-based RCC has not
             yet been established, then this attribute takes the value
             of zero "
        ::= { pnniSvccRccEntry 9 }
```

```
pnniSvccRccVci OBJECT-TYPE
        SYNTAX        INTEGER (0..65535)
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The VCI used at the interface from which the SVCC-based RCC
             leaves the switching system.  If the SVCC-based RCC has not
             yet been established, then this attribute takes the value
             of zero "
        ::= { pnniSvccRccEntry 10 }


-- PTSE table

pnniPtseTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniPtseEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The pnniPtse object contains the attributes that describe
             the most recent instances of PTSEs in a node's topology
             database.  A concatenation of the Node Identifier of the
             local node that received the PTSE, the originating Node's
             Node Identifier and the PTSE Identifier are used to form
             the instance ID for an instance of this object."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.2"
        ::= { pnniMIBObjects 13 }

pnniPtseEntry OBJECT-TYPE
        SYNTAX        PnniPtseEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about a PTSE
             in the topology database of a PNNI logical node in this
             switching system."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.2"
        INDEX         { pnniNodeIndex,
                        pnniPtseOriginatingNodeId,
                        pnniPtseId }
        ::= { pnniPtseTable 1 }

PnniPtseEntry ::=
        SEQUENCE {
                pnniPtseOriginatingNodeId       PnniNodeId,
                pnniPtseId                      Unsigned32,
                pnniPtseType                    INTEGER,
                pnniPtseSequenceNum             Unsigned32,
                pnniPtseChecksum                Unsigned32,
                pnniPtseLifeTime                Unsigned32,
                pnniPtseInfo                    OCTET STRING
                }

pnniPtseOriginatingNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The Node Identifier of the node that originated the PTSE."
        ::= { pnniPtseEntry 1 }
```

```
pnniPtseId OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The value of the PTSE Identifier assigned to the PTSE by
             its originator."
        ::= { pnniPtseEntry 2 }

pnniPtseType OBJECT-TYPE
        SYNTAX        INTEGER  {
                                other(1),
                                nodalStateParameters(96),
                                nodalInformation(97),
                                internalReachableAddresses(224),
                                exteriorReachableAddresses(256),
                                horizontalLinks(288),
                                uplinks(289)
                                }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The type of information contained in the PTSE."
        ::= { pnniPtseEntry 3 }

pnniPtseSequenceNum OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The sequence number of the instance of the PTSE as it
             appears in the local topology database."
        ::= { pnniPtseEntry 4 }

pnniPtseChecksum OBJECT-TYPE
        SYNTAX        Unsigned32 (0..65535)
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The value of the PTSE checksum as it appears in the local
             topology database."
        ::= { pnniPtseEntry 5 }

pnniPtseLifeTime OBJECT-TYPE
        SYNTAX        Unsigned32 (0..65535)
        UNITS         "seconds"
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The value of the remaining lifetime for the given PTSE as
             it appears in the local topology database."
        ::= { pnniPtseEntry 6 }

pnniPtseInfo OBJECT-TYPE
        SYNTAX        OCTET STRING (SIZE(0..65535))
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "An unformatted hexadecimal dump of the PTSE contents in
             full.

             Note: If the size of the PTSE contents is larger than the
             maximum size of SNMP packets then this is truncated."
        ::= { pnniPtseEntry 7 }
```

```
-- pnni map table

pnniMapTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniMapEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A table containing attributes necessary to find and analyze
             the operation of all links and nodes within the PNNI
             hierarchy, as seen from the perspective of a local node.
             An instance of a pnniMap Object describes a link in terms
             of a node at one end of the link.  Normally there will be
             two instances of the pnniMap object in the MIB for each
             horizontal link. The two instances provide information for
             Network management to map port identifiers from the nodes
             at both ends to the link between them.  A concatenation of
             the Local Node Index, Originating Node Identifier and
             Originating Port Identifier are used to form the instance
             ID for this object.

             This entire object is read-only, reflecting the fact that
             the map is discovered dynamically during operation of the
             PNNI protocol rather than configured."
        ::= { pnniMIBObjects 14 }

pnniMapEntry OBJECT-TYPE
        SYNTAX        PnniMapEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing connectivity information
             about a node or link in the PNNI routing domain, as seen
             from the perspective of a PNNI logical node in this
             switching system."
        INDEX         { pnniNodeIndex,
                        pnniMapOriginatingNodeId,
                        pnniMapOriginatingPortId,
                        pnniMapIndex }
        ::= { pnniMapTable 1 }

PnniMapEntry ::=
        SEQUENCE {
                pnniMapOriginatingNodeId        PnniNodeId,
                pnniMapOriginatingPortId        PnniPortId,
                pnniMapIndex                    INTEGER,
                pnniMapType                     INTEGER,
                pnniMapPeerGroupId              PnniPeerGroupId,
                pnniMapAggrToken                PnniAggrToken,
                pnniMapRemoteNodeId             PnniNodeId,
                pnniMapRemotePortId             PnniPortId,
                pnniMapVPCapability             TruthValue,
                pnniMapPtseId                   Unsigned32,
                pnniMapMetricsTag               PnniMetricsTag
                }

pnniMapOriginatingNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The node identifier of the node whose connectivity within
             itself or to other nodes is being described."
```

```
        ::= { pnniMapEntry 1 }

pnniMapOriginatingPortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The port identifier of the port as assigned by the
             originating node, to which the port is attached."
        ::= { pnniMapEntry 2 }

pnniMapIndex OBJECT-TYPE
        SYNTAX        INTEGER (0..65535)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An index into the set of link and nodal connectivity
             associated with the originating node and port.  This index
             is needed since there may be multiple entries for nodal
             connectivity from a specific node and port pair, in
             addition to any entry for a horizontal link or uplink."
        ::= { pnniMapEntry 3 }

pnniMapType OBJECT-TYPE
        SYNTAX        INTEGER {
                                horizontalLink(1),
                                uplink(2),
                                node(3)
                                }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The type of PNNI entity being described by this entry in
             the table."
        ::= { pnniMapEntry 4 }

pnniMapPeerGroupId OBJECT-TYPE
        SYNTAX        PnniPeerGroupId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Identifies the peer group of the originating node."
        ::= { pnniMapEntry 5 }

pnniMapAggrToken OBJECT-TYPE
        SYNTAX        PnniAggrToken
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For horizontal links to/from LGNs and for uplinks, this
             attribute contains the derived aggregation token value for
             this link.  For nodes and for horizontal links between
             lowest-level nodes, this attribute is set to zero."
        ::= { pnniMapEntry 6 }

pnniMapRemoteNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For horizontal links and uplinks, this attribute contains
             the node identifier of the node at the other end of the
             link from the originating node.  If unknown, the PNNI
             protocol entity sets this attribute's value to (all)
```

```
             zero(s).  For nodes, this attribute's value is set to (all)
             zero(s)."
        ::= { pnniMapEntry 7 }

pnniMapRemotePortId OBJECT-TYPE
        SYNTAX         PnniPortId
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
            "For horizontal links and uplinks, this attribute contains
             the port identifier of the port at the remote end of the
             link as assigned by the remote node.  If unknown, the PNNI
             protocol entity sets this attribute's value to zero.

             For nodes, this attribute contains the port identifier of
             the port at the other end of the spoke or bypass from the
             originating port.  When the originating port ID is zero, a
             value of zero indicates the default radius.  When the
             originating port ID is non-zero, a value of zero indicates
             the nodal nucleus."
        ::= { pnniMapEntry 8 }

pnniMapVPCapability OBJECT-TYPE
        SYNTAX         TruthValue
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
            "Indicates whether VPCs
             can be established across the PNNI entity being described
             by this entry in the pnniMapTable."
        ::= { pnniMapEntry 9 }

pnniMapPtseId OBJECT-TYPE
        SYNTAX         Unsigned32
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
            "The value of the PTSE Identifier for the PTSE being
             originated by the originating node which contains the
             information group(s) describing the PNNI entity."
        ::= { pnniMapEntry 10 }

pnniMapMetricsTag OBJECT-TYPE
        SYNTAX         PnniMetricsTag (1..MAX)
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
            "An arbitrary integer that is used to associate a set of
             traffic parameters that are always advertised together.
             Within this set, the parameters are distinguished by the
             service categories and direction to which a set of
             parameters apply."
        ::= { pnniMapEntry 11 }


-- nodal map table

pnniMapNodeTable OBJECT-TYPE
        SYNTAX         SEQUENCE OF PnniMapNodeEntry
        MAX-ACCESS     not-accessible
        STATUS         current
        DESCRIPTION
            "A list of nodes as seen from the perspective of a local
             node.  The pnniMapNodeTable contains all information
```

```
             learned by the local node from nodal information PTSEs.
             This entire object is read-only, reflecting the fact that
             the map is discovered dynamically during operation of the
             PNNI protocol rather than configured."
        ::= { pnniMIBObjects 15 }

pnniMapNodeEntry OBJECT-TYPE
        SYNTAX         PnniMapNodeEntry
        MAX-ACCESS     not-accessible
        STATUS         current
        DESCRIPTION
            "An entry in the table, containing information about a node
             in the PNNI routing domain, as seen from the perspective of
             a logical node in this switching system."
        INDEX          { pnniNodeIndex,
                         pnniMapNodeId }
        ::= { pnniMapNodeTable 1 }

PnniMapNodeEntry ::=
        SEQUENCE {
                pnniMapNodeId                     PnniNodeId,
                pnniMapNodePeerGroupId            PnniPeerGroupId,
                pnniMapNodeAtmAddress             PnniAtmAddr,
                pnniMapNodeRestrictedTransit      TruthValue,
                pnniMapNodeComplexRep             TruthValue,
                pnniMapNodeRestrictedBranching    TruthValue,
                pnniMapNodeDatabaseOverload       TruthValue,
                pnniMapNodeIAmLeader              TruthValue,
                pnniMapNodeLeadershipPriority     INTEGER,
                pnniMapNodePreferredPgl           PnniNodeId,
                pnniMapNodeParentNodeId           PnniNodeId,
                pnniMapNodeParentAtmAddress       PnniAtmAddr,
                pnniMapNodeParentPeerGroupId      PnniPeerGroupId,
                pnniMapNodeParentPglNodeId        PnniNodeId
                }

pnniMapNodeId OBJECT-TYPE
        SYNTAX         PnniNodeId
        MAX-ACCESS     not-accessible
        STATUS         current
        DESCRIPTION
            "Identifies the node whose nodal information is being
             described."
        ::= { pnniMapNodeEntry 1 }

pnniMapNodePeerGroupId OBJECT-TYPE
        SYNTAX         PnniPeerGroupId
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
            "Identifies the peer group of the originating node."
        ::= { pnniMapNodeEntry 2 }

pnniMapNodeAtmAddress OBJECT-TYPE
        SYNTAX         PnniAtmAddr
        MAX-ACCESS     read-only
        STATUS         current
        DESCRIPTION
            "The ATM End System Address of the originating node."
        ::= { pnniMapNodeEntry 3 }

pnniMapNodeRestrictedTransit OBJECT-TYPE
        SYNTAX         TruthValue
        MAX-ACCESS     read-only
```

```
        STATUS        current
        DESCRIPTION
            "Indicates whether the originating node is restricted to
             only allow support of SVCs originating or terminating at
             this node.  A value of `true' indicates that the transit
             capabilities are restricted, i.e., transit connections are
             not allowed, whereas a value of `false' indicates that
             transit connections are allowed.  This attribute reflects
             the setting of the restricted transit bit received in the
             nodal information PTSE of the originating node."
        ::= { pnniMapNodeEntry 4 }

pnniMapNodeComplexRep OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the originating node uses the complex
             node representation.  If the value is `true', the spokes
             and bypasses that make up the complex node representation
             should be found in the pnniMapTable.  This attribute
             reflects the setting of the nodal representation bit
             received in the nodal information PTSE of the originating
             node."
        ::= { pnniMapNodeEntry 5 }

pnniMapNodeRestrictedBranching  OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Indicates whether the originating node is able to support
             additional branches.  If the value is 'false', then it can
             support additional branches.  This attribute reflects the
             setting of the restricted branching bit received in the
             nodal information PTSE of the originating node."
        ::= { pnniMapNodeEntry 6 }

pnniMapNodeDatabaseOverload OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the originating node is currently
             operating in topology database overload state.  This
             attribute has the same value as the Non-transit for PGL
             Election bit in the nodal information group originated by
             this node."
        ::= { pnniMapNodeEntry 7 }

pnniMapNodeIAmLeader OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the originating node claims to be peer
             group leader of its peer group.  This attribute reflects
             the setting of the 'I am Leader' bit received in the nodal
             information PTSE of the originating node."
        ::= { pnniMapNodeEntry 8 }

pnniMapNodeLeadershipPriority OBJECT-TYPE
        SYNTAX        INTEGER (0..255)
        MAX-ACCESS    read-only
```

```
        STATUS        current
        DESCRIPTION
            "The Leadership priority value advertised by the originating
             node."
        ::= { pnniMapNodeEntry 9 }

pnniMapNodePreferredPgl OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Identifies the node which the originating node believes
             should be or is peer group leader of its peer group.  If
             the originating node has not chosen a Preferred PGL, this
             attribute's value is set to (all) zero(s)."
        ::= { pnniMapNodeEntry 10 }

pnniMapNodeParentNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "When the originating node is a peer group leader, indicates
             the node ID of the parent LGN.  If the originating node is
             not peer group leader of its peer group, this attribute's
             value is set to (all) zero(s)."
        ::= { pnniMapNodeEntry 11 }

pnniMapNodeParentAtmAddress OBJECT-TYPE
        SYNTAX        PnniAtmAddr
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "When the originating node is a peer group leader, indicates
             the ATM address of the parent LGN.  If the originating node
             is not peer group leader of its peer group, this
             attribute's value is set to (all) zero(s)."
        ::= { pnniMapNodeEntry 12 }

pnniMapNodeParentPeerGroupId OBJECT-TYPE
        SYNTAX        PnniPeerGroupId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "When the originating node is a peer group leader, indicates
             the node's parent peer group ID.  If the originating node
             is not peer group leader of its peer group, this
             attribute's value is set to (all) zero(s)."
        ::= { pnniMapNodeEntry 13 }

pnniMapNodeParentPglNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "When the originating node is a peer group leader,
             identifies the node elected as peer group leader of the
             parent peer group.  If the originating node is not peer
             group leader of its peer group, this attribute's value is
             set to (all) zero(s)."
        ::= { pnniMapNodeEntry 14 }


-- address map table
```

```
pnniMapAddrTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniMapAddrEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The pnniMapAddr MIB Object contains a list of all reachable
             addresses from each node visible to the local node.  The
             Local Node Index, Advertising Node ID, Advertised Port ID,
             Reachable Address, and Address prefix length are combined
             to form an instance ID for this object.  The entire object
             is read-only, reflecting the fact that reachable addresses
             are discovered during dynamic operation of the PNNI
             protocol rather than configured."
        ::= { pnniMIBObjects 16 }

pnniMapAddrEntry OBJECT-TYPE
        SYNTAX        PnniMapAddrEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about an
             address prefix reachable from a node in the PNNI routing
             domain, as seen from the perspective of a PNNI logical node
             in this switching system."
        INDEX         { pnniNodeIndex,
                        pnniMapAddrAdvertisingNodeId,
                        pnniMapAddrAdvertisedPortId,
                        pnniMapAddrIndex }
        ::= { pnniMapAddrTable 1 }

PnniMapAddrEntry ::=
        SEQUENCE {
                pnniMapAddrAdvertisingNodeId    PnniNodeId,
                pnniMapAddrAdvertisedPortId     PnniPortId,
                pnniMapAddrIndex                INTEGER,
                pnniMapAddrAddress              AtmAddrPrefix,
                pnniMapAddrPrefixLength         PnniPrefixLength
                }

pnniMapAddrAdvertisingNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The node ID of a node advertising reachability to the
             address prefix."
        ::= { pnniMapAddrEntry 1 }

pnniMapAddrAdvertisedPortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The port identifier used from the advertising node to reach
             the given address prefix."
        ::= { pnniMapAddrEntry 2 }

pnniMapAddrIndex OBJECT-TYPE
        SYNTAX        INTEGER (1..MAX)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An arbitrary index that is used to enumerate all of the
```

```
             addresses advertised by the specified node."
        ::= { pnniMapAddrEntry 3 }

pnniMapAddrAddress OBJECT-TYPE
        SYNTAX        AtmAddrPrefix
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The value of the ATM End System Address prefix."
        ::= { pnniMapAddrEntry 4 }

pnniMapAddrPrefixLength OBJECT-TYPE
        SYNTAX        PnniPrefixLength
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The Prefix length to be applied to the ATM End System
             Address prefix."
        ::= { pnniMapAddrEntry 5 }


-- TNS map table

pnniMapTnsTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniMapTnsEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A list of all reachable transit networks from each node
             visible to the local node.  The Local Node Index,
             Advertising Node ID, Advertised Port ID, Transit Network
             Type, Transit Network Plan, and Transit Network ID are
             combined to form an instance ID for this object. The entire
             object is read-only, reflecting the fact that reachable
             transit networks are discovered during dynamic operation of
             the PNNI protocol rather than configured.."
        ::= { pnniMIBObjects 17 }

pnniMapTnsEntry OBJECT-TYPE
        SYNTAX        PnniMapTnsEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about a
             transit network reachable from a node in the PNNI routing
             domain, as seen from the perspective of a PNNI logical node
             in this switching system."
        INDEX         { pnniNodeIndex,
                        pnniMapTnsAdvertisingNodeId,
                        pnniMapTnsAdvertisedPortId,
                        pnniMapTnsType,
                        pnniMapTnsPlan,
                        pnniMapTnsId }
        ::= { pnniMapTnsTable 1 }

PnniMapTnsEntry ::=
        SEQUENCE {
                pnniMapTnsAdvertisingNodeId     PnniNodeId,
                pnniMapTnsAdvertisedPortId      PnniPortId,
                pnniMapTnsType                  TnsType,
                pnniMapTnsPlan                  TnsPlan,
                pnniMapTnsId                    DisplayString
                }
```

```
pnniMapTnsAdvertisingNodeId OBJECT-TYPE
        SYNTAX          PnniNodeId
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "The node ID of a node advertising reachability to the
             transit network."
        ::= { pnniMapTnsEntry 1 }

pnniMapTnsAdvertisedPortId OBJECT-TYPE
        SYNTAX          PnniPortId
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "The port identifier used from the advertising node to reach
             the given transit network."
        ::= { pnniMapTnsEntry 2 }

pnniMapTnsType OBJECT-TYPE
        SYNTAX          TnsType
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "The type of network identification used for this transit
             network."
        ::= { pnniMapTnsEntry 3 }

pnniMapTnsPlan OBJECT-TYPE
        SYNTAX          TnsPlan
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "The network identification plan according to which network
             identification has been assigned."
        ::= { pnniMapTnsEntry 4 }

pnniMapTnsId OBJECT-TYPE
        SYNTAX          DisplayString
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The value of the transit network identifier."
        ::= { pnniMapTnsEntry 5 }



-- pnni metrics table

pnniMetricsTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF PnniMetricsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "This entity's table of PNNI parameters either associated
             with a PNNI entity or for the connectivity between a PNNI
             node and a reachable address or transit network."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3"
        ::= { pnniMIBObjects 18 }

pnniMetricsEntry OBJECT-TYPE
        SYNTAX          PnniMetricsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
```

```
        DESCRIPTION
            "A set of parameters that applies to the connectivity from a
             certain node and port to another node or port or to one or
             more reachable address prefixes and/or transit networks,
             for one (or more) particular service category(s).  Note
             that there can be multiple sets of parameters with the same
             tag, in which case all sets apply to the specified
             connectivity."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3"
        INDEX         { pnniNodeIndex,
                        pnniMetricsTag,
                        pnniMetricsDirection,
                        pnniMetricsIndex }
        ::= { pnniMetricsTable 1 }

PnniMetricsEntry ::=
        SEQUENCE {
                pnniMetricsTag          PnniMetricsTag,
                pnniMetricsDirection    INTEGER,
                pnniMetricsIndex        Integer32,
                pnniMetricsClasses      INTEGER,
                pnniMetricsGcacClp      ClpType,
                pnniMetricsAdminWeight  Unsigned32,
                pnniMetrics1            Unsigned32,
                pnniMetrics2            Unsigned32,
                pnniMetrics3            Unsigned32,
                pnniMetrics4            Unsigned32,
                pnniMetrics5            Unsigned32,
                pnniMetrics6            Unsigned32,
                pnniMetrics7            Unsigned32,
                pnniMetrics8            Unsigned32,
                pnniMetricsRowStatus    RowStatus
                }


pnniMetricsTag OBJECT-TYPE
        SYNTAX        PnniMetricsTag (1..MAX)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An arbitrary integer that is used to associate a set of
             traffic parameters that are always advertised together.
             Within this set, the parameters are distinguished by the
             service categories and direction to which a set of
             parameters apply."
        ::= { pnniMetricsEntry 1 }


pnniMetricsDirection OBJECT-TYPE
        SYNTAX        INTEGER { incoming(1), outgoing(2) }
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The direction, with respect to the advertising node, in
             which the parameters in this entry apply."
        ::= { pnniMetricsEntry 2 }


pnniMetricsIndex OBJECT-TYPE
        SYNTAX        Integer32 (1..MAX)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
```

```
            "An index into the set of parameters associated with the
             given tag and direction."
        ::= { pnniMetricsEntry 3 }


pnniMetricsClasses OBJECT-TYPE
        SYNTAX         INTEGER(0..31)
        MAX-ACCESS     read-create
        STATUS         current
        DESCRIPTION
            "The service categories to which this set of parameters
             applies.  This is an integer used as a bit mask with each
             bit that is set representing a single service category for
             which the resources indicated are available.  Bit 5
             represents CBR, bit 4 represents real-time VBR, bit 3
             represents non-real-time VBR, bit 2 represents ABR, and
             bit 1 (LSB) represents UBR."
        REFERENCE
            "ATM Forum Traffic Management 4.0 Section 2,
             ATM Forum PNNI 1.0 Section 5.8.1.1.3.1"
        ::= { pnniMetricsEntry 4 }

pnniMetricsGcacClp OBJECT-TYPE
        SYNTAX         ClpType
        MAX-ACCESS     read-create
        STATUS         current
        DESCRIPTION
            "Indicates whether the advertised GCAC parameters apply for
             CLP=0 traffic or for CLP=0+1 traffic."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.8.1.1.3.1, 5.13.4.1"
        ::= { pnniMetricsEntry 5 }


pnniMetricsAdminWeight OBJECT-TYPE
        SYNTAX         Unsigned32  (1..MAX)
        MAX-ACCESS     read-create
        STATUS         current
        DESCRIPTION
            "The administrative weight from the advertising node to the
             remote end of the PNNI entity or to the reachable address
             or transit network, for the specified service categories.

             If this metric is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.4"
        DEFVAL     { 5040 }
        ::= { pnniMetricsEntry 6 }


pnniMetrics1 OBJECT-TYPE
        SYNTAX         Unsigned32
        MAX-ACCESS     read-create
        STATUS         current
        DESCRIPTION
            "An alternate routing parameter from the advertising node to
             the remote end of the PNNI entity or to the reachable
             address or transit network, for the specified service
             categories.

             For information learned from PNNI nodes, this is the
             maximum cell rate in cells per second for the specified
             service categories.
```

```
                  If this parameter is not used, its value should be set to
                  0xFFFFFFFF."
             REFERENCE
                 "ATM Forum PNNI 1.0 Section 5.8.1.1.3.7"
             DEFVAL { 'FFFFFFFF'h }
             ::= { pnniMetricsEntry 7 }


pnniMetrics2 OBJECT-TYPE
        SYNTAX          Unsigned32
        MAX-ACCESS      read-create
        STATUS          current
        DESCRIPTION
            "An alternate routing parameter from the advertising node to
             the remote end of the PNNI entity or to the reachable
             address or transit network, for the specified service
             categories.

             For information learned from PNNI nodes, this is the
             available cell rate in cells per second for the specified
             service categories.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.8"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniMetricsEntry 8 }


pnniMetrics3 OBJECT-TYPE
        SYNTAX          Unsigned32
        MAX-ACCESS      read-create
        STATUS          current
        DESCRIPTION
            "An alternate routing parameter from the advertising node to
             the remote end of the PNNI entity or to the reachable
             address or transit network, for the specified service
             categories.

             For information learned from PNNI nodes, this is the
             maximum cell transfer delay in microseconds for the
             specified service categories.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.3"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniMetricsEntry 9 }


pnniMetrics4 OBJECT-TYPE
        SYNTAX          Unsigned32
        MAX-ACCESS      read-create
        STATUS          current
        DESCRIPTION
            "An alternate routing parameter from the advertising node to
             the remote end of the PNNI entity or to the reachable
             address or transit network, for the specified service
             categories.

             For information learned from PNNI nodes, this is the cell
```

```
             delay variation in microseconds for the specified service
             categories.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.2"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniMetricsEntry 10 }


pnniMetrics5 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter from the advertising node to
             the remote end of the PNNI entity or to the reachable
             address or transit network, for the specified service
             categories.

             For PNNI, this is the cell loss ratio for CLP=0 traffic for
             the specified service categories.  The cell loss ratio
             value is computed as 10**(-n) where 'n' is the value
             returned in this variable.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.5"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniMetricsEntry 11 }

pnniMetrics6 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter from the advertising node to
             the remote end of the PNNI entity or to the reachable
             address or transit network, for the specified service
             categories.

             For PNNI, this is the cell loss ratio for CLP=0+1 traffic
             for the specified service categories.  The cell loss ratio
             value is computed as 10**(-n) where 'n' is the value
             returned in this variable.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.6"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniMetricsEntry 12 }


pnniMetrics7 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter from the advertising node to
             the remote end of the PNNI entity or to the reachable
             address or transit network, for the specified service
```

```
             categories.

             For information learned from PNNI nodes, this is the cell
             rate margin in cells per second for the specified service
             categories.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.9"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniMetricsEntry 13 }


pnniMetrics8 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter from the advertising node to
             the remote end of the PNNI entity or to the reachable
             address or transit network, for the specified service
             categories.

             For information learned from PNNI nodes, this is the
             variance factor in units of 2**(-8) for the specified
             service categories.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.10"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniMetricsEntry 14 }


pnniMetricsRowStatus OBJECT-TYPE
        SYNTAX        RowStatus
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "To create, delete, activate and de-activate a set of
             metrics."
        ::= { pnniMetricsEntry 15 }


--
--  PNNI Routing Tables
--

pnniRoutingGroup OBJECT IDENTIFIER ::= { pnniMIBObjects 19 }
pnniRouteBaseGroup OBJECT IDENTIFIER ::= { pnniRoutingGroup 1 }

pnniRouteNodeNumber OBJECT-TYPE
        SYNTAX        Gauge32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The number of current precalculated PNNI routes to PNNI
             nodes that are not invalid."
        ::= { pnniRouteBaseGroup 1 }


pnniRouteAddrNumber OBJECT-TYPE
```

```
        SYNTAX        Gauge32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The number of current PNNI routes from nodes in the PNNI
             routing domain to addresses and transit networks that are
             not invalid."
        ::= { pnniRouteBaseGroup 2 }


-- Table of routes to other nodes

pnniRouteNodeTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniRouteNodeEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "This entity's PNNI Routing table (of routes to other
             nodes)."
        ::= { pnniRoutingGroup 2 }

pnniRouteNodeEntry OBJECT-TYPE
        SYNTAX        PnniRouteNodeEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A particular route to a particular destination node, under
             a particular policy."
        INDEX         { pnniNodeIndex,
                        pnniRouteNodeClass,
                        pnniRouteNodeDestNodeId,
                        pnniRouteNodeDTL }
        ::= { pnniRouteNodeTable 1 }

PnniRouteNodeEntry ::=
        SEQUENCE {
                pnniRouteNodeClass          ServiceCategory,
                pnniRouteNodeDestNodeId     PnniNodeId,
                pnniRouteNodeDTL            Integer32,
                pnniRouteNodeDestPortId     PnniPortId,
                pnniRouteNodeProto          INTEGER,
                pnniRouteNodeTimeStamp      TimeStamp,
                pnniRouteNodeInfo           OBJECT IDENTIFIER,
                pnniRouteNodeGcacClp        ClpType,
                pnniRouteNodeFwdMetricAW    Unsigned32,
                pnniRouteNodeFwdMetric1     Unsigned32,
                pnniRouteNodeFwdMetric2     Unsigned32,
                pnniRouteNodeFwdMetric3     Unsigned32,
                pnniRouteNodeFwdMetric4     Unsigned32,
                pnniRouteNodeFwdMetric5     Unsigned32,
                pnniRouteNodeFwdMetric6     Unsigned32,
                pnniRouteNodeFwdMetric7     Unsigned32,
                pnniRouteNodeFwdMetric8     Unsigned32,
                pnniRouteNodeBwdMetricAW    Unsigned32,
                pnniRouteNodeBwdMetric1     Unsigned32,
                pnniRouteNodeBwdMetric2     Unsigned32,
                pnniRouteNodeBwdMetric3     Unsigned32,
                pnniRouteNodeBwdMetric4     Unsigned32,
                pnniRouteNodeBwdMetric5     Unsigned32,
                pnniRouteNodeBwdMetric6     Unsigned32,
                pnniRouteNodeBwdMetric7     Unsigned32,
                pnniRouteNodeBwdMetric8     Unsigned32,
                pnniRouteNodeVPCapability   TruthValue,
                pnniRouteNodeStatus         RowStatus
```

```
                }

pnniRouteNodeClass OBJECT-TYPE
        SYNTAX        ServiceCategory
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "Indicates the service category with which this forwarding
             table entry is associated."
        ::= { pnniRouteNodeEntry 1 }

pnniRouteNodeDestNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The node ID of the destination node to which this route
             proceeds, and at which the DTL stack for this route
             terminates."
        ::= { pnniRouteNodeEntry 2 }

pnniRouteNodeDTL OBJECT-TYPE
        SYNTAX        Integer32 (1..MAX)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The index into the owning PNNI node's DTL table of the DTL
             stack that goes with this route."
        ::= { pnniRouteNodeEntry 3 }

pnniRouteNodeDestPortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The port ID of the destination node at which the route
             terminates.  A port ID of zero indicates the node nucleus.
             When the destination node is represented by the simple node
             representation, this value should be set to zero."
        DEFVAL { 0 }
        ::= { pnniRouteNodeEntry 4 }


pnniRouteNodeProto OBJECT-TYPE
        SYNTAX        INTEGER {
                                other(1), -- not specified
                                local(2), -- e.g. ilmi
                                mgmt(3), -- configured by management,
                                       -- for example by SNMP or console
                                        -- the following are all dynamic
                                        -- routing protocols
                                pnni(4) -- ATM Forum PNNI
                                }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The routing mechanism via which this route was learned."
        ::= { pnniRouteNodeEntry 5 }

pnniRouteNodeTimeStamp OBJECT-TYPE
        SYNTAX        TimeStamp
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
```

```
            "The time at which this  route  was last  updated  or
             otherwise  determined  to be correct.  Note that no
             semantics of  `too  old' can  be implied except through
             knowledge of the routing  protocol  by  which  the route
             was learned."
        ::= { pnniRouteNodeEntry 6 }

pnniRouteNodeInfo OBJECT-TYPE
        SYNTAX        OBJECT IDENTIFIER
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "A reference to MIB definitions specific to the particular
             routing protocol which is responsible for this route, as
             determined by the value specified in the route's
             pnniRouteNodeProto value.  If this information is not
             present, its value should be set to the OBJECT IDENTIFIER
             zeroDotZero."
        DEFVAL { zeroDotZero }
        ::= { pnniRouteNodeEntry 7 }

pnniRouteNodeGcacClp OBJECT-TYPE
        SYNTAX        ClpType
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "For PNNI, indicates whether any advertised GCAC parameters
             apply for CLP=0 traffic or for CLP=0+1 traffic."
        ::= { pnniRouteNodeEntry 8 }

pnniRouteNodeFwdMetricAW OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The cumulative administrative weight calculated for the
             forward direction of this route.  If this metric is not
             used, its value should be set to 0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.4"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 9 }

pnniRouteNodeFwdMetric1 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the forward direction of
             this route.

             For information learned from PNNI nodes, this is the
             maximum possible cell rate (in cells per second) for the
             forward direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.7"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 10 }

pnniRouteNodeFwdMetric2 OBJECT-TYPE
        SYNTAX        Unsigned32
```

```
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the forward direction of
             this route.

             For information learned from PNNI nodes, this is the
             Available cell rate (in cells per second) for the forward
             direction of the route.  Further information on available
             bandwidth may be obtainable by reference to the nodal
             advertisements of the nodes in the path.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.8"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 11 }

pnniRouteNodeFwdMetric3 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the forward direction of
             this route.

             For information learned from PNNI nodes, this is the
             cumulative Maximum Cell Transfer Delay (in microseconds)
             for the forward direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.3"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 12 }

pnniRouteNodeFwdMetric4 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the forward direction of
             this route.

             For information learned from PNNI nodes, this is the
             cumulative Cell Delay Variation (in microseconds) for the
             forward direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.2"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 13 }

pnniRouteNodeFwdMetric5 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the forward direction of
             this route.
```

```
             For information learned from PNNI nodes, this is the
             cumulative Cell Loss Ratio for CLP=0 traffic for the
             forward direction of the route. The cell loss ratio value
             is computed as 10**(-n) where 'n' is the value returned in
             this variable.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.5"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 14 }

pnniRouteNodeFwdMetric6 OBJECT-TYPE
        SYNTAX         Unsigned32
        MAX-ACCESS     read-create
        STATUS         current
        DESCRIPTION
            "An alternate routing parameter for the forward direction of
             this route.

             For information learned from PNNI nodes, this is the
             cumulative Cell Loss Ratio for CLP=0+1 traffic for the
             forward direction of the route. The cell loss ratio value
             is computed as 10**(-n) where 'n' is the value returned in
             this variable.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.6"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 15 }


pnniRouteNodeFwdMetric7 OBJECT-TYPE
        SYNTAX         Unsigned32
        MAX-ACCESS     read-create
        STATUS         current
        DESCRIPTION
            "An alternate routing parameter for the forward direction of
             this route.

             For information learned from PNNI nodes, this is the Cell
             Rate Margin (in cells per second) for the forward direction
             of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.9"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 16 }

pnniRouteNodeFwdMetric8 OBJECT-TYPE
        SYNTAX         Unsigned32
        MAX-ACCESS     read-create
        STATUS         current
        DESCRIPTION
            "An alternate routing parameter for the forward direction of
             this route.

             For information learned from PNNI nodes, this is the
```

```
             Variance Factor (in units of 2**(-8)) for the forward
             direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.10"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 17 }

pnniRouteNodeBwdMetricAW OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The administrative weight calculated for the backward
             direction of this route.  If this metric is not used, its
             value should be set to 0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.4"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 18 }

pnniRouteNodeBwdMetric1 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the backward direction
             of this route.

             For information learned from PNNI nodes, this is the
             maximum possible cell rate (in cells per second) for the
             backward direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.7"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 19 }

pnniRouteNodeBwdMetric2 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the backward direction
             of this route.

             For information learned from PNNI nodes, this is the
             Available cell rate (in cells per second) for the backward
             direction of the route.  Further information on available
             bandwidth may be obtainable by reference to the nodal
             advertisements of the nodes in the path.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.8"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 20 }

pnniRouteNodeBwdMetric3 OBJECT-TYPE
```

```
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the backward direction
             of this route.

             For information learned from PNNI nodes, this is the
             cumulative Maximum Cell Transfer Delay (in microseconds)
             for the backward direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.3"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 21 }

pnniRouteNodeBwdMetric4 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the backward direction
             of this route.

             For information learned from PNNI nodes, this is the
             cumulative Cell Delay Variation (in microseconds) for the
             backward direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.2"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 22 }

pnniRouteNodeBwdMetric5 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the backward direction
             of this route.

             For information learned from PNNI nodes, this is the
             cumulative Cell Loss Ratio for CLP=0 traffic for the
             backward direction of the route. The cell loss ratio value
             is computed as 10**(-n) where 'n' is the value returned in
             this variable.

             If  this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.5"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 23 }

pnniRouteNodeBwdMetric6 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the backward direction
```

```
             of this route.

             For information learned from PNNI nodes, this is the
             cumulative Cell Loss Ratio for CLP=0+1 traffic for the
             backward direction of the route. The cell loss ratio value
             is computed as 10**(-n) where 'n' is the value returned in
             this variable.

             If  this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.6"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 24 }


pnniRouteNodeBwdMetric7 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the backward direction
             of this route.

             For information learned from PNNI nodes, this is the Cell
             Rate Margin (in cells per second) for the backward
             direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.9"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 25 }

pnniRouteNodeBwdMetric8 OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "An alternate routing parameter for the backward direction
             of this route.

             For information learned from PNNI nodes, this is the
             Variance Factor (in units of 2**(-8)) for the backward
             direction of the route.

             If this parameter is not used, its value should be set to
             0xFFFFFFFF."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.1.3.10"
        DEFVAL { 'FFFFFFFF'h }
        ::= { pnniRouteNodeEntry 26 }

pnniRouteNodeVPCapability OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "This attribute indicates whether a VPC setup on this route
             is possible."
        ::= { pnniRouteNodeEntry 27 }

pnniRouteNodeStatus OBJECT-TYPE
```

```
        SYNTAX        RowStatus
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The row status variable, used according to row installation
             and removal conventions."
        ::= { pnniRouteNodeEntry 28 }


-- Table of DTL stacks for routes to other nodes

pnniDTLTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniDTLEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The set of all DTL stacks used for the pre-computed routes
             maintained by this managed entity."
        ::= { pnniRoutingGroup 3 }

pnniDTLEntry OBJECT-TYPE
        SYNTAX        PnniDTLEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A segment of a DTL stack.  The complete DTL stack is formed
             by traversing the rows of the table for which the
             pnniDTLIndex is the same.  Level transitions are indicated
             using the pnniDLTLinkType column."
        INDEX {
                pnniNodeIndex,
                pnniDTLIndex,
                pnniDTLEntryIndex
                }
        ::= { pnniDTLTable 1 }

PnniDTLEntry ::=
        SEQUENCE {
                pnniDTLIndex      Integer32,
                pnniDTLEntryIndex Integer32,
                pnniDTLNodeId     PnniNodeId,
                pnniDTLPortId     PnniPortId,
                pnniDTLLinkType   INTEGER,
                pnniDTLStatus     RowStatus
                }

pnniDTLIndex OBJECT-TYPE
        SYNTAX        Integer32 (1..65535)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The index in the node's DTL table of this DTL stack."
        ::= { pnniDTLEntry 1 }

pnniDTLEntryIndex OBJECT-TYPE
        SYNTAX        Integer32 (1..200)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The index in the current DTL stack of this entry."
        ::= { pnniDTLEntry 2 }

pnniDTLNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
```

```
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The node which is this hop in the DTL stack."
        ::= { pnniDTLEntry 3 }

pnniDTLPortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The port from the pnniDTLNodeId to use as the exit.  If the
             DTL stack does not care, this is coded as zero."
        ::= { pnniDTLEntry 4 }

pnniDTLLinkType OBJECT-TYPE
        SYNTAX        INTEGER {
                                invalid    (1), -- An invalid link
                                horizontal (2), -- A normal link within
                                             -- the containing peer group
                                uplink     (3), -- A link going up a
                                                -- level
                                last       (4) -- The last entry in the
                                               -- DTL stack
                                }
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The type of link out from this node (pnniDTLNodeId).  This
             is well defined even if the specific port is not
             specified."
        ::= { pnniDTLEntry 5 }

pnniDTLStatus OBJECT-TYPE
        SYNTAX        RowStatus
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The row status variable, used according to row installation
             and removal conventions."
        ::= { pnniDTLEntry 6 }


-- Table of routes from nodes to reachable addresses

pnniRouteAddrTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniRouteAddrEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A table containing all the attributes necessary to
             determine what the PNNI entity believes is reachable in
             terms of ATM End System Addresses and to determine which
             nodes are advertising this reachability.  This table is
             also used to configure static routes to reachable address
             prefixes.  The local node index that received the
             reachability information, reachable address, address prefix
             length, and an index that distinguishes between multiple
             listings of connectivity to a given address prefix from a
             given local node are combined to form an instance ID for
             this object.."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.3"
        ::= { pnniRoutingGroup 4 }
```

```
pnniRouteAddrEntry OBJECT-TYPE
        SYNTAX        PnniRouteAddrEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about a
             reachable address prefix."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.3"
        INDEX         { pnniNodeIndex,
                        pnniRouteAddrAddress,
                        pnniRouteAddrPrefixLength,
                        pnniRouteAddrIndex }
        ::= { pnniRouteAddrTable 1 }
PnniRouteAddrEntry ::=
        SEQUENCE {
                pnniRouteAddrAddress            AtmAddrPrefix,
                pnniRouteAddrPrefixLength       PnniPrefixLength,
                pnniRouteAddrIndex              Integer32,
                pnniRouteAddrIfIndex            InterfaceIndex,
                pnniRouteAddrAdvertisingNodeId  PnniNodeId,
                pnniRouteAddrAdvertisedPortId   PnniPortId,
                pnniRouteAddrType               INTEGER,
                pnniRouteAddrProto              INTEGER,
                pnniRouteAddrPnniScope          PnniLevel,
                pnniRouteAddrVPCapability       TruthValue,
                pnniRouteAddrMetricsTag         PnniMetricsTag,
                pnniRouteAddrPtseId             Unsigned32,
                pnniRouteAddrOriginateAdvertisement     TruthValue,
                pnniRouteAddrInfo               OBJECT IDENTIFIER,
                pnniRouteAddrOperStatus         INTEGER,
                pnniRouteAddrTimeStamp          TimeStamp,
                pnniRouteAddrRowStatus          RowStatus
                }
pnniRouteAddrAddress OBJECT-TYPE
        SYNTAX        AtmAddrPrefix
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The value of the ATM End System Address prefix."
        ::= { pnniRouteAddrEntry 1 }

pnniRouteAddrPrefixLength OBJECT-TYPE
        SYNTAX        PnniPrefixLength
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The prefix length to be applied to the ATM End System
             Address prefix."
        ::= { pnniRouteAddrEntry 2 }

pnniRouteAddrIndex OBJECT-TYPE
        SYNTAX        Integer32 (1..65535)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An index into the set of listings of connectivity to a
             given address prefix from a given local node."
        ::= { pnniRouteAddrEntry 3 }

pnniRouteAddrIfIndex OBJECT-TYPE
        SYNTAX        InterfaceIndex
        MAX-ACCESS    read-create
```

```
        STATUS        current
        DESCRIPTION
            "The local interface over which the reachable address can be
             reached.  The value zero indicates an unknown interface or
             reachability through a remote node.

             This object may only have a non-zero value if the value of
             the corresponding instance of pnniRouteAddrProto is other
             than 'pnni', pnniRouteAddrType is other than 'reject', and
             the node identified by pnniRouteAddrAdvertisingNodeId is
             instantiated within this switching system."
        ::= { pnniRouteAddrEntry 4 }

pnniRouteAddrAdvertisingNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The node ID of a node advertising reachability to the
             address prefix.  If the local node index is zero, then the
             advertising node ID must be set to all zeros."
        ::= { pnniRouteAddrEntry 5 }
pnniRouteAddrAdvertisedPortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The port identifier used from the advertising node to reach
             the given address prefix."
        DEFVAL { 0 }
        ::= { pnniRouteAddrEntry 6 }

pnniRouteAddrType OBJECT-TYPE
        SYNTAX        INTEGER {
                                other(1), -- not specified by this MIB
                                reject(2), -- route which discards
                                           -- traffic
                                internal(3),
                                exterior(4)
                                }
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The type (e.g. internal or exterior) of reachability from
             the advertising node to the address prefix.

             Reject(2) refers to an address prefix which, if matched,
             indicates that the message should be discarded as
             unreachable. This is used in some protocols as a means of
             correctly aggregating routes."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.3"
        DEFVAL { exterior }
        ::= { pnniRouteAddrEntry 7 }

pnniRouteAddrProto OBJECT-TYPE
        SYNTAX        INTEGER {
                                other(1), -- not specified
                                local(2), -- e.g. ilmi
                                mgmt(3), -- configured by management,
                                       -- for example by SNMP or console
                                        -- the following are all dynamic
                                        -- routing protocols
                                pnni(4) -- ATM Forum PNNI
```

```
                                 }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The routing mechanism via which the connectivity from the
             advertising node to the reachable address prefix was
             learned."
        ::= { pnniRouteAddrEntry 8 }


pnniRouteAddrPnniScope OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The PNNI scope of advertisement (i.e. level of PNNI
             hierarchy) of the reachability from the advertising node to
             the address prefix."
        REFERENCE
            "ATM Forum PNNI 1.0 Sections 5.3.6, 5.9.1"
        ::= { pnniRouteAddrEntry 9 }

pnniRouteAddrVPCapability OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Indicates whether VPCs can be established from the
             advertising node to the reachable address prefix."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.14.9.1 Table 5-34"
        ::= { pnniRouteAddrEntry 10 }

pnniRouteAddrMetricsTag OBJECT-TYPE
        SYNTAX        PnniMetricsTag
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The index into the pnniMetricsTable for the traffic
             parameter values that apply for the connectivity from the
             advertising node to the reachable address prefix.  There
             will be one or more entries in the pnniMetricsTable whose
             first instance identifier matches the value of this
             variable.

             If there are no parameters associated with this reachable
             address prefix then the distinguished value zero is used."
        DEFVAL { 0 }
        ::= { pnniRouteAddrEntry 11 }

pnniRouteAddrPtseId OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For reachable addresses learned via PNNI, this attribute
             contains the value of the PTSE Identifier for the PTSE
             being originated by the originating node which contains the
             information group(s) describing the reachable address.  For
             reachable addresses learned by means other than PNNI, this
             attribute is set to zero."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.2"
        ::= { pnniRouteAddrEntry 12 }
```

```
pnniRouteAddrOriginateAdvertisement OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Whether or not the reachable address specified by this
             entry is to be advertised by the local node into its PNNI
             routing domain.

             This object may only take on the value 'true' when the
             value of the corresponding instance of pnniRouteAddrProto
             is other than 'pnni'."
        DEFVAL { true }
        ::= { pnniRouteAddrEntry 13 }

pnniRouteAddrInfo OBJECT-TYPE
        SYNTAX        OBJECT IDENTIFIER
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "A reference to MIB definitions specific to the particular
             routing protocol which is responsible for this reachable
             address prefix, as determined by the value specified in the
             route's pnniRouteAddrProto value.  If this information is
             not present, its value should be set to the OBJECT
             IDENTIFIER zeroDotZero."
        DEFVAL { zeroDotZero }
        ::= { pnniRouteAddrEntry 14 }

pnniRouteAddrOperStatus OBJECT-TYPE
        SYNTAX        INTEGER {
                          inactive(1),
                          active(2), -- i.e. reachability to this
                                     -- prefix exists and is not
                                     -- being advertised in PNNI
                          advertised(3)
                          }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the reachable address prefix is
             operationally valid and whether it is being advertised by
             this node."
        ::= { pnniRouteAddrEntry 15 }

pnniRouteAddrTimeStamp OBJECT-TYPE
        SYNTAX        TimeStamp
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates when the connectivity from the advertising node
             to the reachable address prefix became known to the local
             node."
        ::= { pnniRouteAddrEntry 16 }

pnniRouteAddrRowStatus OBJECT-TYPE
        SYNTAX        RowStatus
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "To create, delete, activate and de-activate a reachable
             address prefix."
        ::= { pnniRouteAddrEntry 17 }
```

```
-- Table of routes from nodes to reachable transit networks

pnniRouteTnsTable OBJECT-TYPE
        SYNTAX        SEQUENCE OF PnniRouteTnsEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "A table containing all the attributes necessary to
             determine what transit networks the PNNI entity believes
             are reachable and to determine which nodes are advertising
             this reachability.  This table is also used to add static
             routes to reachable transit networks.  The local node index
             which received the reachability information, type of
             network identification, network identification plan,
             transit network identifier, and an index that distinguishes
             between multiple listings of connectivity to a given
             transit network from a given local node are combined to
             form an instance ID for this object."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.3.2"
        ::= { pnniRoutingGroup 5 }

pnniRouteTnsEntry OBJECT-TYPE
        SYNTAX        PnniRouteTnsEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An entry in the table, containing information about a
             reachable transit network."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.3.2"
        INDEX         { pnniNodeIndex,
                        pnniRouteTnsType,
                        pnniRouteTnsPlan,
                        pnniRouteTnsId,
                        pnniRouteTnsIndex }
        ::= { pnniRouteTnsTable 1 }

PnniRouteTnsEntry ::=
        SEQUENCE {
                pnniRouteTnsType                TnsType,
                pnniRouteTnsPlan                TnsPlan,
                pnniRouteTnsId                  DisplayString,
                pnniRouteTnsIndex               Integer32,
                pnniRouteTnsIfIndex             InterfaceIndex,
                pnniRouteTnsAdvertisingNodeId   PnniNodeId,
                pnniRouteTnsAdvertisedPortId    PnniPortId,
                pnniRouteTnsRouteType           INTEGER,
                pnniRouteTnsProto               INTEGER,
                pnniRouteTnsPnniScope           PnniLevel,
                pnniRouteTnsVPCapability        TruthValue,
                pnniRouteTnsMetricsTag          PnniMetricsTag,
                pnniRouteTnsPtseId              Unsigned32,
                pnniRouteTnsOriginateAdvertisement      TruthValue,
                pnniRouteTnsInfo                OBJECT IDENTIFIER,
                pnniRouteTnsOperStatus          INTEGER,
                pnniRouteTnsTimeStamp           TimeStamp,
                pnniRouteTnsRowStatus           RowStatus
                }

pnniRouteTnsType OBJECT-TYPE
        SYNTAX        TnsType
```

```
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The type of network identification used for this transit
             network."
        ::= { pnniRouteTnsEntry 1 }

pnniRouteTnsPlan OBJECT-TYPE
        SYNTAX        TnsPlan
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The network identification plan according to which network
             identification has been assigned."
        ::= { pnniRouteTnsEntry 2 }

pnniRouteTnsId OBJECT-TYPE
        SYNTAX        DisplayString
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "The value of the transit network identifier."
        ::= { pnniRouteTnsEntry 3 }

pnniRouteTnsIndex OBJECT-TYPE
        SYNTAX        Integer32 (1..65535)
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION
            "An index into the set of listings of connectivity to a
             given transit network from a given local node."
        ::= { pnniRouteTnsEntry 4 }

pnniRouteTnsIfIndex OBJECT-TYPE
        SYNTAX        InterfaceIndex
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The local interface over which the transit network can be
             reached.  The value zero indicates an unknown interface or
             reachability through a remote node.

             This object may only have a non-zero value if the value of
             the corresponding instance of pnniRouteTnsProto is other
             than 'pnni' and the node identified by
             pnniRouteTnsAdvertisingNodeId is instantiated within this
             switching system."
        ::= { pnniRouteTnsEntry 5 }

pnniRouteTnsAdvertisingNodeId OBJECT-TYPE
        SYNTAX        PnniNodeId
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The node ID of a node advertising reachability to the
             transit network.  If the local node index is zero, then the
             advertising node ID must also be set to zero."
        ::= { pnniRouteTnsEntry 6 }
pnniRouteTnsAdvertisedPortId OBJECT-TYPE
        SYNTAX        PnniPortId
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The port identifier used from the advertising node to
```

```
             reach the given transit network."
        DEFVAL { 0 }
        ::= { pnniRouteTnsEntry 7 }

pnniRouteTnsRouteType OBJECT-TYPE
        SYNTAX        INTEGER {
                                other(1), -- not specified by this MIB
                                exterior(4)
                                }
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The type (e.g. exterior or other) of reachability from the
             advertising node to the transit network."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.1.3"
        DEFVAL { exterior }
        ::= { pnniRouteTnsEntry 8 }

pnniRouteTnsProto OBJECT-TYPE
        SYNTAX        INTEGER {
                                other(1), -- not specified
                                local(2), -- e.g. ilmi
                                mgmt(3), -- configured by management,
                                       -- for example by SNMP or console
                                        -- the following are all dynamic
                                        -- routing protocols
                                pnni(4) -- ATM Forum PNNI
                                }
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "The routing mechanism via which the connectivity from the
             advertising node to the transit network was learned."
        ::= { pnniRouteTnsEntry 9 }


pnniRouteTnsPnniScope OBJECT-TYPE
        SYNTAX        PnniLevel
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "The PNNI scope of advertisement (i.e. level of PNNI
             hierarchy) of the reachability from the advertising node to
             the transit network."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.3.6"
        ::= { pnniRouteTnsEntry 10 }

pnniRouteTnsVPCapability OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Indicates whether VPCs can be established from the
             advertising node to the reachable transit network."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.14.9.1 Table 5-34"
        ::= { pnniRouteTnsEntry 11 }

pnniRouteTnsMetricsTag OBJECT-TYPE
        SYNTAX        PnniMetricsTag
        MAX-ACCESS    read-create
        STATUS        current
```

```
        DESCRIPTION
            "The index into the pnniMetricsTable for the traffic
             parameter values that apply for the connectivity from the
             advertising node to the transit network.  There will be one
             or more entries in the pnniMetricsTable whose first
             instance identifier matches the value of this variable.

             If there are no parameters associated with this transit
             network then the distinguished value zero is used."
        DEFVAL { 0 }
        ::= { pnniRouteTnsEntry 12 }

pnniRouteTnsPtseId OBJECT-TYPE
        SYNTAX        Unsigned32
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "For reachable transit networks learned via PNNI, this
             attribute contains the value of the PTSE Identifier for the
             PTSE being originated by the originating node which
             contains the information group(s) describing the transit
             network.  For reachable transit networks learned by means
             other than PNNI, this attribute is set to zero."
        REFERENCE
            "ATM Forum PNNI 1.0 Section 5.8.2"
        ::= { pnniRouteTnsEntry 13 }

pnniRouteTnsOriginateAdvertisement OBJECT-TYPE
        SYNTAX        TruthValue
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "Whether or not the transit network specified by this entry
             is to be advertised by the local node into its PNNI routing
             domain.

             This object may only take on the value 'true' when the
             value of the corresponding instance of pnniRouteNodeProto
             is other than 'pnni'."
        DEFVAL { true }
        ::= { pnniRouteTnsEntry 14 }

pnniRouteTnsInfo OBJECT-TYPE
        SYNTAX        OBJECT IDENTIFIER
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "A reference to MIB definitions specific to the particular
             routing protocol which is responsible for this transit
             network, as determined by the value specified in the
             route's pnniRouteTnsProto value.  If this information is
             not present, its value should be set to the OBJECT
             IDENTIFIER zeroDotZero."
        DEFVAL { zeroDotZero }
        ::= { pnniRouteTnsEntry 15 }

pnniRouteTnsOperStatus OBJECT-TYPE
        SYNTAX        INTEGER {
                          inactive(1),
                          active(2), -- i.e. reachability to this
                                  -- transit network exists and is
                                  -- not being advertised in PNNI
                          advertised(3)
                          }
```

```
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates whether the reachable transit network is
             operationally valid and whether it is being advertised by
             this node."
        ::= { pnniRouteTnsEntry 16 }

pnniRouteTnsTimeStamp OBJECT-TYPE
        SYNTAX        TimeStamp
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION
            "Indicates how long the connectivity from the advertising
             node to the reachable transit network has been known to the
             local node."
        ::= { pnniRouteTnsEntry 17 }

pnniRouteTnsRowStatus OBJECT-TYPE
        SYNTAX        RowStatus
        MAX-ACCESS    read-create
        STATUS        current
        DESCRIPTION
            "To create, delete, activate and de-activate a reachable
             transit network."
        ::= { pnniRouteTnsEntry 18 }


-- conformance information

pnniMIBConformance
                   OBJECT IDENTIFIER ::= { pnniMIB 2 }
pnniMIBCompliances
                   OBJECT IDENTIFIER ::= { pnniMIBConformance 1 }
pnniMIBGroups
                   OBJECT IDENTIFIER ::= { pnniMIBConformance 2 }


-- compliance statements

pnniMIBCompliance MODULE-COMPLIANCE
        STATUS        current
        DESCRIPTION
            "The compliance statement for entities which implement
             the PNNI MIB.

             Groups of PNNI objects required for management of a minimum
             function node are identified by the suffix MinGroup.

             Groups of PNNI objects required for management of a border
             node are identified by the suffix BorderGroup.

             Groups of PNNI objects required for management of a PGL/LGN
             capable node are identified by the suffix LgnGroup.

             Groups of optional PNNI objects are identified by the
             suffix OptionalGroup."
        MODULE  -- this module
            MANDATORY-GROUPS { pnniGeneralMinGroup,
                               pnniNodeMinGroup,
                               pnniNodePglMinGroup,
                               pnniNodeTimerMinGroup,
                               pnniScopeMinGroup,
                               pnniIfMinGroup,
```

```
                               pnniLinkMinGroup,
                               pnniNbrPeerMinGroup,
                               pnniNbrPeerPortMinGroup }
        OBJECT pnniNodeId
        MIN-ACCESS read-only
        DESCRIPTION
            "Support for manual configuration of node IDs is optional."

        OBJECT pnniNodeLowest
        MIN-ACCESS read-only
        DESCRIPTION
            "Only switching systems that are PGL/LGN capable are allowed
             to provide write/create access to the pnniNodeLowest
             object."

        OBJECT pnniNodeRestrictedTransit
        MIN-ACCESS read-only
        DESCRIPTION
            "Support for the restricted transit capability is optional."

        OBJECT pnniNodeComplexRep
        MIN-ACCESS read-only
        DESCRIPTION
            "The ability to generate the complex node representation is
             only required for PGL/LGN capable switching systems, and is
             otherwise optional."

        OBJECT pnniNodeRowStatus
        SYNTAX INTEGER { active(1) }
        MIN-ACCESS read-only
        DESCRIPTION
            "The ability to create more than one node in a switching
             system is optional."

        OBJECT pnniNodePglLeadershipPriority
        MIN-ACCESS read-only
        DESCRIPTION
            "Only switching systems that are PGL/LGN capable are allowed
             to provide write/create access to the
             pnniNodePglLeadershipPriority object."

        OBJECT pnniIfNodeIndex
        MIN-ACCESS read-only
        DESCRIPTION
            "Write access to the pnniIfNodeIndex object is optional.  It
             only applies when there can be multiple lowest-level nodes
             in the switching system."

        OBJECT pnniIfVPCapability
        MIN-ACCESS read-only
        DESCRIPTION
            "The ability to support switched virtual paths is optional."

        ::= { pnniMIBCompliances 1 }

-- units of conformance

pnniGeneralMinGroup OBJECT-GROUP
        OBJECTS {
                  pnniHighestVersion,
                  pnniLowestVersion,
                  pnniDtlCountOriginator,
                  pnniCrankbackCountOriginator,
                  pnniAltRouteCountOriginator,
```

```
                    pnniRouteFailCountOriginator,
                    pnniRouteFailUnreachableOriginator
                    }
        STATUS current
        DESCRIPTION
            "A collection of general PNNI objects required for
             management of a minimum function switching system."
        ::= { pnniMIBGroups 1 }

pnniGeneralBorderGroup OBJECT-GROUP
        OBJECTS {
                    pnniDtlCountBorder,
                    pnniCrankbackCountBorder,
                    pnniAltRouteCountBorder,
                    pnniRouteFailCountBorder,
                    pnniRouteFailUnreachableBorder
                    }
        STATUS current
        DESCRIPTION
            "A collection of general PNNI objects required for
             management of a border node."
        ::= { pnniMIBGroups 2 }

pnniNodeMinGroup OBJECT-GROUP
        OBJECTS {
                    pnniNodeLevel,
                    pnniNodeId,
                    pnniNodeLowest,
                    pnniNodeAdminStatus,
                    pnniNodeOperStatus,
                    pnniNodeDomainName,
                    pnniNodeAtmAddress,
                    pnniNodePeerGroupId,
                    pnniNodeRestrictedTransit,
                    pnniNodeComplexRep,
                    pnniNodeRestrictedBranching,
                    pnniNodeDatabaseOverload,
                    pnniNodePtses,
                    pnniNodeRowStatus
                    }
        STATUS current
        DESCRIPTION
            "A collection of per node PNNI objects required for
             management of a minimum function switching system."
        ::= { pnniMIBGroups 3 }

pnniNodePglMinGroup OBJECT-GROUP
        OBJECTS {
                    pnniNodePglLeadershipPriority,
                    pnniNodePglInitTime,
                    pnniNodePglReelectTime ,
                    pnniNodePglState,
                    pnniNodePreferredPgl,
                    pnniNodePeerGroupLeader,
                    pnniNodePglTimeStamp,
                    pnniNodeActiveParentNodeId
                    }
        STATUS current
        DESCRIPTION
            "A collection of per node PGL election related PNNI objects
             required for management of a minimum function switching
             system."
        ::= { pnniMIBGroups 4 }
```

```
pnniNodePglLgnGroup OBJECT-GROUP
        OBJECTS {
                  pnniNodeCfgParentNodeIndex,
                  pnniNodePglOverrideDelay
                  }
        STATUS current
        DESCRIPTION
            "A collection of per node PGL election related PNNI objects
             required for management of a PGL/LGN capable switching
             system."
        ::= { pnniMIBGroups 5 }

pnniNodeTimerMinGroup OBJECT-GROUP
        OBJECTS {
                  pnniNodePtseHolddown,
                  pnniNodeHelloHolddown,
                  pnniNodeHelloInterval,
                  pnniNodeHelloInactivityFactor,
                  pnniNodePtseRefreshInterval,
                  pnniNodePtseLifetimeFactor,
                  pnniNodeRxmtInterval,
                  pnniNodePeerDelayedAckInterval,
                  pnniNodeAvcrPm,
                  pnniNodeAvcrMt,
                  pnniNodeCdvPm,
                  pnniNodeCtdPm
                  }
        STATUS current
        DESCRIPTION
            "A collection of per node PNNI objects required for
             management of timers and significant change thresholds in a
             minimum function switching system."
        ::= { pnniMIBGroups 6 }

pnniNodeTimerLgnGroup OBJECT-GROUP
        OBJECTS {
                  pnniNodeHlinkInact
                  }
        STATUS current
        DESCRIPTION
            "A collection of per node PNNI objects required for
             management of timers in a PGL/LGN capable switching
             system."
        ::= { pnniMIBGroups 7 }


pnniNodeSvccLgnGroup OBJECT-GROUP
        OBJECTS {
                  pnniNodeSvccInitTime,
                  pnniNodeSvccRetryTime,
                  pnniNodeSvccCallingIntegrityTime,
                  pnniNodeSvccCalledIntegrityTime,
                  pnniNodeSvccTrafficDescriptorIndex
                  }
        STATUS current
        DESCRIPTION
            "A collection of per node SVCC-based RCC related PNNI
             objects required for management of a PGL/LGN capable
             switching system."
        ::= { pnniMIBGroups 8 }

pnniScopeMinGroup OBJECT-GROUP
        OBJECTS {
                  pnniScopeLocalNetwork,
```

```
                    pnniScopeLocalNetworkPlusOne,
                    pnniScopeLocalNetworkPlusTwo,
                    pnniScopeSiteMinusOne,
                    pnniScopeIntraSite,
                    pnniScopeSitePlusOne,
                    pnniScopeOrganizationMinusOne,
                    pnniScopeIntraOrganization,
                    pnniScopeOrganizationPlusOne,
                    pnniScopeCommunityMinusOne,
                    pnniScopeIntraCommunity,
                    pnniScopeCommunityPlusOne,
                    pnniScopeRegional,
                    pnniScopeInterRegional,
                    pnniScopeGlobal
                    }
        STATUS current
        DESCRIPTION
            "A collection of per node scope mapping related PNNI objects
             required for management of a minimum function switching
             system."
        ::= { pnniMIBGroups 9 }

pnniSummaryLgnGroup OBJECT-GROUP
        OBJECTS {
                    pnniSummaryType,
                    pnniSummarySuppress,
                    pnniSummaryState,
                    pnniSummaryRowStatus
                    }
        STATUS current
        DESCRIPTION
            "A collection of PNNI objects required for controlling
             address summarization."
        ::= { pnniMIBGroups 10 }

pnniIfMinGroup OBJECT-GROUP
        OBJECTS {
                    pnniIfNodeIndex,
                    pnniIfPortId,
                    pnniIfVPCapability,
                    pnniIfAdmWeightCbr,
                    pnniIfAdmWeightRtVbr,
                    pnniIfAdmWeightNrtVbr,
                    pnniIfAdmWeightAbr,
                    pnniIfAdmWeightUbr,
                    pnniIfRccServiceCategory,
                    pnniIfRccTrafficDescrIndex
                    }
        STATUS current
        DESCRIPTION
            "A collection of per interface PNNI objects required for
             management of a minimum function switching system."
        ::= { pnniMIBGroups 11 }

pnniIfBorderGroup OBJECT-GROUP
        OBJECTS {
                    pnniIfAggrToken
                    }
        STATUS current
        DESCRIPTION
            "A collection of per interface PNNI objects required for
             management of a border node."
        ::= { pnniMIBGroups 12 }
```

```
pnniLinkMinGroup OBJECT-GROUP
        OBJECTS {
                  pnniLinkType,
                  pnniLinkVersion,
                  pnniLinkHelloState,
                  pnniLinkRemoteNodeId,
                  pnniLinkRemotePortId,
                  pnniLinkIfIndex,
                  pnniLinkRcvHellos,
                  pnniLinkXmtHellos
                  }
        STATUS current
        DESCRIPTION
            "A collection of per link PNNI objects required for
             management of a minimum function switching system."
        ::= { pnniMIBGroups 13 }

pnniLinkBorderOrLgnGroup OBJECT-GROUP
        OBJECTS {
                  pnniLinkDerivedAggrToken,
                  pnniLinkUpnodeId,
                  pnniLinkUpnodeAtmAddress,
                  pnniLinkCommonPeerGroupId
                  }
        STATUS current
        DESCRIPTION
            "A collection of per link PNNI objects required for
             management of a border node or a PGL/LGN capable switching
             system."
        ::= { pnniMIBGroups 14 }

pnniLinkLgnGroup OBJECT-GROUP
        OBJECTS {
                  pnniLinkSvccRccIndex
                  }
        STATUS current
        DESCRIPTION
            "A collection of per link PNNI objects required for
             management of a PGL/LGN capable switching system."
        ::= { pnniMIBGroups 15 }

pnniNbrPeerMinGroup OBJECT-GROUP
        OBJECTS {
                  pnniNbrPeerState,
                  pnniNbrPeerPortCount,
                  pnniNbrPeerRcvDbSums,
                  pnniNbrPeerXmtDbSums,
                  pnniNbrPeerRcvPtsps,
                  pnniNbrPeerXmtPtsps,
                  pnniNbrPeerRcvPtseReqs,
                  pnniNbrPeerXmtPtseReqs,
                  pnniNbrPeerRcvPtseAcks,
                  pnniNbrPeerXmtPtseAcks
                  }
        STATUS current
        DESCRIPTION
            "A collection of per neighboring peer PNNI objects required
             for management of a minimum function switching system."
        ::= { pnniMIBGroups 16 }

pnniNbrPeerLgnGroup OBJECT-GROUP
        OBJECTS {
                  pnniNbrPeerSvccRccIndex
                  }
```

```
        STATUS current
        DESCRIPTION
            "A collection of per neighboring peer PNNI objects required
             for management of a PGL/LGN capable switching system."
        ::= { pnniMIBGroups 17 }

pnniNbrPeerPortMinGroup OBJECT-GROUP
        OBJECTS {
                  pnniNbrPeerPortFloodStatus
                   }
        STATUS current
        DESCRIPTION
            "A collection of per port to neighboring peer PNNI objects
             required for management of a minimum function switching
             system."
        ::= { pnniMIBGroups 18 }

pnniSvccRccLgnGroup OBJECT-GROUP
        OBJECTS {
                  pnniSvccRccVersion,
                  pnniSvccRccHelloState,
                  pnniSvccRccRemoteNodeId ,
                  pnniSvccRccRemoteAtmAddress,
                  pnniSvccRccRcvHellos,
                  pnniSvccRccXmtHellos,
                  pnniSvccRccIfIndex,
                  pnniSvccRccVpi,
                  pnniSvccRccVci
                  }
        STATUS current
        DESCRIPTION
            "A collection of per SVCC-based RCC PNNI objects required
             for management of a PGL/LGN capable switching system."
        ::= { pnniMIBGroups 19 }

pnniPtseOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniPtseType,
                  pnniPtseSequenceNum,
                  pnniPtseChecksum,
                  pnniPtseLifeTime,
                  pnniPtseInfo
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional per PTSE PNNI objects."
        ::= { pnniMIBGroups 20 }


pnniMapOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniMapType,
                  pnniMapPeerGroupId,
                  pnniMapAggrToken,
                  pnniMapRemoteNodeId,
                  pnniMapRemotePortId,
                  pnniMapVPCapability,
                  pnniMapPtseId,
                  pnniMapMetricsTag
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to create a map
             of nodes and links in the PNNI routing domain."
```

```
        ::= { pnniMIBGroups 21 }

pnniMapNodeOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniMapNodePeerGroupId,
                  pnniMapNodeAtmAddress,
                  pnniMapNodeRestrictedTransit,
                  pnniMapNodeComplexRep,
                  pnniMapNodeRestrictedBranching,
                  pnniMapNodeDatabaseOverload,
                  pnniMapNodeIAmLeader,
                  pnniMapNodeLeadershipPriority,
                  pnniMapNodePreferredPgl,
                  pnniMapNodeParentNodeId,
                  pnniMapNodeParentAtmAddress,
                  pnniMapNodeParentPeerGroupId,
                  pnniMapNodeParentPglNodeId
                 }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to create a map
             of nodes in the PNNI routing domain."
        ::= { pnniMIBGroups 22 }

pnniMapAddrOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniMapAddrAddress,
                  pnniMapAddrPrefixLength
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to create a map
             of reachable addresses in the PNNI routing domain."
        ::= { pnniMIBGroups 23 }

pnniMapTnsOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniMapTnsId
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to create a map
             of reachable transit networks in the PNNI routing domain."
        ::= { pnniMIBGroups 24 }

pnniMetricsOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniMetricsClasses,
                  pnniMetricsGcacClp,
                  pnniMetricsAdminWeight,
                  pnniMetrics1,
                  pnniMetrics2,
                  pnniMetrics3,
                  pnniMetrics4,
                  pnniMetrics5,
                  pnniMetrics6,
                  pnniMetrics7,
                  pnniMetrics8,
                  pnniMetricsRowStatus
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to manage
             metrics and attributes associated with PNNI entities."
```

```
        ::= { pnniMIBGroups 25 }

pnniRouteGeneralOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniRouteNodeNumber,
                  pnniRouteAddrNumber
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects."
        ::= { pnniMIBGroups 26 }

pnniRouteNodeOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniRouteNodeDestPortId,
                  pnniRouteNodeProto,
                  pnniRouteNodeTimeStamp,
                  pnniRouteNodeInfo,
                  pnniRouteNodeGcacClp,
                  pnniRouteNodeFwdMetricAW,
                  pnniRouteNodeFwdMetric1,
                  pnniRouteNodeFwdMetric2,
                  pnniRouteNodeFwdMetric3,
                  pnniRouteNodeFwdMetric4,
                  pnniRouteNodeFwdMetric5,
                  pnniRouteNodeFwdMetric6,
                  pnniRouteNodeFwdMetric7,
                  pnniRouteNodeFwdMetric8,
                  pnniRouteNodeBwdMetricAW,
                  pnniRouteNodeBwdMetric1,
                  pnniRouteNodeBwdMetric2,
                  pnniRouteNodeBwdMetric3,
                  pnniRouteNodeBwdMetric4,
                  pnniRouteNodeBwdMetric5,
                  pnniRouteNodeBwdMetric6,
                  pnniRouteNodeBwdMetric7,
                  pnniRouteNodeBwdMetric8,
                  pnniRouteNodeVPCapability,
                  pnniRouteNodeStatus
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to manage
             precalculated routes to nodes in the PNNI routing domain."
        ::= { pnniMIBGroups 27 }

pnniDTLOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniDTLNodeId,
                  pnniDTLPortId,
                  pnniDTLLinkType,
                  pnniDTLStatus
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to manage
             precalculated routes to nodes in the PNNI routing domain."
        ::= { pnniMIBGroups 28 }

pnniRouteAddrOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniRouteAddrIfIndex,
                  pnniRouteAddrAdvertisingNodeId,
                  pnniRouteAddrAdvertisedPortId,
```

```
                  pnniRouteAddrType,
                  pnniRouteAddrProto,
                  pnniRouteAddrPnniScope,
                  pnniRouteAddrVPCapability,
                  pnniRouteAddrMetricsTag,
                  pnniRouteAddrPtseId,
                  pnniRouteAddrOriginateAdvertisement,
                  pnniRouteAddrInfo,
                  pnniRouteAddrOperStatus,
                  pnniRouteAddrTimeStamp,
                  pnniRouteAddrRowStatus
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to manage routes
             to reachable addresses in the PNNI routing domain."
        ::= { pnniMIBGroups 29 }

pnniRouteTnsOptionalGroup OBJECT-GROUP
        OBJECTS {
                  pnniRouteTnsIfIndex,
                  pnniRouteTnsAdvertisingNodeId,
                  pnniRouteTnsAdvertisedPortId,
                  pnniRouteTnsRouteType,
                  pnniRouteTnsProto,
                  pnniRouteTnsPnniScope,
                  pnniRouteTnsVPCapability,
                  pnniRouteTnsMetricsTag,
                  pnniRouteTnsPtseId,
                  pnniRouteTnsOriginateAdvertisement,
                  pnniRouteTnsInfo,
                  pnniRouteTnsOperStatus,
                  pnniRouteTnsTimeStamp,
                  pnniRouteTnsRowStatus
                  }
        STATUS current
        DESCRIPTION
            "A collection of optional PNNI objects used to manage routes
             to reachable transit networks in the PNNI routing domain."
        ::= { pnniMIBGroups 30 }


END
```

15. Appendix A: Examples of Message Sequences

15.1 General

This section describes the relationship between the Call States, Connection Establishment/Release messages and timers used in the symmetric procedures.

15.2 Finite State Machine

Figure 15-1 shows the message sequence chart of PNNI for normal ATM connection establishment and release.

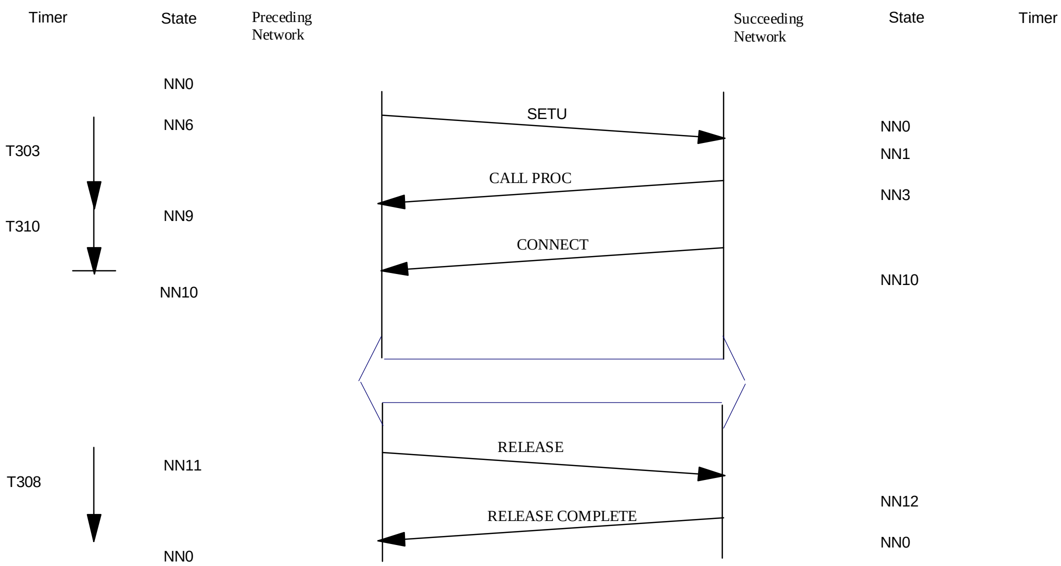


Figure 15-1: PNNI Finite State Machine

15.3 Example of Message Sequences

The flows shown below describe, in a simplified manner, the initiation of a Call establishment and call clearing across multiple PNNIs.

15.3.1 Successful Call Establishment

Figure 15-2 shows an example of the message sequence across the PNNI when a call is initiated from TE X to TE Y for a successful call.

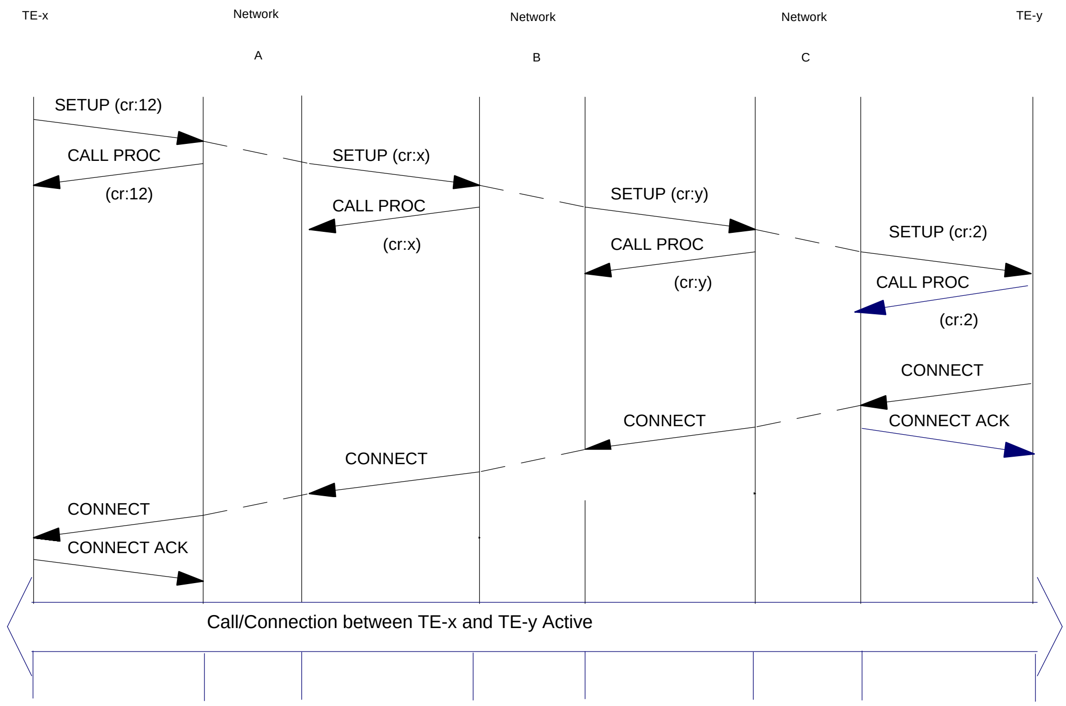


Figure 15-2: Setup, successful call

15.3.2 Unsuccessful Call Establishment

Figure 15-3 shows an example of the message sequence across the PNNI when a call is initiated from TE X to TE Y for a unsuccessful call (called user rejection).

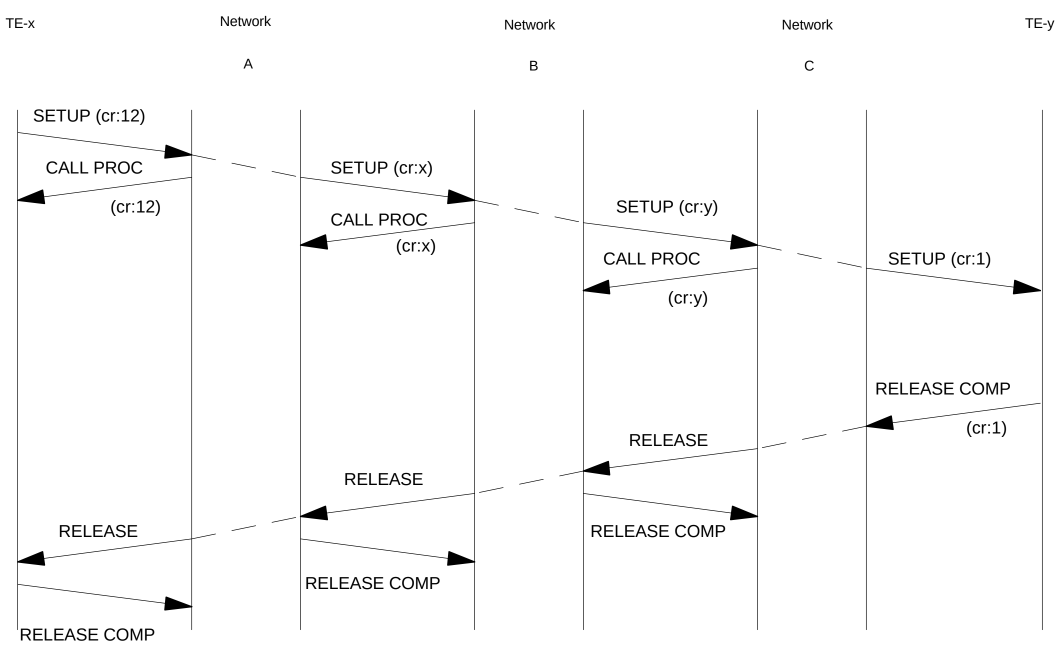


Figure 15-3: Setup, unsuccessful call

15.3.3 Normal call clearing (from originator)

Figure 15-4 shows an example of call clearing from the active state, initiated by TE x.

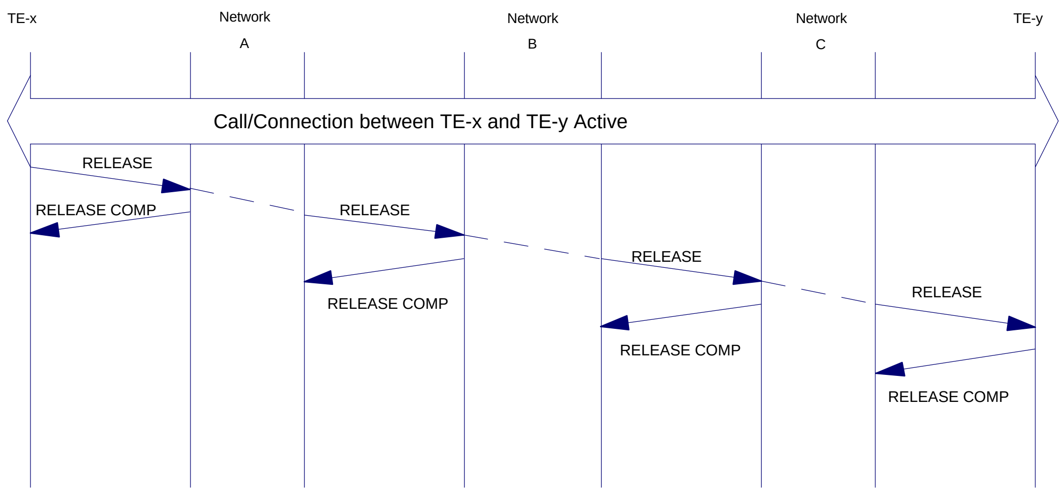


Figure 15-4: Normal call clearing by originator

15.3.4 Call termination by a Private network

Figure 15-5 shows an example of a Private network terminating a call (for some reason, but not normal clearing) which is in the active state.

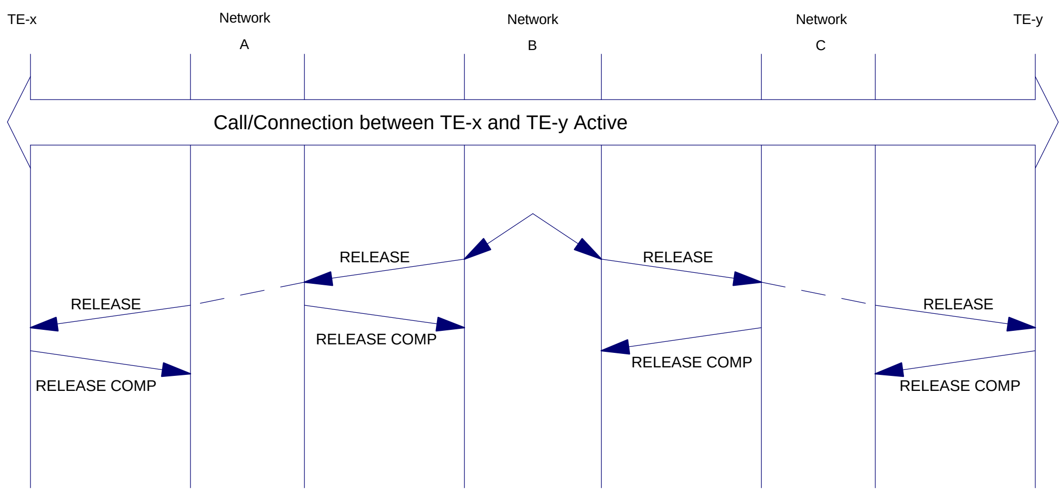


Figure 15-5: Call termination by Network B

16. Appendix B: Derivation of Generic Connection Admission Control

As part of providing quality of service (QoS) and throughput guarantees, a switching system performs connection admission control (CAC) during connection setup phase to determine if a connection request can be accepted without violating existing connections' QoS and throughput requirements. To enable routing to produce paths that will likely be accepted, it is necessary for switching systems to advertise some information about their internal CAC states. Requiring switching systems to expose detailed and up-to-date CAC information, however, may result in unacceptably high routing traffic. Furthermore, CAC is not subject for standardization in PNNI, and so it is not practical for the switching systems to advertise their potentially different detailed CAC states. Generic connection admission control (GCAC) solves this problem by allowing switching systems to advertise CAC information that is generic (i.e., independent of the actual CAC used in the switching systems) and compact, but yet rich enough to support any CAC.

GCAC defines a set of parameters to be advertised and a common admission interpretation of these parameters. This common interpretation is in the form of a generic CAC algorithm to be performed during path selection to determine if a link or node can or cannot be included for consideration. The algorithm uses the advertised GCAC parameters (available from the topology database) and the characteristics of the connection being requested (available from signalling) to determine if a link/node will likely accept or reject the connection. A link/node is included if the GCAC algorithm determines that it will likely accept the connection, and excluded otherwise.

16.1 Generic CAC for CBR and VBR Service Categories

GCAC for CBR and VBR service categories uses the following parameters:

ACR - Available Cell Rate, a measure of effective available bandwidth
CRM - Cell Rate Margin, a measure of the difference between the allocated bandwidth and the aggregate sustained rate of existing connections
VF - Variance Factor, a relative measure of the cell rate margin normalized by the variance of the aggregate rate

The assumption behind the generic CAC is that the ratio between CRM, which represents the safety margin the switching system is putting above the aggregate sustained cell rate, and the standard deviation of the aggregate rate (to be defined below) does not change significantly as one connection is added on the link. Using this assumption, each link advertises the current CRM and CRM-to-standard-deviation ratio. Any ingress switch doing path selection can then compute the new standard deviation of the aggregate rate (from the old value and the connection's traffic descriptors) and an estimate of the new CRM. From this, the increase in bandwidth required to carry the new connection can be computed and compared to ACR.

To expand on the discussion above, let AAC denote the actual allocated capacity, i.e., the amount of bandwidth that has been allocated to existing connections by the actual CAC used in the switching system. CRM is the difference between AAC and the aggregate sustained rate (ASR) of the existing connections. ASR can be either the sum of existing connections' declared sustainable cell rates (SCRs) or a smaller - possibly measured or estimated - value. Let MaxCR denote the maximum cell rate that is usable by either CBR or VBR connections (or both). The following diagram illustrates the relationship among MaxCR, AAC, ACR, ASR and CRM:

```
              |<-- CRM -->|<---- ACR ----->|
|-------------|-----------|----------------|
0            ASR         AAC             MaxCR
```

The assumption is that CRM is proportional to some measure of the burstiness of the traffic generated by the existing connections, this measure being the standard deviation of the aggregate traffic rate defined as the square root of the sum of SCR_excon(PCR_excon-SCR_excon) over all existing connections, where SCR_excon and PCR_excon are declared sustainable and peak cell rates, respectively. This assumption is based on the simple argument that AAC needs to be some multiple of the standard deviation above the mean aggregate traffic rate to guarantee some levels of cell loss ratio and cell queuing time. Depending on the actual CAC used, the CRM-to-standard-deviation ratio may vary as connections are established and taken down. It is reasonable to assume, however, that around some sufficiently large value of AAC, this ratio will not vary significantly. What this means is a link can advertise its current CRM-to-standard-deviation ratio (actually in the form of VF, which is the square of this number), and the GCAC algorithm can use this number to estimate how much bandwidth is required to carry an additional connection.

The GCAC algorithm is derived as follows: Consider a new connection with peak cell rate PCR and sustainable cell rate SCR, and a link with the following advertised GCAC parameters: ACR_link, CRM_link, and VF_link. Denote the variance (i.e., square of standard deviation) of the aggregate traffic rate by VAR_link (not advertised). Denote other unadvertised GCAC quantities by AAC_link and ASR_link. Then,

$$VAR\_link = \sum_{\substack{\text{over existing}\\ \text{connections excon}}} SCR\_excon*(PCR\_excon-SCR\_excon) \qquad (1)$$

and

$$VF\_link = \frac{CRM\_link**2}{VAR\_link}. \qquad (2)$$

Using the above equation, VAR_link can be computed from the advertised VF_link and CRM_link as VAR_link = CRM_link**2/VF_link.

Let Delta_CR be the additional capacity needed to carry the new connection. The GCAC algorithm basically computes Delta_CR from the advertised GCAC parameters and the new connections traffic descriptors, and compare it with ACR_link. If Delta_CR =< ACR_link then the link is included for path selection consideration; otherwise, it is excluded, i.e.,

Include link
ACR_link >= Delta_CR (3)
<
Exclude link

Let CRM_new denote the cell rate margin if the new connection were accepted. Denote other 'new' quantities by AAC_new, ASR_new, and VAR_new. Then,

$$Delta\_CR = CRM\_new - CRM\_link + SCR \qquad (4)$$

since CRM_new = AAC_new - ASR_new, CRM_link = AAC_link - ASR_link, and ASR_new - ASR_link = SCR. Substituting (4) into (3), rearranging terms, and squaring both sides yield

Include link
[ACR_link + CRM_link - SCR]**2 >= CRM_new**2 (5)
<
Exclude link

Using the GCAC assumption made earlier, CRM_new**2 can be computed as

CRM_new**2 = VF_link * VAR_new, (6)

where

VAR_new = VAR_link + SCR*(PRC-SCR). (7)

Substituting (2), (6) and (7) into (5) yields

```
                              Include link
[ACR_link+CRM_link-SCR]**2 >= CRM_link**2 + VF_link*SCR(PCR-SCR),        (8)
                               <
                              Exclude link
```

and moving CRM_link**2 to the left-hand side and rearranging terms yield

```
                                             Include link
[ACR_link-SCR] * [ACR_link-SCR+2*CRM_link] >= VF_link*SCR(PCR-SCR).      (9)
                                              <
                                             Exclude link
```

In general Delta_CR is between SCR and PCR. So the above test is not necessary for the cases ACR_link >= PCR and ACR_link < SCR. In the former case, the link is included; in the latter case, the link is excluded.

```
  Exclude                     Include
|<-- link  --->|<--- Test (9) --->|<-- link
|--------------|------------------|--------------->  ACR
              SCR                PCR
```

16.2 Generic CAC for UBR and ABR Service Categories

In UNI 3.0 Sec. 3.6.2.4, it is stated that connection admission control shall not reject a UBR call based on bandwidth availability but it may reject based on other reasons such as number of UBR calls exceeding a chosen threshold. GCAC defines only one parameter for UBR service category - maximum cell rate (MaxCR) - to advertise how much capacity is usable for UBR connections. The purpose of advertising this parameter is twofold: MaxCR can be used for path optimization, and MaxCR = 0 is used to indicate that a link is not accepting any (additional) UBR connections.

17. Appendix C: Guidelines for Topology Aggregation

To preserve the flexibility of the complex node representation for PNNI Routing, no requirements are specified for when and how the nodal state parameters constituting the complex node representation should be generated. Nevertheless, the following guidelines are provided for discretionary use only.

17.1 Diameter and Radius

For each nodal state parameter, the “diameter” of a logical node is defined as an aggregation of all parameter values in a full-mesh representation of the logical node. Each “diameter” must be converted to a “radius” for the default node representation in the PNNI complex node representation. For a nodal metric, the “radius” is simply half the “diameter”. For a nodal attribute, the “radius” is the same as the “diameter”.

17.2 Presumed Nodal State Parameters

PNNI routing does not specify how presumed nodal state parameters are determined for the default node representation. A conservative approach determines the entire set of parameters based on a single path selection criterion, whereas an aggressive approach determines each of them based on a possibly different path selection criterion. The degree of aggressiveness is left to the discretion of the advertisers.

17.3 Significant Exceptions

Exceptions are useful if they provide “significantly different” topology information than that revealed by the default node representation in the PNNI complex node representation. For example, one could designate a connectivity between a port and the nucleus on a default node representation as an exception to expose an outlier, which is a node whose exclusion from the peer group it belongs to would significantly improve the accuracy and simplicity of the aggregation of the remainder of the peer group topology.

17.4 Useless Exceptions

It is useful to identify conditions under which a link is not to be designated as an exception. For example, we may consider the following condition. A link is said to be dominated by a path if at least one parameter value of the link is less desirable than the corresponding parameter value of the path, and all other parameter values are pair-wise equal. Since PNNI Routing selects internal paths through a logical group node from the best concatenation of links in the complex node representation in a normal manner, it is not useful to designate a logical link to be an exception if it would be dominated by at least one alternate path in the advertised topology.

17.5 Number of Exceptions

While PNNI Routing does not specify a hard limit on the number of exceptions, it is recommended that the number of exceptions used to configure a complex node representation of a logical group node be normally kept smaller than 3 times the number of ports in the logical group node. For logical group nodes with a small number of ports, it is acceptable for as many as all the port-to-port connectivities to be designated as exceptions.

18. Appendix D: Multi-Peer Group Systems

There are a number of cases in which it is useful for a single physical system to implement the functions of multiple PNNI nodes. This allows a single physical system to be included in multiple peer groups. This does not require any change in the PNNI protocol, and therefore cannot in fact be prohibited. This capability also significantly expands upon the flexibility and usefulness of the PNNI Routing Protocol.

This section describes some of these cases, and is intended as explanatory material. This section does not require nor propose any actual change in the PNNI protocol.

The following situations have been identified in which one physical system may want to directly participate in the operation of multiple peer groups:

1. Peer group leader (in a child and a parent peer group)

2. Border systems (in two bordering peer groups)

3. Core systems in branchy networks

4. Backbone and local area topology

18.1 Technical Details

18.1.1 Peer Group Leader

The most obvious case of a single real physical system operating in multiple peer groups is the peer group leader (PGL). The PGL operates as a normal member of a peer group, and also, as the elected leader of the peer group, operates as a normal member of the parent peer group.

The PGL is required to feed information between the parent and child peer group. In order to allow nodes within the child peer group to route to destinations outside of the peer group, the PGL feeds higher level PTSEs into the peer group. In order to allow nodes outside of the peer group to route to destinations within the peer group, the PGL announces (in its PTSEs transmitted into the parent peer group) reachability to one or more address prefixes which summarize the destinations reachable within the child peer group.

18.1.2 Border Systems

In some cases, a single node may sit on the boundary between two neighboring peer groups. This situation is illustrated in Figure 18-1. Here a single real physical system "X" sits on the boundaries between peer groups A and B.

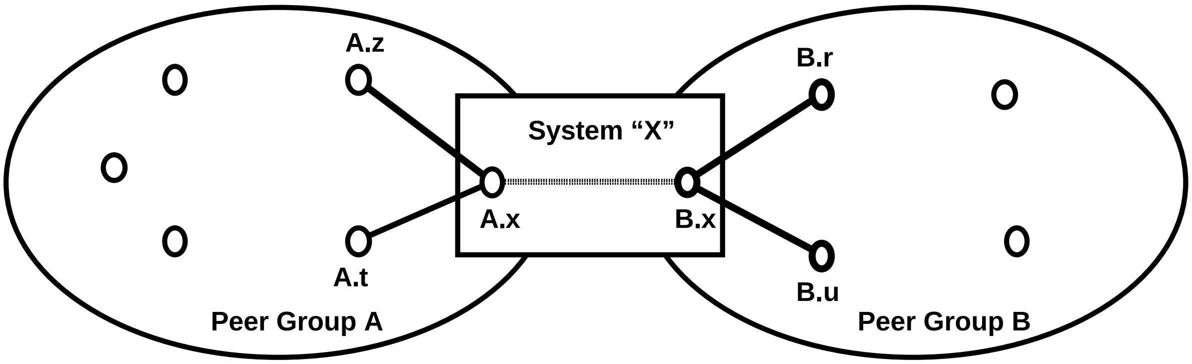


Figure 18-1: Border Node between Peer Groups A and B

The situation illustrated in Figure 18-1 may occur in the normal operation of PNNI within a single larger ATM Private Network which is also running PNNI. In this case, the border system X operates just as if it were two separate systems A.x and B.x, using the normal PNNI protocol mechanisms. The information exchange corresponding to the dotted line internal to system X corresponds to the normal exchange of PNNI information between neighboring PNNI systems in different peer groups. System X may choose to run PNNI internally in this case, or may use a proprietary implementation as long as the external behavior conforms to normal PNNI behavior for border nodes.

This situation may occur at administrative boundaries between different organizations. In this case, separate instances of PNNI routing may be used on each side of the administrative boundaries. It may be necessary to carefully control the exchange of information between these two instances. This is similar to the exchange of routing information between OSPF and BGP at domain boundaries in an IP network. This allows PNNI to be used both as an intra-domain "interior" routing protocol in a private organization, and as an inter-domain routing protocol between organizations.

In this latter case, the exchange of routing information between different instances of PNNI may be referred to as "inter-domain PNNI". A "Router Management" function allows the inter-domain routing exchange to follow specific administrative requirements.

The details of the router management function is for further study (this is a common function in the IP Internet, and we may use current Internet practice as a start in definition of a router management function for PNNI). Typically, the routing manager allows configuration of which summary addresses to advertise as reachable, and which summary addresses to accept if the other system advertises these as reachable.

In many inter-domain cases, the system may choose to announce reachability to specific address prefixes, plus metrics describing the cost of reaching addresses which match that prefix, without being able to (or choosing to) announce sufficient detail to allow creation of associated DTLs. This implies that in some cases the DTL may terminate at a boundary system. In this case, if PNNI routing is used in both domains, the system may create a new DTL to continue the call into the other domain.

In many cases, organizations want to maintain control over their own networks, including management of border nodes. Thus the situation illustrated in Figure 18-1, in which one border node is shared between two routing domains, may be unacceptable.

A further extension of the border case is illustrated in Figure 18-2. Here the two domains A and B do not share a common border system. Rather, domain A has control over border system X, and domain B has control over border system Y. Systems X and Y communicate via a simple instance of PNNI Routing, which comprises one peer group and two nodes. This peer group may be called an "interdomain peer group", in that its sole purpose is to interconnect the two domains A and B.

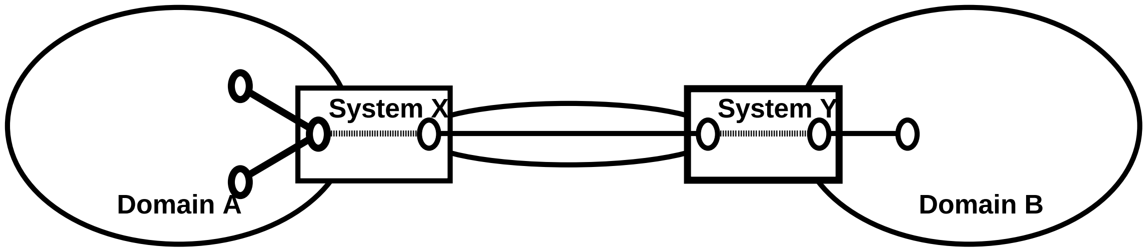


Figure 18-2: Two Domains with Individual Border Nodes

The operation of the border nodes in Figure 18-2 is very similar to the operation of the single border node in Figure 18-1.

In this case, note that it is not necessary for both (or either) domains to run PNNI routing internally. For example, Domain A may represent a private ATM network which is running PNNI internally. System X is therefore configured to control the exchange of reachability information between the private PNNI domain internal to Domain A, and the inter-domain instance of PNNI. On the other hand, Domain B may be either another private domain, or a public domain, which is running some other routing protocol internally. Thus, PNNI provides one possible way to exchange information between domains, and can be used even when one or both domains is not running PNNI internally.

18.1.3 Core System in a Very Branchy Network.

Some networks are characterized by a relatively moderate number of high performance "core systems", each of which has links to a large number of smaller (generally lower performance) non-transit stub systems. The core systems may be attached in a high- performance backbone. The stubs receive connectivity to other sites (including other stubs) via the backbone.

Where there are a large number of non-transit stubs attached to a single common core system, any one stub does not need to know the detail regarding the other stubs. This implies that the large number of stubs attached to a single core system should not be considered part of a single common peer group. This implies that the core system needs to operate as a member of multiple peer groups.

18.1.4 Backbone and Local Area Topology

Figure 18-3 illustrates another common topology, in which a network has a central backbone, plus multiple local areas. This is somewhat similar to the branch topology described above, except that in this case the core systems are each attached to only one (or a small number of) local areas. In Figure 18-3, there are six high performance backbone nodes A through F, connected via high bandwidth links (shown as heavy lines in our example). There are also three local areas X, Y, and Z, each of which contains a number of lower speed switches and lower bandwidth links (only the internal structure of area X is illustrated in Figure 18-3).

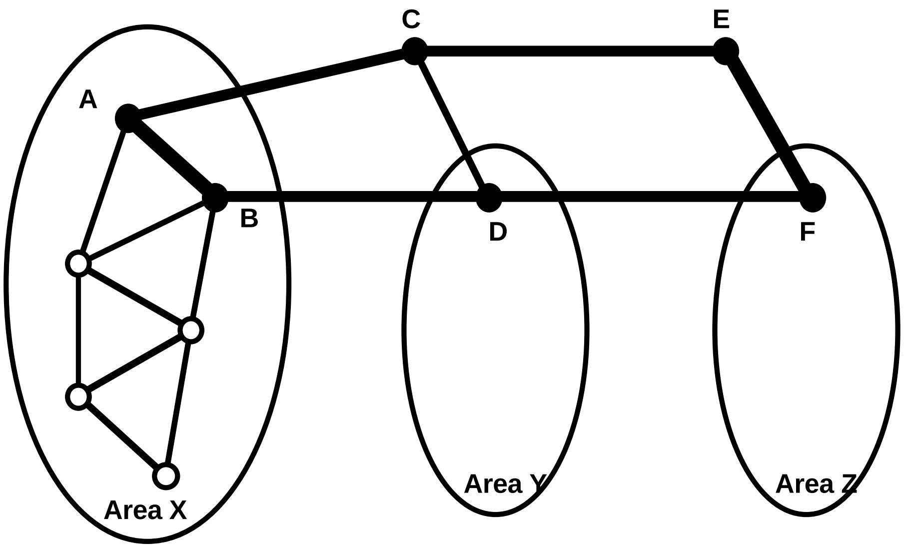


Figure 18-3: Backbone and Local Area Topology

In topologies such as this, a sensible way to manage the network is for each local area (areas X, Y, and Z in our example) to be configured as low level peer groups, and the backbone (comprising nodes A through F, plus the high bandwidth links) to be configured as the higher level parent peer group for each of the lower level peer groups. This implies, for example, that systems A and B are part of both the peer group consisting of Area X, and also the backbone peer group.

18.2 Proposed Disposition Of This Section

This section has pointed out a number of cases in which a single real physical system will need to participate in multiple peer groups, and therefore implement the functions of multiple logical nodes (one logical node in each peer group). It is suggested that this is a relatively normal case, and that the flexibility and usability of PNNI routing will be enhanced by allowing for these cases. It is suggested that the precise interactions between the multiple logical nodes represented within a single physical system may depend upon the topology of the network. Thus, the PNNI Subworking group should identify specific topologies of interest, and should specify the proper operation in each case.

The text contained in Section 18.1 may be thought of as an outline for explanatory material. This is intended to point out ways that PNNI routing may be used. This material does not require any change in the PNNI protocol. however, this material may serve as useful guidance to implementers and users of PNNI Routing.

19. Appendix E: Allocating Resources in Transit Peer Groups

When a switch calculates a DTL (either in advance or in response to a call request) the calculations make assumptions about the amount of resources that peer groups along the selected path will consume. For example, the calculator assumes that the peer group will be prepared to use up only as much delay as it has advertised. For any cumulative metric, the total of the values from each traversed peer group are then compared with what the caller requested.

If now a transit peer group chooses a path with a longer delay than the advertised value, it is quite possible for the call to become blocked by exceeding the delay request somewhere further along the path.

This is a particular problem if the transit peer group could have chosen a lower delay path, but chose not to. The crankback message will come past the transit peer group, and that group will never be told that there was a delay problem, and that it should pick a lower delay path.

An attractive first cut solution was to communicate the optimization objective that was used originally. However, it can easily be seen that the problem is related to the values used, and not the optimization objective. For example, the initial optimization may have been on the basis of administrative weight. The calculated path may just barely meet the needed delay. Depending on how the transit peer group derived its announced delay metric, it may well be that an administrative-weight optimized path will not meet the needs. Further, it can be seen that communicating the objective function would constrain systems from implementing new and different optimization functions, when such local flexibility is intended as a feature of this protocol.

If that will not solve the problem, what will? The following are heuristics that will increase the probability of a transit peer group picking a useful internal path. These are not mandatory behaviors. Several are included because they address several cases and have differing complexity. In the following, what is being described is the behavior of a switch at a peer group border upon receiving a call from outside the peer group with a DTL which transits the peer group. The concern is what delay budget to allow when calculating a path across the peer group. The assumption in the following is that the call did have a delay bound, and therefore the amount of delay consumed in this transit peer group is significant.

1) Use the advertised value. If the peer group was advertised with a single nodal metric for delay (for the relevant category of service), then use the advertised value as the target. If a path can not be calculated which meets the other call objectives and meets the advertised delay, then calculate a path with the best possible delay meeting the call.
2) Use the call and the path. The call has an indication of how much of the requested delay has already been consumed. It also has a DTL stack. Examining the DTL stack, the switch can attempt to calculate how much delay would be used by the remainder of the call. Adding this to the amount consumed, and subtracting from the original request gives a good target for the value which this peer group should attempt to meet. This algorithm refers to attempting to calculate the consumed delay due to complex nodal metrics. If there are no complex nodal metrics along the path, then a value equivalent to that used by the originating switch will be generated (equivalent up to any changes in advertised information). However, if there are nodes with complex metrics, the switch will have to pick a path, which appears to meet the call, across the peer group. This parallels what the originating switch did, but could pick a different crossing path. Thus, even here there is the potential for mismatching behavior and unfortunate crankback.

   Once the delay target has been selected, the same use can be made of it as in (1) above.

20. Appendix F: PNNI Packet Sizes

This appendix provides formulae for conservative estimates of the octet counts in the PNNI packets.

20.1 PNNI Hello Packet (PHP)

PHPs are exchanged between neighbor nodes over each physical link and VPC in order to discover and verify the identity of neighbor nodes and to determine the status of the links to those nodes. In addition, Hellos are also exchanged between neighbor LGNs over each SVC-based RCC to identify port IDs and determine the status of horizongal links.

Let L denote the number of known higher levels with respect to a neighbor of a given node.

Let C denote the number of service categories. Each RAIG consists of 32 octets of required information and 12 octets of optional GCAC related information. Thus, the maximum number of octets in an RAIG is 44. The bit-mask encoding in each RAIG allows for one IG to be used to specify resource availability information for several service categories which share common resource availability information. Separate RAIGs must be used for service categories which do not share common resource availability information. In this respect, a conservative estimate (upper bound) for the total number of octets in RAIGs for all service categories is 44C.

The contents of a PHP are listed in Table 20-1.

Table 20-1: Contents of PNNI Hello Packet

Information Entities	Number of Octets
Hello Common Part	100
Aggregation Token IG	8
Nodal Hierarchy List IG	12 + 56L
Uplink Information Attribute IG	8 + 44C
LGN Horizontal Links Extension IG	X

Let H denote the number of horizontal links. In Table 20-1, X = 0 at the lowest hierarchical level where there is no LGN, and X = 8 + 12H at higher levels.

Total number of octets in a PHP transmitted within a peer group = 100 + X.

Total number of octets in a PHP transmitted across a peer group boundary = 128 + 56L + 44C.

20.2 PNNI Topology State Packet (PTSP)

PTSPs are used to distribute topology information throughout a peer group. A PTSP originated by a node captures the topology information about the node, including its complex node representation, as well as all horizontal links and uplinks attached to the node. In PNNI Routing, a complex node representation is a collection of nodal state parameters that provide detailed state information associated with a logical group node.

Let P denote the number of ports in the complex node representation associated with the node that originates the PTSP.

Let D be denote the number of octets for next higher level binding information. D = 84 if there is a higher level, and D = 0 otherwise.

Let N denote the number of octets in a Network Identification (one octet per IA5 character).

Let W denote the number of octets in an Uplink Information Attribute IG, where W <= 8 + 44C octets.

The contents of a PTSP are listed in Table 20-2.

Table 20-2: Contents of PNNI Topology State Packet

Information Entities	Number of Octets
PTSP Header	44
PTSE Header	20
Nodal State Parameter IG	16 + 44C
Nodal IG	48 + D
Internal Reachable ATM Address (IRA) Information	A
Exterior Reachable ATM Address (ERA) Information	B
IRA IG Overhead	16 + 2*44C
ERA IG Overhead	16 + 2*44C
Transit Network ID	7 + N
Horizontal Links IG	40 + 44C
Uplinks IG	72 + 44C + W

Note that IRA information and ERA information each includes the reachable address prefix as well as 1 octet indicating prefix length.

In Table 20-2, padding must be included, if necessary, in the estimates for the IRA information, the ERA information, and the Transit Network ID.

Total number of octets in a PTSP = 44 + 20 * (Number of PTSEs in the PTSP)
+ (Total payload of PTSEs in the PTSP)

Total payload of PTSEs in the PTSP = (16 + 44C) * f(P)
+ 48 + D
+ (16 + 2*44C) * (Number of IRA Igs)
+ A * (Total Number of IRA Prefixes)
+ (23 + 2*44C + N) * (Number of ERA Igs)
+ B * (Total Number of ERA Prefixes)
+ (40 + 44C) * (Number of Horizontal Links)
+ (72 + 44C + W) * (Number of Uplinks)

where f(P) is the number of nodal state parameters in the complex node representation.

The above estimate is obtained with the following assumptions:
a) All IRA prefixes with padding have the same length.
b) All ERA prefixes with padding have the same length.
c) All transit network identifications with padding have the same length.

With the above assumptions, one can determine an upper bound on the total payload of PTSEs in a PTSP by letting the constants respectively take on conservative estimates.

20.3 PTSE Acknowledgement Packet (PTSE_AP)

PTSE_APs are used to acknowledge the receipt of PTSEs from a neighbor node. Each PTSE_AP consists of multiple Nodal PTSE Acknowledgement IGs, and each of these IGs is used to acknowledge the receipt of multiple PTSEs with the same originating node and transmitted by the same neighbor node.

The contents of a PTSE_AP are listed in Table 20-3.

Table 20-3: Contents of PTSE Acknowledgement Packet

Information Entities	Number of Octets
PTSE_AP Header	8
Nodal PTSE Acknowledgement IG	28
PTSE Acknowledge Overhead	12

Total number of octets in a PTSE_AP sent by a given node = 8
+ 28 * (Total Number of PTSE Ack Sets in the PTSE_AP)
+ 12 * (Total Number of PTSE Acks in the PTSE_AP)
<= 8 + 40 * (Total Number of PTSE Acks in the PTSE_AP)

The upper bound is obtained by assuming the worst case where each PTSE Ack Set contains only one PTSE Ack.

20.4 Database Summary Packet (DBSP)

DBSPs are used during the initial database exchange process and contains the header information of all PTSEs in a node’s topology database. Each DBSP consists of multiple Nodal PTSE Summaries IGs, and each of these IGs contains the PTSE header information of multiple PTSEs with the same originating node and transmitted by the same neighbor node.

The contents of a DBSP are listed in Table 20-4.

Table 20-4: Contents of Database Summary Packet

Information Entities	Number of Octets
DBSP Header	16
Nodal PTSE Summary IG	44
PTSE header Overhead	16

Total number of octets in a DBSP sent by a given node
= 16 + 44 * (Total Number of PTSE Summary Sets in the DBSP)

+ 16 * (Total Number of PTSE Headers in the DBSP)
<= 16 + 60 * (Total Number of PTSE Headers in the DBSP)

The upper bound is obtained by assuming the worst case where each DB Summary Set contains only one PTSE Header.

20.5 PTSE Request Packet (PTSE_RP)

PTSE_RPs are used during database synchronization to request from a neighboring peer those PTSEs that have been newly discovered or that have been found to be obsolete. Each DBSP consists of multiple Requested PTSEs IGs, and each of these IGs contains multiple request PTSEs with the same originating node and transmitted by the same neighbor node.

The contents of a PTSE_RP are listed in Table 20-5.

Table 20-5: Contents of PTSE Request Packet

Information Entities	Number of Octets
PTSE_RP Header	8
Requested PTSE IG	28
Requested PTSE Overhead	4

Total number of octets in a PTSE_RP sent by a given node
= 8 + 28 * (Total Number of PTSE Request Sets in the PTSE_RP)
+ 4 * (Total Number of PTSE Requests in the PTSE_RP)
<= 8 + 32 * (Total Number of PTSE Requests in the PTSE_RP)

The upper bound is obtained by assuming the worst case where each PTSE Request Set contains only one PTSE Request.

21. Appendix G. Flooding Parameter Selection Guidelines

The objective of flooding is the dissemination of information in the fastest possible way reliably in the network. The flooding speed will depend on many factors like link speeds, node processing speed, network topology etc. It seems very likely that the flooding bottleneck will be the switching system processing speed. One way to reduce the processing load on the switches is to acknowledge a group of PTSE's in a single packet. This is done by appropriately selecting the PeerDelayedAckInterval. The PeerDelayedAckInterval is the time interval between consecutive checks of the PeerDelayedAcks list to verify if we have any outstanding unacknowledged PTSE's. If there are any unacknowledged PTSE's their headers are bundled in an acknowledgment packet and sent out the link.

21.1 Basic Parameter Relations

In order for the ATM network to function without excessive retransmissions PTSERetransmissionInterval and PeerDelayedAckInterval need to satisfy the following basic inequality:

PTSERetransmissionInterval > PeerDelayedAckInterval + Rtdelay

where Rtdelay is the round trip delay suffered by the PTSP. This delay might be negligible for links at the lowest level of the hierarchy, and could be important as we move up the hierarchy.

21.2 Higher Hierarchical Levels

When we move up the hierarchy, parameters like PTSERetransmissionInterval will have to be configured with higher values due to the fact that links are represented by virtual circuits and nodes by Peer Groups. Logically this is equivalent to having a network with links and nodes that are slower (more delay while traversing the virtual circuit forming the logical link, and nodes that are Logical Group nodes). More the cell loss ratio on such links and nodes might be increased which makes it more likely for retransmissions to occur while flooding. The PTSERetransmissionInterval for a selected hierarchical level has to take these facts into consideration for the ATM network to behave efficiently.

21.3 Varying the PeerDelayedAckInterval

The selection of the PeerDelayedAckInterval will affect the average number of PTSE acknowledgments grouped in a single packet. The longer the PeerDelayedAckInterval the more likely we will gather a larger number of acknowledgments and reduce the number of packets flooded in the network. We can increase the PeerDelayedAckInterval as long as we satisfy the inequality stated above. In the case of intensive flooding we might reach a scenario where we have gathered many elements in the Delayed Acknowledgment list and we would like to acknowledge them. In such a case we can achieve the grouping effect with a small PeerDelayedAckInterval and could risk running out of buffers if we continued to delay the acks. As we can see the selection of the PeerDelayedAckInterval might be influenced by the switch performance and memory. More it could be related to the frequency at which new PTSPs are generated in the network (intensive flooding due to frequent significant changes in the nodal and link metrics).

21.4 Default PeerDelayedAckInterval Value

Viewing the reasoning presented above we need to establish a default value for the PeerDelayedAckInterval. We can make a rather safe assumption that the Rtdelay for a worst case scenario at the highest level of the hierarchy would not be greater than 2 seconds. In such a case a default PTSERetransmissionInterval could be in the order of 5-10 seconds. If we assume PTSERetransmissionInterval 5 seconds as default, then the PeerDelayedAckInterval should not be greater than 1 or 2 seconds to assure some security margin in the inequality stated above.

Thus the default values of 5 second for PTSERetransmissionInterval and 1 second for PeerDelayedAckInterval would be independent of the hierarchy level (worse case scenario). The user might select to modify these values to tune its network flooding parameters for specific hierarchical levels viewing the guidelines presented.

22. Appendix H: A Sample Algorithm for Route Generation

As part of the PNNI specification, it is desirable to specify sample algorithms which generate useable routes in a fashion compliant with the formal specification.

There are two general categories of route generation algorithms. One category is an algorithm used to generate route(s) to all possible destination. The sample given here does this for a single service category and a single optimization criteria. A straightforward process might be to run this in the background for each service category and for each single optimization criteria within that category. This would produce a set of precalculated routes to use for an incoming call. In addition, one needs an algorithm to calculate a route for a given call. This is needed when the precalculated routes do not meet the callers parameters.

It should be noted that in these algorithms the structure of the received information is assumed (as it will) to reflect the hierarchical structure, and therefore the algorithm does not need proceed iteratively through hierarchical levels. However, when the path is convert into DTLs, level analysis is needed.

In order to describe this algorithm, we need an abstract description of the database. We will use arrays for must of the description, with arrays being indexed from 1.

Nodecount = The number of nodes in the database. For this purpose, a node with complex metrics counts as one node.

AllNodeCount = The count of all nodes, with ports which are identified in complex metrics counting as well.

Linkcount = count of links, include links described in complex metrics. Classcount= The number of supported service categories

AllNodes[AllNodeCount] = The array of nodal structures. This includes the reachability information and, for nodes without complex metrics, (or default nodes in the complex case) the nodal metrics. Subfields are:

- LinkCount: The number of links out of this node.
- Classinfo[classcount]: The metric info for each class. The metric info includes whether the class is supported.
- Reachability: The list of prefixes reachable from this node.
- IsComplex: True for everything except the default-entry of a complex node
- CenterNode: If this is a node created by a complex metric, this points to the default node for reachability purposes.
- ParentNode: The index of the Parent, or -1 if there is none.

Links[Linkcount] = The attributes of the link. These include

- SrcNode, DestNode: The indices of the starting and ending node of the link.
- IsUplink: True if this is an uplink from a node.
- PairedLink: The link in the obverse direction
- Classinfo[Classcount]: metrics for the service categories
- IsInternal: True if this link is generated by a complex node metric.

For an Uplink, there will be a backlink. However, no node will point to the backlink as an outward-going link. It is only used as the reverse of an uplink. If one wishes to calculate from destinations towards the source, then one would put the DownLinks as the actual link, and the uplinks would only be reachable as the obverse of the DownLinks. It is necessary to avoid having both, or looping paths can result.

MaxOutLink = The maximum number of outward links from a node. Note that in a practical implementation this is unlimited and a different structure should be used.

OutLinks[AllNodeCount, MaxLinkCount] = The index into "Links" of the links out of a given node.

This = This index of the starting node in AllNodes

We will also use a data type of "path". This represents node traversal sequence. It also has cumulative values of the metrics in use.

Description of Generate_All_Routes_Q(C)

C is the service category, and Q is the metric index to optimize. This must be an additive metric, so that it can be the target of a Dijkstra. To generate all possible routes for a given service category C, with an optimization criterion Q, we will need some auxiliary information. For purposes of this sample, the optimization is on a single forward-direction metric. It could as easily be a backwards-direction metric. This is independent of whether paths are calculated from the source or from the destination. In fact, any mathematical function of the forward and reverse direction metrics and attributes could be used, so long as the result of the function is a single value such that when elements are added to a path, the single positive non-zero value for that element is added to the value for the path. (So, one could use a scaled average of the forward and backwards administrative weight and delay if one so chose.) In most cases, simply optimizing once for the forward administrative weight, and once for the forward delay, will give reasonable pre-computed paths.

BestCosts[AllNodeCount]: This is the best cost to each destination node. While we are actually only interested in basic nodes, this array must have size equal to AllNodeCount for accumulation purposes. This is initialized to -1 to indicate that no paths have been found.

To initialize BestCosts, set the cost for "This" node and all its containing nodes to 0. This represents the fact that we are already here and do not need to find a way here.

BestPaths[AllNodeCount]: This is an array of paths select to the destinations. It is initialized to Empty. If multiple paths are to be recorded, this will be a double array, [AllNodeCount, pathcount]

Tent: A list of paths sorted by the optimization criteria. To initialize this, create the set of paths consisting of "This" Node and each link out of "This" Node.

Then, the algorithm will proceed to run a Dijkstra on the database. The Dijkstra will use whatever optimization criteria (an accumulating metric) has been selected. If desired, paths within a cost "epsilon" of best may also be retained. Epsilon must be small enough that looping is impossible. (For administrative cost, this turns out to be 2 since we internally suppress immediate backtracks.)

PseudoCode:

A number of utility routines are used in the algorithm:

Admit_Link_All(Link, C, Q):

This routine tests whether a link can be used for a given service category and optimization criteria. It checks that the category is supported, and that a non-zero value exists for the metric.

Create_Path(...):
This routine creates a path data structure, with the elements given.

```
Make_Path(Node, Link, C, Q)
        path p;
        p = Create_Path(Node, Link, Links[Link].DestNode);
        p.cost = AllNodes[Node].Classinfo[C].Metric[Q] + Links[Link].Classinfo[C].Metric[Q] +
                        AllNodes[Links[Link].DestNode].Classinfo[C].Metric[Q];
        return(p);

Append_Path(Path, Link, C, Q)
        path p;
        integer i;

        p = Create_Path(Path, Link, Links[Link].DestNode);
        p.cost = Path.cost +
                        Links[Link].Classinfo[C].Metric[Q] +
                        AllNodes[Links[Link].DestNode].Classinfo[C].Metric[Q];
        /* accumulate all metrics */
        for (i = 1 to metric_count)
                        path.metric[i] = Path.metric[i] + Links[Link].Classinfo[C].Metric[i] +
                        AllNodes[Links[Link].DestNode].Classinfo[C].Metric[i];
        /* Now combine attributes using the appropriate min, max, mask,
            or other rule depending on the attribute */
        for (i = 1 to attribute_count)
                path.attribute[i] = combine_attribute(path.attribute[i],
                        Links[Link].Classinfo[C].attribute[i],
                        AllNodes[Links[Link].DestNode].Classinfo[C].attribute[i]);
        return
```

Add_to_Tent(Path, C, Q, Tiebreakers):
Insert the path into the tent, sorted by C/Q and the Tiebreakers so that one can remove entries from the tent in sorted order.

Remove_Smallest(Tent):
returns the lowest cost path in tent, and removes it from the tent.

Last_Node(Path):
returns the index of the last node in the path.

Last_Link(Path):
returns the index of the last link of the path.

Tent_Empty(Tent):
tests whether the tent is empty.

Record_Path(Path, C, Q, Tiebreakers):
Check whether the path is good enough to record. If so, record it, updating Best_Costs and Best_Paths.

If multiple paths are being allowed, with a metric tolerance, this routine enforces that tolerance. It uses Tiebreakers to decide between otherwise equally good paths. In order to prevent paths from re-entering the same node through different complex metrics, a special check is done in Record_Path.

When a path with given "cost" to a given "node" is being considered, and "node" is a complex node, and the last link is not an internal link (e.g. this is an arrival), then the path is admitted against the list for the real node, as well as the individual complex node. This results in the node as a whole being used for external paths, while still generating internal paths.

This behavior, while it prevents re-entry, also restricts finding certain classes of alternate paths. Specifically, it can not find paths which have a more expensive path to a certain transit node with a complex nodal metric, and then a cheaper transit cost leading to a cheaper total cost. Using the approach above gives robust path generation. All switches are required to follow the DTLs exactly, whether they have been generated with such loop avoiding behavior or not. As such, implementers may consider removing the special check described above, or relaxing it.

```
Generate_all_routes(C, Q, Tiebreakers)
   /* For Service Category C, with optimization criteria Q, generate optimal and
   nearly optimal paths.  Tiebreakers helps order otherwise equal paths/links */

    integer i, j, k;
    path p, q;

    for (i = 1 to AllNodeCount)
            BestCosts[i] = -1;
            for (j = 1 to Pathcount)
                    BestPaths[i, j] = NIL;

    Tent = Empty;

    i = This;
    while (i > 0)
            BestCosts[i] = 0;
            i = AllNodes[i].ParentNode;

    for (i = 1 to AllNodes[This].LinkCount)
            /* Make sure that the links can be used for this service category
                 and optimization parameter */
            if (Admit_Link_All(OutLinks[This,i], C, Q))
                    /* The Path carries the cumulative cost for the prime condition */
                    p = Make_Path(This, OutLinks[This, i], C, Q);
                    Add_to_Tent(p, C, Q, Tiebreakers);

    while (not Tent_Empty(Tent))
            p = Remove_Smallest(Tent);
            i = Last_Node(p);
            k = Last_Link(p);

            /* Check if the cost is good enough to be useable */
            if (Record_Path(p, i, C, Q, Tiebreakers))

                 /* Now examine the outlinks, and put them into the tent */
                 for (j = 1 to AllNodes[i].LinkCount)
                         /* check if the link is useable */
```

```
                if (Links[k].PairedLink != j /* do not backtrack directly */ &&
                        Admit_Link_All(OutLinks[i, j], C, Q))
                    q = Append_Path(p, OutLinks[i, j], C, Q);
                    Add_to_Tent(q);
```

When this is complete, there will exists a set of paths to each destination node for the given service category and optimization criteria. These can then be used to create candidate DTLs for locally originated calls to the destinations reachable at the given node. The behavior in Record_Path of associating node entries with the node as a whole, combined with the fact that reachability is associated with the node as a whole, means that nodal reachability is associated with any entry, and therefore is not affected by internal partitions, spanning tree representations, or other issues. If we associated reachabilities with entry ports at some later time, then again the association will work.

It should be noted that the data structure for recording the selected pre-calculated paths should obey the rule of "longest match" whereby a longer match is better than a shorter one, no matter what the metric.

There is one caveat to be borne in mind. The paths generated thus far are NOT directly suitable for use in generating paths to obey a received DTL. If we are in node B.1 and we receive a DTL which specified (A, B, C) then we must generate a path across B, to C. That path may not traverse an additional top level peer group D. In contrast, when we originate a call, it is perfectly ok to go through D to C, even when C is a neighbor.

To handle this, one additional data structure should be pregenerated.
Go through the list of links, and find all Uplinks. create two tables, DownCount[AllNodeCount] and DownLinks[AllNodeCount, MaxOccurences]; DownCount tells how many uplinks there are to a given node. DownLinks contains the index of each uplink. When given a DTL to obey, look up the destination. If DownCount is 0, give up. Otherwise, the set of candidate paths are the paths to the sources of the DownLinks[DTL_Entry, ...]. These are all within the peer group, and therefore are valid destinations. By the re-entry suppression of the generation algorithm, the paths are all valid.

On demand calculation

When the set of precalculated paths does not meet the need, it is necessary to attempt on demand calculation of a route. This can be done either for originating a call, or for propagating a call with a DTL. If a DTL is being processed, then the database must be reduced to those nodes within the scope of the current DTL entry, and the single node that is the next hop on the DTL.

The algorithm proceeds almost precisely as before except that a different link admission test is used, and a check for completion is added. (Path recording must also actively accumulate all additive metrics in a forward and backwards direction.

The new utility routine is:

```
Admit_Link(Call, Link, C, Q, path_metrics)
    /* The service category comes from the call, but it is simpler to pass it
       as a parameter */
     if (not Admit_Link_All(Link, C, Q)) return (false);

     /* Then test whether the call will fit on the link */
     if (not GCAC(Call.bandwidth, Link, C)) return false;
     if (not GCAC(Call.reverse_bandwidth, Links[Link].PairedLink, C))
```

```
            return (false);

        /* now test the cumulative metrics */
        foreach (cumulative_metric m)
                    if (Outside_Range (Call.m, Cumulate(Links[Link].Classinfo[C].m,
                            path_metrics[m])))
                    return (false);

                if (Outside_Range (Call.m,
                Cumulate(Links[Links[Link].PairedLink].Classinfo[C].m, path_metrics[reverse+m])))
                    return (false);

        foreach (attribute a)
                if (Invalid_attr(Call.a, Links[Link].Classinfo[C].a)) return (false);
                if (Invalid_attr(Call.a,
                            Links[Links[Link].PairedLink].Classinfo[C].a)) return (false);

        return (true);
```

Reachable(Destination, Node):
If destination is a node (from a DTL) Reachable returns true of Node is that destination. If Destination is an address, Reachable returns true if Node advertises reachability to a prefix of the destination address.

```
Generate_routes(Dest, Call, C, Q, Tiebreakers)
      /* Generate routes to the destination.  The service category has been determined to be C, and
      Q/Tiebreakers are the selected optimization criteria */

      integer i, j, k;
      path p, q;

        /* check if we are already there */
        if (Reachable(Dest, This))
                /* return NIL; */
        /* Due to longest match: */
        save_path(NULL);

        for (i = 1 to AllNodeCount)
                BestCosts[i] = -1;
                for (j = 1 to Pathcount)
                        BestPaths[i, j] = NIL;

        Tent = Empty;

        i = This;
        while (i > 0)
                BestCosts[i] = 0;
                i = AllNodes[i].ParentNode;

        for (i = 1 to AllNodes[This].LinkCount)
                /* Make sure that the links can be used for this service category
                      and optimization parameter */
                if (Admit_Link(Call, OutLinks[This,i], C, Q, NIL))
```

```
                /* The Path carries the cumulative cost for the prime condition */ p =
                Make_Path(This, OutLinks[This, i], C, Q);
                Add_to_Tent(p, C, Q, Tiebreakers);

    while (not Tent_Empty(Tent))
        p = Remove_Smallest(Tent);
        i = Last_Node(p);
        k = Last_Link(p);

        /* check if we have gotten there */
        if (Reachable(Dest, i))
            save_path(p);
            /* it would be desirable to exit at this point.  However,
               in order to enforce longest match, it is either necessary to
               know which node has the longest match for the destination
               address, or to generate all of them */

        /* Check if the cost is good enough to be useable */
        else if (Record_Path(p, i, C, Q, Tiebreakers))

           /* Now examine the outlinks, and put them into the tent */
           for (j = 1 to AllNodes[i].LinkCount)
               /* check if the link is useable */
               if (Links[k].PairedLink != j /* do not backtrack directly */ &&
                Admit_Link(Call, OutLinks[i, j]), C, Q, metrics(p))
                   q = Append_Path(p, OutLinks[i, j], C, Q);
                   Add_to_Tent(q);
```

It should be noted that while this example runs the precalculation on the entire database, some implementations may choose to run it only on the lowest level peer group, or only for some sub-group of the destinations which by configuration are known to be significant. The on-demand portion of the algorithm can then be used to fill in.

It should also be noted that the correct optimization criteria for pre-calculated routes is very sensitive to the actual calls that the users are setting up. Implementers may want to consider allowing some customer choice in the selection of what to pre-calculate.

DTL Generation:

Having generated reachability paths, and inverted them into reachability tables, it is still necessary to transform a path selection into a DTL stack.
The procedure is very straight forward, and works for all known path generation algorithms:

1) Create a DTL stack with a single DTL containing only the starting node at the lowest level. Set a pointer to the start of the generated route.

Repeat the following two steps:

2) A) If the current path entry is a regular node, add it to the DTL.
   B) If the current path entry is a complex nodal metric entry, add the regular node it is part of to the DTL if it is not already the current entry in the DTL.

3) A) If the link from the current entry is a horizontal link then

a) if it points to a complex nodal metric node, put the port number of this link in the current DTL entry

b) advance the pointer in the route

B) The link from the current entry in the route is an uplink:

a) End the current DTL

b) for each hierarchical level between the current one and the one containing the next route entry, create a single entry DTL containing the nodeid for the peer group which contains the starting node at that level

c) Create a new DTL with an entry for the peer group which contains the starting node at the level of the next entry in the route, and advance the route pointer

C) If there is no next entry in the route, you are done

This will give a DTL with those links explicitly identified which the technical analysis has said need to be called out, and no others.

23. Appendix I: Using PNNI To Connect Non-PNNI Routing Domains

PNNI may be used to connect routing domains running non-PNNI routing protocols into larger PNNI routing domains. One way for a non-PNNI routing domain to appear in a larger PNNI routing domain is by representing the non-PNNI routing domain as a single node. It can appear either as a lowest-level node, or as a logical group node. In either case, the complex node representation may be used to represent the connectivity across the non-PNNI routing domain.

The mechanisms required in non-PNNI routing protocols in order for non-PNNI routing domains to be represented as PNNI nodes are outside the scope of this specification. However, this appendix lists several requirements of PNNI nodes that may affect the operation of such non-PNNI routing domains.

If a routing domain running a non-PNNI routing protocol appears as a lowest level node in a larger PNNI routing domain:

- Connectivity must be established between all ports on the node. If the non-PNNI routing domain partitions, each partition must either become a separate lowest level node, or all links from the partition must not be advertised as reachable from the PNNI routing domain.
- The lowest-level node must be capable of exchanging topology information through the Hello protocol (over physical links and VPCs), Database Synchronization, and Flooding. The information flow over all ports must be consistent, based on the same topology database. The lowest-level node must participate in the Peer Group Leader Election.
- The lowest-level node will appear in DTLs. SETUP or ADD PARTY messages entering the lowest-level node must be routed across the current port specified in the DTL, if any, to the next node in the DTL stack.
- If the lowest-level node is capable of becoming peer group leader of its peer group, the switching system instantiating the parent logical group node must be capable of establishing and receiving SVC-based RCCs through any of its ports, and of exchanging topology information over those SVCs through the SVC-based RCC Hello protocol, the Horizontal Link Hello Protocol, Database Synchronization, and Flooding.

If a routing domain running a non-PNNI routing protocol appears as a logical group node in a larger PNNI routing domain:

- Connectivity must be established between all ports on the logical group node. If the non-PNNI routing domain partitions, each partition must either become a separate logical group node, or all links from the partition must not be advertised as reachable from the PNNI routing domain.
- Lowest-level Hellos on PNNI physical links and VPCs must appear as if they are attached to lowest-level border nodes contained in the logical group node.
- The lowest-level border nodes only appear in terms of the operation of outside Hellos. They must be able to create and to process nodal hierarchy lists, and to create ULIAs for those links. They do not appear in DTLs, and so connectivity between the lowest-level border nodes is not required, except for satisfying the above conditions.
- The physical system instantiating the logical group node must be capable of establishing and receiving SVC-based RCCs, and of exchanging topology information over those SVCs through the SVC-based RCC Hello protocol, the Horizontal Link Hello Protocol, Database Synchronization, and Flooding. The LGN must participate in the Peer Group Leader Election.
- The logical group node will appear in DTLs. SETUP or ADD PARTY messages entering the logical group node must be routed across the current port specified in the DTL, if any, to the next node in the DTL stack.

Other methods for representing non-PNNI routing domains in larger PNNI routing domains are possible. Representation of a non-PNNI routing domain as multiple connected PNNI nodes is not described in this specification.