The ATM Forum

## Technical Committee

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

**af-pnni-0055.000**
**Letter Ballot**
**March 1996**

(c) 1996 The ATM Forum. All Rights Reserved. No part of this publication may be reproduced in any form or by any means.

The information in this publication is believed to be accurate as of its publication date. Such information is subject to change without notice and the ATM Forum is not responsible for any errors. The ATM Forum does not assume any responsibility to update or correct any information in this publication. Notwithstanding anything to the contrary, neither the ATM Forum nor the publisher make any representation or warranty, expressed or implied, concerning the completeness, accuracy, or applicability of any information contained in this publication. No liability of any kind shall be assumed by The ATM Forum or the publisher as a result of reliance upon any information contained in this publication.

The receipt or any use of this document or its contents does not in any way create by implication or otherwise:

- Any express or implied license or right to or under any ATM Forum member company's patent, copyright, trademark, or trade secret rights which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor
- Any warranty or representation that any ATM Forum member companies will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor
- Any form of relationship between any ATM Forum member companies and the recipient or user of this document.

Implementation or use of specific ATM standards or recommendations and ATM Forum specifications will be voluntary, and no company shall agree or be obliged to implement them by virtue of participation in the ATM Forum. The ATM Forum is a non-profit international organization accelerating industry cooperation on ATM technology. The ATM Forum does not, expressly or otherwise, endorse or promote any specific products or services.

# Table of Contents

**1. INTRODUCTION ..... 1**

1.1 OVERVIEW ..... 1
1.2 REFERENCE MODELS ..... 1
1.3 PNNI STATUS ..... 2
1.4 DOCUMENT ORGANIZATION ..... 2
1.5 REFERENCES ..... 2

**2. TERMINOLOGY ..... 4**

2.1 ABBREVIATIONS ..... 4
2.2 DEFINITIONS ..... 5
2.3 NOTATION AND CONVENTIONS ..... 12

**3. PNNI ROUTING DESCRIPTION ..... 13**

3.1 PHYSICAL NETWORK ..... 13
3.2 THE LOWEST HIERARCHICAL LEVEL ..... 14
*3.2.1 Peer Groups and Logical Nodes ..... 14*
*3.2.2 Logical Links And Their Initialization ..... 15*
*3.2.3 Information Exchange in PNNI ..... 15*
*3.2.3.1 Nodal Information ..... 16*
*3.2.3.2 Topology State Information ..... 16*
*3.2.3.3 Reachability Information ..... 16*
*3.2.3.4 Initial Topology Database Exchange ..... 16*
*3.2.3.5 Flooding ..... 17*
*3.2.4 Peer Group Leader ..... 17*
3.3 ONE HIERARCHICAL LEVEL UP ..... 18
*3.3.1 Logical Group Nodes ..... 18*
*3.3.2 Feeding Information Up The Hierarchy ..... 20*
*3.3.3 Feeding Information Down The Hierarchy ..... 20*
*3.3.4 Uplinks (Revisiting The Hello Protocol) ..... 20*
*3.3.5 PNNI Routing Control Channels ..... 22*
*3.3.6 Revisiting Initial Topology Database Exchange ..... 23*
*3.3.7 Horizontal Links ..... 23*
*3.3.8 Topology Aggregation ..... 24*
*3.3.8.1 Link Aggregation ..... 24*
*3.3.8.2 Nodal Aggregation ..... 24*
3.4 COMPLETING THE PNNI ROUTING HIERARCHY ..... 26
*3.4.1 Progressing To The Highest Level Peer Group ..... 27*
*3.4.2 Uplinks And Horizontal links Revisited ..... 27*
*3.4.3 Recursion in the Hierarchy ..... 28*
3.5 ADDRESS SUMMARIZATION & REACHABILITY ..... 29
*3.5.1 Address Scoping ..... 32*
*3.5.1.1 An Example of Address Summarization with Address Scoping ..... 32*
3.6 SINGLE NODE PERSPECTIVE ..... 35
3.7 PATH SELECTION ..... 36
*3.7.1 Generic Connection Admission Control ..... 38*

**4. PNNI SIGNALLING DESCRIPTION ..... 39**

4.1 INTRODUCTION ..... 39
4.2 OVERVIEW ..... 39
4.3 ASSOCIATED SIGNALLING ..... 40

4.4 DESIGNATED TRANSIT LISTS ....40
4.5 CRANKBACK ....40
4.6 SOFT PVPCS AND PVCCS ....40
4.7 AN EXAMPLE OF CONNECTION SETUP WITH CRANKBACK ....41
4.8 SUPPORTING ANYCAST WITH SCOPING ....45

**5. PNNI ROUTING SPECIFICATION ....47**

5.1 GENERAL OPERATIONAL PROCEDURES ....47
*5.1.1 Timers ....47*
*5.1.2 Packet Transmission ....47*
*5.1.3 Packet Validation ....47*
5.2 ADDRESSING ....48
*5.2.1 ATM Addressing Convention ....48*
*5.2.2 Node Addresses ....49*
*5.2.3 End System Addresses ....49*
5.3 IDENTIFIERS AND INDICATORS ....49
*5.3.1 Level Indicators ....49*
*5.3.2 Peer Group Identifiers ....50*
*5.3.3 Node Identifiers ....50*
*5.3.4 Port Identifiers and Logical Links ....51*
*5.3.5 Aggregation Tokens ....51*
*5.3.6 Scope of Addresses ....52*
5.4 LOGICAL LINKS ....52
5.5 PNNI ROUTING CONTROL CHANNELS ....52
*5.5.1 Specification of the AAL used by the PNNI Routing Control Channel ....53*
*5.5.2 Traffic Contract for RCCs over Physical Links ....53*
*5.5.3 Traffic Contract for RCCs over VPCs ....53*
*5.5.4 SVCC-Based RCC Connection Establishment Overview ....54*
5.5.4.1 SETUP Message Contents ....54
5.5.4.1.1 AAL Parameters ....54
5.5.4.1.2 ATM Traffic Descriptor ....55
5.5.4.1.3 Broadband Bearer Capability ....55
5.5.4.1.4 Broadband Low Layer Information ....56
5.5.4.1.5 QoS Parameter ....56
5.5.4.1.6 Extended QoS Parameters ....56
5.5.4.1.7 End-to-End Transit Delay ....57
5.5.4.1.8 Called Party Number ....57
5.5.4.1.9 Calling Party Number ....57
5.5.4.1.10 Connection Identifier ....57
5.5.4.1.11 DTL ....57
5.5.4.2 CONNECT Message Contents ....57
5.5.4.2.1 AAL Parameters ....58
5.5.4.2.2 Broadband Low Layer Information ....58
5.5.4.2.3 Connection Identifier ....58
5.5.4.3 RELEASE and RELEASE COMPLETE Message Contents ....58
*5.5.5 SVCCs when One End is not an LGN ....58*
*5.5.6 Establishing and Maintaining SVCCs between Logical Group Nodes ....59*
5.5.6.1 SVCC Establishment ....59
5.5.6.2 Inconsistent Information and Partitioned Neighbor Peer Groups ....60
5.5.6.3 Detailed Mechanisms for Establishment and Maintenance of SVCCs ....60
5.6 THE HELLO PROTOCOL ....62
*5.6.1 PNNI Version Negotiation ....62*
*5.6.2 Hellos Over Physical Links and VPCs ....62*
5.6.2.1 The Hello State Machine ....62
5.6.2.1.1 The Hello Data Structure ....63
5.6.2.1.2 Hello States ....65

5.6.2.1.3 Events causing Hello state changes ..... 66
5.6.2.1.4 Description of The Hello State Machine ..... 68
5.6.2.2 Sending Hellos ..... 73
5.6.2.2.1 Originating and sending ULIAs in Hellos on Outside Links ..... 74
5.6.2.3 Receiving Hellos ..... 74
5.6.2.3.1 Processing ULIAs received in Hellos on outside links ..... 75
*5.6.3 The LGN Hello Protocol ..... 75*
5.6.3.1 SVCC-Based RCC Hello Protocol ..... 76
5.6.3.1.1 The SVCC-Based RCC Hello Data Structure ..... 77
5.6.3.1.2 SVCIntegrityTimer ..... 77
5.6.3.1.3 Loss of Neighbor Relationship ..... 78
5.6.3.2 LGN Horizontal Link Hello Protocol ..... 78
5.6.3.2.1 The LGN Horizontal Link Hello Data Structure ..... 79
5.6.3.2.2 LGN Horizontal Link States ..... 79
5.6.3.2.3 LGN Horizontal Link Events ..... 80
5.6.3.2.4 Description of The Horizontal Link Hello State Machine ..... 82
5.6.3.3 Overall Procedures ..... 84
5.7 DATABASE SYNCHRONIZATION ..... 85
*5.7.1 The Neighboring Peer Data Structure ..... 86*
*5.7.2 Neighboring Peer States ..... 88*
*5.7.3 Events causing neighboring peer state changes ..... 89*
*5.7.4 The Neighboring Peer State Machine ..... 90*
*5.7.5 Sending Database Summary Packets ..... 92*
*5.7.6 Receiving Database Summary Packets ..... 93*
*5.7.7 Sending PTSE Request Packets ..... 95*
*5.7.8 Receiving PTSE Request Packets ..... 96*
5.8 TOPOLOGY DESCRIPTION AND DISTRIBUTION ..... 96
*5.8.1 Topology Information ..... 96*
5.8.1.1 Topology State Parameters ..... 96
5.8.1.1.1 Link State Parameters ..... 97
5.8.1.1.2 Nodal State Parameters ..... 97
5.8.1.1.3 Resource Availability Information Group (RAIG) ..... 97
5.8.1.1.3.1 Resource Availability Information Group Flags ..... 98
5.8.1.1.3.2 Cell Delay Variation (CDV) ..... 98
5.8.1.1.3.3 Maximum Cell Transfer Delay (maxCTD) ..... 98
5.8.1.1.3.4 Administrative Weight (AW) ..... 98
5.8.1.1.3.5 Cell Loss Ratio for CLP=0 ($CLR_0$) ..... 99
5.8.1.1.3.6 Cell Loss Ratio for CLP=0+1 ($CLR_{0+1}$) ..... 99
5.8.1.1.3.7 Maximum Cell Rate (maxCR) ..... 99
5.8.1.1.3.8 Available Cell Rate (AvCR) ..... 99
5.8.1.1.3.9 Cell Rate Margin (CRM) ..... 100
5.8.1.1.3.10 Variance Factor (VF) ..... 100
5.8.1.2 Nodal Information ..... 100
5.8.1.2.1 ATM End System Address of the Node ..... 100
5.8.1.2.2 Leadership Priority ..... 100
5.8.1.2.3 Nodal Information Flags ..... 100
5.8.1.2.4 Preferred Peer Group Leader Node ID ..... 101
5.8.1.2.5 Next Higher Level Binding Information ..... 102
5.8.1.2.5.1 Parent Logical Group Node ID ..... 102
5.8.1.2.5.2 Parent Logical Group Node's ATM End System Address ..... 102
5.8.1.2.5.3 Parent Peer Group ID ..... 102
5.8.1.2.5.4 Node ID of PGL of Parent Peer Group ..... 102
5.8.1.3 Reachability Information ..... 102
5.8.1.3.1 Internal Reachable Addresses ..... 102
5.8.1.3.2 Exterior Reachable ATM Addresses ..... 103
*5.8.2 PTSPs and PTSEs ..... 104*
5.8.2.1 PTSPs ..... 104
5.8.2.2 PTSEs ..... 105

5.8.2.2.1 PTSE Header Contents ..... 105
5.8.2.2.2 PTSE Checksum algorithm ..... 106
5.8.2.2.3 PTSE identification and PTSE instance identification ..... 106
5.8.2.2.4 Comparison of PTSE instances ..... 106
*5.8.3 Flooding ..... 107*
5.8.3.1 Overview ..... 107
5.8.3.2 Sending PTSPs ..... 107
5.8.3.3 Receiving a PTSP ..... 108
5.8.3.4 Processing of Peer Retransmission List ..... 110
5.8.3.5 Sending of PTSE Acknowledgments ..... 110
5.8.3.6 Receiving Acknowledgment Packets ..... 110
5.8.3.7 Origination of a New PTSE or a new PTSE Instance ..... 110
5.8.3.8 Expiration of a PTSE's Lifetime ..... 111
5.8.3.9 Removal of PTSEs from the topology database ..... 111
*5.8.4 Aging PTSEs in the Topology Database ..... 112*
5.8.4.1 Computing the Remaining Lifetime of a PTSE ..... 112
5.8.4.2 Refreshing Self-originated PTSEs ..... 112
*5.8.5 Triggered Updates of Topology Information ..... 112*
5.8.5.1 Significant Change Events for PNNI Topology State Elements ..... 112
5.8.5.2 Processing Significant Changes to PTSEs ..... 113
5.8.5.2.1 Changes in Nodal Information Groups ..... 113
5.8.5.2.2 Changes in Internal Reachable ATM Addresses IGs ..... 113
5.8.5.2.3 Changes in Exterior Reachable ATM Addresses IGs ..... 113
5.8.5.2.4 Changes in other information groups ..... 114
5.8.5.2.5 Changes to Resource Availability Information ..... 114
5.8.5.2.5.1 Administrative Weight ( AW ) ..... 114
5.8.5.2.5.2 Cell Loss Ratio ( $CLR_0$ ) ..... 114
5.8.5.2.5.3 Cell Loss Ratio ( $CLR_{0+1}$ ) ..... 114
5.8.5.2.5.4 Available Cell Rate ( AvCR ) ..... 114
5.8.5.2.5.5 Maximum Cell Transfer Delay ( maxCTD ) ..... 115
5.8.5.2.5.6 Cell Delay Variation ( CDV ) ..... 116
5.8.5.2.5.7 Maximum Cell Rate ( maxCR ) ..... 117
5.8.5.2.5.8 Cell Rate Margin ( CRM ) and Variance Factor ( VF ) ..... 117
5.9 ADVERTISING AND SUMMARIZING REACHABLE ADDRESSES ..... 117
*5.9.1 Scope of Advertisement of Addresses ..... 117*
*5.9.2 Summary Address and Suppressed Summary Address ..... 118*
*5.9.3 Native Addresses ..... 118*
*5.9.4 Foreign Addresses ..... 118*
*5.9.5 Group Addresses ..... 119*
*5.9.6 Exterior Reachable Addresses ..... 119*
5.10 HIERARCHY ..... **Error! Bookmark not defined.**
*5.10.1 Peer Group Leader ..... 119*
5.10.1.1 Peer Group Leader Election Algorithm ..... 119
5.10.1.1.1 The PGL Election Data Structure ..... 119
5.10.1.1.2 PGL Election States ..... 120
5.10.1.1.3 Events Causing Neighboring Peer State Changes ..... 123
5.10.1.1.4 The PGL Election State Machine ..... 124
5.10.1.1.5 Sending a Nodal Information PTSE ..... 128
5.10.1.1.6 Choosing Preferred PGL ..... 128
5.10.1.2 Leadership Priority ..... 129
5.10.1.2.1 Group Leader Increment ..... 129
5.10.1.2.2 Distinguished Values and Range Restrictions ..... 129
5.10.1.2.3 Range Recommendations ..... 129
5.10.1.2.4 Implications of Hierarchy ..... 129
5.10.1.3 Default Configuration Consistency ..... 130
5.10.1.4 Operation Without a Peer Group Leader ..... 130
5.10.1.5 Exchange of Nodal Hierarchy Information on Outside Links ..... 130
*5.10.2 Advertising Uplinks ..... 131*

5.10.2.1 Advertising Uplinks from Lowest Level Nodes ..... 131
5.10.2.2 Reverse Direction Information in Uplinks ..... 131
*5.10.3 Topology Aggregation ..... 132*
5.10.3.1 Link Aggregation and Binding Information ..... 132
5.10.3.2 Simple Node Representation ..... 134
5.10.3.3 Complex Node Representation ..... 134
*5.10.4 Feeding Information Down the Hierarchy ..... 136*
*5.10.5 Flushing PTSEs in a Multi-Level Hierarchy ..... 138*
5.11 PEER GROUP PARTITIONS ..... 138
*5.11.1 Representation of the partition in the hierarchy ..... 138*
*5.11.2 Routing in the Presence of Partitions ..... 139*
*5.11.3 What to Aggregate as Peer Group Leader ..... 139*
5.12 OPERATION OF A NODE WHEN IN A TOPOLOGY DATABASE OVERLOAD STATE ..... 140
*5.12.1 Requirement for operation in topology database overload state ..... 140*
5.12.1.1 Minimum state required ..... 140
5.12.1.2 Disallowed Functions ..... 140
5.12.1.3 PGL election ..... 140
5.12.1.4 PTSP reception and transmission ..... 140
5.12.1.5 Periodic resynchronization ..... 141
*5.12.2 Recommendations for operation in topology database overload state ..... 141*
5.12.2.1 Call processing ..... 141
5.12.2.2 Topology database handling ..... 142
5.13 PATH SELECTION ..... 142
*5.13.1 Eligible Entities for Path Selection ..... 143*
*5.13.2 Link Constraints, Node Constraints, and Path Constraints ..... 143*
*5.13.3 CLR Selection for CBR andVBR Service ..... 144*
*5.13.4 Generic CAC Algorithm for CBR and VBR Services ..... 144*
5.13.4.1 Parameter Choice for GCAC ..... 144
5.13.4.2 Complex GCAC ..... 146
5.13.4.2.1 Algorithm ..... 146
5.13.4.2.2 Special Cases ..... 146
5.13.4.2.3 Optional Improvement to Algorithm ..... 146
5.13.4.3 Simple GCAC ..... 146
*5.13.5 Generic CAC Algorithm for Best-Effort Service ..... 147*
5.13.5.1 Minimum Acceptable ATM Traffic Descriptor ..... 147
5.14 PACKET FORMATS ..... 147
*5.14.1 Notation ..... 148*
*5.14.2 Information Group Tags ..... 148*
5.14.2.1 Mandatory Tag ..... 148
5.14.2.2 Summarizing Tag ..... 149
5.14.2.3 Transitive Tag ..... 149
5.14.2.4 Mandatory Non-Transitive Information Groups ..... 149
5.14.2.5 Examples ..... 150
5.14.2.5.1 Advisory Metrics and Attributes ..... 150
5.14.2.5.2 Local Information Groups ..... 150
5.14.2.5.3 User lists ..... 150
5.14.2.5.4 Security Labels ..... 150
5.14.2.6 Information Group Tag Bit Definitions ..... 150
*5.14.3 Information Group Summary ..... 152*
*5.14.4 PNNI Packet Header ..... 154*
*5.14.5 The Resource Availability Information Group ..... 154*
*5.14.6 Uplink Information Attribute ..... 155*
*5.14.7 Transit Network ID ..... 156*
*5.14.8 PNNI Hellos ..... 157*
5.14.8.1 The Aggregation Token IG ..... 158
5.14.8.2 The Nodal Hierarchy List IG ..... 158
5.14.8.3 The LGN Horizontal Link Extension IG ..... 159

*5.14.9 PNNI Topology State Packets (PTSP)* ..... *159*
5.14.9.1 Restricted Information Groups ..... 161
5.14.9.1.1 The Nodal state parameters IG ..... 161
5.14.9.1.2 The Nodal Information Group ..... 161
5.14.9.1.3 The Internal Reachable ATM Addresses Information Group ..... 163
5.14.9.1.4 The Exterior Reachable ATM addresses Information Group ..... 163
5.14.9.1.5 The Horizontal Links Information Group ..... 164
5.14.9.1.6 The Uplinks Information Group ..... 165
5.14.9.2 Unrestricted Information Groups ..... 165
5.14.9.3 Unknown Information Groups ..... 166
5.14.9.4 Processing PTSEs Violating Restrictions ..... 166
5.14.9.5 PTSE Error Scenarios ..... 166
5.14.9.5.1 Nodal Info Group in Nodal Info PTSE ..... 167
5.14.9.5.2 Next higher Level Binding TLV in Nodal Info TLV ..... 167
5.14.9.5.3 Nodal State Parameters TLV ..... 167
5.14.9.5.4 Horizontal Link TLV ..... 168
5.14.9.5.5 Uplink TLV ..... 168
5.14.9.5.6 ULIA ..... 168
5.14.9.5.7 RAIGs ..... 168
5.14.9.6 Nodal Info PTSE and its connection to other PTSEs issued by the node ..... 168
*5.14.10 PTSE Acknowledgment Packets* ..... *169*
*5.14.11 Database Summary Packets* ..... *169*
*5.14.12 PTSE Request Packets* ..... *171*
*5.14.13 System Capabilities* ..... *172*

**6. PNNI SIGNALLING SPECIFICATION ..... 173**

6.1 PNNI MODEL ..... 173
*6.1.1 Definitions* ..... *173*
*6.1.2 Protocol model* ..... *173*
6.1.2.1 Signalling layer ..... 174
6.1.2.2 Signalling adaptation layer ..... 174
6.1.2.3 ATM layer ..... 175
6.2 OVERVIEW OF CALL/CONNECTION CONTROL ..... 175
*6.2.1 ATM Point-to-point call states* ..... *175*
6.2.1.1 Null (NN0) ..... 175
6.2.1.2 Call Initiated (NN1) ..... 175
6.2.1.3 Call Proceeding Sent (NN3) ..... 175
6.2.1.4 Alerting Delivered (NN4) ..... 175
6.2.1.5 Call Present (NN6) ..... 175
6.2.1.6 Alerting Received (NN7) ..... 176
6.2.1.7 Call Proceeding Received (NN9) ..... 176
6.2.1.8 Active (NN10) ..... 176
6.2.1.9 Release Request (NN11) ..... 176
6.2.1.10 Release Indication (NN12) ..... 176
*6.2.2 States associated with global call reference* ..... *176*
6.3 MESSAGE FUNCTIONAL DEFINITIONS AND CONTENTS ..... 176
*6.3.1 Messages for ATM point-to-point call and connection control* ..... *177*
6.3.1.1 ALERTING ..... 177
6.3.1.2 CALL PROCEEDING ..... 178
6.3.1.3 CONNECT ..... 178
6.3.1.4 RELEASE ..... 179
6.3.1.5 RELEASE COMPLETE ..... 180
6.3.1.6 SETUP ..... 180
6.3.1.7 STATUS ..... 182
6.3.1.8 STATUS ENQUIRY ..... 182
6.3.1.9 NOTIFY ..... 183
*6.3.2 Additional or modified messages for the support of 64kbit/s based ISDN circuit mode services* 183
6.3.2.1 ALERTING ..... 184

6.3.2.2 CONNECT ..... 184
6.3.2.3 PROGRESS ..... 185
6.3.2.4 RELEASE ..... 185
6.3.2.5 SETUP ..... 186
*6.3.3 Messages used with the global call reference ..... 186*
6.3.3.1 RESTART ..... 186
6.3.3.2 RESTART ACKNOWLEDGE ..... 187
*6.3.4 Messages for Point-to-multipoint call and connection control ..... 187*
6.3.4.1 ADD PARTY ..... 188
6.3.4.2 ADD PARTY ACKNOWLEDGE ..... 189
6.3.4.3 PARTY ALERTING ..... 189
6.3.4.4 ADD PARTY REJECT ..... 190
6.3.4.5 DROP PARTY ..... 191
6.3.4.6 DROP PARTY ACKNOWLEDGE ..... 191
6.4 GENERAL MESSAGE FORMAT AND INFORMATION ELEMENT CODING ..... 192
*6.4.1 Overview ..... 192*
*6.4.2 Protocol discriminator ..... 192*
*6.4.3 Call reference ..... 192*
*6.4.4 Message type and message length ..... 192*
6.4.4.1 Message type ..... 192
6.4.4.2 Message length ..... 193
*6.4.5 Variable length information elements ..... 193*
6.4.5.1 Coding rules ..... 193
6.4.5.2 Extension of codesets ..... 194
6.4.5.3 Broadband locking shift procedures ..... 194
6.4.5.4 Broadband non-locking shift procedures ..... 194
6.4.5.5 ABR additional parameters ..... 194
6.4.5.6 ABR setup parameters ..... 195
6.4.5.7 Alternative ATM traffic descriptor ..... 195
6.4.5.8 ATM adaptation layer parameters ..... 195
6.4.5.9 ATM traffic descriptor ..... 195
6.4.5.10 Broadband bearer capability ..... 196
6.4.5.11 Broadband high layer information ..... 196
6.4.5.12 Broadband low layer information ..... 196
6.4.5.13 Broadband repeat indicator ..... 196
6.4.5.14 Call state ..... 196
6.4.5.15 Called party number ..... 197
6.4.5.16 Called party subaddress ..... 197
6.4.5.17 Calling party number ..... 198
6.4.5.18 Calling party subaddress ..... 198
6.4.5.19 Cause ..... 198
6.4.5.20 Connected number ..... 198
6.4.5.21 Connected subaddress ..... 198
6.4.5.22 Connection identifier ..... 198
6.4.5.23 Connection scope selection ..... 198
6.4.5.24 End-to-end transit delay ..... 198
6.4.5.25 Extended QoS parameters ..... 200
6.4.5.26 Minimum acceptable ATM traffic descriptor ..... 200
6.4.5.27 Notification indicator ..... 200
6.4.5.28 Quality of service parameter ..... 200
6.4.5.29 Restart indicator ..... 200
6.4.5.30 Transit network selection ..... 200
6.4.5.31 Generic identifier transport ..... 201
*6.4.6 Information elements specific to PNNI ..... 201*
6.4.6.1 Calling party soft PVPC or PVCC ..... 201
6.4.6.2 Called party soft PVPC or PVCC ..... 202
6.4.6.3 Crankback ..... 203
6.4.6.4 Designated transit list ..... 206

*6.4.7 Information Elements for the support of 64 kbit/s based circuit mode services* ..... *208*
6.4.7.1 Narrow-band bearer capability ..... 208
6.4.7.2 Narrow-band high layer compatibility ..... 208
6.4.7.3 Narrow-band low layer compatibility ..... 208
6.4.7.4 Progress indicator ..... 208
*6.4.8 Information Elements for Point-to-Multipoint Call/connection Control* ..... *208*
6.4.8.1 Endpoint reference ..... 208
6.4.8.2 Endpoint state ..... 208
6.5 CALL/CONNECTION CONTROL PROCEDURES FOR ATM POINT-TO-POINT CALLS ..... 208
*6.5.1 Establishment of a signalling AAL* ..... *209*
*6.5.2 Call/connection establishment* ..... *209*
6.5.2.1 Call/connection request ..... 209
6.5.2.2 Connection identifier allocation/selection ..... 210
6.5.2.2.1 Associated signalling ..... 210
6.5.2.2.2 Non-associated signalling ..... 211
6.5.2.2.2.1 Allocation for switched virtual channels ..... 211
6.5.2.2.2.2 Allocation for switched virtual paths ..... 212
6.5.2.2.3 Use of VPCIs ..... 212
6.5.2.2.4 VPCI and VCI ranges ..... 213
6.5.2.3 Service category, traffic parameter, and QoS selection procedures ..... 213
6.5.2.3.1 Determination of service category ..... 213
6.5.2.3.2 Allowed combination of bearer capabilities, traffic parameters, and QoS ..... 213
6.5.2.3.3 Traffic parameter selection procedures ..... 213
6.5.2.3.4 Procedures for negotiation of traffic parameters during call/connection setup ..... 214
6.5.2.3.5 QoS parameter selection procedures ..... 215
6.5.2.3.6 Traffic parameter selection procedures for ABR connections ..... 217
6.5.2.3.7 Processing of Cumulative RM fixed round-trip time parameter for ABR connections ..... 219
6.5.2.4 Call/connection proceeding ..... 219
6.5.2.5 Call/connection alerting ..... 220
6.5.2.6 Call/connection connected ..... 220
6.5.2.6.1 Procedures for traffic parameter selection during call/connection acceptance ..... 220
6.5.2.6.2 Procedures for negotiation of traffic parameters during call/connection acceptance ..... 221
6.5.2.7 Failure of call establishment ..... 221
6.5.2.8 Generic identifier transport procedures ..... 221
*6.5.3 Call/connection clearing* ..... *221*
6.5.3.1 Terminology ..... 222
6.5.3.2 Exception conditions ..... 222
6.5.3.3 Clearing ..... 222
6.5.3.4 Clear collision ..... 222
*6.5.4 Call/connection collisions* ..... *223*
*6.5.5 Restart Procedures* ..... *223*
6.5.5.1 Sending RESTART ..... 223
6.5.5.2 Receipt RESTART ..... 223
6.5.5.3 Restart collision ..... 223
*6.5.6 Handling of error conditions* ..... *223*
6.5.6.1 Protocol discrimination error ..... 224
6.5.6.2 Message too short ..... 224
6.5.6.3 Call reference error ..... 224
6.5.6.3.1 Invalid call reference format ..... 224
6.5.6.3.2 Call reference procedural errors ..... 224
6.5.6.4 Message type or message sequence errors ..... 224
6.5.6.5 Message length error ..... 225
6.5.6.6 General information element errors ..... 225
6.5.6.6.1 Information element sequence ..... 225
6.5.6.6.2 Duplicated information elements ..... 225
6.5.6.6.3 Coding standard error ..... 225
6.5.6.7 Mandatory information element error ..... 225
6.5.6.7.1 Mandatory information element missing ..... 225
6.5.6.7.2 Mandatory information element content error ..... 225

6.5.6.8 Non-mandatory information element errors ..... 225
6.5.6.8.1 Unrecognized information element ..... 225
6.5.6.8.2 Non-mandatory information element content error ..... 226
6.5.6.8.3 Unexpected recognized information element ..... 226
6.5.6.9 Signalling AAL reset ..... 226
6.5.6.10 Signalling AAL failure ..... 226
6.5.6.11 Status enquiry procedure ..... 227
6.5.6.12 Receiving a STATUS message ..... 227
*6.5.7 Error procedures with explicit action indicator ..... 227*
6.5.7.1 Unexpected or unrecognized message type ..... 227
6.5.7.2 Information element error ..... 227
*6.5.8 Handling of messages with insufficient information ..... 228*
*6.5.9 Network node call control requirements ..... 228*
*6.5.10 Notification procedures ..... 228*
*6.5.11 Notification of interworking ..... 228*
*6.5.12 List of Timers ..... 228*
6.6 CALL/CONNECTION CONTROL PROCEDURES FOR POINT-TO-MULTIPOINT CALLS ..... 228
*6.6.1 Setup of the initial party ..... 229*
*6.6.2 Adding a party ..... 229*
*6.6.3 Party dropping ..... 230*
*6.6.4 Restart procedures ..... 230*
*6.6.5 Handling of error conditions ..... 230*
*6.6.6 List of timers for point-to-multipoint ..... 230*

**7. ANNEX A: DESIGNATED TRANSIT LIST ..... 231**

7.1 TERMINOLOGY ..... 231
7.2 INITIAL DTL PROCESSING ..... 231
*7.2.1 Procedures at the DTL originator ..... 231*
*7.2.2 Procedures at an entry border node ..... 232*
*7.2.3 Procedures at a node which is not the DTL originator or an entry border node ..... 233*
7.3 NEXT STEP IN DTL PROCESSING ..... 234
7.4 RECOMMENDATIONS ON INCLUSION OF PORT IDS IN DTLS ..... 235
7.5 PSEUDOCODE OF DTL PROCEDURES ..... 235
7.6 EXTERIOR ROUTES AND DTLS ..... 237

**8. ANNEX B: CRANKBACK PROCEDURES ..... 238**

8.1 SCOPE OF CRANKBACK PROCEDURES ..... 238
8.2 CRANKBACK CAUSE ..... 238
*8.2.1 Reachability Errors ..... 238*
8.2.1.1 Destination unreachable and transit network unreachable ..... 239
8.2.1.2 Next node unreachable ..... 239
*8.2.2 Resource Errors ..... 240*
8.2.2.1 Resource Errors Due To Service Category ..... 240
8.2.2.2 Resource Errors Due to Traffice and QoS Parameters ..... 240
8.2.2.3 Resource Errors Due to VPCI/VCI Allocation ..... 240
*8.2.3 DTL Processing Errors ..... 241*
8.3 PROCEDURES FOR CRANKBACK LEVEL AND BLOCKED TRANSIT ..... 241
*8.3.1 Procedures at the Point of Blocking ..... 241*
8.3.1.1 Blocking at a node ..... 241
8.3.1.2 Blocking at the preceding end of a link ..... 241
8.3.1.3 Blocking at the succeeding end of a link ..... 242
*8.3.2 Receiving a clearing message with a Crankback information element ..... 242*
8.3.2.1 Receiving a clearing message indicating blocking at this interface ..... 242
8.3.2.2 Receiving a clearing message at the entry border node at the crankback level ..... 243
8.3.2.2.1 Alternate routing ..... 243
8.3.2.2.2 Crankback to next higher level ..... 243

8.3.2.3 Additional procedures for point-to-multipoint calls/connections ....244
*8.3.3 Pseudocode of Crankback Procedures* ....*244*
8.4 CARRYING UPDATED TOPOLOGY STATE PARAMETERS IN CRANKBACK INFORMATION ELEMENT ....246

**9. ANNEX C: SOFT PERMANENT VIRTUAL CONNECTION PROCEDURES ....248**

9.1 MESSAGES ....248
9.2 PROCEDURES FOR POINT-TO-POINT SOFT PVPC/PVCCS ....248
*9.2.1 Initiating PVPC/PVCC establishment* ....*248*
*9.2.2 Receiving a CONNECT message at the calling NI* ....*249*
*9.2.3 Receiving a SETUP message at the called NI* ....*249*
9.2.3.1 Last PVPC segment VPI allocation/selection ....249
9.2.3.2 Last PVCC segment VPI/VCI allocation/selection ....249
9.3 PROCEDURES FOR POINT-TO-MULTIPOINT PVCCS ....250
*9.3.1 Adding a party at the originating interface* ....*250*
*9.3.2 Set up of the first party* ....*250*
9.3.2.1 Receiving a CONNECT message at the calling NI ....250
9.3.2.2 Receiving a SETUP message at the called NI ....250
*9.3.3 Adding a party* ....*250*
9.3.3.1 Receiving an ADD PARTY ACKNOWLEDGE message at the calling NI ....251
9.3.3.2 Receiving an ADD PARTY message at the called NI ....251
9.3.3.2.1 Last PVCC segment VPI/VCI and Endpoint reference allocation/selection ....251

**10. ANNEX D: ARCHITECTURAL CONSTANTS ....252**

**11. ANNEX E: ARCHITECTURAL VARIABLES ....253**

**12. ANNEX F: CONFIGURATION OF THE PNNI HIERARCHY ....255**

**13. ANNEX G: PNNI MINIMUM SUBSETS ....257**

**14. ANNEX H: THE PNNI MANAGEMENT INFORMATION BASE ....259**

**15. APPENDIX A: EXAMPLES OF MESSAGE SEQUENCES ....339**

15.1 GENERAL ....339
15.2 FINITE STATE MACHINE ....339
15.3 EXAMPLE OF MESSAGE SEQUENCES ....339
*15.3.1 Successful Call Establishment* ....*339*
*15.3.2 Unsuccessful Call Establishment* ....*340*
*15.3.3 Normal call clearing (from originator)* ....*341*
*15.3.4 Call termination by a Private network* ....*341*

**16. APPENDIX B: DERIVATION OF GENERIC CONNECTION ADMISSION CONTROL ....342**

16.1 GENERIC CAC FOR CBR AND VBR SERVICE CATEGORIES ....342
16.2 GENERIC CAC FOR UBR AND ABR SERVICE CATEGORIES ....344

**17. APPENDIX C: GUIDELINES FOR TOPOLOGY AGGREGATION ....345**

17.1 DIAMETER AND RADIUS ....345
17.2 PRESUMED NODAL STATE PARAMETERS ....345
17.3 SIGNIFICANT EXCEPTIONS ....345
17.4 USELESS EXCEPTIONS ....345
17.5 NUMBER OF EXCEPTIONS ....345

**18. APPENDIX D: MULTI-PEER GROUP SYSTEMS ....346**

18.1 TECHNICAL DETAILS ....346

*18.1.1 Peer Group Leader* ..... *346*
*18.1.2 Border Systems* ..... *346*
*18.1.3 Core System in a Very Branchy Network.* ..... *348*
*18.1.4 Backbone and Local Area Topology.* ..... *348*
18.2 PROPOSED DISPOSITION OF THIS SECTION ..... 349

**19. APPENDIX E: ALLOCATING RESOURCES IN TRANSIT PEER GROUPS ..... 350**

**20. APPENDIX F: PNNI PACKET SIZES ..... 351**

20.1 PNNI HELLO PACKET (PHP) ..... 351
20.2 PNNI TOPOLOGY STATE PACKET (PTSP) ..... 351
20.3 PTSE ACKNOWLEDGEMENT PACKET (PTSE_AP) ..... 353
20.4 DATABASE SUMMARY PACKET (DBSP) ..... 353
20.5 PTSE REQUEST PACKET (PTSE_RP) ..... 354

**21. APPENDIX G. FLOODING PARAMETER SELECTION GUIDELINES ..... 355**

21.1 BASIC PARAMETER RELATIONS ..... 355
21.2 HIGHER HIERARCHICAL LEVELS ..... 355
21.3 VARYING THE PEERDELAYEDACKINTERVAL ..... 355
21.4 DEFAULT PEERDELAYEDACKINTERVAL VALUE ..... 355

**22. APPENDIX H: A SAMPLE ALGORITHM FOR ROUTE GENERATION ..... 357**

**23. APPENDIX I: USING PNNI TO CONNECT NON-PNNI ROUTING DOMAINS ..... 365**

# List of Figures

*Figure 1-1: Switching System Architectural Reference Model* ..... *1*
*Figure 3-1: Private ATM network with 26 switching systems and 33 bi-directional links.* ..... *13*
*Figure 3-2: Partially configured PNNI hierarchy showing lowest level nodes.* ..... *14*
*Figure 3-3: Partially configured PNNI hierarchy showing lowest hierarchical levels.* ..... *19*
*Figure 3-4: Uplinks* ..... *22*
*Figure 3-5: Complex node representation of LGN A.4 from Figure 3-3.* ..... *25*
*Figure 3-6: Example of a complete PNNI hierarchically configured network.* ..... *26*
*Figure 3-7: Uplinks revisited.* ..... *28*
*Figure 3-8: Address summarization in a PNNI hierarchy.* ..... *30*
*Figure 3-9: Sample Network* ..... *33*
*Figure 3-10: View of the network from nodes A.3.3, A.3.2, A.3.1, and A.3.4.* ..... *35*
*Figure 4-1: ATM Interfaces* ..... *39*
*Figure 4-2: A Sample Hierarchical Network* ..... *41*
*Figure 4-3: An Example of Crankback and Alternate Routing* ..... *42*
*Figure 5-1: NSAP Address Structure and Address Prefixes* ..... *48*
*Figure 5-2: Peer Group Identifier 14 octet encoding* ..... *50*
*Figure 5-3: An SVCC-Based RCC and horizontal links between a Lowest-Level Node and an LGN* ..... *59*
*Figure 5-4: Hello State Changes* ..... *65*
*Figure 5-5: LGN Horizontal Link Hello FSM.* ..... *80*
*Figure 5-6: Neighboring Peer State Changes (Database Synchronization)* ..... *88*
*Figure 5-7: PGL Election FSM.* ..... ***Error! Bookmark not defined.***
*Figure 5-8: Peer Group with Five Border Nodes* ..... *135*
*Figure 5-9: Default Node Representation* ..... *136*
*Figure 5-10: Complex Node Representation with Exceptions* ..... *136*
*Figure 6-1 PNNI interface* ..... *173*
*Figure 6-2 PNNI Control Plane* ..... *174*
*Figure 6-3 ALERTING Message Contents* ..... *178*
*Figure 6-4 CALL PROCEEDING Message Contents* ..... *178*
*Figure 6-5 CONNECT Message Contents* ..... *179*
*Figure 6-6 RELEASE message content* ..... *180*
*Figure 6-7 RELEASE COMPLETE message content* ..... *180*
*Figure 6-8 SETUP message content* ..... *182*
*Figure 6-9 STATUS message content* ..... *182*
*Figure 6-10 STATUS ENQUIRY message content* ..... *183*
*Figure 6-11 NOTIFY message content* ..... *183*
*Figure 6-12 ALERTING message contents* ..... *184*
*Figure 6-13 CONNECT message contents* ..... *185*
*Figure 6-14 PROGRESS message contents* ..... *185*
*Figure 6-15 RELEASE message content* ..... *186*
*Figure 6-16 SETUP message content* ..... *186*
*Figure 6-17 RESTART message content* ..... *187*
*Figure 6-18 RESTART ACKNOWLEDGE message content* ..... *187*
*Figure 6-19 ADD PARTY message content* ..... *189*
*Figure 6-20 ADD PARTY ACKNOWLEDGE message content* ..... *189*
*Figure 6-21 PARTY ALERTING message content* ..... *190*
*Figure 6-22 ADD PARTY REJECT message content* ..... *190*
*Figure 6-23 DROP PARTY message content* ..... *191*
*Figure 6-24 DROP PARTY ACKNOWLEDGE message content* ..... *192*
*Figure 6-25 Call state information element* ..... *197*
*Figure 6-26 End-to-end transit delay information element* ..... *199*
*Figure 6-27 Calling party soft PVPC or PVCC Information element* ..... *201*

*Figure 6-28 Called party soft PVPC or PVCC Information element* ..... *202*
*Figure 6-29 Crankback Information element* ..... *203*
*Figure 6-30 Designated transit list information element* ..... *207*
*Figure 13-1: PNNI Minimum Subsets* ..... *257*
*Figure 15-1: PNNI Finite State Machine* ..... *339*
*Figure 15-2: Setup, successful call* ..... *340*
*Figure 15-3: Setup, unsuccessful call* ..... *341*
*Figure 15-4: Normal call clearing by originator* ..... *341*
*Figure 15-5: Call termination by Network B* ..... *341*
*Figure 18-1: Border Node between Peer Groups A and B* ..... *347*
*Figure 18-2: Two Domains with Individual Border Nodes* ..... *348*
*Figure 18-3: Backbone and Local Area Topology* ..... *349*

# List of Tables

*Table 3-1: Configured Summary Addresses* ..... *29*
*Table 3-2: Summary Address List* ..... *30*
*Table 3-3: Advertised Reachable Address Prefixes* ..... *31*
*Table 3-4: Advertised Reachability Information* ..... *31*
*Table 5-1: Default UNI Scope to PNNI Level Mappings* ..... *52*
*Table 5-2: AAL Parameters* ..... *54*
*Table 5-3: ATM User Cell Rate/ATM Traffic Descriptor* ..... *55*
*Table 5-4: Broadband Bearer Capability* ..... *56*
*Table 5-5: Broadband Low Layer Information* ..... *56*
*Table 5-6: QoS Parameter* ..... *56*
*Table 5-7: Extended QoS Parameters* ..... *57*
*Table 5-8: Called Party Number (ATM End System Addresses)* ..... *57*
*Table 5-9: Cause I.E.* ..... *58*
*Table 5-10: Hello FSM* ..... *69*
*Table 5-11: Horizontal Link Hello FSM* ..... *82*
*Table 5-12: Neighboring Peer FSM* ..... *90*
*Table 5-13: Topology States Parameters* ..... *97*
*Table 5-14: PGL Election FSM Part 1.* ..... ***Error! Bookmark not defined.***
*Table 5-15: PGL Election FSM Part 2.* ..... *126*
*Table 5-16: PCR and SCR values for CLP=0* ..... *144*
*Table 5-17: PCR and SCR values for CLP=1* ..... *146*
*Table 5-18: Information Group Summary* ..... *152*
*Table 5-18: Information Group Summary continued* ..... *153*
*Table 5-19: Information Groups in PNNI Packets* ..... *153*
*Table 5-20: The PNNI Packet Header* ..... *154*
*Table 5-21: PNNI Packet Types* ..... *154*
*Table 5-22: The Resource Availability Information Group* ..... *155*
*Table 5-23: RAIG Flags* ..... *155*
*Table 5-24: The Uplink Information Attribute* ..... *156*
*Table 5-25: Transit Network ID* ..... *156*
*Table 5-26: Network Identification Data* ..... *156*
*Table 5-27: The PNNI Hello* ..... *157*
*Table 5-28: The Aggregation Token IG* ..... *158*
*Table 5-29: The Nodal Hierarchy List* ..... *158*
*Table 5-30: The LGN Horizontal Link Extension IG* ..... *159*
*Table 5-31: The PTSP Header* ..... *159*
*Table 5-32: The PTSE Header* ..... *160*
*Table 5-33: The Nodal State Parameters IG* ..... *161*
*Table 5-34: Flags* ..... *161*
*Table 5-35: The Nodal IG* ..... *162*
*Table 5-36: Nodal Flags* ..... *162*
*Table 5-37: The Internal Reachable ATM Address IG* ..... *163*
*Table 5-38: The Exterior Reachable ATM Address IG* ..... *164*
*Table 5-39: The Horizontal Links IG* ..... *165*
*Table 5-40: The Uplinks IG* ..... *165*
*Table 5-41: The PTSE Acknowledgement Packet* ..... *169*
*Table 5-42: The Database Summary Packet* ..... *170*
*Table 5-43: Database Summary Packet Flags* ..... *170*
*Table 5-44: The PTSE Request Packet* ..... *171*
*Table 5-45: The Systems Capabilities IG* ..... *172*
*Table 6-1 Messages for ATM Call and Connection Control* ..... *177*

*Table 6-2 Messages for ATM call/ connection control....183*
*Table 6-3 Messages Used with the Global Call Reference....186*
*Table 6-4 Messages used with ATM Point-to-multipoint call/connection control....187*
*Table 6-5 Information elements used in PNNI....193*
*Table 13-1: Mandatory and Optional Capabilities....257*
*Table 20-1: Contents of PNNI Hello Packet....351*
*Table 20-2: Contents of PNNI Topology State Packet....352*
*Table 20-3: Contents of PTSE Acknowledgement Packet....353*
*Table 20-4: Contents of Database Summary Packet....353*
*Table 20-5: Contents of PTSE Request Packet....354*

# 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:

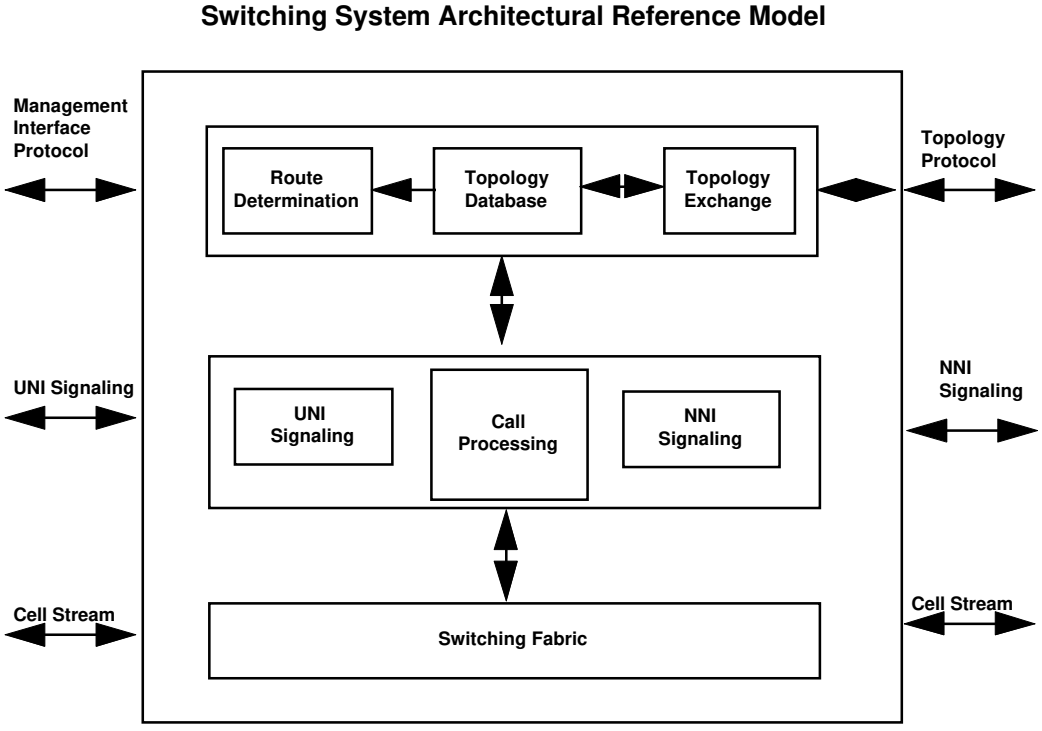



**Figure 1-1: Switching System Architectural Reference Model**

### 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.<br><br>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.<br><br>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.

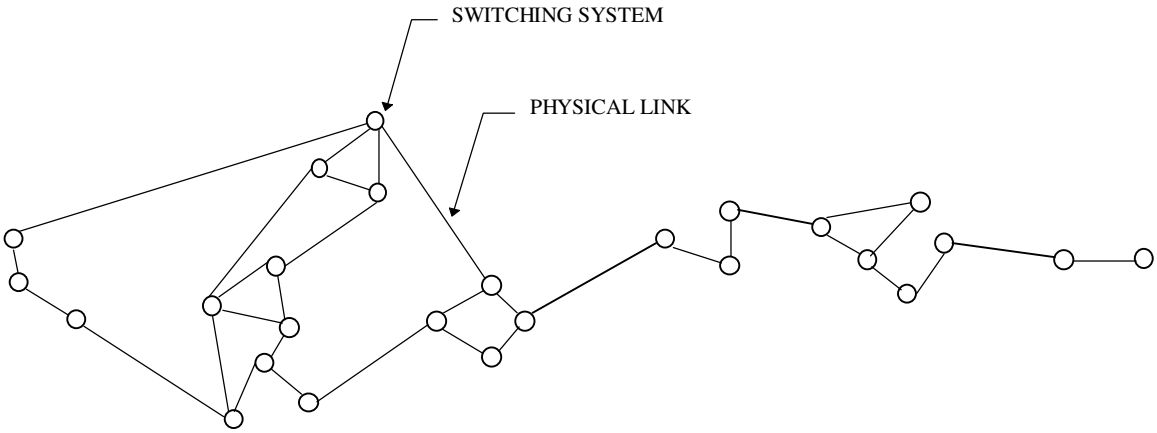



**Figure 3-1: Private ATM network with 26 switching systems and 33 bi-directional links.**

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.

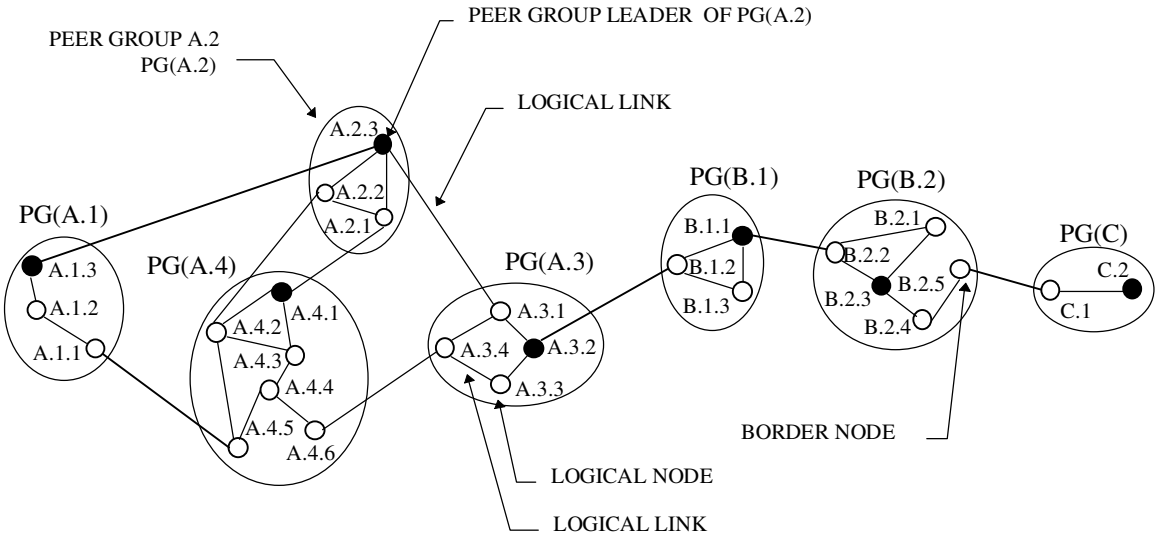



**Figure 3-2: Partially configured PNNI hierarchy showing lowest level nodes.**

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.

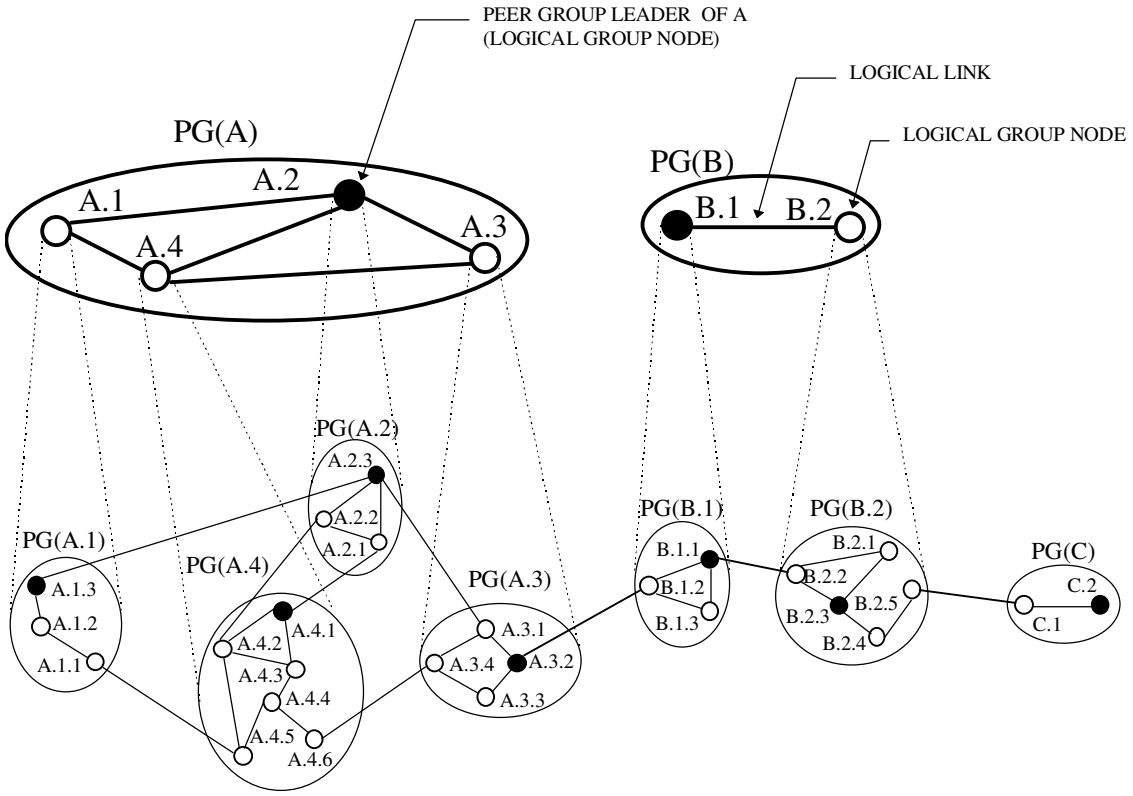



**Figure 3-3: Partially configured PNNI hierarchy showing lowest hierarchical levels.**

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.

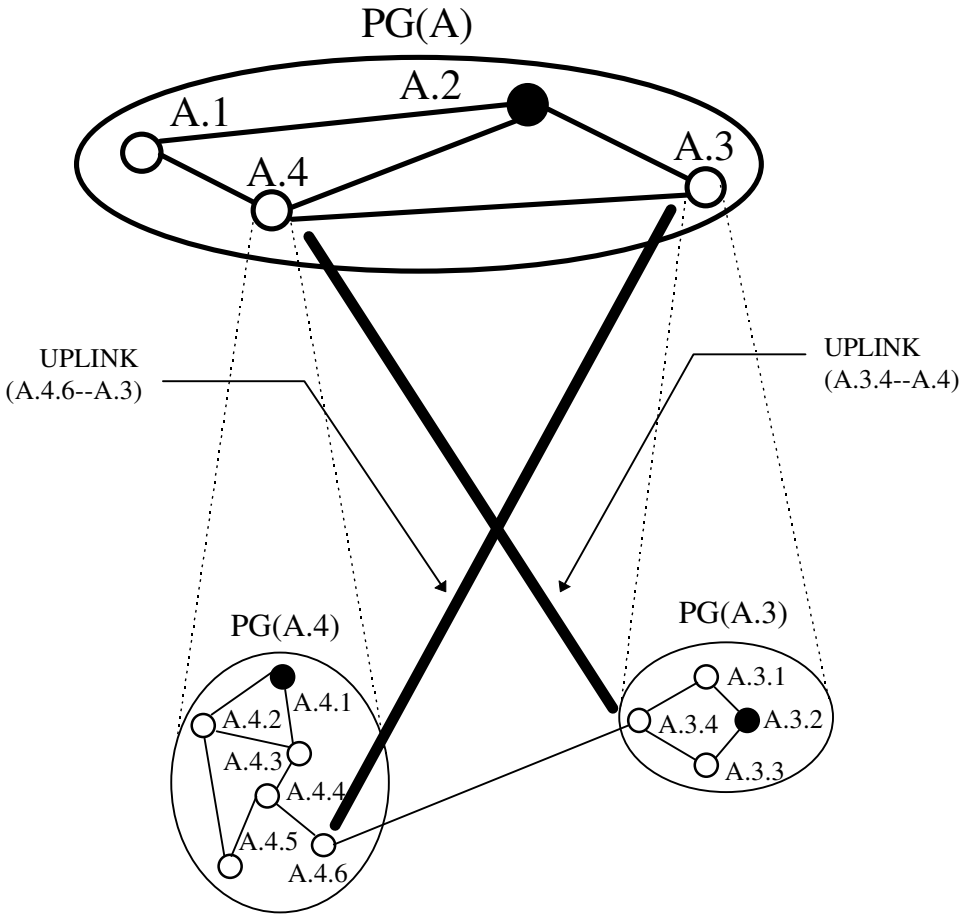



**Figure 3-4: Uplinks.**

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.

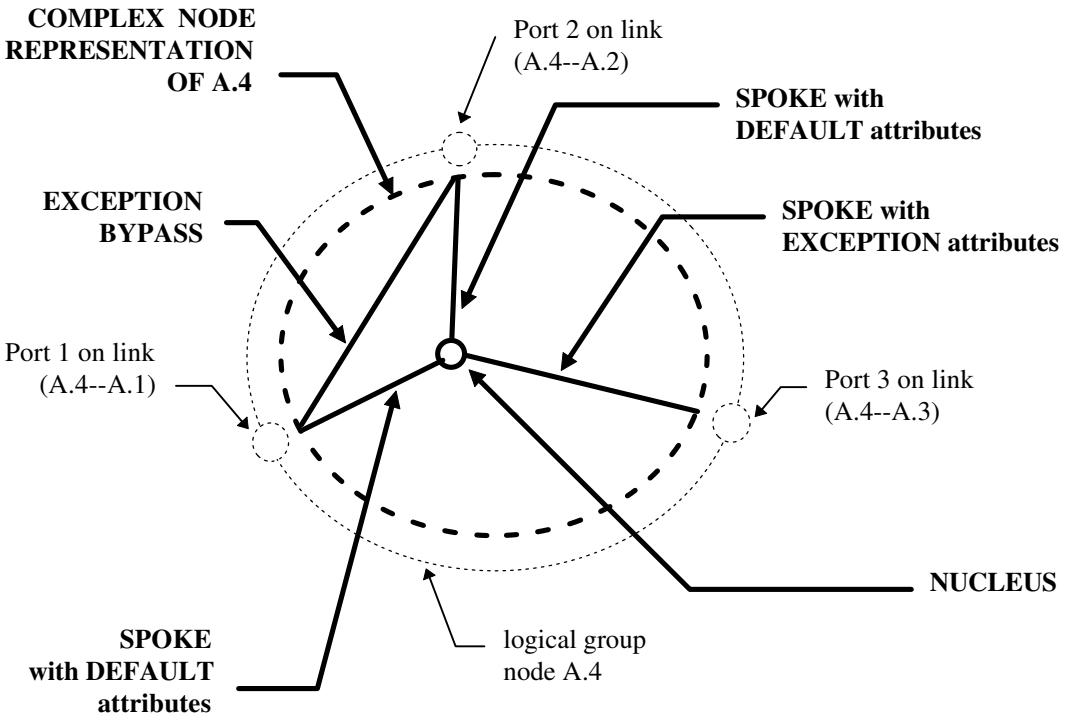



**Figure 3-5: Complex node representation of LGN A.4 from Figure 3-3.**

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.

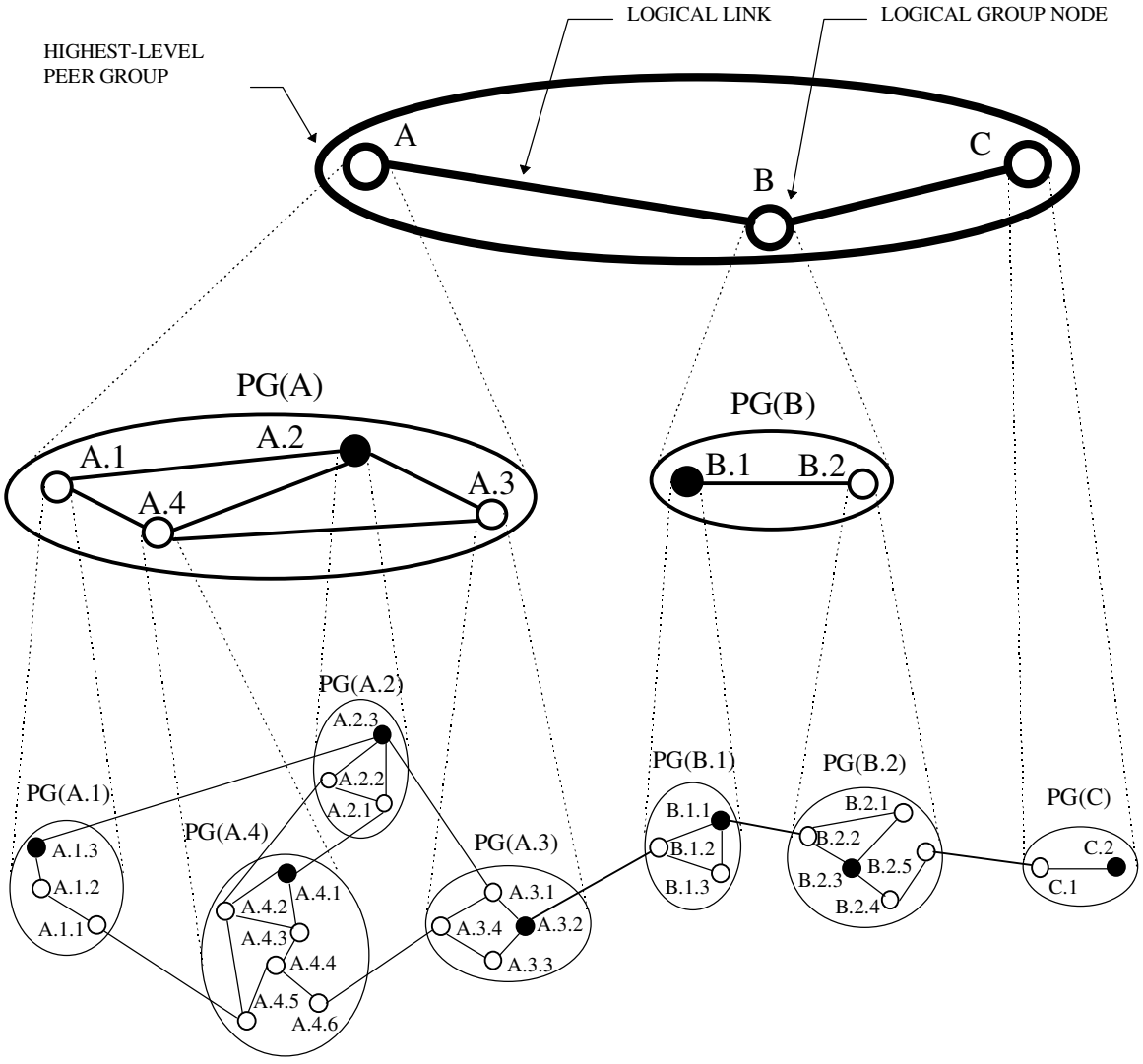



**Figure 3-6: Example of a complete PNNI hierarchically configured network.**

### 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.

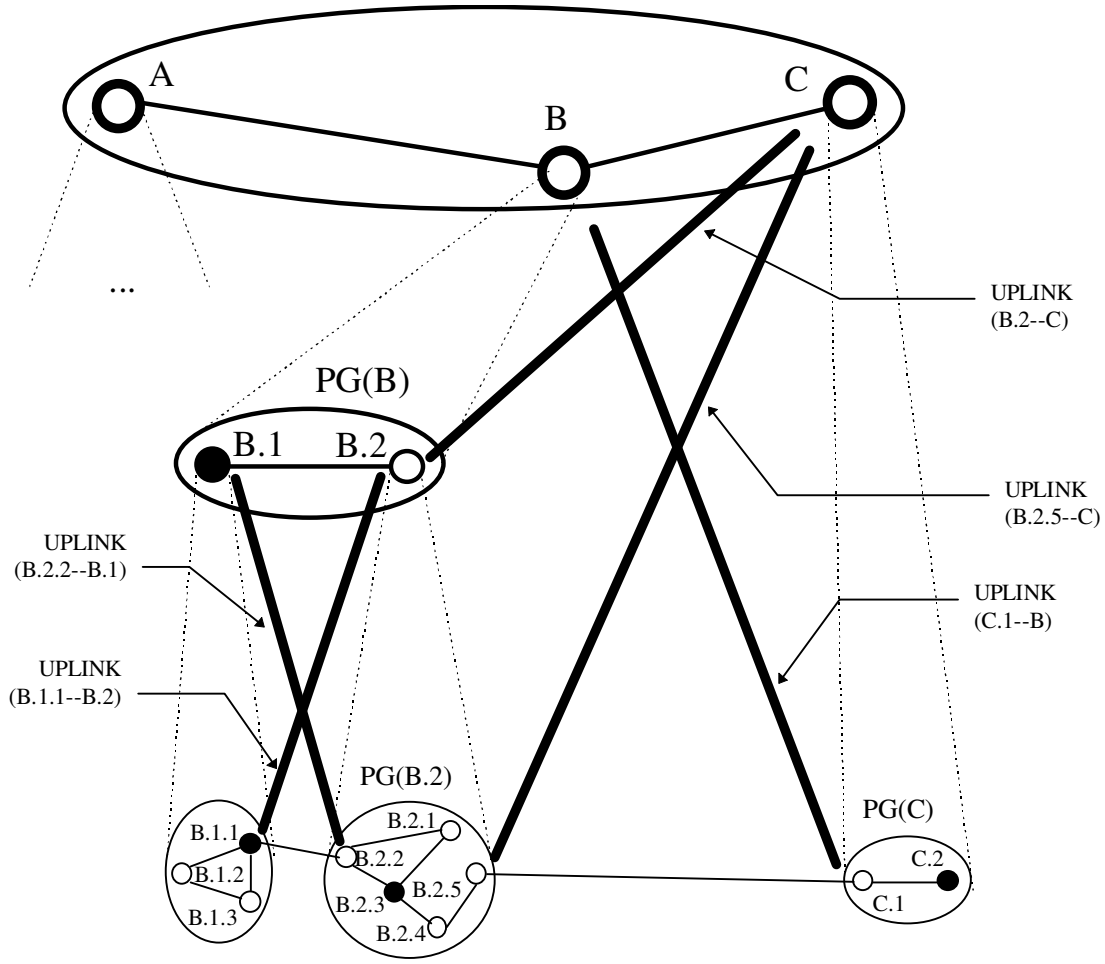



**Figure 3-7: Uplinks revisited.**

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)<br>P<Y.2> (configured) | P<Y.1> (configured)<br>P<Z.2> (configured) | P<A.2.3> (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.

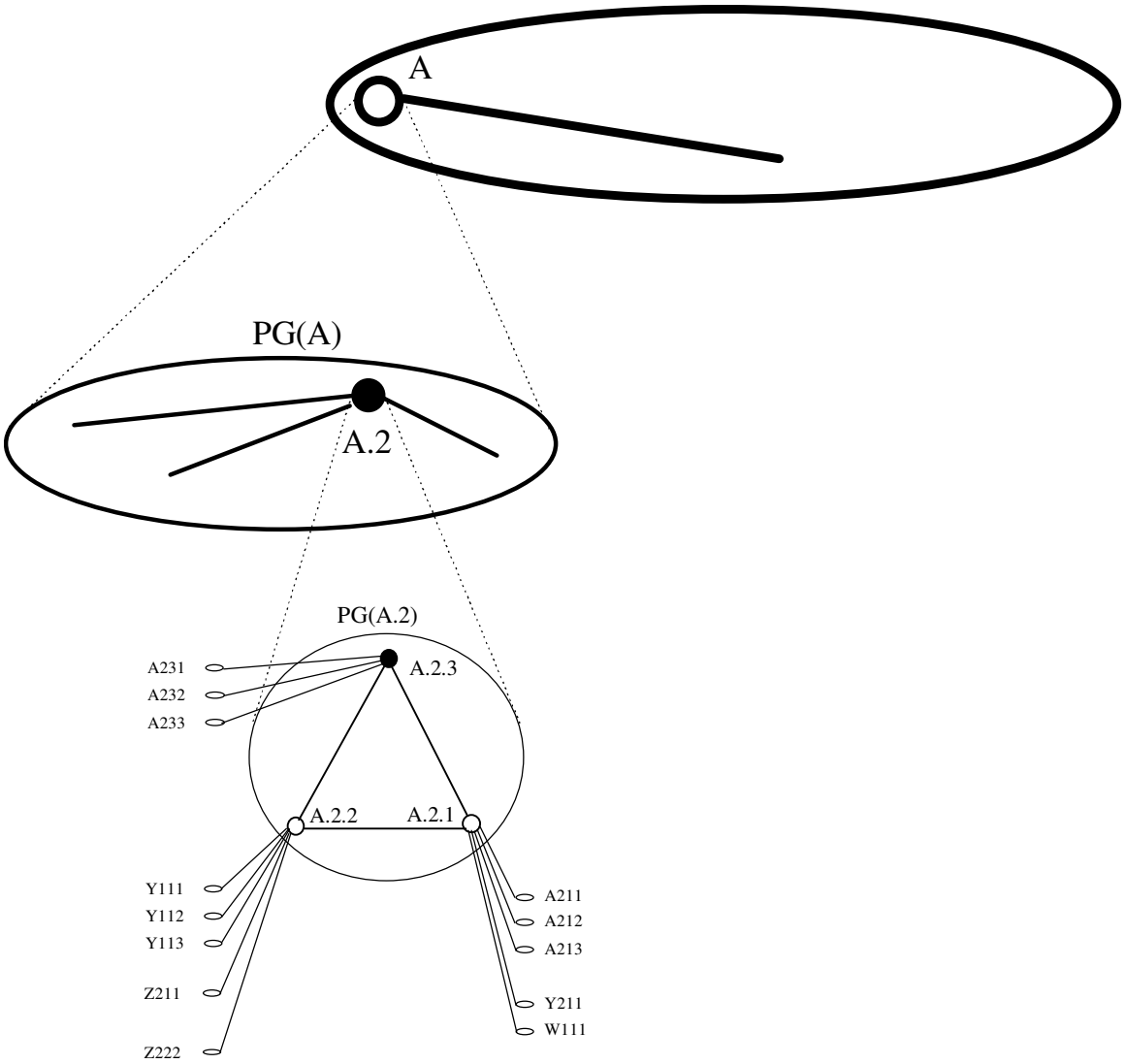



**Figure 3-8: Address summarization in a PNNI hierarchy.**

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)<br>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.

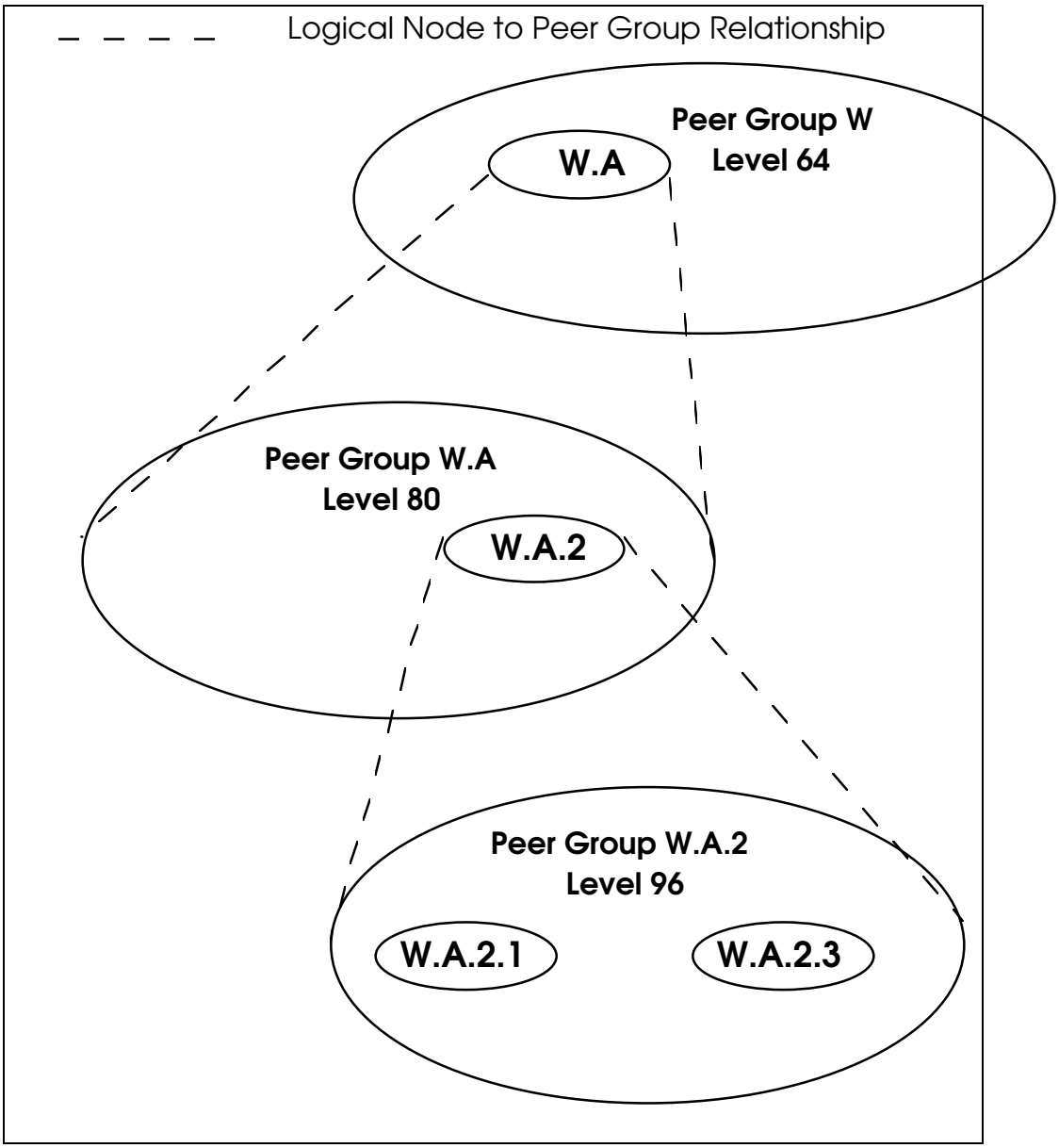



**Figure 3-9: Sample Network**

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.

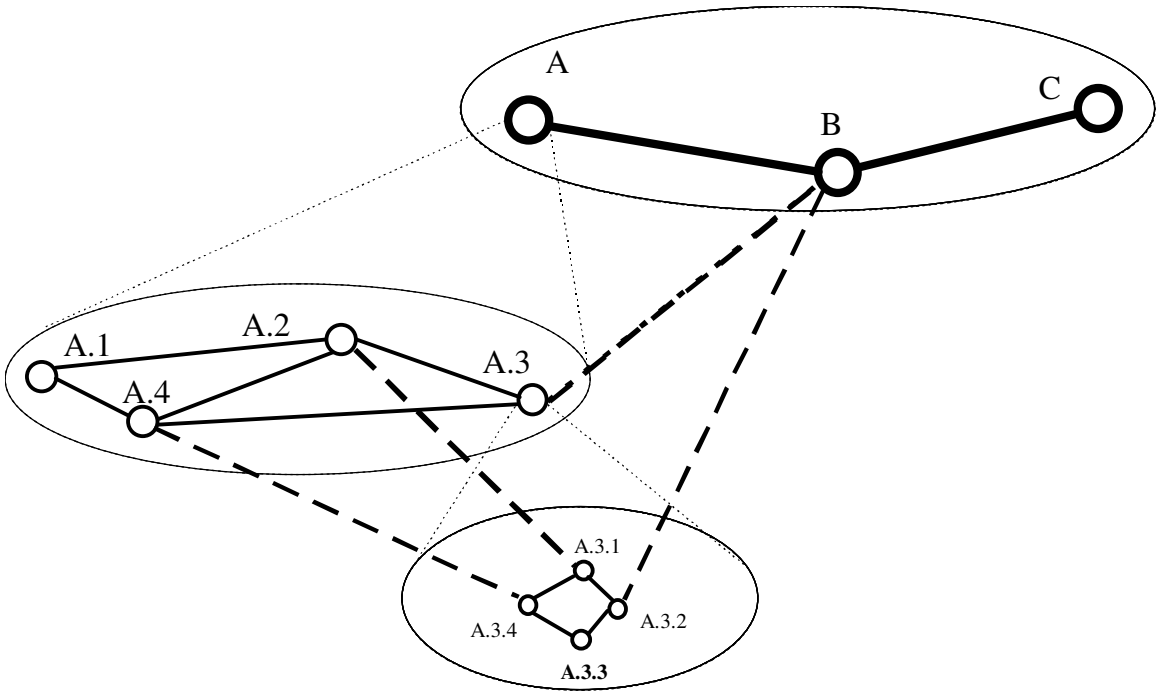



**Figure 3-10: View of the network from nodes A.3.3, A.3.2, A.3.1, and A.3.4.**

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.

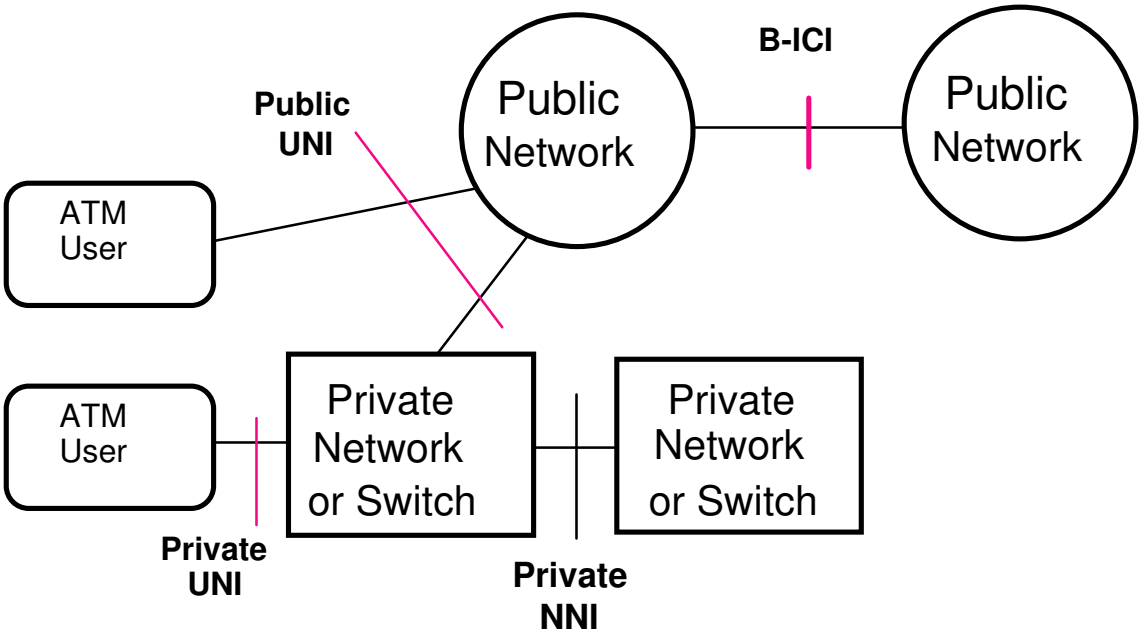
A generic CAC is needed in the path selection process to determine if a link or node is likely to have enough resources available to support the proposed connection. A generic CAC should include a link or node if the switching system is likely to accept the proposed connection. Conversely, the generic 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.

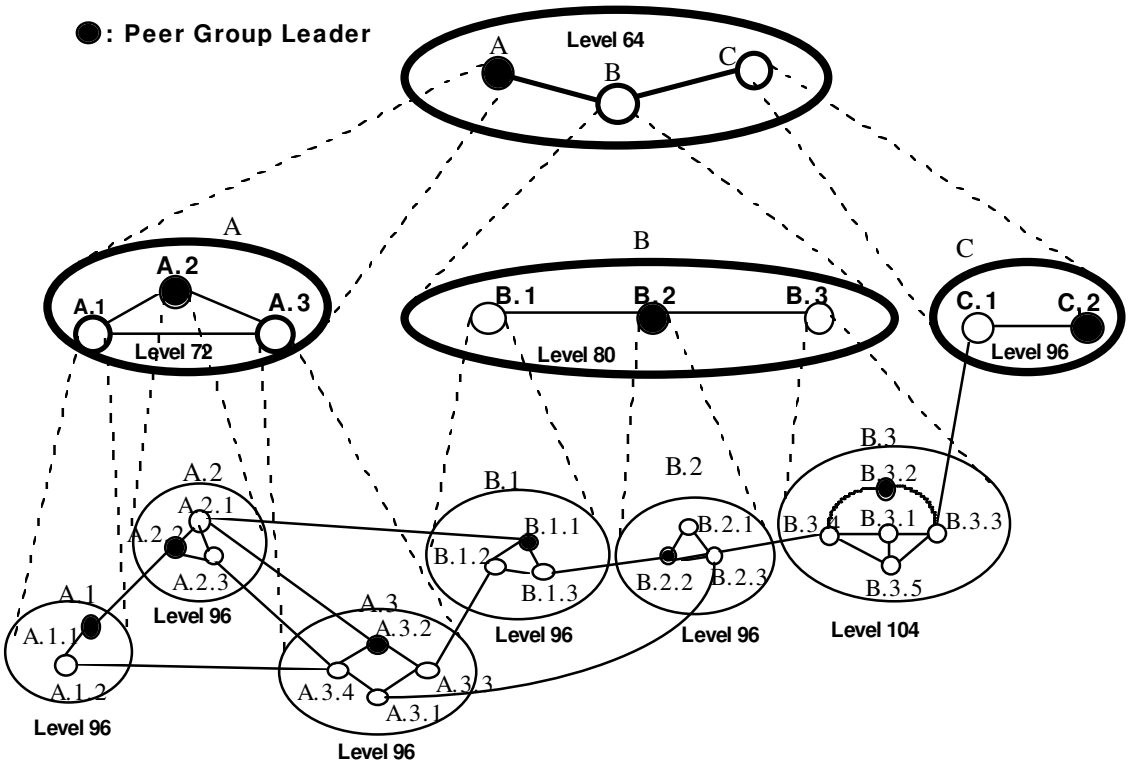



**Figure 4-2: A Sample Hierarchical Network**

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.

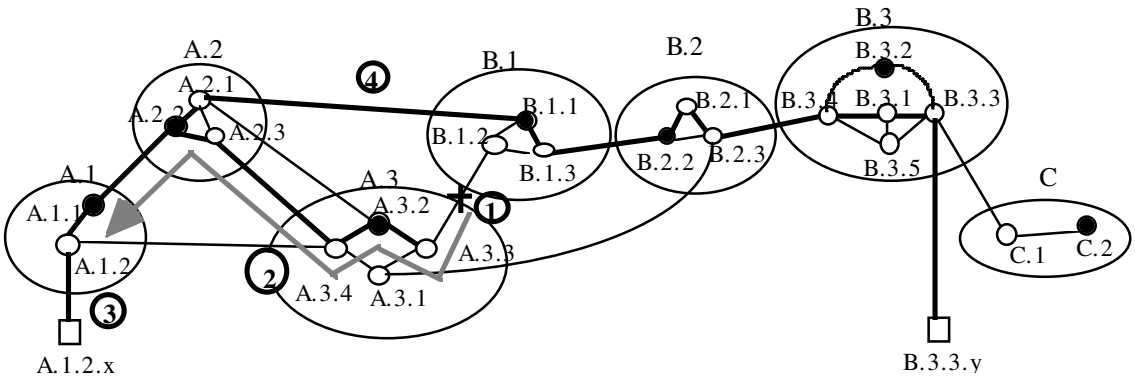



**Figure 4-3: An Example of Crankback and Alternate Routing**

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

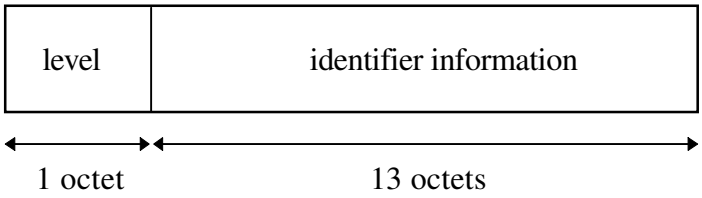



**Figure 5-2: Peer Group Identifier 14 octet encoding**

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.

**T**he 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<br>Real time VBR<br>Constant Bit Rate<br>Available Bit Rate<br>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)<br>(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<br>3 | (ITU-TS), used for cause 16 and 31<br>ATM Forum, used for cause 53 |
| I.E. Instruction Field | 0 | (not significant) |
| Location | 0 | (User) |
| Cause Value | 16<br>31<br>53 | Normal call clearing<br>Normal, Unspecified<br>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.

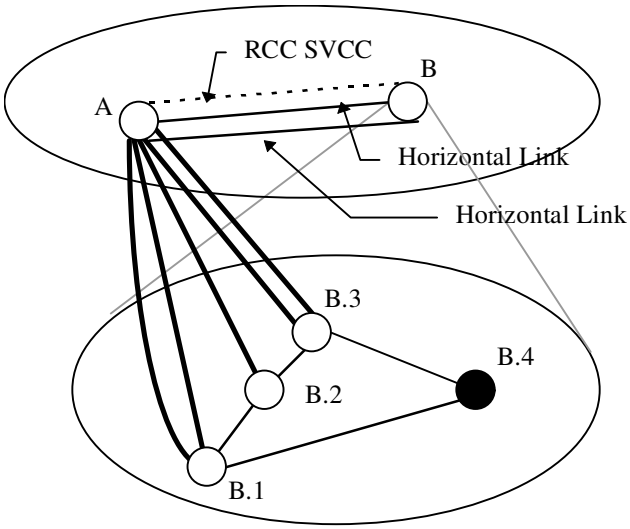



**Figure 5-3: An SVCC-Based RCC and horizontal links between a Lowest-Level Node and an LGN.**

### 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.

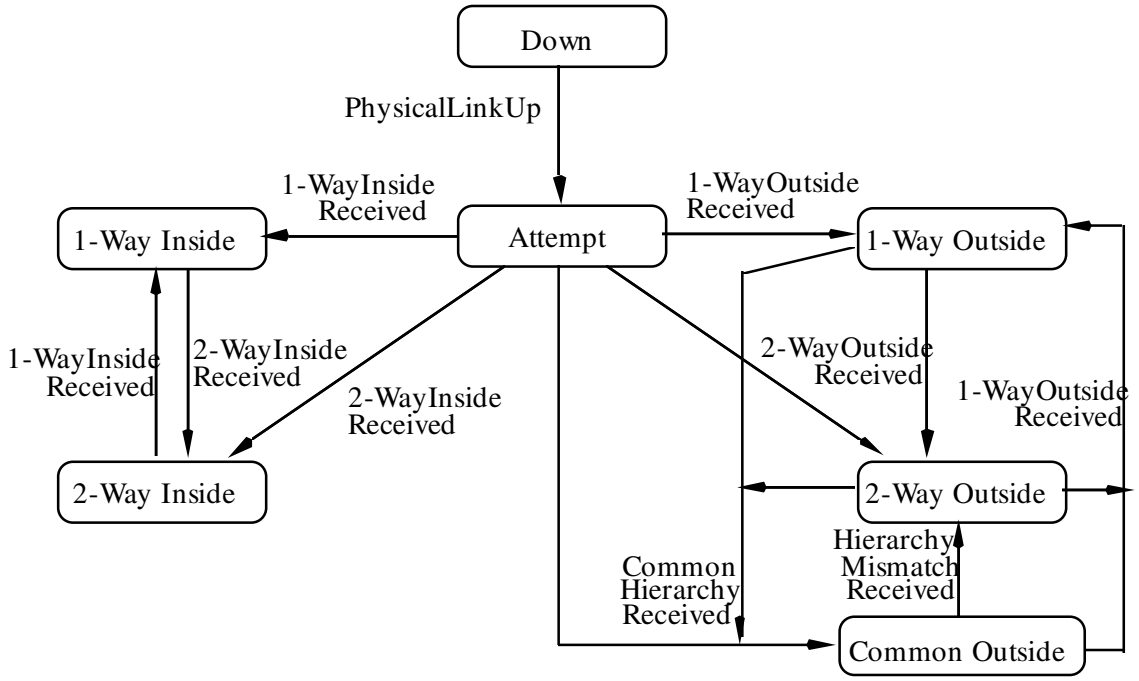
Derived Link Aggregation Token
: The algorithmically derived value of the link aggregation token for this link.

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.

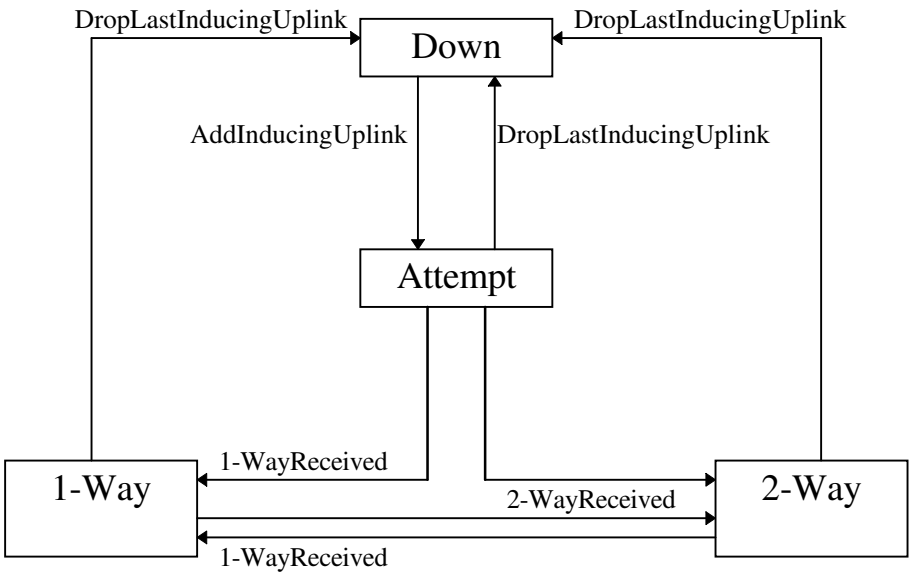



**Figure 5-5: LGN Horizontal Link Hello FSM.**

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

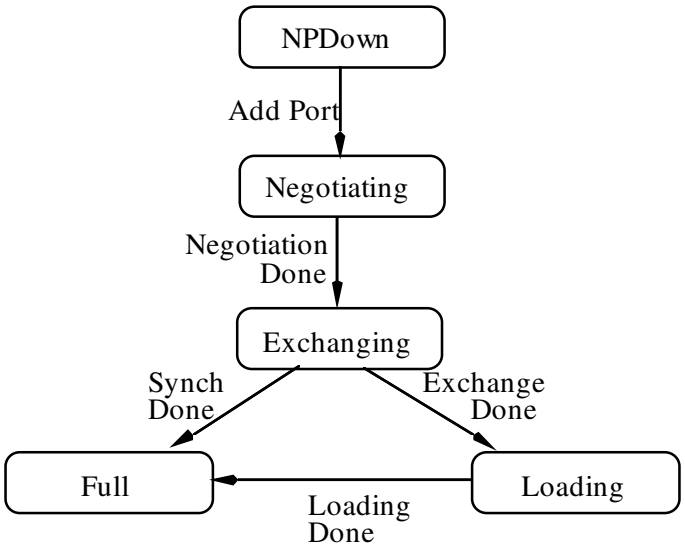
The amount of time, in seconds, before a node sends a new PTSE Request Packet requesting PTSEs of the last PTSE Request Packet that have not been received yet.

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.

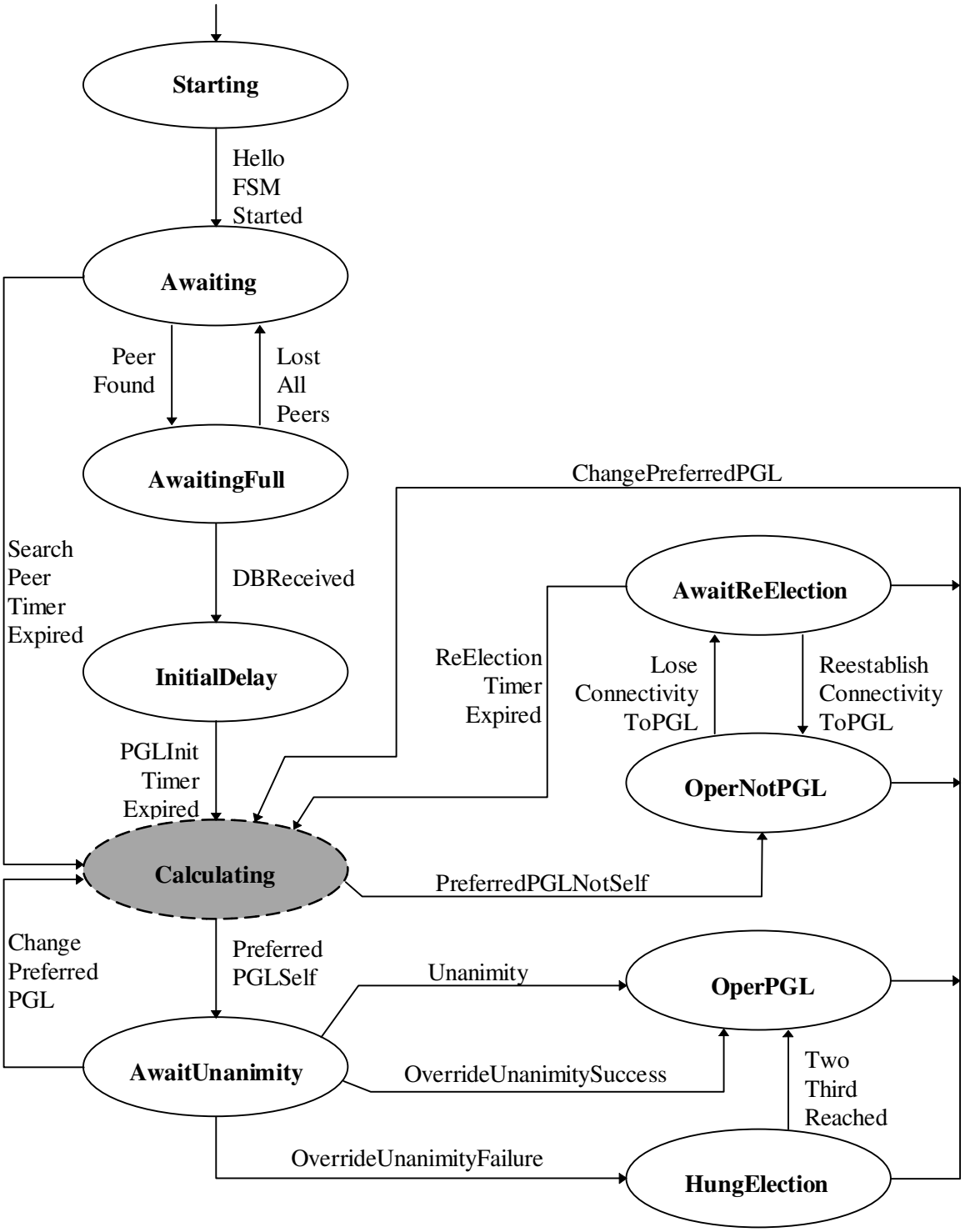



**Figure 5-7: PGL Election FSM.**

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.

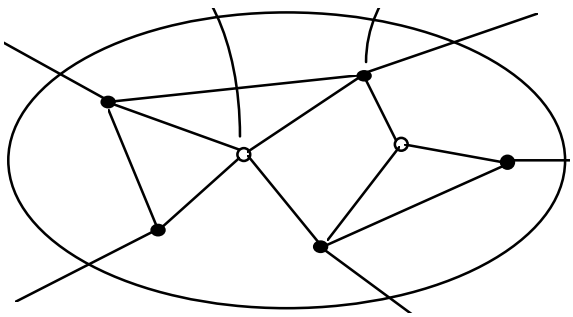


**Figure 5-8: Peer Group with Five Border Nodes**

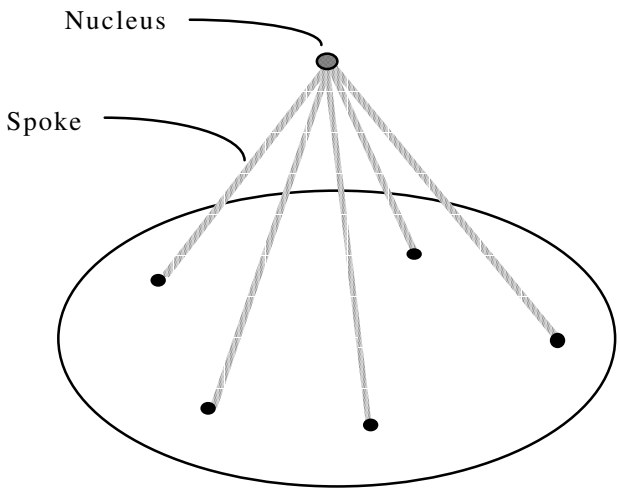



**Figure 5-9: Default Node Representation**

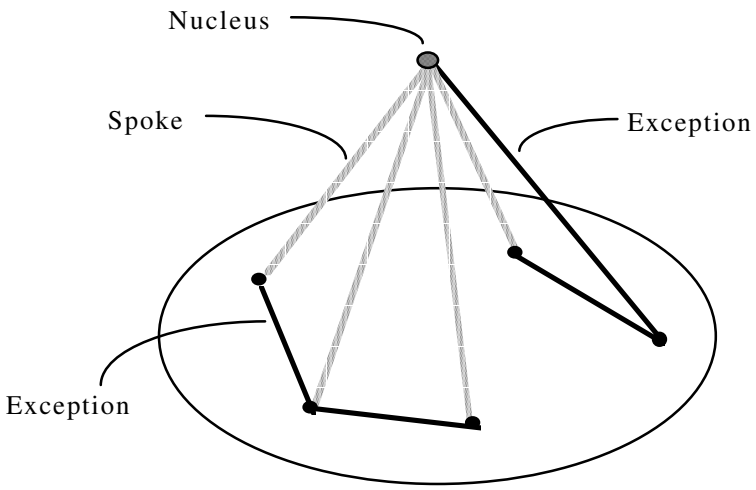



**Figure 5-10: Complex Node Representation with Exceptions**

### 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