INTERNET-DRAFT
<draft-schulter-ipv6atm-framework-00.txt>

P. Schulter
Digital Equipment Corp.
November 21, 1995

A Framework for IPv6 Over ATM

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its ˝areas and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use ˝Internet-Drafts as reference material or to site them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "lid-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.os.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

This Internet Draft Expires May 25, 1996.

Abstract

Work in specifying the IPv6 [IPv6-SPEC] has now progressed to a point that work on IPv6 implementations is beginning at many organizations. So far, the IPv6 specifications have dealt with broadcast media LANs [IPv6-ETHER]with very little attention to ATM. The IP over ATM WG is now starting to look at IPv6 over ATM [IP-IPV6ND]. Many of the problems encountered in running IPv4 over ATM [IP-FRAME, IP-ATM, IP-ATMU, IP-ATMMC] must also be dealt with when running IPv6 over ATM. Since ATM networks are ˝point-to-point networks that offer no connectionless broadcast or ˝multicast capabilities, many of the IPv6 protocols can't be directly ˝applied to the ATM network. Instead, some software framework built on ˝top of the ATM network must be built to handle those protocols or functions which rely on connectionless multicast capabilities. Among these functions are neighbor discovery and address configuration. Another difficulty with ATM is that ATM networks have a flat address space, and are expected to become very large (even global). It is desirable to logically partition a ˝large ATM network into IPv6 subnets and to be able to maintain IPv6 subnet semantics while still maintaining the ability to interconnect any two nodes on the ATM network so that ATM QoS services can be provided within the framework of IPv6 networking.

This document proposes a framework for performing IPv6 Neighbor Discovery and address configuration on ATM networks. This framework also permits the logical partitioning of ATM networks into logical groups of nodes so that the semantics of IPv6 link-local and site-local addresses are maintained. Finally, this framework permits any two systems on separate subnets to directly

connect so that virtual circuits with specific QoS requirements can be established.

Status of this Memo....1
Abstract....1
1. Introduction and Background....4
1.1. Requirements....5
2. Definitions of Terms....5
2.1. IPv6 Terminology....5
2.2. ATM Terminology....7
2.3. New Terminology....7
2.4. Addresses....8
3. IPv6 Over ATM Framework Description....8
3.1. NDS Trees....14
3.1.1. Manual NDS Tree Configuration....15
3.1.2. Autoconfiguration of Trees....17
3.2. Forming Logical Links....18
3.2.1. Neighborhoods....21
3.3. Forming Sites....21
3.4. Beyond Sites....22
4. NDS Operations....23
4.1. Router Advertisements....25
4.2. Router Solicitations....31
4.3. Neighbor Solicitation....31
4.4. Neighbor Advertisements....33
4.5. Anycast Addresses....33
4.6. Neighbor Unreachability Detection....35
5. Address Configuration....35
5.1. Link-Local Addresses....36
5.2. Stateless Address Configuration....36
5.3. Stateful Address Configuration....37
5.4. Manual Address Configuration....37
5.5. Node Relocation....37
5.6. Duplicate Address Detection....39
6. Multicasting....40
7. VC Characteristics....40
7.1. NDS VCs....40
7.2. Data VCs....41
8. Security....41
Acknowledgments....41
Authors' Address....41
References....41

1. Introduction and Background

The basic problem in adapting IPv6 to ATM networks is that ATM networks do not provide connectionless broadcast or multicast capabilities, and therefore, do not inherently provide IPv6 to ˝ATM address resolution capabilities. Further, ATM networks can be very large (even global in scope) which can result in networks that are composed of thousands or even hundreds of thousands of nodes. Unlike broadcast networks such as Ethernet or FDDI, connectivity on ATM networks is not restricted by network segmentation. Generally, all nodes on an ATM network can establish connectivity to any other node on the same network. Additionally, ATM even provides subaddressing so nodes on different ATM networks which are interconnect (generally two private networks connected through a public network) can directly connect. ATM networks provides no physical or logical partitioning of reachability. While this reachability scheme can result in much greater flexibility in creating logical networks within a larger ATM network (for example, nodes in geographically diverse locations can be logically grouped into logical ˝networks), it also makes the partitioning of systems on the larger ATM network into a set of smaller logical networks more difficult.

IP protocols currently rely on connectionless broadcast and multicast capabilities of legacy LAN technologies to perform functions such as IP to physical address resolution. Further, IPv6 protocols generally rely on the physical partitioning of networks to establish the boundaries of subnetworks. To adapt IP and its protocols to ATM the following mechanisms need to be defined:

- IPv6 to ATM address resolution
- how the Neighbor Discovery protocols are to be used in an ATM network.

  the logical partitioning of nodes into small, autonomous groups with limited IPv6 addressing scopes (subnetworks)

While providing these mechanisms for IPv6, the framework should meet the following design goals:

- the concept of the subnetwork must be maintained
- the use of link-local and site-local addressing must be maintained
- the resulting protocols and architecture must scale to arbitrarily large networks
- provide a way for nodes to join multiple IPv6 subnets and multiple ATM node groupings
- allow nodes belonging to a particular subnet to join the subnet regardless of their physical location
- permit (but not require) that nodes in different logical grouping or subnets can establish connectivity directly through the ATM network without going through a router. This is needed so that connections with specific QoS requirements can be established so that IPv6 can make full use of ATM QoS capabilities
- built-in fault tolerance and redundancy

Optionally, any IPv6 over ATM architecture should provide for the highest degree of autoconfiguration as the ATM and IPv6 protocols will allow. Ideally, all elements of an IPv6 over ATM network should be self configuring with as little human intervention and management as possible.

Finally, any protocols developed for IPv6 over ATM should be defined for both current UNI 3.1 and future UNI 4.0 networks. Since it is expected that UNI 4.0 will be widely deployed by the time IPv6 goes into wide use, the new capabilities and features ˝of UNI 4.0 should be taken into account in places where they can improve the functioning of the protocols. However, since initial implementations will be written to existing UNI 3.0/3.1 networks, the protocols must also work under UNI 3.0/3.1.

1.1. Requirements

Throughout this document, the words that are used to define the significance of the particular requirements are capitalized. These words are:

- **MUST** — This word or the adjective **"REQUIRED"** means that the item is an absolute requirement of this specification.
- **MUST NOT** — This phrase means the item is an absolute prohibition of this specification.
- **SHOULD** — This word or the adjective **"RECOMMENDED"** means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
- **SHOULD NOT** — This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- **MAY** — This word or the adjective **"OPTIONAL"** means that the item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example, an other vendor may omit the same item.

2. Definitions of Terms

Relevant terminology from the IPv6 Protocol [IPV6-SPEC], IPv6 Addressing Architecture [IPV6-ADDR], IPv6 Stateless Address Autoconfiguration [IPV6-ADDCONF], IPv6 Neighbor Discovery [IPV6-ND], then Classical IP and ARP over ATM [IP-ATM], UNI 3.0 [ATM-UNI30], UNI 3.1 [ATM-UNI31], UNI 4.0 [ATM-UNI40], and PNNI [ATM-PNNI] will be provided, and finally new terminology introduced by this document will be given.

2.1. IPv6 Terminology

- **IPv6** — Internet Protocol Version 6.
- **IPv4** — Internet Protocol Version 4.

- **Node** — a device that implements IPv6. In this document all nodes implement IPv6 over ATM. Node are always connected to an ATM network and may be connected to any number of other networks.
- **Router** — A node that forwards IP packets not explicitly addressed to itself.
- **Host** — Any node that is not a router
- **link** — A communications facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6. Examples are Ethernet (simple or bridged), ATM networks, PPP links, X.25, Frame Relay, and FDDI; and internet (or higher) layer "tunnels", such as tunnels over IPv6 or IPv6 itself.
- **Neighbors** — Nodes attached to the same link.
- **Interface** — A node's attachment to a link
- **address** —An IP layer identifier for an interface or a set of interfaces
- **packet** — An IP header plus a payload.
- **Communication** — Any packet exchange between nodes that requires that the address of each node used in the exchange remain the same for the duration of the exchange. Examples are a TCP connection or UDP request/response.
- **Unicast-address** — An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.
- **Multicast-address** — an identifier for a set of interfaces (typically belonging to different nodes). A packet sent to ˝a multicast address is delivered to all interfaces identified by that address.
- **Link-layer address** — a link-layer identifier for an interface. Examples are an IEEE 802 address for Ethernet links, and ATM NSAP or E.164 addresses (with optional subaddresses) for ATM networks.
- **link-local address** — An address having link-only scope that can be used to reach neighboring nodes attached to the same link. All interfaces have a link-local address.
- **Site-local address** — An address having scope that is limited to the local site
- **global address** — an address with unlimited scope
- **interface token** — A link dependent identifier for an interface this is (at least) unique per link.
- **Anycast address** — an identifier for a set of interfaces (typically belonging to different nodes). A packet sent to ˝an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols measurement of distance).
- **On-link** — An address that is assigned to an interface on a specified link. A node considers an address to be on-link if:
  - it is covered by one of the link's prefixes
  - a neighboring router specifies the address as the target of a Redirect message
  - a Neighbor Advertisement message is received for the (target) address
  - a Neighbor Discovery message is received from the address
- **off-link** — the opposite of "on-link"; and address that is not assigned to any interfaces on the specified link.

2.2. ATM Terminology

- **Point-to-Point (PtP)** — An ATM connection (virtual circuit) connecting exactly two ATM nodes. The same set of nodes can have multiple PtP connections to each other.
- **Point-to-Multipoint (PtM)** — An ATM connection that connects one node to many nodes. PtM connections are unidirectional with data flowing from the "root" node to many "leaf" nodes. PtM connections are created by placing a special call to each node to be included in the PtM connection. The same set of nodes can have multiple PtM connections to each other.
- **Call** — The process of establishing a connection (virtual circuit) between two nodes (parties). One node actively initiates connection establishment (calling party) and the other node passively receives the call (called party). ˝Circuit traffic parameters are established and negotiated during the call.
- **Release** — The process of terminating a single connection between two more nodes. A node is always notified if a connection is released for any reason.
- **Group Address** — This type of address is specific to UNI 4.0 (UNI 3.0/3.1 does not support group addressing). An address which identifies a set of ATM nodes. Calls to a group address are routed by the ATM network to exactly one of the nodes identified by the address. Group addresses are assigned a specific scope when they are registered with the network. ˝This scope defines which nodes within the ATM network routing hierarchy are able to reach the specific nodes identified by the group address. Group addresses are also referred to as anycast addresses in the UNI 4.0 specification, but the term group address is used in this document to avoid confusion with IPv6 anycast addresses.
- **Address registration** — The process by which an ATM node informs the ATM network of its link-layer addresses. ATM nodes can register multiple addresses and the addresses can take several forms. A node must register an address to be able to ˝establish connections to other nodes.
- **UNI** — User-Network Interface. The ATM interface and protocols between the nodes and the network.
- **PNNI** — Private Network-Network Interface. The ATM interface and protocols between switching elements of an ATM network. The also defines how an ATM network topology is build and how calls are routed within that topology.

2.3. New Terminology

- **Logical Link (LL)** — a set of ATM nodes which can reach each other using only link-local addresses and which can broadcast to all other nodes in the LL. This is analogous to a LAN segment.
- **Neighbor Discovery Server (NDS)** — An entity which provides a service for performing IPv6 Neighbor Discovery in an ATM network.
- **NDS(n)** — This notation denotes a Neighbor Discovery server at a specific level in the NDS hierarchy.
- **GA(n)** - This notation denotes a specific ATM group address which is used to place calls to an NDS at level n.

- **"Broadcast"** – In this document this term refers to physically transmitting an IPv6 packet to all nodes or other entities within a specific scope. This term does not refer to any function of the ATM network, and is used in quotes so as not to be confused with a hardware broadcast function.
- **Site** – A collection of Logical Links in which all nodes share a common IPv6 site address prefix. All nodes in a Site can reach any other node in the Site using IPv6 site addresses. Since this term is used to specify a specific logical grouping of LLs and nodes, and does not directly relate to the ATM UNI 4.0 site group addressing scope, it is capitalized to distinguish it from the ATM term "site".

2.4. Addresses

- **all-nodes multicast address** – The link-local scope address to reach all nodes. FF02::1
- **All-routers multicast address** – the link-local scope address to reach all routers. FF02::2
- **Solicited-node multicast address** – a link-local scope address that is computed as a function of the solicited target's address. See [IPV6-ND]
- **link-local address** – a unicast address having link-only scope that can be used to reach neighbors. All interfaces **MUST** have a link-local address. Routers **MUST NOT** forward packets with link-local source addresses. See [IPV6-ADDR]
- **unspecified address** – A reserved address value that indicates the lack of an address (e.g., the address is unknown). It is never used as a destination address. See [IPV6-ND] and [IPV6-ADDR].
- **DHCP Server/Relay Agent address** – the link local scope address used to reach all IPv6 DHCP Servers and Relay Agents. ˝FF02::1:0 [IPV6-DHCP].
- **tentative address** – an address that has been assigned to an interface, but which has not yet been verified using duplicate address discovery. A node can not use a tentative address as the source address in a packet, or receive packets which contain the tentative address as the destination, until duplicate address detection has been performed [see IPV6-ADDCONF].

3. IPv6 Over ATM Framework Description

The IPv6 over ATM framework proposed here manages both logical network partitioning and node reachability via a collection of servers which are used for neighbor discovery and address configuration. These servers, called Neighbor Discovery Servers (NDS) are organized in a tree structure rather than a flat structure. Organizing the framework into a tree structure ˝follows from the IPv6 address architecture and scoping, facilitates logical partitioning of the network and hierarchical management of neighbor discovery operations, as well as providing scalability, redundancy, and failover capabilities. A tree structure also makes network autoconfiguration in a UNI 4.0 environment, where group addresses and reachability scopes can be used, possible.

All ATM nodes on an ATM network are all technically "on-link" since they can all connect to any other node and exchange data. An ATM network can be logically partitioned (potentially across arbitrary geographic boundaries if needed) into many IPv6 subnetworks. This is necessary to logically group nodes sharing some organizational relationship together into different groups which have some degree of isolation from other groups. However, even after logically partitioning ATM networks into logical IPv6 subnetworks, all ATM nodes are still physically "on-link" to all other ATM nodes. The ability for any two nodes to connect has ˝not been affected at the link-layer, only network layer reachability has been affected. Note that this case is different from the normal case with legacy LANs. Using legacy LANs, IPv6 ˝subnetworks are generally physically as well as logically disjoint. With ˝ATM, subnetworks are logically disjoint, but not physically ˝disjoint.

Given this type of partitioning of an ATM network, it is possible to define two levels of "on-link". The first is the physical level where a node is considered "on-link" if it is capable of receiving a call from another node. This level of "on-link" is determined by the physical connections of the ATM network(s). ˝ The second level is the logical level where nodes within the same logical grouping are considered "on-link" at the network layer. Nodes which are not logically "on-link" are still physically "on-link". Hosts which are physically "off-link" must be logically "off-link" also.

The framework proposed here manages these two levels of "on-link" in such a way as to maintain the subnetwork model (the logical "on-link" level) while also maintaining the ability for any two nodes to directly connect (the physical "on-link" level). This allows existing IPv6 semantics to be maintained while still allowing "VC cut through" between nodes on different subnetworks.

To do this, the concept of a Logical Link (LL) is used. A ˝logical link has semantics similar to those of a legacy LAN physical ˝link. That is, a logical link is a set of nodes connected to some media which can communicate to each other through the media, and a LL provides a broadcast domain for all nodes attached to it. Most importantly, a Logical Link provides the isolation and logical grouping of a set of nodes into a "community" which can be managed like a LAN. Thus, Logical Links control the logical ˝determination of "on-link" at the network level, not at the physical level.

The Logical Link concept differs somewhat from the IP subnetwork concept. A Logical Link defines node groupings, but does not necessarily need to equate with IP subnetting. Since the operation of a Logical Link mainly provides the semantics of a physical LAN, it is possible that many IP subnets could be configured on the same Logical Link. This, a LL could support multiple Logical IP Subnets (LISs).

The Logical Link is the basic unit of this framework. At a minimum, some set of nodes must be grouped together into a ˝Logical Link. Multiple LLs can exist on the same ATM network, and will function as disjoint logical networks, requiring routers for ˝nodes on different LLs to communicate. However, Logical Links may be grouped together to form larger node groupings, call Sites. A Site can consist of any number of LLs. Each LL in a Site functions just like an isolated LL, with the exception that the Site grouping can manage logical "on-link" determination between LLs. When LLs are grouped into Sites, nodes on each LL are allowed to directly connect to nodes on any other LL via the ATM network. However, logical reachability is managed by IPv6 ˝address

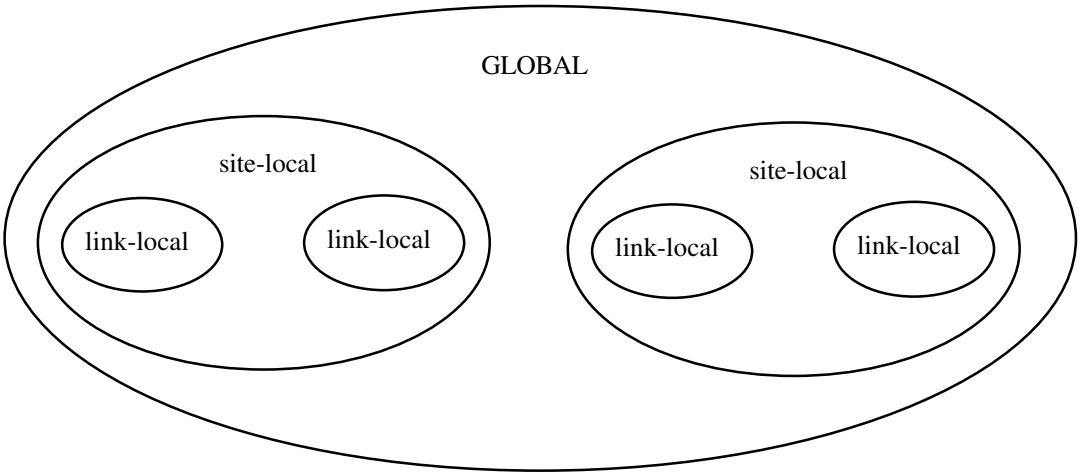
scoping. That is, nodes on one LL can only reach nodes on ˝another LL by using an IPv6 address of appropriate scope.

Sites can be further grouped into larger units as needed to provide managed connectivity between any set of nodes. As with Sites, logical reachability between nodes in different Sites is managed by IPv6 address scoping.

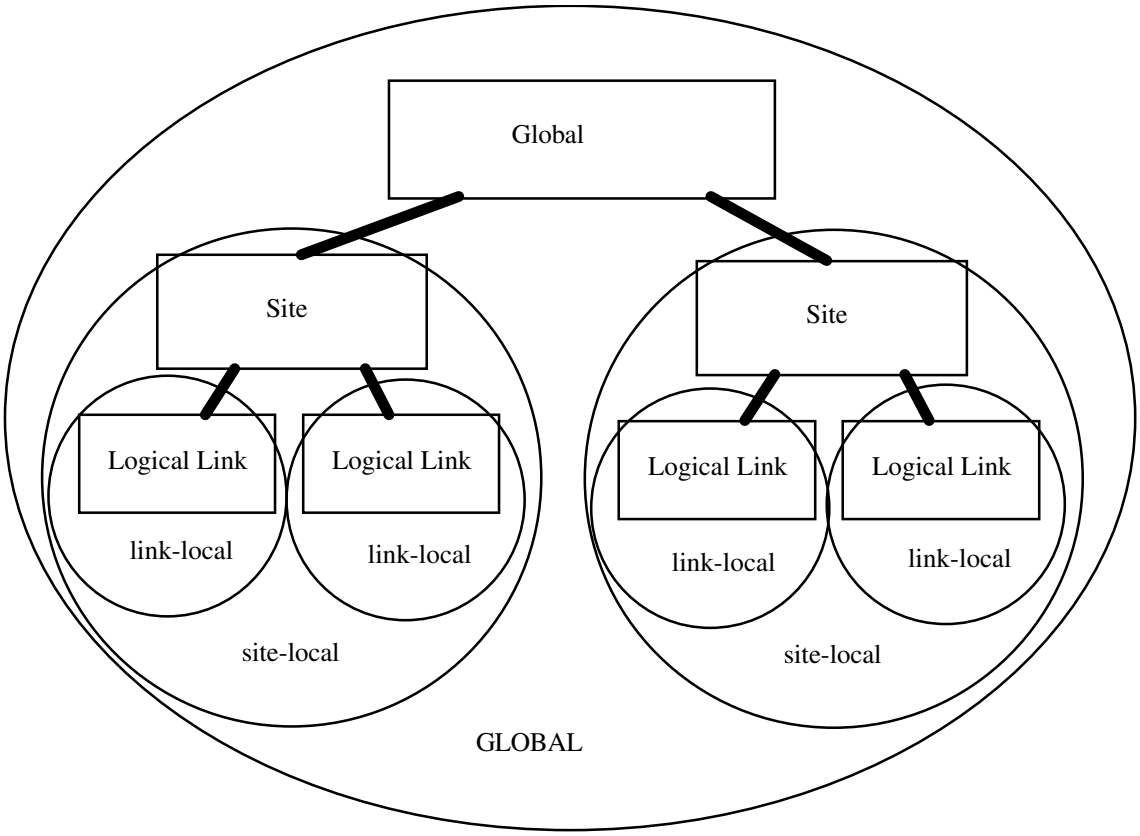
The specification of the NDS server hierarchy proceeds from these IPv6 address scoping rules. These address form a simple ˝hierarchy due to their scoping rules. The IPv6 hierarchy has three ˝levels:

- **link-local** — the lowest level which is limited to the local network link or Logical Link.
- **Site-local** — the "middle" level which spans multiple links, but is not globally visible.
- **Global** — the highest level which has unlimited scope.

By maintaining these concepts in this framework, IPv6 concepts ˝and addressing structure can be mapped onto ATM.
IPv6 address scopes can be viewed as:



In the proposed NDS tree structure, these IPv6 addressing scopes would be mapped as shown in the following figure:



In its simplest form, the NDS tree has one NDS server for each level of IPv6 addressing hierarchy. That is, there is a server ˝at the link-local level, a server at the Site level, and a server at the Global level. Servers at lower levels all connect to the server at the next higher level. All nodes are connected at the link-local level. The NDS servers control the flow of Neighbor Discovery messages between levels based on the addressing scope ˝of the message's target address. This then defines three NDS servers, each belonging to a specific logical level (denoted by the notation NDS(*level*) as follows) :

- **NDS(LL)** — The NDS server which serves the Logical Link and handles all ND messages with link-local scope.
- **NDS(Site)** — The NDS server which serves the Site level and handles all messages with site-local scope.
- **NDS(Root)** — The NDS server which serves the global address space and handles all ND messages with global scope.

Additionally, a fourth server is defined, which will be explained in the following sections. This is

- **NDS(N)** — The NDS neighborhood server which serves a subset of a logical link. Only NDS(N) servers can accept connections from individual nodes.

Note that the NDS(Root) server need not be separate from any of the other servers as the tree extends only to a particular ˝level.

While IPv6 specifies three addressing scopes, having only three layers in the NDS hierarchy may be insufficient on large ATM networks. On large networks, the number of nodes that may need ˝to be connected to a specific NDS may be larger than the system running the NDS can handle. Also, it may be desirable to distribute the NDS tasks to more servers to better distribute the load on the NDS servers. To accommodate this, and to allow the tree hierarchy to scale, the number of NDS levels in the tree ˝does not need to correspond to the number of levels in the IPv6 addressing hierarchy. Instead, the NDS tree is specified in ˝terms of both the number of logical levels (which is always three or fewer), and the number of physical levels (which can be any ˝number as necessary to achieve the scalability and connectivity requirements of a specific ATM network). The logical levels always correspond exactly to the IPv6 addressing scopes. Thus, within the link-local logical level (the Logical Link), there could be any number of physical NDS server levels (the height of the subtree), each composed of any number of NDS servers as ˝needed (the width of a level of the subtree).

Because each level of NDS server filters out ND packets based on target address (only passing up packets with higher addressing scope) the amount of traffic flowing between servers should be lower at the higher levels of the tree. This should enable the ˝NDS tree to scale to any size, even global.

Note that all nodes in an ATM network need not be connected via the same tree. Multiple NDS trees can exist in any ATM network ˝so that the nodes in each tree are logically isolated from each other. Because of this logical isolation, nodes in each tree can not directly contact nodes in another tree at the IPv6 layer ˝(even though ATM connections are still possible at the physical layer). Nodes managed by one tree **MUST** communicate to nodes in another tree via a router.

Finally, the NDS tree is meant to handle only Neighbor Discovery protocols and address configuration protocols. It is not meant, and **MUST NOT** be used to carry general multicast and broadcast data. The NDS tree handles only multicast data to a specific set of multicast addresses. A separate multicast mechanism must be used to implement IPv6 multicast capabilities (see [IP-ATMMC]). Packets to the following multicast addresses **MUST** be sent via the NDS tree:

- all-nodes multicast address
- all-routers multicast address
- solicited node multicast address
- DHCP server multicast address

These multicast address differ from more generic multicast functions in that they are used as part of some discovery mechanism which usually involves a unicast response to the multicast. Other messages that follow this model could also be carried by the tree. For example, router protocols such as RIP, or service location protocol messages [IP-SRVLOC]. While this framework does not explicitly describe the transmission of such packets through the NDS hierarchy, it does not prohibit the use ˝of the hierarchy by protocols other than ND protocols. Using the ˝NDS tree to carry any other multicast protocol **MUST** be done only with IETF agreement for specific purposes. Nodes **MUST NOT** use the NDS tree to carry multicast traffic not explicitly specified for the NDS tree by some IETF standard.

The framework proposed here requires no modification to the existing neighbor discovery and address configuration protocols. They will continue to operate as they do on legacy LANs. ˝However, some additional protocols are required for node registration and NDS server-to-server communications, but these are isolated to ˝the layer between IPv6 and ATM. Also, this framework does not depend on any central repository of information for the network to operate. While some network state and reachability information ˝is maintained by the servers, this state relates to subnet reachability rather than node specific information. All node specific information is obtained through the existing neighbor discovery mechanisms rather than from servers. This makes the NDS easier to implement, more efficient, and failover easier.

Implementing the physical NDS levels under UNI 3.0/3.1 and UNI ˝4.0 will require that each entity (either a node or NDS server) at a specific level in the tree be configured with the ATM link-layer address of the NDS server (or servers if redundant or distributed NDS severs are used) immediately above it. Of course, this will require entering a great deal of link-layer addresses into every entity on the tree. Not only is this time consuming and error prone, but since ATM link-layer addresses are not fixed, some changes in ATM network topology could cause some NDS server's address to change, requiring reconfiguration of some number of nodes or servers.

The UNI 4.0 [ATM-UNI40] specification defines an ATM anycast capability which can be applied to this framework to the NDS tree to be completely self configuring. The UNI 4.0 anycast ˝capability allows ATM nodes to register group addresses (anycast addresses) with the network. A group address is registered with a specific reachability scope which determines which nodes will be able to connect to the registering node using the registered group address. This reachability is determined by the ATM network hierarchy and the PNNI [ATM-PNNI] call routing mechanisms. The UNI 4.0 (and PNNI) specification define two types of group ˝address scoping. One type is referred to as PNNI routing scope and is directly related to the PNNI network routing hierarchy. This scope has 104 level, but is very closely tied to the network topology. The second defined scope is called the organization scope. This scoping is a logical scoping which has been designed to more naturally map on to human organizations. This type of scope has only 16 levels. This framework uses organizational scope for the NDS autoconfiguration functions because it represents a much more manageable number of levels, and because ˝it not as dependent on ATM network topology as PNNI scoping. The UNI 4.0 and PNNI specification define the following scopes:

- Local Network (LN)
- Local Network plus one (LN+1)
- Local Network plus two (LL)
- Site Minus two (S-2)
- Site Minus one (S-1)
- Site (S)
- Site Plus one (S+1)
- Organization minus one (O-1)
- Intra-Organization (O)
- Organization Plus one (O+1)
- Community minus one (C-1)
- Intra-community (C)

- Community plus one (C+1)
- Regional (R)
- Inter-regional (IR)
- Global (G)

When an ATM node places a call to a group address it must specify the highest scope through which the call will be routed. The caller can limit the scope within which a call will be routed, ˝and thus, what set of target node can be reached. A call will only ˝be delivered to a called node if the scope of the advertised group address includes the calling node, and if the routing scope specified in the call includes the target node's group address scope. A call to a group address is routed to the "nearest" node based on the ATM network topology.

Using UNI 4.0 anycast addressing, it will be possible to make the NDS tree autoconfiguring by having every entity (either a node or an NDS server depending on the level) at one level in the tree connect to the NDS server above it simply by placing a call to ˝the server's group address using the appropriate scope in the call. Using this mechanism, an NDS tree would form based largely on the topology of the ATM network and the placement of servers within that topology. In this way, only a small number of well known addresses would be required to configure a very large tree, or even multiple trees. Effectively, only one well known address at each of the 16 possible levels of the physical tree would be required.

In UNI 4.0 networks, where group addresses are utilized, there **MUST** be one unique group address at each level of the hierarchy which servers at each level register (for a total of 16 assigned group addresses). Multiple group addresses are required since ˝the routing of group address calls is always to the "nearest" (in terms of the ATM network topology) node which has registered that address with the network. Because of the way ATM anycast call routing is specified, it would not be possible to autoconfigure the hierarchy using only a single group address and relying on ˝the anycast routing scope rules to direct the call to the intended recipient. Effectively, the "nearest" node that registers a group address "masks" the presence of servers who advertised the same group address with a greater scope. The group address used by each level of NDS server is denoted by GA(*level*) where *level* is one of the organizational scopes described above. Each GA(*level*) is registered with the network with the scope corresponding to ˝the level (that is, GA(C) would be registered with the community scope).

Group address are assigned by the ATM Forum. To support this framework with IETF would request a set of group addresses from the ATM Forum and assign them for use by IPv6 over ATM. These group addresses would be well-known addresses permanently ˝assigned to IPv6 over ATM.

The following sections describe how this hierarchical tree of NDS servers is organized and configured both manually and automatically.

3.1. NDS Trees

An NDS server tree or subtree is formed by interconnecting some number of physical NDS servers is a specific manner. The

connections between nodes and NDS servers at any level of the ˝tree use the same procedures, and set up the same type and number of connections. This results in a tree in which each level is organized exactly like any other level, and in which only one procedure for interconnection of the levels must be used. The manner is which the NDS tree is connected is done to permit a specific type of dataflow within the tree.

The NDS tree is composed of a number of physical NDSs at some number of levels connected with the following ATM VCs:

- Connected to the NDS at the next higher level with a PtP VC.
- Connected to the NDS at the next higher level by a PtM VC which is rooted in the NDS at the next higher level.

With these connections, each NDS has a PtP connection over which it can communicate with each individual NDS below it, and a PtM ˝VC over which it can "broadcast" to each NDS below it. This connection structure, when extended to some number of levels, provides a way for packets to be "broadcast" to a set of NDSs or nodes on the tree (or to all nodes).

The general dataflow in the tree is that packets are always unicast up the tree towards the root. At some point an NDS can "broadcast" the packets back down the tree as required by the protocols. An NDS can also unicast down the tree to a specific server under it according to the processes described later in ˝this document.

The physical NDS tree can be of any height or width as needed. However, at some point on the tree the boundaries for a Logical Link and Site must be defined. The specification of the NDS(LL) or NDS(Site) servers can be at any layer in the tree based on ˝node connectivity requirements. Thus, the establishment of an NDS(LL) or NDS(Site) server is done entirely by configuration rather than placing these servers at some arbitrary level in the tree. ˝Once a physical NDS has been designated as an NDS(LL) or NDS(Site), then that NDS must perform additional special functions. Also, all ˝NDS servers above the NDS(LL) servers must be configured as being outside a LL so they can perform additional ND message routing functions as described later (servers below the NDS(LL) do not perform these functions).

The part of the NDS tree below the NDS(LL) server is referred to as the LL subtree. The part of the NDS tree below the NDS(Site) server is referred to as the site subtree.

In each LL there **MUST** be exactly one NDS(LL) server. In each Site, there **MUST** be exactly one NDS(Site) server. In the whole NDS tree there **MUST** be exactly one NDS(Root) server. The NDS(Root) server can be the same as the NDS(LL) or NDS(Site) servers if the tree terminates at that level.

3.1.1. Manual NDS Tree Configuration

In UNI 3.0/3.1 or UNI 4.0 environments where manual configuration is used, each node and NDS must be configured with at least one ATM link-layer address of the NDS to which it must connect. An NDS that has no configured NDS address **MUST** be configured as the NDS(Root) server.

Each node or NDS will be configured with the ATM link-layer addresses of a set NDS servers to which it can connect. Each NDS **MUST** connect to exactly one NDS server above it unless it is the NDS(Root) server (which by definition has no NDS server above ˝it). Nodes may connect into multiple LLs. A separate table of NDS server addresses **MUST** be maintained for each LL. Each node **MAY** also make more than one connection to a given LL.

All nodes **MUST** create and bind to at least one ATM NSAP for each LL connection they will make. This NSAP is used to connect to ˝the NDS tree, to receive the PtM connection from the called NDS server, and for receiving data connections from other nodes. The node **MAY** bind to separate NSAPs for each of these connections as long as it keeps track of which NSAPs it will use for each type ˝of connection. All NDS servers, other than the root server, **MUST** also create this (these) binding(s).

When a node or server is configured to connect to more than one NDS, the ATM link-layer addresses configured in the NDS server table **SHOULD** be placed in order to prioritize the order in which NDS servers should be used. This will provide load distribution between multiple NDS servers at the same level, while still permitting tree function and failover in the event one server fails.

All NDS servers **MUST** bind to an ATM NSAP over which they will accept calls from nodes or NDS servers below them. This NSAP **SHOULD** be as deterministic as possible since it **MUST** be entered in the table of NDS addresses for servers or nodes below. If this NSAP changes all servers or nodes which connect to this NDS must be reconfigured.

The process for connecting any node or NDS into the NDS tree is the same.

- Starting with the first valid ATM link-layer address in its table, the node or NDS places a PtP call to that address. If the call fails then the next address is tried. If calls to all addresses fail then the node or NDS backs off for some random period of time [TBD] and repeats the process until a call succeeds.
- When the PtP VC is established the node or NDS will perform a registration process [TBD] with the called NDS.
- When the registration process is completed successfully, the called NDS server will add the calling party to a PtM connection which it uses to “broadcast” packets to all connected nodes or NDSs.

This process is performed by every node and NDS on the tree until the entire tree is formed.

The tree **MAY** also be statically configured using PVCs.

If an NDS fails the nodes or NDSs to which it is connected will immediately receive an ATM RELEASE notification, and all the server’s VCs will be torn down by the network. When the nodes or servers below the failed server detect this condition **MUST** initiate the initial connection process described above. The NDS server above the failed NDS server or node **MUST** destroy any cached information associated with the failed server or node, but **MUST NOT** take any action to re-establish the connection.

3.1.2. Autoconfiguration of Trees

Autoconfiguration of the NDS tree is only possible on UNI 4.0 ATM networks which implement anycast addressing. In fully autoconfiguring NDS trees, the number of physical levels in the tree is limited to 16 by the UNI 4.0 organizational scoping. If more levels are needed they must be manually configured.

Not all 16 levels need exist in an autoconfigured tree. Only those levels necessary to provide the needed level of ATM connectivity and scalability have to exist.

Note that autoconfiguration only works on private ATM networks since anycast addressing has not been defined for public ˝networks. An NDS tree on a public network would have to be configured manually (probably via PVCs for Internet providers).

In autoconfiguration environments, each NDS registers GA(X) for its level in the hierarchy. For example, the NDS at level LL would register GA(LL). The scope of the registered GA(X) depends on the height of the tree and the connection scope required by each NDS. Each NDS can register any reachability scope with ˝GA(X) as required to provide the needed connectivity. The only restrictions in group address scoping are that those NDS servers below the NDS(LL) server **MUST NOT** register their GA(X) with a scope wider than the GA(X) of the NDS(LL) server. Similarly, NDS servers below the NDS(Site) server **MUST NOT** register their GA(X) addresses with a scope wider than the NDS(Site) server or less than the NDS(LL) server. Finally, NDS servers between the NDS(Site) server and the NDS(Root) server (if they are not the same) **MUST NOT** register their GA(X) addresses with a scope wider than that of the NDS(Root) server, or lower than the NDS(Site) server.

These scoping requirements permit some backup and redundancy in various levels of the tree. For example, if all nodes in an LL are connecting through NDS servers at the LN level, each NDS(LN) server could register GA(LN) with a scope greater than LN and ˝less than or equal to the scope used by the NDS(LL) server (which ˝would have to be at level LN+1 or higher). With this type of scoping, nodes would normally connect to the closest NDS(LN) server based on the ATM network topology. However, if that server were to fail, nodes would the automatically reconnect to the next nearest NDS(LN) server.

To perform autoconfiguration all nodes and servers create and ˝bind to all NSAPs as specified for manual configuration. NDS servers **MUST** always have unicast NSAPs through which they can be reached. In addition to the NSAPs described for manual configuration, all NDSs **MUST** register GA(X) for its level with appropriate scope and bind to that NSAP to receive calls.

Each node and NDS on the tree **MUST** be configured with its level on the tree. These levels are numbered 0 through 16, with 0 being a node, and 16 being the highest level NDS server permitted.

To configure the tree, the following process is used. Each node and NDS **MUST** have a table with the list of GA(X) for each level X.

- The node or NDS at level X places a call to GA(X+1) with a call routing scope of X+1. If the connection fails then a call to GA(X+2) with call routing scope of X+2 is tried. If the end of the GA table is reached the node or NDS backs off for some random period of time [TBD] and then repeats the procedure.
- If the call succeeds, the calling party performs a registration process with the called NDS. Part of this process involves providing the called NDS with the calling party's level in the hierarchy. If the calling party's level in the hierarchy is not appropriate for the called party (i.e., a node reaching an NDS above the NDS(LL) server), the connection is released, the index into the caller's GA table is reset, the process is repeated until an NDS at the appropriate level is reached.
- If the called NDS learns through the registration process that it has connections from more than one level in the hierarchy, it will release the connection from the node or NDS at the lowest level. NDS servers **MUST NOT** permit connections from more than a single level below (otherwise the tree structure would not be regular).
- Once the registration process completes successfully, the called NDS server adds the calling party to its PtM connection which it uses to "broadcast" to all calling parties.

If an NDS server fails, the nodes or NDSs to which it is ˝connected will immediately receive an ATM RELEASE notification, and all the server's VCs will be torn down by the network. When the nodes or servers below the failed server detect this condition they **MUST** initiate the initial connection process described above. The NDS server above the failed NDS server or node **MUST** destroy any cached information associated with the failed server or node, but **MUST NOT** take any action to re-establish the connection.

As stated earlier, there **MUST** be exactly one NDS server at the NDS(LL), NDS(Site), and NDS(Root) levels. This means that only one physical NDS(X) server which functions as one of these ˝special servers is allowed to register GA(X) at any given time. This restriction is required to avoid fragmenting a LL or Site, which is what would occur if there were more than one server which registers a specific GA with the same scope. By restricting ˝these levels to a single NDS sever, the same procedures used achieve redundancy and failover at other levels (where multiple servers are permitted) can not be used. Thus, some new procedure must be defined to achieve failover at these levels. This procedure is only necessary for the servers which function as the NDS(LL), NDS(Site), and NDS(Root) servers.

This procedure is TBD.

3.2. Forming Logical Links

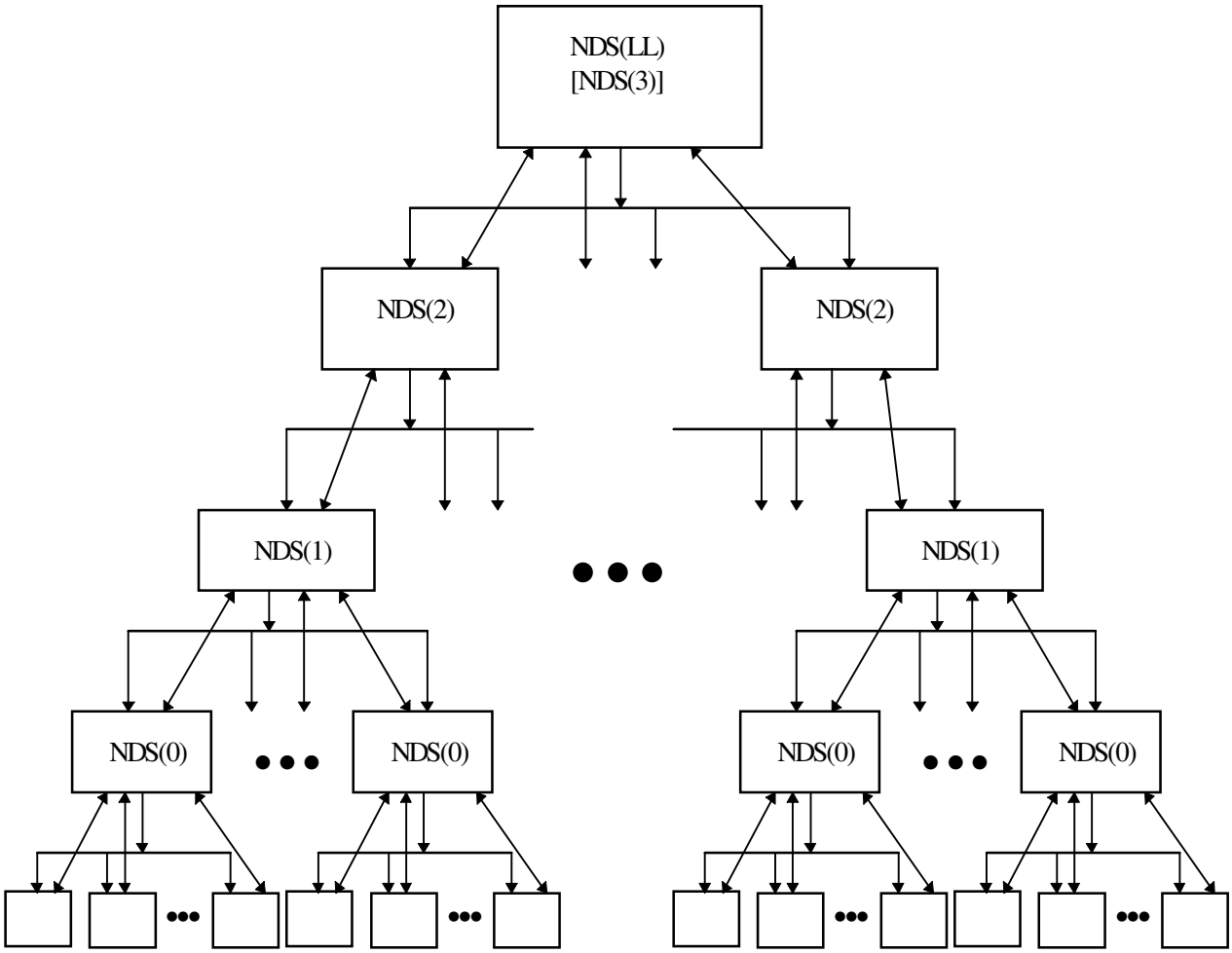
Logical Links are the basic unit of the NDS tree. Like LAN segments, they form the basis of an IPv6 ATM "internet". Logical Links act very much like LAN segments, and are the point to which all nodes connect.

At a minimum, a Logical Link consists of one NDS(LL) server to which one or more nodes are connected. In a Logical Link there ˝is always exactly one NDS(LL) server. This server directs the ND message traffic flow within the LL, controls which messages are forwarded outside the LL to the Site, and controls what messages are admitted into the LL from the Site.

Physically, the NDS(LL) server does not have to be the server to which all nodes are connected, nor does it have to be the only server on the LL. All that is required is that the single ˝NDS(LL) server be located at the root of the LL physical NDS subtree. ˝The LL physical NDS subtree can be as wide or deep as needed to achieve the desired node connectivity and LL scalability. This means that there could be any number of NDSs within the LL, but that the NDS which is the root of the LL subtree must be explicitly identified as the NDS(LL) since it must perform ˝special functions.

The underlying characteristic of an LL is that all nodes on the ˝LL are members of the same IP subnets (they use the same prefixes to configure their addresses), and that any node on the LL can use link-local addresses to reach any other node on the LL. Link-local addresses are limited in scope to Intra-LL communications and are not visible outside the LL. Finally, an LL forms a "broadcast" domain in which any node can "broadcast" data to all other nodes on the same LL (but not to other LLs on the same ATM network).

The following figure shows an example of a four level LL. The nodes are all connected to NDS(0) servers. These are connect to NDS(1) servers, which are then connected to NDS(2) server. The NDS(2) servers are finally connected to an NDS(3) server. The NDS(3) server is also configured as the NDS(LL) server. In UNI 4.0 environments where autoconfiguration is enabled, the scope of the group addresses GA(0), GA(1), and GA(2) must be less than or equal to the scope of address GA(3). Thus, the scope of the ˝LL on the ATM network would be the scope with which the GA(3) ˝address is registered. All nodes within this scope would connect to this ˝LL.



A Logical Link has the following properties:

- All nodes in a LL can establish direct ATM virtual circuits to any other node on the LL
- All nodes in a LL can perform Neighbor Solicitation to ˝discover the ATM address of any other node on the LL using link-local addresses or global addresses
- Host on one LL can not be discovered or contacted using link-local addresses by nodes on an other LL

Node can connect to multiple LLs. This places one level of abstraction between the IP layer and the ATM layer, essentially virtualizing the ATM layer into multiple logical links. In manually configured environments the ATM NSAP of an NDS server to which a node connects for each LL that a node could join would have to be configured on the node. In a UNI 4.0 ˝autoconfiguration environment the node could join only the closest LL using autoconfiguration, but could join other LLs using manual configuration. However, if a node wanted to join a specific subnet (as opposed to a specific LL) this could be handled via a special protocol which will be described later.

In most cases, each LL will be associated with a single IP ˝subnet. In this case a LL will equate to an LIS. However, just as it is permissible to configure multiple subnets on a physical LAN segment, it will also be possible to configure multiple subnets ˝on an LL. This framework places no restrictions on how IP subnet ˝can be formed when compared to existing LAN technologies.

Just as with legacy LANs, IP subnets can not span multiple LLs. With legacy LANs, IP subnets can not be spread across multiple, physically disjoint LAN segments (without further subnetting). This restriction remains when using LLs. Multiple LLs can not be aggregated at the link level. Aggregation occurs at a higher level which does not permit the sharing of subnet addresses by ˝two LLs.

3.2.1. Neighborhoods

There is one last special class of NDS server. This is called ˝the neighborhoods server, or NDS(N). The NDS(N) server is the server to which individual nodes **MUST** connect. NDS(N) server **MUST NOT** accept connections from other NDS server. Thus, the registration process used by the calling party connecting to the NDS will provide the information the NDS needs to determine the identity ˝of the connecting party.

The NDS(N) servers perform some special functions involved in connecting nodes to the NDS tree and perform some special ND message filtering. These functions will be described in more detail in the sections covering the ND protocol operations on the NDS tree.

In the figure above, the NDS(N) servers are the NDS(0) ˝servers.

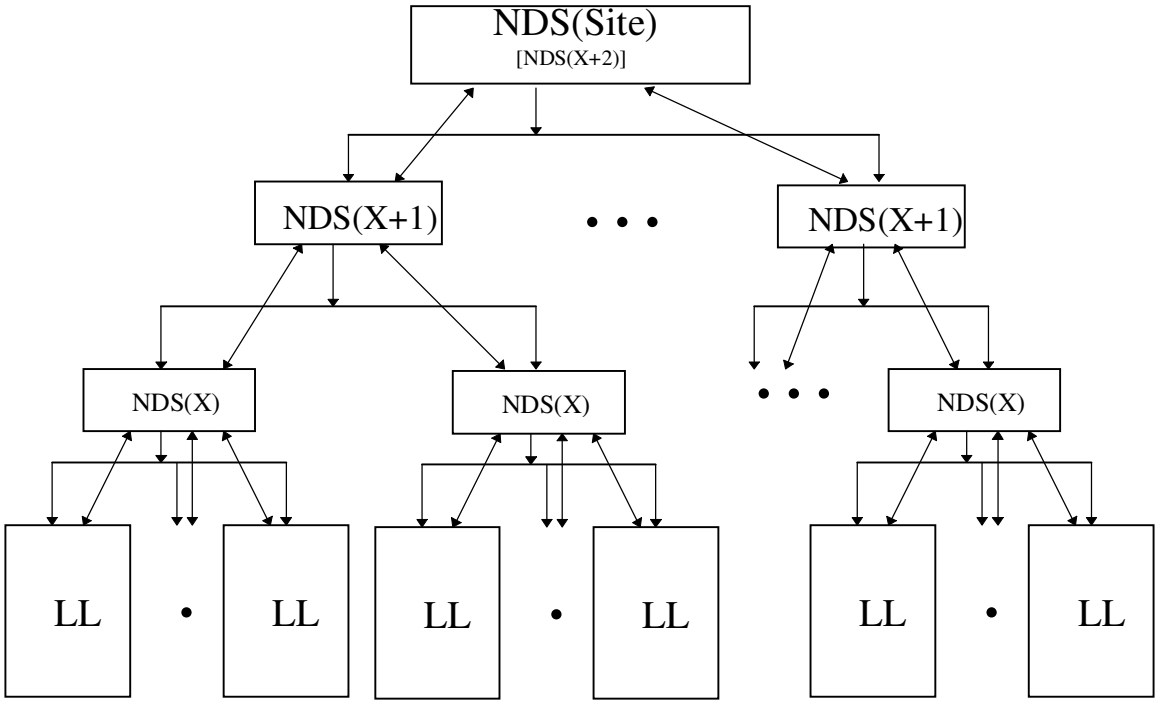
3.3. Forming Sites

A site is simply a collection of a set of LLs whose nodes can connect using IPv6 site-local addresses. A site is built from multiple LLs by connecting the NDS(LL) servers to an NDS(Site) through zero or more intermediate NDS servers. Each site terminates at exactly one NDS(S) server. The NDS(Site) level server must exist to establish a site. The number of levels in the Site tree can be extended to provide the needed levels of connectivity and scalability.

The NDS(S) server performs filtering of ND messages entering and leaving the site. It will not pass any site-local solicitations outside the site, thus limiting the scope of site local ˝addresses.

The site layers of the hierarchy are illustrated in the following figure. This figure illustrates a site composed of three levels of NDS servers above the LL layer. The site terminates at a single NDS(Site) server three levels above the NDS(LL) servers. The NDS(X+2) server is configured as the NDS(Site) server. ˝In UNI 4.0 autoconfiguration environments, the NDS server between the NDS(LL) servers and NDS(Site) server **MUST** register their respective global addresses with a scope greater than the scope ˝of the NDS(LL) servers, and less than or equal to the scope of the NDS(Site) server. Thus, when using autoconfiguration, the scope

of the site is the scope with which the NDS(Site) server ˝registers its group address. The site will include all nodes within this scope as determined by the ATM network topology. The scope of ˝the Site NDS(Site) server can be selected so that it includes all the nodes which are to be members of the Site.



A site has the following properties:

- All nodes in the site can connect to all other nodes in the site by creating an ATM virtual circuit
- All nodes in a site can perform Neighbor Solicitation to discover the ATM address of any other node on the site using site-local address or global addresses
- Hosts in one site can not be discovered or contacted using site-local addresses by nodes in another site

3.4. Beyond Sites

The tree structure can be extended by more levels as needed. Sites can be grouped together to create larger groupings of ˝nodes. However, only global IPv6 address can be used between nodes of different Sites. The tree can be extended up as far as is necessary and can be made as wide as necessary to include all the Sites which will be allowed to communicate.

A full tree is used to manage reachability of a set of nodes on ˝an ATM network. These nodes are logically grouped into logical ˝links and sites, but are capable of directly establishing a connection to each other. Multiple trees can exist within the same ATM network. In this case, nodes managed by different trees would be required to go through a router to communicate even if they were capable of creating direct connections. A tree consists entirely of nodes which are allowed to directly connect. A node can not ˝be part of a tree if it can not connect to every other node on the tree.

There is no limit to the number of LLs they may join, or restrictions on which LLs they may join. A node can connect to many different LLs, either within the same NDS tree, or in different NDS trees. Hosts that must belong to a specific LL, ˝but which are not connected to the ATM network within the scope of an NDS(N) server with that LL, can connect to specific LL in two ways:

1. Connect directly to an NDS(N) in the target LL by manually configuring the unicast address of the NDS(N) server(s) in that LL.
2. Allow the NDS tree to relocate the node using the protocol described under manual IPv6 address configuration. For the node to be automatically relocated, the node must be assigned an IPv6 address (via either DHCP or manual IPv6 address configuration), which belongs to another LL.

In UNI 4.0 autoconfiguration environments the configuration of ˝the tree will be determined by the ATM network topology. That is, ˝the topology of the network will determine which systems can interconnect using group addresses of a given scope. Thus, NDS server systems should be chosen taking their location on the ATM network into account. Having various NDS servers register their address with a higher scope will also relieve some of the topological restrictions.

Finally, any system can perform NDS server functions at multiple levels. There is no restriction (other than possibly ATM topological restrictions) on how many levels can be served by a single physical system. All that has to be done is to configure the system appropriately. The system would then register ˝multiple GAs and the appropriate scopes. However, there is no provision made in this specification for providing Intra-system cut-through in cases where the same system is serving adjacent levels. In this case, all communications between NDSs at adjacent levels ˝will be carried across the ATM network (though only to the server's local switch).

4. NDS Operations

Once formed, the NDS tree provides a way for reachability information to be circulated between LLs as well as a way of "broadcasting" or "multicasting" certain information. IPv6 addressing scopes are mapped on to the layers of the tree by designating specific NDS servers a Neighborhood, Logical Link, Site, and root servers. These NDS servers perform the specific functions necessary to implement IPv6 addressing semantics within the tree. All NDS servers work together to maintain IPv6 ˝Neighbor

Discovery semantics within the tree. Thus, the tree structure maps IPV6 onto ATM in a consistent manner.

The flow of data through the tree has the following characteristics:

- data from the leaves of the tree is unicast through the NDSs towards the root
- data coming from the root can be "broadcast" towards the leaves of the tree
- data from the root can also be unicast towards specific leaves or branches (this is also referred to as "directing" the data down the tree)

This dataflow provides full ability to "broadcast" data from any node to all other nodes at any level in the hierarchy. Because ´of this feature, it is possible to use the existing IPv6 ND and DHCPv6 protocols with no modifications. Also, since the ability to "broadcast" solicitation and other packets is provided, there is no need for any central server to maintain node reachability information. Host reachability information can continue to ´reside in each node, and can be queried by any other node connected to the tree. This enables the NDS tree to provide the same ´semantics as ND expects on legacy LANs.

The IPv6 ND and DHCPv6 protocols require that nodes join various multicast groups. In the NDS tree, multicasting reception of ND packets is handled simply by filtering incoming packets which arrive on the node's PtM connection from the NDS(N) server. That is, all multicast packets are really "broadcast", and each node must perform appropriate address filtering. For example, all ´non-router nodes must discard any packets received which are ´addressed to the all-routers multicast address. Since this framework specifies only a specific set of multicast addresses which are to be treated in this fashion, and since packets addressed to these addresses will only arrive on specific VCs, this filtering can be provided without affecting general multicast functions ´[IP-ATMMC].

The tree can be used to broadcast to the following multicast addresses:

- The all-nodes address
- The all-routers address
- The DHCPv6 server/Relay Agent address
- The solicited-node address

The tree is not intended to handle the general multicast case and must not be used for this purpose (it would be undesirable to use it for this purpose since the load on the NDSs and latencies ´would be unacceptable for sending data). The broadcast mechanism of ´the LL should be used only for sending broadcast data that is used ´to determine node reachability, service discovery, or routing protocol information.

The IPv6 ND solicitation/advertisements mechanisms work as follows:

- all solicitations and advertisements sent to a multicast or broadcast address are sent to all nodes on an LL

- Unicast advertisements in reply to a solicitation are send via a VC which the solicited node creates to the soliciting node.

The following sections describe how the existing IPv6 protocols are applied and how any new protocols will be used.

4.1. Router Advertisements

Router advertisements are sent by routers to the all-nodes multicast address by all routers at periodic intervals. These advertisements not only contain information about the router ˝being advertised, but also address configuration information which is used by each node to perform stateless address configuration and to determine which prefixes are on-link. The router ˝advertisement can also contain information about the network such as what the default MTU size should be for communications on the subnet, and whether stateful address configuration should be used.

Router advertisements are handled specially by the NDSs since ˝they contain information about the local network (LL) configuration. There are no required changes to either router or nodes for ATM. They will continue to use router advertisements as described in the ND specification. The special work is done entirely by the NDSs and is transparent to IPv6 entities.

Like any node, a router must first connect to an NDS(N) server to become part of a Logical Link. Part of the process of joining a logical link is the registration of the node with the NDS(N) server. When a router connect to the NDS(N) server and ˝registers, it **MUST** identify its self as a router. This information is required by the NDS(N) server to handle certain types of failure and recovery conditions described later. Otherwise, the connection of a router to an NDS(N) server is exactly the same as the connection of any node to the NDS(N) server.

When a router sends out an advertisement, it will send it to the all-nodes multicast address. To do this, it must send it advertisement out on its PtP VC that connects to its NDS(N) server. The NDS(N) server will forward the message up the tree. The message will continue up the tree until it arrives at the NDS(LL) server (which could be the same as the NDS(N) server). The message is moved up the tree by each NDS server receiving the message on one of its PtP VCs from the server below, then forwarding the message to its PtP VC to the server above. When ˝the NDS(LL) server receives the router advertisement it performs the following actions:

- Records the information about the router. This includes the router lifetime, max. hop limit, M and O bits, reachability time, and restrans timer.
- Records the MTU option if present.
- Records the source link-layer address if present
- Records all prefix information including the prefix length, L and A bits, and lifetimes.

All this information is placed in a cache which ages using a TBD algorithm so that the information in the cache is never older ˝than the same information that is cached on nodes (that is, nodes must

never receive information from the NDS tree which is older than their information).

The NDS(LL) server **MUST** then take the router advertisement packets and send them (unmodified) via its PtP connection to the NDS server "above" it (i.e., out into the Site towards the root of the tree) if the LL is connected to a Site. This is how each NDS(LL) server makes other NDS servers and LLs aware of which prefixes ˝are available on the tree, and thus, which are "on-link". As each NDS server above the NDS(LL) server receives the router ˝advertisements from the server below, it **MUST** record all the prefix information in the advertisement and associate the prefix list with the PtP ˝VC on which the advertisement was received. This forms a prefix cache in each NDS server that is used to route other neighbor discovery messages through the NDS tree. The NDS server's prefix cache entries **MUST** be timed out and removed to assure that they do not contain stale information. The prefix caches **MUST** be updated when new router advertisements are received. Each prefix in the cache **MUST** be aged and flushed individually. All entries from a single router advertisement **MUST NOT** be aggregated and aged together.

When router advertisement messages arrive at the NDS(Root) ˝server, this server **MUST** "broadcast" down the tree via its PtM VC connecting it to all NDS servers directly below it. The router advertisements **MUST** then be relayed by each NDS server to its PtM connection down to the next level. Each NDS server **SHOULD NOT** record any information about router advertisements flowing in the downward (from the root) direction. The router ˝advertisements are relayed down the NDS tree until they are received by all NDS(LL) servers. The NDS(LL) servers **MUST** then record all the prefix information contained in the messages, and record these prefixes as being "external" to the LL. The NDS(LL) servers **MUST** also eliminate the redundant information caused by the receipt of ˝their own router advertisements. The purpose of forwarding all router advertisement messages to all NDS(LL) servers on an NDS tree ˝is to distribute prefix information, not to distribute router information. This information is eventually used by the nodes to determine reachability within the tree, not to locate routers ˝that are not connected to the same LL. In this framework, nodes can not reach routers that are not on the same LL since nodes must ˝use link-local addresses to communicate with routers. Since link-local addresses can not be used between nodes on different LLs, ˝no node can reach a router on a remote LL.

In some cases a router will not advertise all prefixes on the LL, or the LL could be entirely manually configured. In these cases, a list of prefixes which are to be considered on-link by all ˝other nodes on the NDS tree **MUST** be configured into the NDS(LL) server. The NDS(LL) server **MUST** create a router advertisement message and send it up towards the root of the tree at the same interval as router advertisement messages would normally be sent from ˝routers.

In some cases administrators may wish to restrict which local prefixes are advertised to the tree. In such cases, the NDS(LL) server would be restricted to sending only those prefixes which are manually configured on the server. To permit this, NDS(LL) servers **MUST** have a configuration variable which can be set to disable the sending of local router advertisements outside the ˝LL. When the NDS(LL) server is configured to disable forwarding of router advertisement messages outside the LL, the list of ˝prefixes to be advertised is configured manually on the server by an administrator.

The NDS(LL) does not relay any received router advertisements to the nodes on its LL. Instead, it creates its own router advertisements using all the information it receives from both local and remote routers. Thus, the NDS(LL) server is the single source of all router advertisements received by nodes on LLs. NDS(LL) servers will also advertise other subnet configuration information in router advertisements (information such as statically configured subnet information and MTU information as described later).

To create the router advertisement messages for the LL, the NDS(LL) server composes new router advertisement messages by "appending" prefix information received from "outside" the LL to each router advertisement message received from within the LL. Thus, each node in an LL will receive a router advertisement from each local router, but each advertisement will contain all external prefix information in addition to any internal prefix information. After composing a new router advertisement message, the NDS(LL) server will send these out the PtM to the NDS ´servers (or nodes) below it so the message is "broadcast" to all nodes on the LL.

When the NDS(LL) server creates a router advertisement message to be sent to all nodes on the LL, it **MUST** create a new router advertisement message as follows:

- Max Hop Limit is unchanged
- M bit is unchanged
- O bit is unchanged
- Router Lifetime value is adjusted to account for the time that has elapsed between the receipt of the router advertisement and the current time
- Reachable Time is unchanged
- Retrans Timer is unchanged
- MTU option (if received) is sent unchanged
- Source link layer address option (if received) is sent unchanged
- All prefix information options (if received) are sent as follows:
    - Prefix Length is unchanged
    - L bit is unchanged
    - A bit is unchanged
- Invalidation Lifetime is adjusted to account for the time that has elapsed between the receipt of the router advertisement and the current time
- Deprecation Lifetime value is adjusted to account for the time that has elapsed between the receipt of the router advertisement and the current time
- Prefix is unchanged

Additionally, prefix information from router advertisements received from other LLs **MUST** be appended to the prefix list as follows (all other information in external router advertisements **MUST NOT** be included when creating local router advertisement messages):

- Prefix Length is unchanged
- L bit is set to 1

- A bit is set to 0
- Invalidation Lifetime is adjusted to account for the time that has elapsed between the receipt of the router advertisement and the current time
- Deprecation Lifetime value is adjusted to account for the time that has elapsed between the receipt of the router advertisement and the current time
- Prefix is unchanged.

Finally, any needed security information **MUST** be added, and the packet checksum computed, prior to transmission.

The adjustment of the various timeout values must be included in the router advertisement creation process because router advertisements can be created at any time in response to router solicitations as described later, and because router ˝advertisement creation can be trigger at any time by the receipt of an internal or external router advertisement.

Anytime the NDS(LL) server receives new or updated router advertisements from outside the LL it **MUST** recreate the internal router advertisement messages and broadcasts them down the LL. ˝If there is more than one router connected to the LL, the NDS(LL) server **MUST** recreate an internal router advertisement message for each internal router, and **MUST** append the entire external prefix list to each router advertisement message created. This means nodes on an LL will receive duplicate, but consistent ˝information.

The NDS(N) servers **MUST** locally cache each router advertisement packet received from the NDS(LL) servers. These cached messages are used in the node registration process and directly respond to any router solicitation messages received from node.

When an NDS(N) server initially connects to the tree its cache of router advertisement messages will be empty. It **MUST NOT** attempt to fill the cache, but **MUST** wait for either the periodic advertisement packets to arrive, or for an node to solicit router information.

When the NDS(N) server receives the first router solicitation ˝from a node (the first since the NDS(N) server configured) the server will use this event to initiate the filling of its cache. To do this it **MUST** forward the router solicitation up the tree towards the NDS(LL) server (or process it locally if it also the NDS(LL) server). When the NDS(LL) server receives the solicitation it **MUST** perform its normal process of creating router advertisement messages and "broadcasting" them down the tree. This will cause all NDS(N) servers to refresh their router and prefix information caches. When the NDS(LL) responds to a router solicitation, it **MUST NOT** transmit any router advertisements outside the LL as part of the response.

When a node connects to an NDS(N) server, the NDS(N) server **MUST** send all its cached advertisements to the node via the node's PtP VC. This is done as part of the node registration process.

When an NDS(LL) server comes on line, the time necessary for all external routers to advertise their prefix lists may be unacceptably long (or non-deterministic). To get an immediate update of external prefix information, an NDS(LL) server **MUST** transmit a router solicitation message up the tree. This message

need not contain any specific information as it is only the message that has significance, not the message contents. When an NDS(LL) server receives a router solicitation message from the ˝NDS server above it, it **MUST** compose router advertisements containing all local router information (no external prefixes) and send ˝those advertisements up the tree towards the root NDS server. To facilitate this, NDS(LL) servers **MAY** cache individual internal router advertisement messages and send those rather than creating new messages, though it is acceptable for an NDS(LL) server to construct a new router advertisement message as long as the new message contains the correct information. As the router advertisement messages propagate through the tree they will ˝update the caches of all NDS servers, including the soliciting server.

When any NDS server outside an LL comes on line it **MUST** send a router solicitation to request that the prefix information on the subtree below it. To do this, the NDS server **MUST** send a router solicitation message on each NDS server below it as the servers connect and register. The solicitation is transmitted on the connected server's PtP VC. In this way, the solicitations are sent to each subtree under the recovering NDS server so that the recovering NDS server can update its table of "internal" prefixes.

Anytime the NDS(LL) server receives a new or updated router advertisement from a local router it **MUST** send the router advertisement out towards the root of the tree if this function ˝is enabled. This will provide all other LLs timely updates of reachability information. It **MUST** also recreate the router advertisement messages to send down to the nodes with the current information contained in the latest message received from the router.

With this process, the NDS(LL) server has all information about which prefixes are locally on-link and which are remotely ˝on-link. That is, it has a list of "internal" and "external" prefixes. This information is used in routing other ND packets. The ˝NDS(LL) server **MUST** timeout all router information and prefix information based on the information contained in the original router advertisement packets received from the source.

The prefix information received at each NDS server above the LL level **MUST** be used to control routing of certain neighbor discovery messages through the NDS servers. This is done to minimize traffic between NDS servers, and the number of NDS servers which must process specific messages. The prefix cache maintained by each NDS server between (and including) the NDS(LL) server and the NDS(Root) server is used to control how specific messages are routed through the tree. With this cache, each NDS server has a list of prefixes which are valid within one of the subtrees below it. When an NDS server receives certain neighbor discovery messages, they **MUST** compare the prefix of the message's target address to the prefixes in their prefix caches. If a ˝match is found then the message is relayed on the PtP VC associated ˝with the prefix. If no match is found then the message **MUST** be relayed to the next NDS towards to the root of the tree.

The same routing mechanism **MUST** also be used to route neighbor discovery messages that are moving down the tree. When an NDS server receives neighbor discovery messages from the server above it, it **MUST** perform a comparison of the target address prefix to the entries in its prefix cache. If a match is found then the message **MUST** then be relayed to the specific PtP VC associated

with the matching cached prefix. If a no match is found then the message **MUST** be transmitted down via the NDS server's PtM VC connecting all server's below.

NDS servers within an LL **MUST NOT** direct any ND messages, but **MUST** transmit them via their PtM VC.

Using this method of routing NDS traffic flow in both the up and down directions, messages only travel up the NDS tree until they arrive on the first NDS server that recognizes the target's prefix. From there, the message is directed to a specific LL.

All NDS servers outside an LL **MUST** time out all router and prefix information based on information received in the original router advertisement messages. This **MUST** be done to assure consistency of prefix information used and advertised throughout the tree.

Under some circumstances it will be necessary to flush the router advertisement caches on the NDS(N) servers. This has to be done so that router solicitations from nodes will always cause current network information to be returned. Specifically, information ˝for a router that is no longer available (due to router or network failure) should not be returned.

When a node initially connects to an NDS(N) server it will ˝perform a registration process. As part of this process, the node must identify itself as an end-system node, or a router (or both). If the node identifies itself as a router, the NDS must perform special actions when it detects that the node has gone off-line (i.e., the VC to the node is released). When an NDS(N) server detects that a router is disconnected (through the release of the VC), it **MUST** send a cache flush message (a new NDS message whose format is TBD) up towards the NDS(LL) server. When the NDS(LL) server receives the cache flush message it **MUST** flush its own local router advertisement cache and then "broadcast" the cache flush message down towards all NDS(N) servers on the LL. ˝When the NDS(N) servers receive a cache flush message they **MUST** remove all entries from their router advertisement caches. Additionally, they **MUST** forward the cache flush message to any nodes which have registered as routers.

When a router receives a cache flush message it **MUST** resend its router advertisement to the all-nodes multicast address. This will cause the NDS(LL) server, and then all NDS(N) servers to receive updated information from all remaining routers on the ˝LL.

When a router initially connects to its NDS(N) server it **MUST** send a router advertisement to the all-nodes multicast address. This is required since routers do not receive router solicitation messages from nodes.

If any server between a node and the NDS(LL) server fails (including the NDS(LL) server), all router advertisement ˝caches in all NDS servers on the LL **MUST** be flushed to remove the entries for any routers on the disconnected branch of the tree. Notification of failures is carried through the LL NDS subtree ˝via a new protocols message called a cache flush message. If any NDS server in an LL looses the VC to the NDS server above it, then it **MUST** send a cache flush message via its PtM connection to NDS servers below it. This will cause all NDS(N) servers below the failed link to flush their router advertisement caches. If an ˝NDS

server looses the VC to the NDS server below it (or a router ˝below it), it **MUST** send a cache flush message up towards the NDS(LL) server. This will cause all nodes outside the scope of the ˝failed link to flush their router advertisement caches.

If the NDS(LL) server fails then each NDS server below it **MUST** send a cache flush message to all their NDS(N) servers. When the NDS(LL) server is restored each NDS server **MUST** "broadcast" a cache flush message to their NDS(N) servers after they have fully connected and registered with the NDS(LL) server. This will insure that the NDS(LL) server will be sent updated router advertisements. When the NDS(LL) server is restored and receives a connection and registration from each NDS server below, it **MUST** send any router advertisements already received to the newly attached NDS server via the server's PtP connection. This will update each subtree's router caches as they reconnect. Any new router advertisements received by the NDS(LL) server are always "broadcast" to all currently connected NDS(LN+1) servers.

4.2. Router Solicitations

Router solicitations are sent by nodes to discover information about routers or to obtain a list of prefixes for stateless address configuration and for on-link determination. When an ˝node registers with the NDS(N) server it is sent a copy of the cached router advertisement messages which the NDS(N) server has received. This will bring the node up to date immediately. If the node sends a router solicitation to the all-routers multicast address this **MUST** be intercepted by the NDS(N) server. The NDS(N) server **MUST** then send all router advertisement messages which it currently has cached in reply. Thus, router solicitation will never be send beyond the NDS(N) servers. This will prevent the situation of multiple routers creating multiple VCs back to the solicitor to send the replies (which would not contain the additional external prefix information). If the NDS(N) ˝server has not already sent a single router solicitation message to the NDS(LL) server, it **MUST** forward the first one it receives from an node to trigger a refresh of its router discovery cache.

4.3. Neighbor Solicitation

Neighbor solicitation are sent by nodes to discover the ˝link-layer address of other nodes. Neighbor solicitation is normally done the first time data packets are to be exchanged between nodes. The information provided by the neighbor solicitation mechanism enables each node to unicast data to the other node. In an ATM environment, the result of successful neighbor solicitation ˝should be the establishment of a PtP VC between the soliciting and solicited nodes over which data can be exchanged. By default, this VC will carry only best effort traffic. On ATM networks, nodes **SHOULD** establish multiple VCs with traffic parameters and QoS setting to carry differing classes of traffic. The initial ˝VC setup between nodes **SHOULD** be used to carry all traffic with no QoS requirements.

Neighbor solicitation is used by nodes with no modifications for ATM (other than the format of the link-layer address). Any node on a LL can reach any other node on the LL using link-local

addresses. Any node in a site can reach any other node in the site using site-local addresses. Any node on the tree can reach any other node on the tree using global addresses. Because all nodes on the tree appear to be both physically and logically "on-link", any two nodes can communicate using IPv6 addresses of the appropriate scope. All communications with nodes that are not "on-link" (i.e., not on the same NDS tree) must take place through a router. Routers to external networks **MUST** exist within the same LL as the node sending packets to the external network since ˝link-local addresses are used to establish connectivity to the ˝router.

When a node needs to communicate with another node it will create a neighbor solicitation message and send it via the PtP VC to its attached NDS(N) server. The neighbor solicitation message will be forwarded up the tree to the NDS(LL) server. The NDS(LL) server **MUST** examine the target address of each neighbor solicitation messages which it receives to determine how the solicitation message is to be routed. The NDS(LL) server examines the target address of each solicitation message and determines if the scope of the target address is link-local. If the target address scope is link-local, the NDS(LL) server **MUST** relay the neighbor solicitation message to all NDS servers in it LL via its PtM connection. This "broadcasts" the solicitation down the tree until it reaches all nodes on the tree. The node who's address matches the target address reply to the solicitation as described in the next section.

If the scope of the target address in a neighbor solicitation message is site-local or global, the NDS(LL) server **MUST** compare the prefix of the target address with its list of prefixes which are on its LL. If a match is found then the NDS(LL) server will relay the message down its LL subtree as described above. If no match is found then the NDS(LL) server **MUST** send the neighbor solicitation message to the NDS server above it via its PtP VC to that server.

As each neighbor solicitation message travels up the tree towards the root, each NDS server **MUST** compare the prefix of the target address in the message with its list of prefixes in it's prefix cache. If a match is found, the NDS server **MUST** relay the neighbor discovery message down the tree on the VC associated ˝with the matched prefix. If no match is found then the neighbor discovery message **MUST** be sent up to the next level of the tree towards the root.

As each neighbor solicitation message moves down the tree, each NDS server at the LL level an above **MUST** compare the prefix of the target address in the message with the list of prefixes in its prefix cache. If a match is found, then the NDS server **MUST** relay the neighbor solicitation message out the PtP VC associated with the matched prefix. If no match is found then the NDS server **MUST** transmit the message on the PtM VC connecting to all servers below.

If a neighbor solicitation message reaches the root of the tree the NDS(Root) server **MUST** discard the message without resending it. In cases where an NDS server has just come up and has not ˝yet filled it's prefix cache, the standard retry mechanism for neighbor solicitation messages will cause the soliciting node to retry the solicitation at regular intervals.

4.4. Neighbor Advertisements

There are two types of neighbor advertisements: solicited and unsolicited. The handling of these two types of advertisements ˝by the NDS tree is very different.

When a node receives a neighbor solicitation which contains its address in the target address field, it must send a reply to the soliciting node so that each node will be aware of the other's link-layer address. However, unicasting a message through the tree structure is very difficult, and probably unnecessary. When a node receives a neighbor solicitation it can assume that its being solicited because the soliciting node is ready to communicate with it. Thus, establishing a connection to the soliciting node is an optimization that will save traffic and ˝work on the tree. Thus, when any node **MUST** respond to a neighbor solicitation it will do so by placing an ATM call to the soliciting node and sending the neighbor advertisement via the newly established VC. This VC will be of the default type (best effort), with the default MTU (9188) unless the two nodes negotiate a different MTU using UNI signaling when the call is placed.

Unsolicited neighbor advertisements are sent by nodes when they change their link-layer addresses. Unsolicited neighbor advertisements are sent to the all-nodes multicast address. ˝These messages **MUST** be sent via the VC which connects the node to its NDS(N). When a neighbor advertisement message arrives at the NDS(LL) server, the server **MUST** send the advertisement back down the LL via its PtM connection to NDS servers below. This will "broadcast" the unsolicited advertisement to all nodes on the sender's LL. Additionally, the advertisement **MUST** be sent out on every PtP VC which connects the sender to other nodes. This must be done to assure that every node the sender is in contact with receives the sender's updated link-layer so that any cached link-layer data can be updated.

4.5. Anycast Addresses

The current IPv6 address architecture draft [IPV6-ADDR] restricts the use of anycast addresses to routers. This is done because ˝the use and routing of anycast messages is not yet well defined or understood for legacy LANs. However, the ATM environment and NDS tree offer a way to properly handle anycast addresses. Thus, ˝this framework includes a specification for the handling of anycast addresses within the NDS tree

Anycast addressing is handled in a hierarchical fashion, and follows the layout of the tree. Anycast addresses will have a "scope" within the tree, just as any other address. However, anycast addresses scope can be as small as a neighborhood. ˝When a node needs to connect to an anycast address it will reach the ˝node "closest" to it, based on the tree hierarchy.

When a node registers its anycast address on a node, the node **MUST** be configured to know that the address is an anycast address. This information **MUST** be passed to the IPv6 ATM interface so that the ATM code can take special actions. When an anycast address ˝is assigned to an IPv6 ATM interface, a special anycast ˝advertisement message **MUST** be sent out to the NDS tree. This message will

follow the format of all other new inter-NDS messages. This is a required extension for ATM only to enable anycast address ˝handling to work efficiently within the NDS tree. All anycast advertisements travel up the tree as far as permitted by the IPv6 scope of the anycast address type (anycast addresses use unicast address formats). As they travel up the tree ALL NDSs **MUST** record the anycast address along with the VC of the server (or node) ˝from which the anycast advertisement arrived. When the root NDS ˝server receives the anycast advertisement it **MUST** record the anycast address, and **MUST NOT** resend the neighbor advertisement back down the tree.

When a node needs to connect to an anycast address it **MUST** send a neighbor solicitation with the anycast address as the target address. This message is relayed up towards the root. Because IPv6 anycast addresses are assigned from the normal unicast address space the NDS servers will not be able to distinguish a solicitation for an anycast address from one for a unicast address. Thus, ALL neighbor solicitation messages **MUST** be processed using the following procedure. Since some environments will not need to have (or want) anycast capabilities. To allow anycast capabilities to be disabled, every NDS server **MUST** have a configuration variable that can be set to disable anycast functions. Anycast capabilities **MAY** be disabled on a per-LL or per-subtree basis by disabling NDS server anycast processing on the appropriate set of NDS servers.

When a neighbor solicitation arrives at each NDS server, each server **MUST** examine the target address of the solicitation and compare the solicited address with the list of anycast advertisements it has received. If the solicitation's target address matches an anycast address in the NDS server's list, then the NDS server **MUST** relay the neighbor solicitation message to the VC from which the anycast advertisement was received. If the target address does not match an anycast address in the NDS server's table, the server **MUST** then perform the normal neighbor solicitation message processing as described earlier. Thus, the neighbor solicitation is sent up through the NDS tree normally, only going up the tree hierarchy as far as is necessary.

Nodes which advertise anycast addresses **MUST** resend their advertisements on a periodic basis.

If an NDS fails and recovers, it will loose any cached anycast data. In this case the server **MUST** forward all solicitations as it normally would during the server recovery process.

It may be possible that two nodes will advertise an anycast address within the same scope of the NDS tree. That is, some NDS server will receive two or more advertisements for the same anycast address, but from different VCs. When this occurs, it means that there is more than one "nearest" node advertising the anycast address within the subtree below the server. In such cases, the server **SHOULD** direct neighbor solicitations to alternate branches of the subtree so as to distribute the connections to the nodes which registered the anycast address. The method for doing this is up to specific implementations, but the minimum of distributing solicitations in a round-robin ˝fashion **MUST** be implemented.

As the result of NDS server failures or other conditions, it may be possible for a neighbor solicitation to reach more than one

node that has advertised an anycast address. In this case, the soliciting node will receive more than one VC and neighbor advertisement from the solicited nodes. It is then up to the soliciting node to select one of the responding nodes for communications, either with or without the co-operation of the responding nodes. The choice of a responding node depends on the specific protocol or service which is connecting, and it is ˝beyond the scope of this document to specify how responding nodes are to be selected.

When a node that has advertised an anycast address fails, the NDS(N) server to which it has connected will detect the failure when it receives the VC RELEASE notification from the ATM ˝network. When an NDS(N) server receives a RELEASE for a VC for which it ˝has recorded anycast advertisement data, it **MUST** send an anycast flush message up the NDS tree towards the root. This message **MUST** contain the anycast address being flushed. All nodes that ˝receive the anycast flush message **MUST** remove the entry in their anycast table for the address being flushed associated with the VC on which the flush is received.

If a node de-configures an anycast address on its IPv6 ATM interface, it **MUST** send an anycast flush message to the NDS tree.

Using this method, "closest" means a node which can be reached by ascending the minimum number of levels up the tree. For example, if one system in every neighborhood advertises the same anycast address anycast solicitations to that address would never leave the neighborhood since the NDS(LL) server would know how to ˝direct the anycast neighbor solicitation. If one node in each LL advertised the same anycast address all other nodes in the same ˝LL would reach them and their solicitations would never leave the neighborhood (under normal conditions, except when the NDS(LL) server had just recovered).

4.6. Neighbor Unreachability Detection

Neighbor Unreachability detection will work with no changes in behavior other than nodes will get immediate notification when communications with other nodes fail because of a network failure (i.e., a RELEASE on the VC(s) will be received). Nodes can then attempt to re-call the other node using cached link-layer addresses, or the can re-solicit the remote nodes. In the case ˝of re-calling, the other node should be considered unreachable after some number of retries. In the re-solicitation case, the other node is considered unreachable as specified in the Neighbor Discovery spec.

5. Address Configuration

There are three types of address configuration that may be used ˝in IPv6: stateless, stateful, and manual. These methods of address configuration are used to assign IPv6 addresses with scope beyond the local link to IPv6 end-system nodes. If nodes do not require connectivity beyond the local link then only link-local address are needed. In stateless address configuration, nodes receive a list of prefixes which may be used for address configuration in router advertisement messages. Nodes then create address by combining the prefixes with the node's host-token. In stateful

address configuration nodes are assigned addresses by a central address administration authority via DHCPv6. In manual configuration, nodes are configured with IPv6 addresses in much the same way as IPv4 networks are now. IPv6 over ATM must ˝support all these address configuration methods.

IPv6 nodes can use any combination of address configuration methods.

5.1. Link-Local Addresses

A link-local address is an address which can be used only to ˝reach nodes on the same local link (or LL for ATM). Link-local addresses are never routed and are unique only within the same local link. All nodes have link local addresses. Nodes can create link-local addresses independent of any other network entity. Link-local addresses are completely autonomous.

All nodes within an LL can generate link-local addresses and use them in the normal way. Link advertisements and solicitations ˝for link-local addresses are never relayed outside the LL by the NDS(LL) server. Nodes on one LL cannot reach any node on another LL using link local addresses. These addresses must only be unique within an LL.

A link local address is formed by appending the node's host-token to the link-local address prefix [IPV6-PVCs]. For ATM, a host-token format for both ATM NSAP and E.164 addresses has already been proposed in IP-IPV6ND. The use of this host-token format ˝is recommended.

5.2. Stateless Address Configuration

Stateless address configuration requires that there be at least one router on the local-link which has been configured with a ˝list of prefixes which may be used by local nodes to create IPv6 addresses. Generally, these prefixes are "loaned" for a specific period of time, after which nodes must stop using the addresses created from the prefixes unless the loan time is extended. In IPv6, the prefix list is carried in router advertisement ˝messages. If multiple routers are present on a local-link then the must advertise consistent information, otherwise nodes may not be able to configure addresses and create connections off the ˝local-link.

Router advertisement messages contain two types of prefixes. One type is marked as available for address configuration, the other is marked as not available for address configuration but may be used in determining if an address is "on-link" (i.e., the address is reachable without going through a router).

As described in the section on router advertisements, information advertisements send by local routers (routers on the LL) will always be preserved and sent to all nodes by the NDS(LL) server. The information in the local advertisements could be augmented with information the NDS(LL) server receives from outside the LL. However, only information from internally generated router advertisements will be used in stateless address configuration. Information from external router advertisement information will ˝be used only in on-link determinations. Thus, stateless address

configuration on an LL will continue to work just as it does on legacy LANs. The only change is in defining "on-link" as being on the same ATM network rather than on the same LL. However, this change is transparent to nodes and will require no special processing by nodes.

5.3. Stateful Address Configuration

Stateful address configuration uses an address management authority to assign IPv6 addresses to nodes using a registration protocol [IPV6-DHCP]. This requires that there be a registration agent on the local-link that either assigns addresses directly, ˝or is capable of contacting the entity which can assign addresses.

To use stateful address configuration, there must be at least one DHCPv6 server or DHCPv6 relay agent on the Logical Link. To use stateful address configuration, a node must first contact the local DHCPv6 agent on the LL. To do this, nodes transmit a message to the DHCP multicast address. Messages broadcast to the DHCPv6 multicast address **MUST** be carried by the NDS tree hierarchy. Nodes which are functioning as DHCPv6 agents **MUST** accept packets received for the DHCPv6 multicast address. All other nodes **MUST** discard these packets.

5.4. Manual Address Configuration

Manual address configuration requires no special processing. ˝When nodes are brought on-line they are manually configured with their IPv6 addresses. These nodes then use the standard IPv6 duplicate address discovery mechanisms to make sure that they have been assigned a unique unicast address.

5.5. Node Relocation

In cases of stateful or manual address configuration, nodes may ˝be assigned an address whose prefix is not valid for the LL to which the node is connected. This is more likely to happen under UNI 4.0 autoconfiguring environments since a change in ATM network topology could change the LL to which a node automatically connects. In manually configured environments, the only cause of node joining the incorrect LL will be configuration errors.

If a node joins an incorrect LL, this can be dealt with in two ways. First, nothing can be done and the node can be allowed to connect to the LL. This case is analogous to the same case in legacy LAN environments. The node will be able to communicate using link-local addresses, but may not be able to communicate using site-local or global address. The second way of handling this is to automatically detect that a node has registered with the wrong LL, and then have the NDS relocate the node to the correct LL.

When a node joins a LL, it must first register with the NDS(N) server to which it connects, and then it must perform the normal ND processes to configure its addresses on the interface. Part ˝of the address configuration process is to perform duplicate address discovery. At the time duplicate address discovery is performed,

the node has not yet fully configured the address on the ˝interface (the address is tentative). While the address is tentative, the node **MAY** also require relocation to another LL. Each node **MUST** have a configuration variable which enables and disables the relocation request function. If enabled, relocation request is done at the same time duplicate address discovery takes place.

To request a possible relocation, a node **MUST** send a relocation request message containing its tentative address and the link-layer address of the interface. Only one address **MUST** be included in each message. When an NDS(N) server receives a relocation request message it **MUST** forward it to the NDS(LL) server. When the NDS(LL) server receives a relocation request message it **MUST** compare the prefix of the address in the message to the list on local prefixes (i.e., those received in local router ˝advertisement messages). If a match is found then the node requires no relocation and the relocation request is discarded. If no match is found then the node has been configured with an address which does not belong on the local LL. In this case, the NDS(LL) **MUST** forward the relocation request message up towards the root of the NDS tree. The routing of this message through the NDS tree will follow the same process as the routing of neighbor solicitation message. Thus, the message will be directed to the LL on which the node's address would be valid.

When a relocation request message arrives at an NDS(LL) server from the server above, the NDS(LL) server **MUST** compare the prefix of the address in the relocation request message to its list of local prefixes. If no match is found then the NDS(LL) server ˝**MUST** discard the address. If the prefix is determined to be a local prefix the NDS(LL) server **MUST** forward the relocation request message to exactly one NDS server below it (or process the ˝massage its self if there are no lower NDS servers). The NDS(LL) server **SHOULD** distribute the relocation request messages among all NDS server below it to distribute the connection loads on those servers.

As the relocation request travels down the LL subtree from the NDS(LL) server towards an NDS(N) server, all intermediate NDS servers **MUST** send the message to only one NDS server below them. This must be done to distribute the load and assure that the relocation request arrives at only one NDS(N) server on the ˝LL.

When the relocation message arrives at an NDS(N) server, the NDS(N) server **MUST** place a call to the node that sent the relocation request. The relocating node can be called via the link-layer address included in the relocation request message. When the NDS(N) server connects to the relocating node it **MUST** send a relocation reply message consisting of the IPv6 address that the relocating node sent in the relocation request message, along with the link-layer address that the relocating node can ˝use to reach the NDS(N) server to connect and register. The relocating node can use this information to determine to which relocation message the NDS(N) server is replying, and to ˝cache the new NDS(N) server's address in the event the connection to the server is broken. The relocating node **MUST** then initiate the registration process using the PtP VC from the new NDS(N) server as the initial connection to the new LL. The relocating node **MUST** also release all VCs to the original LL. The relocating node **MUST** de-configure any addresses (link-local, stateful and stateless) which it registered when connected to the old LL.

Since nodes can join multiple LLs, nodes **MUST** have virtual interfaces, one for each LL they join. Each virtual interface **MUST** have a unique ATM link-layer address so that each virtual interface can act autonomously. This will make it possible for nodes to join multiple LLs by connecting to a single LL with all virtual interfaces and then permitting those virtual interfaces which must relocate to do so automatically. This will reduce the amount of configuration necessary in both manual and autoconfiguration environments.

Relocation request messages **MUST** be resent three (3) times at one second intervals to make sure at least one reaches the possible target LL. If no response is received after a three tries, the sender can assume it is connected to the correct LL.

5.6. Duplicate Address Detection

All nodes must perform duplicate address detection for all ˝unicast addresses assigned to an interface (or a virtual interface to ˝a LL in the case of ATM). This is done by transmitting special neighbor solicitation messages to determine if another node is also using the same unicast address. A unicast address can not ˝be assigned to an interface until its uniqueness has been verified. A unicast address in the process of duplicate address detection, but not assigned to a node, is referred to as a tentative ˝address.

To perform duplicate address detection, a node **MUST** first join the all-nodes multicast group and then the solicited-node multicast address for the tentative address. When a node joins a LL it becomes a leaf of a single PtM connection which is used by the NDS(N) server to "broadcast" to all nodes to which it is serving. There is not a separate PtM connection for all multicast groups. Instead, nodes **MUST** filter incoming packets from the PtM circuit based on destination address to assure that only packets that should be delivered to that node are passed up to the IPv6 input routine. Thus, joining any of the all-nodes, all-routers, or solicited-node multicast groups only requires application of node packet filtering. Packets to all these multicast groups are treated specially by the NDSs but are generally "broadcast" to all nodes within some given scope.

To perform duplicate address detection, a node must compose a special neighbor solicitation packet with the tentative address being the target address, and the unassigned address as the ˝source address. This packet is sent to the NDS(N) server which will relay it up the tree towards the NDS(LL). If the target address is a link-local address the NDS(LL) server "broadcasts" the solicitation back to every node on the LL. If the address is a site-local or global address the NDS(LL) server relays the solicitation towards the root of the tree if the prefix of the target address of the solicitation is not on the local LL. The solicitation is sent towards the root unless an NDS recognizes ˝the prefix of the target address as one being below it and directs ˝the packet back down the tree. Eventually, the packet arrives at all nodes on one or more LLs.

When a node receives a neighbor solicitation to the ˝solicited-node address it **MUST** determine if the node is a member of that multicast group. If it is then the packet **MUST** be processed. If not, the packet **MUST** be discarded. Otherwise, the solicitation is process normally as specified in the address configuration ˝spec.

6. Multicasting

While the NDS tree does provide some multicast capabilities, the NDS tree is to be used only for multicasting ND and other specified protocols. It is not meant to provide a general multicast capability.

This framework recommends the multicast method proposed in IP-ATMMC to provide general multicast capabilities.

7. VC Characteristics

There are two classes of VCs specified in this framework: the VCs used to interconnects elements of the NDS tree (including nodes), and VCs used for data communications between nodes. These two types of VCs should share as many characteristics as possible, particularly packet framing. Unless otherwise specified, framing should be LLC/SNAP encapsulation as specified in IP-ATMND. Optionally, each party of a call **MAY** negotiate null encapsulation.

7.1. NDS VCs

VCs between elements of the NDS tree, including the VCs between nodes and the NDS(N) servers, should all be set up with the same characteristics. Since these VCs are not used to carry data between nodes, they have no QoS or bandwidth requirements.

The characteristics of the PtP VCs interconnecting elements of ˝the NDS tree are:

- Best effort traffic type in UNI 3.X environments, and ABR in UNI 4.0 environments.
- An MTU of 9188 bytes. The MTU **MAY** be negotiated at call setup time but **MUST NOT** be negotiated to a value of less than 1508 bytes (Ethernet MTU plus 8 bytes of LLC/SNAP encapsulation).
- LLC/SNAP encapsulation as specified in IP-ATMND. Optionally, null encapsulation **MAY** be negotiated at call setup time.

The characteristics of the PtM connections which connect NDS servers to the NDS servers or nodes below them are:

- Best effort traffic type in UNI 3.X environments, and ABR in UNI 4.0 environments.
- An MTU of 9188. This **MUST NOT** be negotiated since all leaves of the PtM connection may not be able to handle an MTU negotiated by the root of the PtM VC and the initial leaf. All entities **MUST** handle an MTU of 9188 on this connection.
- LLC/SNAP encapsulation. Encapsulation **MUST NOT** be negotiated since all leaves of the PtM connection may not be able to handle alternate encapsulation formats

negotiated by the root of the PtM call and the first leaf. All entities **MUST** handle LLC/SNAP encapsulation.

7.2. Data VCs

The VCs used for data communications between nodes can be set up according to QoS requirements of the application driving the connection. Since the purpose of this framework is to provide an environment where nodes on the same ATM network can directly communicate to utilize ATM QoS features, the use of multiple VCs with different traffic parameters is encouraged. However, the default circuits which are established between nodes (for ˝example, the VC established in response to a neighbor solicitation) should have the following characteristics:

- Best effort traffic type in UNI 3.X environments, and ABR in UNI 4.0 environments.
- An MTU of 9188 bytes. The MTU **MAY** be negotiated at call setup time to any value permitted by IPv6. The MTU **MUST NOT** be negotiated to a value less than that supported by IPv6.
- LLC/SNAP encapsulation as specified in IP-ATMND. Optionally, null encapsulation **MAY** be negotiated at call setup time.

Additional VCs can be created as needed with any characteristics as needed as long as both communicating nodes agree on all VC characteristics, and the VC parameters do no violate any IPv6 or ATM specification.

8. Security

The security implications of the NDS tree is still undetermined. This will be part of ongoing work for later revisions of this document.

Acknowledgments

The author wishes to gratefully acknowledge the help and suggestions provided by Markus Jork, Geraldine Harter, Gerd Hoelzing, and Jim Bound.

Authors' Address

Peter A. Schulter
Digital Equipment Corporation
ZK03-3/U14
110 Spit Brook Road
Nashua, NH 03062
Phone: +1 603 881 2920
FAX: +1 603 881 2257
email: schulter@zk3.dec.com

References

[IPV6-SPEC]
S. Deering and R. Hinden, "Internet Protocol Version 6 [IPv6] Specification Internet Draft, June 1995 <draft-ietf-ipngwg-ipv6-spec-02.txt>

[IPV6-ADDR]
R. Hinden, S. Deering, Editors, "IP Version 6 Addressing Architecture" Internet Draft, November 1995 <draft-ietf-ipngwg-ipv6-addr-arch-03.txt>

[IPV6-ADDRCONF]
S. Thomson, T. Narten, "IPv6 Stateless Address Autoconfiguration" Internet Draft, November 1995 <draft-ietf-addrconf-ipv6-auto-05.txt>

[IPV6-ND]
T. Narten, E. Nordmark, and W. A. Simpson, "IPv6 Neighbor Discovery" Internet Draft, September 1995 <draft-ietf-ipngwg-discovery-02.txt>

[IPV6-ETHER]
M. Crawford, "A Method for the transmission of IPv6 Packets over Ethernet Networks", Internet Draft, October 1995 <draft-ietf-ipngwg-ethernet-ntwrks-01.txt>

[IPV6-SA]
R. Atkinson, "Security Architecture for the Internet Protocol" RFC 1825, August 1995

[IPV6-AUTH]
R. Atkinson, "IP Authentication Header" RFC 1826, August 1995

[IPV6-DHCP]
J. Bound, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" Internet Draft, November 1995 <draft-ietf-dhc-dhcpv6-03.txt>

[IP-FRAME]
R. Cole, D. Shur, "IP Over ATM: A Framework Document" Internet Draft, October 1995 <draft-ietf-ipatm-framework-doc-06.txt>

[IP-ATM]
M. Laubach, "Classical IP and ARP over ATM" RFC 1577, January 1993

[IP-ATMU]
M. Laubach, "Classical IP and ARP over ATM Update (Part Deux)" Internet Draft, August 1995 <draft-ietf-ipatm-classic2-00.txt>

[IP-ATMMC]
G. Armitage, "Support for Multicast over UNI 3.0/3.1 based ATM Networks" Internet Draft, October 1995 <draft-ietf-ipatm-ipmc-08.txt>

[IP-IPV6ND]
G. Armitage, "IPv6 and Neighbor Discovery over ATM" Internet Draft, August 1995 <draft-ietf-ipatm-ipv6nd-00.txt>

[ATM-UNI30]

ATM Forum, "ATM User Network Interface Specification Version 3.0" ISBN 0-13-225863-3, Prentice Hall, Englewood Cliffs, NJ, 1993

[ATM-UNI31]
ATM Forum, "ATM User Network Interface Specification (UNI) Version 3.1" ISBN 0-13-393828-X, Prentice Hall, Englewood Cliffs, NJ, June 1995

[ATM-UNI40]
ATM Forum, "ATM User-Network Interface (UNI) Signalling Specification Version 4.0", ATM Forum/95-1434R6, December 1995

[ATM-PNNI]
ATM Forum, "PNNI Draft Specification", ATM Forum 94-0471R14

This Internet Draft expires May 25, 1996.