Department of Veteran's Affairs
OEAMS Enterprise Architecture Services (EAS)

A discussion of IPv6 Transition Mechanisms

Department of Veterans Affairs
810 Vermont Avenue NW
Washington, DC 20420

Version 2.0

DOCUMENT CHANGE HISTORY

The table below identifies changes that have been incorporated into this document.

Date	Modified By	Version	Description
03/24/06	A. Bland	1.0	Initial Draft Document Completed
04/07/06	K. Hall	1.0.1	Tier-1 Peer Review (Technical) Completed
04/14/06	A. Bland	2.0	Edit of Draft Completed
04/20/06	W. Yale	2.0	Tier-2 Peer Review
04/20/06	A. Zuech	2.0	Tier-2 Peer Review (Supervisor)
04/21/06	A. Bland	2.0	Released for publication in EA v4.1.1

Table of Contents

1 Introduction 4
1.1 Intended Audience 5
1.2 Conventions Used in This Paper 5
2 Abbreviations 5
3 Terminology 6
3.1 Types of Nodes 6
3.2 Types of IPv6 Addresses 6
3.3 Techniques Used in the Transition 7
4 IPv6 – The Protocol and History 7
4.1 IPv6 Header Format 7
4.2 Expanded Addressing Capabilities 8
4.3 Simplification of Packet Headers 8
4.4 Support for Extensions and Options 9
4.5 Flow Labeling Capability 9
4.6 Header and Extension headers processing 9
4.7 Other improvements 10
5 IPv6 Addressing 10
5.1 Address categories 10
5.2 Address notation 11
5.3 Prefix notation 12
5.4 Aggregate global unicast address 13
5.5 International registry services and current address allocations 14
6 Deployment Mechanisms 15
6.1 Single Stack 15
6.1.1 Dual Stack 15
6.2 Endpoint Address Configuration 16
6.2.1 Address Configuration for "Single Stack" Endpoints 16
6.2.2 Address Configuration for "Dual Stack" Endpoints 16
6.3 Network-Centric Deployment Mechanisms 17
6.3.1 Overview 17
6.4 IPv6 over the Backbone 18
6.5 Selection of Backbone Strategy 18
6.6 Native IPv6 19
7 IPv4 to IPv6 Migration Mechanisms 19
7.1 Tunneling 20
7.1.1 Selection of a Tunneling Strategy 20
7.1.2 Tunneling – the 6in4 versus 6over4 debate 21
7.1.3 Manually Established Tunnels 21
7.1.3.1 Manually Configured Tunneling (MCT) 21
7.1.3.2 IPv6 over GRE 23
7.1.3.3 Tunnel Broker and Tunnel Server 24

7.1.3.4 Teredo (Shipworm) Tunnel 24
7.2. Automatically Established Tunnels 24
7.2.1 ISATAP 24
7.2.2 IPv6 over IPv4 Tunnels 26
7.2.3 RFC 3056 "6to4" Tunnel 27
8 Protocol Translation 28
8.1 Independent Application Layer Gateway (ALG) 29
8.2 RFC 2767 Bump in the Stack (BIS) 29
8.2 RFC 3089 SOCKS-Based IPv6/IPv4 Gateway 29
8.4 RFC 3142 TCP-UDP Relay 29
8.5 Dual-Stack Translation Mechanism (DSTM) 29
8.6 RFC 3089 SOCKS-Based IPv6/IPv4 Gateway 29
8.7 RFC 2766 NAT-PT 30
9 Other IPv6 Transition Considerations 32
9.1 IPv4 Firewalls and ACLs 32
9.2 MTU and Bandwidth 32
9.3 NAT 32
9.4 Multicast 32
10 Conclusion 33
Appendix A IPv6-Related RFCs 34
References 37

1 Introduction

In August of 2005, the Office of Management budget issued Memorandum 05-22, establishing the goal of transitioning all Federal government agency network backbones to the next generation of the Internet Protocol Version 6 (IPv6) by June 2008.[1]

The current version of the Internet Protocol, Version 4 (IPv4), supports four (4) billion IP addresses, which inherently limits the number of devices that can be assigned a globally unique and routable address. IPv6 provides an almost unlimited number of available IP addresses as well as significantly enhanced mobility features. It is, therefore, imperative that IPv6 be embraced in order to support the continued growth of the Internet and the development of new applications that will leverage mobile Internet connectivity.

This paper was developed by the VA Office of Enterprise Architecture Management, in its responsibility to support the VA IPv6 Transition Office and VA's forthcoming transition effort. The purpose of this paper is to provide the reader with an overview of the technologies and mechanisms currently available to assist with the transition from IPv4 to IPv6.

The migration to IPv6 is a complex problem; however, the IETF (Internet Engineering Task Force) has taken this into consideration with the design of IPv6 and has provided mechanisms to assist with the transition. The transition mechanisms presented in this document are described in greater detail within the following RFCs and drafts:[1]

- RFC 1933 - Transition Mechanisms for IPv6 Hosts and Routers
- RFC 2529 - Transmission of IPv6 over IPv4 Domains without Explicit Tunnels
- RFC 2765 - Stateless IP/ICMP Translation Algorithm (SIIT)
- RFC 2766 - Network Address Translation - Protocol Translation (NAT-PT)
- Draft-ietf-ngtrans-dstm-03
- Draft-ietf-ngtrans-socks-gateway-05

The key to a successful IPv6 transition is compatibility with the large installed base of IPv4 hosts and routers. Maintaining compatibility with IPv4 while deploying IPv6 will streamline the task of transitioning the Internet to IPv6. This specification defines a set of mechanisms that IPv6 hosts and routers may implement in order to be compatible with IPv4 hosts and routers.

The mechanisms in this document are designed to be employed by IPv6 hosts and routers that need to interoperate with IPv4 hosts and utilize IPv4 routing infrastructures. We expect that most nodes in the Internet will require this compatibility for the **indefinite future**.

However, IPv6 may be used in some environments where interoperability with IPv4 is not required. IPv6 nodes that are designed to be used in such environments *need not use or even implement these mechanisms*. Choosing between mechanisms is part of the Technical Architecture and Design process.

[1] The section of this paper entitled "*Appendix A IPv6-Related RFCs*" includes additional references the reader may find useful.

1.1 Intended Audience

The intended audience of this document is network professionals that are involved with IP communications and who will design, plan and deploy IP networks and services. This paper will also be of interest to network administrators, system administrators, and IT Managers. This paper is **not** intended for an executive-level audience.

1.2 Conventions Used in This Paper

This paper contains both *footnotes* and *endnotes*. Footnotes are denoted through the use of ***bold, italics, and underline***. The endnotes are found at the rear of this document under the references section 8, and form the bibliography of this document.

2 Abbreviations

ALG:	Application Layer Gateway
API:	Application Programming Interface
APNIC:	Asia Pacific Network Information Centre
ARIN:	American Registry for Internet Numbers
ARPANET:	Advanced Research Projects Agency Network
BIA:	Bump-in-the-API
BIS:	Bump-in-the-Stack
CIDR:	Classless Inter-Domain Routing
CPU:	Central Processing Unit
DARPA:	Defense Advanced Research Projects Agency
DHCP:	Dynamic Host Configuration Protocol
DNS-ALG:	Domain Name System-Application Layer Gateway
ESP:	Encapsulating Security Payload
EUI:	Extended Unique Identifier
FTP:	File Transfer Protocol
GPRS:	General Packet Radio Service
GSM:	Global System for Mobile communications
HTTP:	Hypertext Transfer Protocol
IANA:	The Internet Assigned Number Authority
ICMP:	Internet Control Message Protocol
ICS:	Internet Connection Sharing
IEEE:	Institute for Electrical and Electronics Engineers.
IETF:	Internet Engineering Task Force
IIJ:	Internet Initiative Japan
IMAP:	Internet Message Access Protocol
INWG:	Internetworking Working Group
IPng:	Next Generation IP
IPTO:	The Information Processing Techniques Office
IPv4:	Internet Protocol version 4
IPv6:	Internet Protocol version 6
IR:	Infrared
ISATAP:	Intra-Site Automatic Tunnel Addressing Protocol
ISP:	Internet Service Provider
MAC:	Media Access Control
MCT:	Manually Configured Tunnel

MPLS:	Multi Protocol Label Switching
NAPT-PT:	Network Address Port Translator – Protocol Translator
NAT:	Network Address Translator
NAT-PT:	Network Address Translator – Protocol Translator
NCP:	Networking Control Protocol
NLA:	The Next-Level Aggregation identifier
QOS:	Quality of Service
R&D:	Research & Development
RF:	Radio Frequency
RFC:	Request For Comments
RIR:	Regional Internet Registries
RPC:	Remote Procedure Call
SIIT:	Stateless IP/ICMP Translation
SLA:	The site-level aggregation identifier
SOCKS64:	SOCKS-based IPv6/IPv4 Gateway Mechanism
SSL:	Secure Socket Layer
sTLA:	sub TLA
TCP:	Transmission Control Protocol
TLA:	The Top-level Aggregation identifier
TOS:	Type of Service
TOTD:	Trick-Or-Treat Daemon
TRT:	Transport Relay Translator
TTL:	Time to Live
UDP:	User Datagram Protocol
UMTS:	Universal Mobile Communications System
VPN:	Virtual Private Network

3 Terminology

The following terms are used in this document:

3.1 Types of Nodes

IPv4-only node: A host or router that implements only IPv4. An IPv4-only node does not understand IPv6. The installed base of IPv4 hosts and routers existing before the transition begins are IPv4-only nodes.

IPv6/IPv4 node: A host or router that implements both IPv4 and IPv6.

IPv6-only node: A host or router that implements IPv6, and does not implement IPv4. The operation of IPv6-only nodes is not addressed here.

IPv6 node: Any host or router that implements IPv6. IPv6/IPv4 and IPv6-only nodes are both IPv6 nodes.

IPv4 node: Any host or router that implements IPv4. IPv6/IPv4 and IPv4-only nodes are both IPv4 nodes.

3.2 Types of IPv6 Addresses

IPv4-compatible IPv6 address: An IPv6 address, assigned to an IPv6/IPv4 node, which bears the high-order 96-bit prefix 0:0:0:0:0:0, and an IPv4 address in the low-order 32-bits. IPv4-compatible addresses are used by the automatic tunneling mechanism.

IPv6-only address: The remainder of the IPv6 address-space. An IPv6 address that bears a prefix other than 0:0:0:0:0:0.

3.3 Techniques Used in the Transition

IPv6-in-IPv4 tunneling: The technique of encapsulating IPv6 packets within IPv4 so that they can be carried across IPv4 routing infrastructures (6in4).

IPv6-in-IPv4 encapsulation: IPv6-over-IPv4 tunneling (6in4).

Configured tunneling: IPv6-in-IPv4 tunneling where the IPv4 tunnel endpoint address is determined by configuration information on the encapsulating node.

Automatic tunneling: IPv6-in-IPv4 tunneling where the IPv4 tunnel endpoint address is determined from the IPv4 address embedded in the IPv4-compatible destination address of the IPv6 packet.

4 IPv6 – The Protocol and History

Prior to any discussion of IPv4 to IPv6 migration tools, it is first necessary to familiarize the reader with the history and structure of IPv6.

IPv6 was recommended by the IPng Area Directors of the IETF at the Toronto IETF meeting of July 25, 1994 in RFC 1752. The recommendation was approved by the Internet Engineering Steering Group and made a Proposed Standard on November 17, 1994. The core set of IPv6 protocols were made an IETF Draft Standard on August 10, 1998.[2]

The initiative to make a new version of the Internet protocol was mainly caused by the shortage of address space; however, the additional motivation of quality of service (QoS) was a factor as well.

IPv6 is designed to run on high performance networks (e.g. Gigabit Ethernet) and at the same time be efficient on low bandwidth networks, such as wireless. Also, scalability was a design requirement of IPv6 as it was assumed that new technologies will appear in the future and IPv6 is designed to integrate with these new technologies as they become available.

IPv6 includes transition mechanisms, which are designed so that users would be able to deploy IPv6 in a manner that provides interoperability between IPv4 and IPv6.

IPv6 will also add improvements in the areas of routing and network auto-configuration. IPv6 is expected to gradually replace IPv4, with the two coexisting for a number of years during a transition period.[3]

4.1 IPv6 Header Format

The reasons for a next generation of IP are best illustrated by looking at the new header format for IPv6, shown in figure 1.

Explanation of IP Header fields**:**

- The 4-bit *Version* field = 6, for IPv6
- The 8-bit *Traffic Class* field is available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets.

- The 20-bit *Flow Label* field may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service.
- The 16-bit *Payload Length* field is a 16-bit unsigned integer, which indicates the length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets. (The length of extensions is included).
- The 8-bit *Next Header* field is an 8-bit selector that identifies the type of header immediately following the IPv6 header. (The values are the same as those in the IPv4 Protocol field).
- The 8-bit *Hop Limit* field is an 8-bit unsigned integer that decrements by 1 by each node that forwards the packet. The packet is discarded if Hop Limit decrements to zero.
- The 128-bit *Source Address* field contains the address of the packet's originator.
- The 128-bit *Destination Address* field contains the address of the packet's recipient.[4]

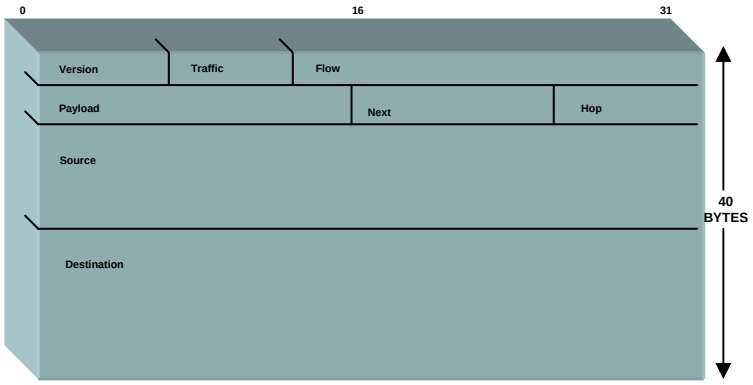


Figure 1 IPv6 Header Format

4.2 Expanded Addressing Capabilities

IPv6 increases the IP address size from 32 bits to 128 bits. This results in IPv6 being able to support more levels of addressing hierarchy. IPv6 allows a much greater number of machines to connect to the Internet and the auto-configuration of IP addresses is simplified. The scalability of multicast routing is improved, by adding a scope field to multicast addresses. And a new type of address called an "anycast address" is defined, which is used to send a packet to any one in a group of nodes.[5]

4.3 Simplification of Packet Headers

The number of fields in the header has been reduced from 12 fields in IPv4 down to 8 in IPv6. Additionally, the behavior of the IPv4 "options" field was changed to ensure that IPv6 headers are a fixed length, allowing for faster packet processing.

The minimum IPv4 header size is 20 bytes, but the "options" field could increase the header size to a maximum of 60 bytes. In practice, however, the IPv4 header rarely exceeds 20 bytes. In contrast, the IPv6 header is exactly 40 bytes long. For traffic engineering purposes, this has little effect upon packets less than the Maximum Transmission Unit (MTU) of the traffic path, but could affect the performance of applications currently tuned for a 1500 byte MTU. [2]

[2] For additional details, see http://www.tcpipguide.com/free/t_IPv6DatagramSizeMaximumTransmissionUnitMTUFragment.htm.

4.4 Support for Extensions and Options

IPv4 options are integrated into the basic IPv4 header. In IPv6, the options are handled as **extension headers**. Extension headers are optional, fixed-length fields inserted between the IPv6 header and the payload only when necessary. This allows a flexible and extensible means to define optional packet properties at a future date.

4.5 Flow Labeling Capability

IPv6 adds labeling on packets which enable packets of a certain type to get special handling on sender's request. This is for packets with non-default quality of service, such as real-time services.

4.6 Header and Extension headers processing

The IPv6 header has a total size of 40 bytes, which is twice the size of the IPv4 default header. But on a closer look the IPv6 header is simplified compared to the IPv4 header as the address-space alone consumes 32 bytes in IPv6. This leaves only 8 bytes with other header information.[6] This means that only 8 bytes will be processed at each router, which means process time decreases. In comparison to IPv4, IPv6 does not extend the header, but makes use of e*xtension* headers. This is a key improvement as these are a part of the *payload* instead of the header itself and therefore does not slow the processing time. The way that IPv6 has designed extension headers is that theoretically, there are no limits as to how many extension headers can be allowed together within a packet. This makes it easy to add new extension headers for new services in the future.

The current IPv6 specification defines six Extension headers

1. Hop-by-hop Options header
2. Routing header
3. Fragment header
4. Destination Options header
5. Authentication header
6. ESP (Encapsulating Security Payload) header

There is **not always** an extension header with every header. There may be just one or there may be more than one between the IPv6 header and the Upper-Layer Protocol header, which is always the last header in an IP packet. It all depends on the requirements of the processing of the payload of the packet. Each extension header is identified in the next header field of the preceding header.

Only the destination node (and none of the other nodes between the source and the destination) processes the Extension headers. If the destination is a multicast address, all the nodes that belong in that specific multicast group process the extension headers. The extension headers must be processed in the order they appear within the packet header.

The only exception to the rule about the destination node being the only one to process the extension headers is when the Hop-by-hop Options header is in use. All nodes in the route towards the destination node must process this. The Hop-by-hop header must therefore always immediately follow the IPv6 header.

If more than one Extension header is present in a single packet the following order is recommended:[7]

1. IPv6 Header
2. Hop-by-hop Options header
3. Destination Options header[3]
4. Routing header
5. Fragment header
6. Authentication header
7. ESP header
8. Destination Options header[4]
9. Upper-layer header

At most the extension headers should occur once with the exception of the Destination header, which may occur twice (once before the Routing header and once before the upper-layer header).

In cases when the upper-layer header is another IPv6 header, such as in the case of IPv6 being tunneled over or encapsulated in IPv6, the upper-layer header can be followed by its own extension headers, which are separately subject to the same ordering recommendations.[8]

The IPv6 header is therefore expected to reduce the cost of header processing between the source and destination node.

4.7 Other improvements

IPv6 supports automatic configuration, which means a host can be made Internet-access-ready without laborious manual entry of address information. IPv6 allows better plug and play and thereby reduces the amount of time it takes to get a large number of machines online with IPv6.[9]

IPv6 includes IP Security (IPSec) for sender authentication and data encryption by default, whereas it is an optional extension to IPv4.[10]

5 IPv6 Addressing

IP Addresses in Version 6 have both substantial similarities and differences from their IPv4 counterparts. The following sections specifically address how the IPv4 and IPv6 addresses compare.

5.1 Address categories

An IPv6 address can be classified into one of three categories:

1. Unicast.
2. Multicast.
3. Anycast.

IPv6 addresses are assigned to interfaces in much the same manner as in IPv4. Each interface of a node requires at least one unicast address. A single interface can be assigned multiple IPv6

[3] For options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header.
[4] For options to be processed only by the final destination of the packet.

addresses of any type (unicast, multicast and anycast). A node can therefore be identified by any of its interfaces. It is also possible to assign one unicast address to multiple interfaces for load-sharing reasons, if the hardware and drivers support it.

The **unicast** address uniquely identifies an interface of an IPv6 node. An object sent to a unicast address is delivered to the interface identified by that address.

The **multicast** address identifies a group of IPv6 interfaces. A packet sent to a multicast address is processed by all members of the multicast group.

The **anycast** address is assigned to multiple interfaces (usually multiple nodes). A packet sent to an anycast address is delivered to only one of these interfaces, usually the nearest one.

Figure *2* below shows a typical IPv6 address, which consists of three parts – the global routing prefix, the subnet ID and the interface ID.

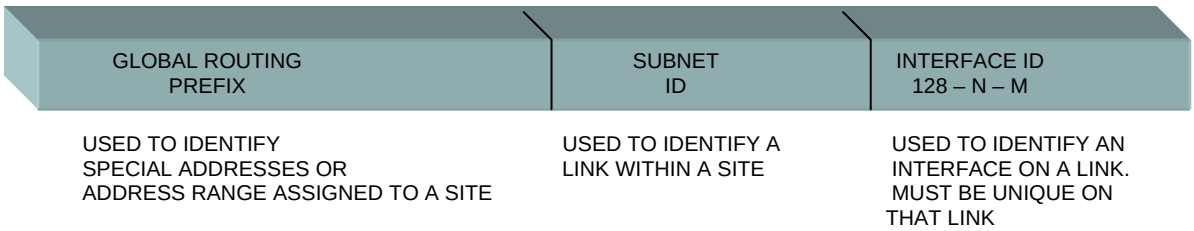


Figure 2 IPv6 general address format

The global routing prefix is used to identify a special address, such as multicast, or an address range assigned to a site. A subnet ID (also referred to as "subnet prefix" or just "subnet") is used to identify a link within a site. An interface ID is used to identify an interface on a link and needs to be unique on that link.

5.2 Address notation

Before any further description of the IPv6 address it is important to give a basic introduction on the IPv6 address notation.

An IPv6 address has 128 bits (16 bytes). The address is divided into eight, 16 bit hexadecimal blocks, separated by colons. For example:

FE80:0000:0000:0000:0202:B3FF:FE1E:8329

There are several ways to write these addresses, many zeros can be avoided, as there will be a lot of these in the address field. The first example shows the above address:

FE80:0:0:0:202:B3FF:FE1E:8329

To make the address even shorter a double colon can replace consecutive zeros, or leading or trailing zeros, within the address. If this rule applies the above address will look like this:

FE80::202:B3FF:FE1E:8329

The double colon can only appear once in an address, as the computer always uses the full 128 bits. Where the double colon is present the computer fills the address with zeros so that the address reaches the full length of 128 bits.

In an environment where both IPv4 and IPv6 nodes are mixed, there is another convenient form of IPv6 address notation. The IPv4 address can be inserted in the end of the IPv6 address in its original form:

IPv4 address example: *128.39.0.2*

IPv6 address for above example: *0:0:0:0:0:0:128.39.0.2* or *::128.39.0.2*

If preferred it also can be written in hexadecimals: *2002:8027:2::*

5.3 Prefix notation

The prefix notation is an important part of understanding the more complex hierarchy of IPv6 addresses than used in IPv4. IPv4 mainly divided the addresses into A-, B- and C-class addresses. The IPv6 prefix structure allows a larger range of network or subnetwork splits in the address.

A *format prefix* (also referred to as global routing prefix) is the high-order bits of an IPv6 address used to identify the subnet or a specific type of address. The notation appends the prefix length, written as a number of bits with a slash:

IPv6 address/prefix length

This is very similar to the CIDR notation for IPv4 or for subnetted IPv4 addresses. The prefix length indicates how many of the left most bits are a part of the prefix. The prefix is used by routers to identify which subnet the address belongs to. The packet is then forwarded using the value of the prefix only.

Example prefix notation:

2E78:DA53:12::/40

Figure 3 illustrates the hexadecimal prefix notation with its corresponding binary conversion.

HEX NOTATION	BINARY NOTATION	NUMBER OF BITS
2E78	0010111001111000	16 BITS
DA53	1101101001010011	16 BITS
12	00010010	40 BITS TOTAL

Figure 3 Example prefix notation

Note that in the address notation the address would have to be written 2E78:DA53:1200::, but as it is only the 40 left most bits that are of interest, the double colon (::) will replace the remaining bits with zeros until the address reaches 128 bits.

The format prefixes that are used to identify special addresses, such as link-local addresses or multicast addresses are reserved prefixes, as shown in figure 4 below.

ALLOCATION	PREFIX	PREFIX	FRACTION OF ADDRESS
RESERVED	**0000 0000**	**::0/8**	**1/256**
RESERVED FOR NSAP	**0000 001**	**::2/7**	**1/128**
ALLOCATION	0000 010	::4/7	1/128
RESERVED FOR IPX ALLOCATION	001	::20/3	1/8
AGGREGATABLE GLOBAL UNICAST ADDRESSES	1111 1110 10	FE80::/10	1/1024
LINK-LOCAL UNICAST ADDRESSES	1111 1110 11	FEC0::/10	1/1024
SITE-LOCAL UNICAST ADDRESSES	1111 1111	FF00::/8	1/256

Figure 4 List of assigned prefixes

Some of the special addresses are assigned out of the reserved address space with the binary prefix 0000 0000. These include the *unspecified address*, the *loopback address* and the IPv6 addresses with the embedded IPv4 addresses.

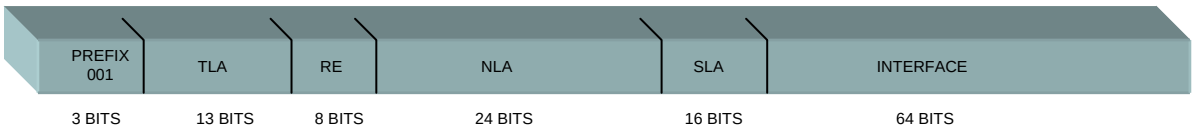
Unicast addresses can be distinguished from multicast addresses by their prefix. Globally unique unicast addresses have a high-order byte starting at 001. An IPv6 address with a high-order byte of 1111 1111 (FF in hex) is always a multicast address.

Anycast addresses are taken from the unicast address space, so it is not possible to identify these only by looking at the prefix. If a unicast address is assigned to multiple interfaces, which makes it an anycast address, the interfaces need to be configured to let them all know that the address is an anycast address.

Addresses in the prefix range 001 to 111 should use the 64-bit interface identifier that follows the EUI-64 (Extended Unique Identifier) format. The EUI-64 is a unique identifier defined by the IEEE.[11]

5.4 Aggregate global unicast address

As mentioned earlier the unicast addresses are identified by the prefix 001. The initial address specification defined *provider-based addresses;* the name has been changed to *aggregate global unicast address*. The name change reflects the addition of an ISP-independent means of aggregation called *exchange-based aggregation.* The prefix is followed by five components, as shown in figure 5.



PREFIX 001, PREFIX FOR AGGREGATABEL GLOBAL UNICAST ADDRESS
TLA ID TOP-LEVEL AGGREGATION IDENTIFIER
RES RESERVED FOR FUTURE USE
NLA ID NEXT-LEVEL AGGREGATION IDENTIFIER
SLA ID SITE-LEVEL AGGREGATION IDENTIFIER
INTERFACE ID INTERFACE IDENTIFIER

Figure 5 Format of the aggregate global unicast address

The format prefix 001 is assigned to the aggregate global unicast address range. The Top-level Aggregation identifier (TLA) contains the highest level of routing information about the address. Its size of 13 bits limits the number of top-level routes to 8192. In the earlier specification, the TLA was the provider-based identifier. It was later assigned to the RIR; American Registry for Internet Numbers (ARIN) in North America, Réseau IP Européens - Network Coordination Centre in Europe (RIPE - NCC), and Asia Pacific Network Information Centre (APNIC) by IANA (The Internet Assigned Number Authority), see figure 6 below. With this change in the specification, the commercial aspect of the TLA has been removed and the focus is now on routing optimization; the TLA does not have to be an ISP. At the core of the Internet, the routing tables need only one route entry per TLA, which means that the 13-bit TLA is large enough.

ISPs and exchange points use the next-level aggregation identifier (NLA). These network access providers are usually public, and they will further structure the address space assigned by the TLA with route topology optimization as a priority.

The site-level aggregation identifier (SLA) is the address space assigned to organizations, used for internal network structure. It can be subnetted further within the organization.

The last part of the IPv6 address is used for the 64 bit interface identifier, as discussed earlier in section 5.3 (EUI-64).

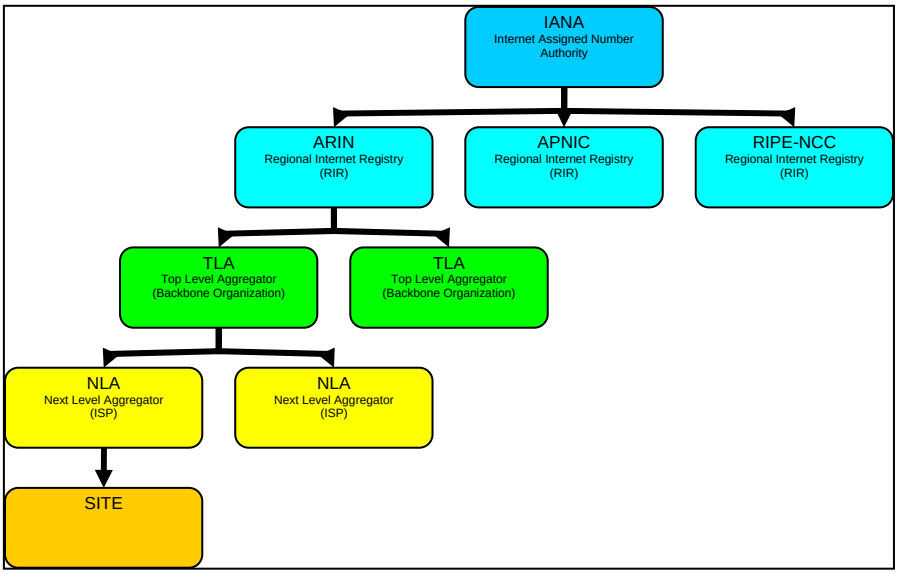


Figure 6 IPv6 address distribution hierarchy

From July 1999 the RIRs could allocate **subTLA**s (sTLA) to ISPs. IANA will assign small blocks e.g. a few hundred of sTLA IDs to RIRs. The registries will assign the sTLA ID's to organizations meeting the requirements of certain specifications. When the registries have assigned all of their sTLA ID's they can request that the IANA give them another block. The blocks do not have to be contiguous. The IANA may also assign sTLA ID's to organizations directly. This includes the temporary TLA assignment for testing and experimental usage for activities such as the 6bone or new approaches like exchanges.[12]

5.5 International registry services and current address allocations

Several TLA allocations have been made, as listed in figure 7.

PREFIX	ALLOCATION	RFC
2001::/16	SUB-TLA ASSIGNMENTS ARIN 2001:0400::/29 APNIC 2001:0200/29	RFC2450
2002::/16	6to4	RFC3056
3FFE::/16	6BONE TESTING	RFC2471

Figure 7 Current TLA allocations

ISPs in Europe will therefore have access to information about the regional registry of IPv6 addresses through RIPE-NCC web pages. For end users, the IPv6 address allocation is managed by their ISP. ISPs in other parts of the world find their information at their regional registries.

Note: *one of the most significant differences between IPv4 and IPv6 is that router interface IP address assignment is received from adjacent routers and maintained within neighbor adjacency tables. Network operation procedures should be updated to reflect this change.*

6 Deployment Mechanisms

Each network segment must be assessed to determine whether any legacy IPv4 traffic is necessary to maintain proper network operations. This may be the case if either legacy applications or hardware cannot be upgraded and must be replaced. Determining whether all endpoints within a particular network segment run "single stacks" or "dual stacks" is a threshold question that affects which migration tools will be available to assist with the IPv6 conversion strategy.

6.1 Single Stack

Fundamentally, endpoint nodes and hosts run a single network protocol (i.e. "single stack"). This is the easiest stack to configure and maintain from an endpoint perspective, assuming that the network design can support this configuration. If the endpoint runs a "single stack" then it is only able to send and receive traffic.

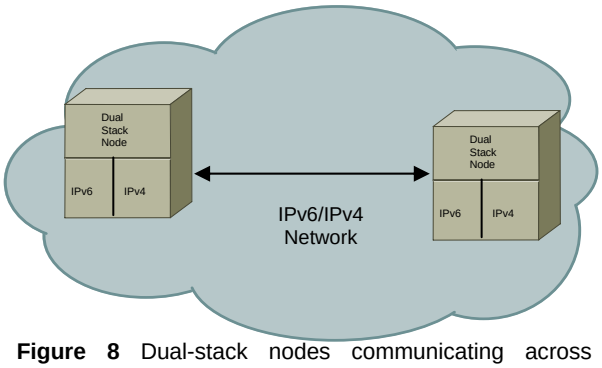


Figure 8 Dual-stack nodes communicating across an IPv6/IPv4 Network

Conversion directly from an IPv4 single stack to an IPv6 single stack can be risky if there are unknown applications on the endpoints. Note that regardless of the endpoint configurations, the routers should be configured for dual stack operation for the duration of the conversion process.

6.1.1 Dual Stack

Dual Stack, also referred to as Dual-Layer in RFC 1930, allows for the parallel usage of IPv4 and IPv6 in a single machine or network.

The most straightforward way for IPv6 nodes to remain compatible with IPv4-only nodes is by providing a complete IPv4 implementation. IPv6 nodes that provide a complete IPv4 implementation in addition to their IPv6 implementation are called "IPv6/IPv4 nodes," as shown in figure 8. IPv6/IPv4 nodes have the ability to send and receive both IPv4 and IPv6 packets. They can directly interoperate with IPv4 nodes using IPv4 packets, and directly interoperate with IPv6 nodes using IPv6 packets.

The dual IP layer technique may or may not be used in conjunction with the IPv6-over-IPv4 tunneling techniques.

An IPv6/IPv4 node that supports tunneling may support only configured tunneling, or both configured and automatic tunneling. Therefore, three configurations are possible:

1. IPv6/IPv4 node that does not perform tunneling.
2. IPv6/IPv4 node that performs configured tunneling only.
3. IPv6/IPv4 node that performs configured tunneling and automatic tunneling.

6.2 Endpoint Address Configuration

IP address assignments for IPv6 endpoints are substantially similar to how IPv4 endpoints receive their addresses. The major differences that exist are described within the following sections.

Because they support both protocols, IPv6/IPv4 nodes may be configured with both IPv4 and IPv6 addresses. Although the two addresses may be related to each other, this is not required. IPv6/IPv4 nodes may be configured with IPv6 and IPv4 addresses that are unrelated to each other.

6.2.1 Address Configuration for "Single Stack" Endpoints

IPv4 and IPv6 nodes may either use the stateless address configuration mechanism[13] (typically provided via UDP by a router) or stateful DHCP [14] (typically provided via TCP by a server) to acquire an address. Manual address assignment is also possible, but is highly undesirable in IPv6 since it prevents a new feature, stateless autoconfiguration, from operating. Note that stateless autoconfiguration applies only to endpoints and routers utilize neighbor adjacency advertisements to receive their addressing.

6.2.2 Address Configuration for "Dual Stack" Endpoints

Because they support both protocols, IPv6/IPv4 "dual stack" nodes may be configured with both IPv4 and IPv6 addresses. Although the two addresses may be related to each other, this is not required. IPv6/IPv4 nodes may be configured with IPv6 and IPv4 addresses that are unrelated to each other.

Nodes that perform automatic tunneling are configured with IPv4-compatible IPv6 addresses. These may be viewed as single addresses that can serve both as IPv6 and IPv4 addresses. The entire 128-bit IPv4-compatible IPv6 address is used as the node's IPv6 address, while the IPv4 address embedded in low-order 32-bits serves as the node's IPv4 address.

These mechanisms may provide either IPv4-compatible or IPv6-only IPv6 addresses. IPv6/IPv4 nodes may use IPv4 mechanisms to acquire their IPv4 addresses. IPv6/IPv4 nodes that perform automatic tunneling may also acquire their IPv4-compatible IPv6 addresses from another source: IPv4 address configuration protocols. A node may use any IPv4 address configuration mechanism to acquire its IPv4 address, then "map" that address into an IPv4-compatible IPv6 address by pre-pending it with the 96-bit prefix 0:0:0:0:0:0. This mode of configuration allows IPv6/IPv4 nodes to "leverage" the installed base of IPv4 address configuration servers. It can be particularly useful in environments where IPv6 routers and address configuration servers have not yet been deployed. The specific algorithm for acquiring an IPv4-compatible address using IPv4-based address configuration protocols is as follows:

1) The IPv6/IPv4 node uses standard IPv4 mechanisms or protocols to acquire its own IPv4 address. These include:

- The Dynamic Host Configuration Protocol (DHCP)[15]

- The Bootstrap Protocol (BOOTP)[16]
- The Reverse Address Resolution Protocol (RARP)[17]
- Manual configuration
- Any other mechanism which accurately yields the node's own IPv4 address

2) The node uses this address as its IPv4 address.

3) The node prepends the 96-bit prefix 0:0:0:0:0:0 to the 32-bit IPv4 address that it acquired in step 1. The result is an IPv4-compatible IPv6 address with the node's own IPv4-address embedded in the low-order 32-bits. The node uses this address as its own IPv6 address.

6.3 Network-Centric Deployment Mechanisms

Although the endpoint options are currently limited, there are several strategies and techniques to consider when deploying IPv6 within the network. Based upon the particular technique selection, certain connectivity capabilities may be limited or require workarounds to ensure network operations remain uninterrupted. The following sections provide an overview of these network-related issues to consider.

6.3.1 Overview

Various IPv6 deployment mechanisms were developed to deal with integration challenges such as traversing IPv4 NATs or scaling to large number of sites. Table 1 summarized the IPv6 deployment techniques currently in use along with their key features.

Table 1 IPv6 Unicast Deployment Techniques

Mechanism		Scope	Where Used	Key Features	Limitations
Native IPv6	Routed	Host/ Inter-site	Access/ Backbone	Simplest method in terms of setup	Support on all network elements in the path
	Bridged (IPv6 RBE)	Host	Access	IPv6 RBE routes the IPv6 traffic out of a bridged ATM encapsulation	
	PPP	Host/ Inter-site	Access		
IPv6 over MPLS	Dedicated links/circuit	Inter-site	Backbone	It is a native method where a physical or virtual link interconnects two (2) sites	
	6PE	Inter-site	Backbone	MPSL infrastructure – IPv4 Label Switch Path setup – IPv6 transport over MPLS	Applicable to MPLS infrastructure only – IPv4 and IPv6 traffic follow the same paths
	IPv6 MPLS	Inter-site	Backbone	MPLS infrastructure – Native IPv6 Label Switch Path setup – IPv6 transport over MPLS	**MPLS only – LDP and RSVP IPv6 implementation not available**
Tunnels	Configured	Host/ Inter-site	Access/ Backbone	Static – supported by most IPv6	

				implementations – **Tunnel endpoints can be secured with IPv4 IPSEC**	
	Tunnel broker and tunnel server	Host	Access	Provides the means to **automatically setup configured tunnels upon request** – A dedicated device or a router performs this function	**Potential security implications**
	Teredo	Host	Access	**Works Through multiple NATs**	Management overhead
	GRE	Inter-site	Access/ Backbone	Static – It supports the transport of layer 2 multicast, which is important for the control plane of some protocols such as IS-IS	
	ISATAP	Host/ Inter-site	Access/ Backbone	Automatic tunnel	Performance – No solution for multicast
	6 to 4	Inter-site	Access Backbone	Automatic tunnel – Reserved address space: 2002::/16	**Return path selection not optimized – Security issue if not secure through IPSEC**

6.4 IPv6 over the Backbone

There are three (3) basic approaches for deploying Ipv6 over a network backbone:

1. **Dual-stack routers and links** – IPv4 and IPv6 share the same link layer and core routers run both IPv4 and IPv6.
2. **IPv6 (and IPv4) users a separate link layer** – Some service provider backbones are simply using a layer 2 technology such as Frame Relay, ATM, or Packer over SONET (PoS) to transport IPv4 traffic between edge routers. To establish IPv6 connectivity, routers attached to the WAN or MAN can use the same layer 2 infrastructure as IPv4, but over separate ATM or Frame Relay PVCs (Permanent Virtual Circuits), or over different optical lambda. Layer 2 links are terminated at the Internet exchange point or on the autonomous system boundaries where IPv6 (or dual-stack) routers handle layer 3 connectivity to the Internet or other networks.
3. **IPv6 on the edges and tunnels to traverse the backbone**

6.5 Selection of Backbone Strategy

Long-term objectives should drive tactical as well as strategic choices. In most scenarios where there is a dominance of IPv4 networks, tactical choices will typically evolve around tunnel deployments, whereas dual-stack backbones maybe seen as the long-term strategy. Tunnels tend to be less expensive – at least initially, to deploy because they deal with a subset of the backbone nodes. On the other hand, they deliver suboptimal paths in most cases while causing packet overhead in the form of an additional header. Tunnels also carry some configuration and

management overhead, which grows as the number of tunnels increase to a point where it becomes more cost-effective to migrate to a dual-stack backbone. When tunnel mechanisms are used to transport IPv6 traffic over the backbone, tunnel endpoints can always be configured between edge-routers pairs (PE) or between customer-router pairs (CE).

IPv6 connectivity across the backbone can be set up with multiple segments managed independently using a variety of IPv6 transport mechanisms As an example, the enterprise could deploy a dual-stack network and interface with a MPLS ISP providing native IPv6 access through an IPv6 Provider Edge Router (6PE). This ISP is then connected to a second ISP providing IPv6 over IPv4 tunnels. An enterprise will have to make its own choices about strategy for transporting IPv6 traffic; it will also have to deal with choices and strategies of other networks with which it is peering.

6.6 Native IPv6

The backbone network can provide the IPv6 connectivity by enabling IPv6 on all routers and the links interconnecting them. In theory, there nothing preventing the network backbone from being IPv6 only; however, it is highly unlikely that all IPv4 networks will disappear in the near future. Therefore, there will be a requirement for existing IPv4 networks to coexist with IPv6 networks. There are several strategies to choose from in deploying an IPv6 network:

- Upgrade all existing routers to dual stack and configure the network to provide IPv6 connectivity.
- Upgrade a subset of the existing routers to dual stack, establish IPv6 dedicated circuits between them, and configure the IPv6 network as an overlay network on top of the IPv4 network. The circuits can be ATM or Frame Relay, using PVCs (Permanent Virtual Circuits) or SVCs (Switched Virtual Circuits).
- Build an IPv6 network as a parallel network to the existing IPv4 network. This scenario includes the case where an IPv6 network is built from scratch, without considerations on transition or coexistence.

With each of these strategies, the IPv6 topology is likely to be different from the IPv4 topology especially if using a dual-stack deployment scheme. With respect to routing protocols, the extra configuration and operating costs of running two (2) sets of routing protocols in the backbone are worth mentioning in the dual-stack approach.

Although the dual-stack approach carries twice the complexity of single stack, it appears to be the logical choice in the long run. It is the only approach that allows separating IPv4 and IPv6 topologies and services in the core. It is also the only approach that does not have to concern itself about traversing NATs (when leaving the enterprise boundaries), and it supports IPv6 multicast properly.

Tunneling mechanisms offer a reasonable intermediate step and many tunneling mechanisms exist. Dual-stack deployments do not compete with tunneling; instead, they complement each other during the transitioning period.

7 IPv4 to IPv6 Migration Mechanisms

The IETF has provided migration tools to assist in the migration from an IPv4 backbone to an IPv6 backbone. These tools are of two (2) distinct types: Tunneling and Protocol Translation. Both of these techniques are discussed in detail within the following sections.

7.1 Tunneling

Until all routers understand IPv6, the Internet will continue to be partitioned into IPv6 networks and IPv4 networks. Since much of the IPv6 distribution infrastructure is under development, isolated IPv6 networks will still need to utilize IPv4 infrastructure to reach the IPv6 backbone. Tunneling IPv6 traffic across IPv4 network infrastructure is one means of accomplishing this goal. Figure 9 illustrates the communications links between these IPv6 islands.

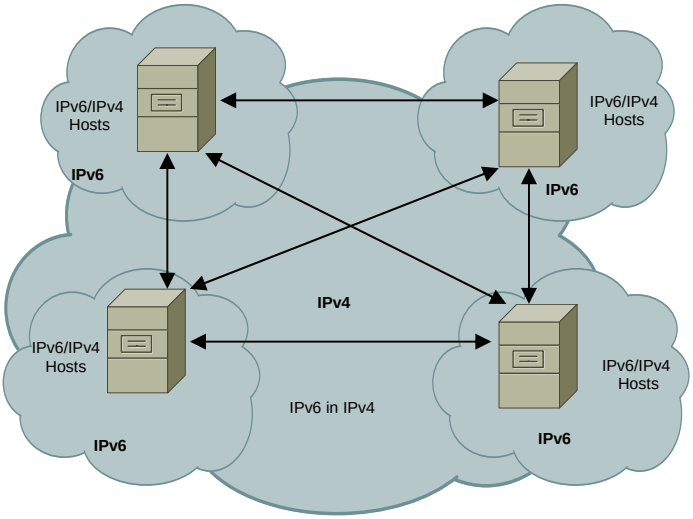


Figure 9 IPv6 in IPv4 Tunnels

7.1.1 Selection of a Tunneling Strategy

When the backbone network does not support IPv6, some tunneling mechanism is required. IPv6 in IPv4 tunnel encapsulates IPv6 traffic into IPv4 packets to traverse the IPv4 backbone. Many tunneling mechanisms are available for deploying IPv6. These mechanisms differ by the location of the tunnel endpoints and the manner in which these endpoints are determined. Each of these tunneling mechanisms requires that the endpoints be dual-stacked.

Tunneling can be implemented using four (4) different techniques:

1. Host-to-host
2. Host-to-router
3. Router-to-router
4. Router-to-host

RFC 2529, 1853, 1933, 2893, 2473, and 2529 describes IPv6 in IPv4 tunneling. The IPv6 in IPv4 tunnel behaves like a single link in an IPv6 network, decrementing the hop limitation in the IPv6 header by one (1). In this manner the existence of the tunnel is hidden.

IPv6 in IPv4 tunnels can be used over various segments of a network and the reasons for using them fall into one or more of the following categories:

- Tunnels represent a quick and inexpensive way to provide IPv6 connectivity. The endpoints will have to be upgraded to *dual stack*, but the rest of the traversed network is unaffected.
- They offer a solution to transporting IPv6 traffic across network domains that are not managed by the same entities.
- They offer the means to transport the IPv6 traffic over portions of the network where the deployment of native IPv6 is not technically possible. Either the underlying layer does not fully support IPv6 or the network elements do not support the new protocol.

At the Access Layer (AL) all of the aforementioned reasons can justify the use of tunnels. Tunneling is attractive at this layer because its interface with devices of various types and capabilities, devices that cannot be easily upgraded to support IPv6 natively.

Various tunneling mechanisms were developed to deal with deployment challenges such as traversing IPv4 NATs or scaling to large number of sites.

7.1.2 Tunneling – the 6in4 versus 6over4 debate

Before discussing tunneling, confusion over the use of the terms “6in4” and 6over4” requires clarification. 6in4 and 6over4 are quite different technical mechanisms. 6over4 is a transition mechanism which uses 6in4 as the procedure for encapsulating IPv6 packets in an IPv4 multicast environment. For the purposes of this document and because the industry has yet to standardized on the use of these terms, the terms 6in4 and 6over4 will be used interchangeably to describe this process.

7.1.3 Manually Established Tunnels

There are two primary mechanisms to manually establish IPv6 tunnels over IPv4 infrastructure; the Manually Configured Tunnel (MCT) and the GRE tunnel. The following sections provide additional depth regarding each of these mechanisms.

7.1.3.1 Manually Configured Tunneling (MCT)

There are two primary mechanisms to manually establish IPv6 tunnels over IPv4 infrastructure; the Manually Configured Tunnel (MCT) and the GRE tunnel. The following sections provide additional depth regarding each of these mechanisms. The manually configured tunnel (MCT) was one of the first transition mechanisms developed for IPv6.

For this reason, MCT is supported by most IPv6 implementations, making it a safe choice in a heterogeneous vendor environment. MCT is a static point-to-point tunnel. It terminates on dual stack routers. The tunnels endpoint’s IPv4 addresses have to be routable in the transitioned domain. A fixed IPv6 prefix has to be configured on the tunnel interfaces.

In the case of the network AL, the tunnel is typically set up between a CPE or the user’s host and a router within the AL.

Figure 10 illustrates IPv6-in-IPv4 tunneling and relevant router configuration for the MCT and provides an example of a configured tunnel between a CPE with IPv4 address 200.15.15.1 and the provider access router with IPv4 address 200.13.13.1. It should be noted that within the Cisco IOS, the RA feature is turned off by default on tunnel interfaces. If the RAs need to be exchanged over the tunnel interface, the feature should be re-enabled with the command **no ipv6 nd suppress**-ra.

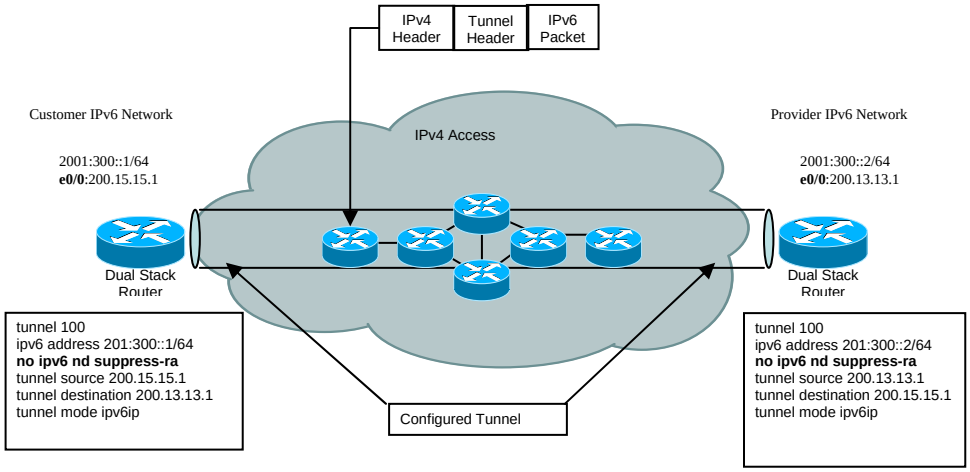


Figure 10 Manually Configured Tunneling

In this method the tunnels IPv4 endpoint address is decided by configuration information at the tunnel's encapsulating node. The encapsulating node must store the tunnel endpoint address for each tunnel. The tunnel endpoints are manually configured at both ends. In order for this to function, the tunnel endpoints must be *dual stack* nodes. NAT can not be used between the endpoints, as the IPv4 address must be reachable.

The IPv6 packets that are to be tunneled, is usually decided by routing information at the encapsulating node. This is usually done via a routing table, which directs packets based on their destination address using the prefix mask and match techniques.

The tunnel can be either unidirectional or bi-directional. If it is bi-directional, the tunnel behaves like a virtual point-to-point link.

A *default* configured tunnel can be set up to allow an IPv4/IPv6 host that has no reachability to any IPv6 router to communicate with an IPv6 network. The IPv4 address of an IPv4/IPv6 border-router to the IPv6 backbone has to be known, and can be used as the tunnel endpoint address. When this sort of tunnel is set up as default, all IPv6 destination addresses will match the route and can use the tunnel. *A **default** configured tunnel is only used if there are no other routes that match the destination address.*

One major design issue that differs between IPv4 and IPv6 when manually establishing tunnels includes allocating the tunnel's IP address. A /127 prefix in IPv6 can cause duplicate address

detection (DAD) mechanisms to fail. Only /128, /126, or /64 prefix lengths are recommended for point-to-point links.[5]

7.1.3.2 IPv6 over GRE

IPv6 over Generic Routing Encapsulation (GRE) tunnels represent another type of MCT. IPv6 over GRE enables IPv6 traffic forwarding over an existing IPv4 infrastructure, with minimum changes. Tunnel endpoints can be set up between provider edges. The tunnel endpoints are dual-stack routers, which encapsulates IPv6 traffic into IPv4, with both tunnel source and destination configured manually for GRE. Tunnels over GRE, specified in RFC 2473 and RFC 1701, have extra encapsulation header (GRE header), with the protocol type of the encapsulated protocol (0x86DD for IPv6). This enables protocols other than IP to use the same tunnel.

GRE tunnels are point-to-point tunnels. The number of tunnels needed for the service grows with the number of endpoints, making them difficult to scale. This is a typical problem with manual tunneling mechanisms, and for this reason they are usually used just to interconnect a few sites.

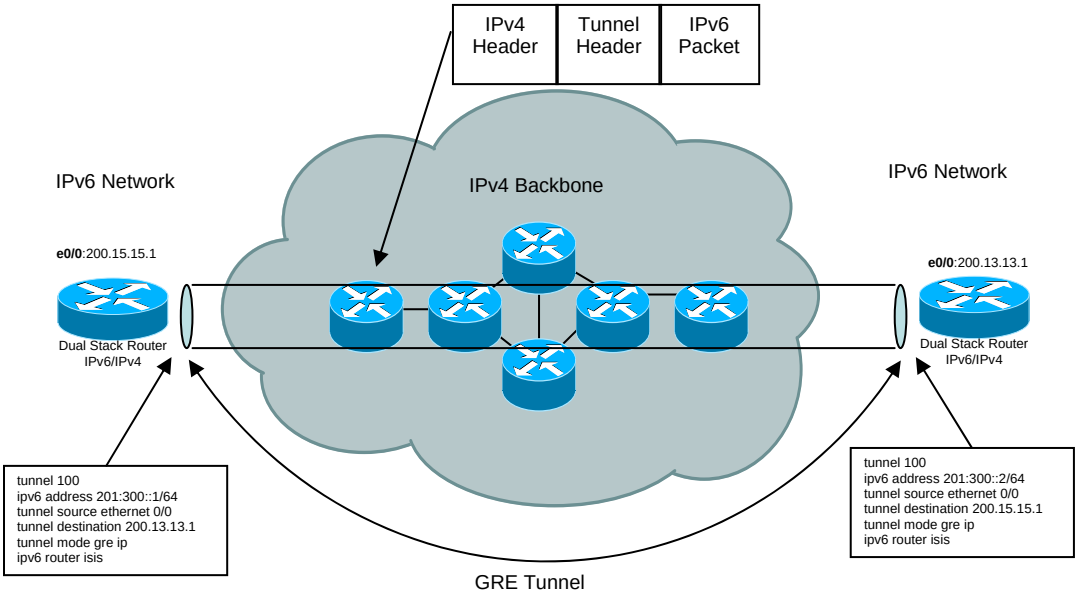


Figure 12 GRE Tunnel

The MCT is suitable for the static setup of interconnecting IPv6 locations. However, if used for connectivity of a large number of isolated hosts, static management of the MCT would encounter scalability issues. A mechanism that is available to address this issue is the Tunnel Broker and Tunnel Server.

[5] Note that the /64 prefix is the safest choice, although /128 and /126 are technically feasible. For additional details, see RFC 3177, IAB/IESG Recommendations on IPv6 Address Allocations to Sites, http://www.ietf.org/internet-drafts/draft-savola-ipv6-127-prefixlen-04.txt, and ftp://ftp.rfc-editor.org/in-notes/rfc3627.txt.

7.1.3.3 Tunnel Broker and Tunnel Server

The tunnel broker is a resource (outside of the routers terminating the MCT) that can automatically configure on a router the endpoint of the tunnel requested by a host. The host communicates with the tunnel broker over IPv4.

The tunnel server is an additional feature of the tunnel broker where the broker functionality is performed by the same device that terminates the tunnel. Both these mechanisms are attempting to provide means to scale the MCT solution when delivering IPv6 unicast service to a large number of isolated hosts. **Neither the tunnel broker nor the tunnel server is supported in the Cisco IOS software.**

7.1.3.4 Teredo (Shipworm) Tunnel

NAT is a pervasive fact of today's IPv4 networks. Most IPv6 in IPv4 tunneling mechanisms cannot transit NAT because of the protocol number used (41 for a configured tunnel). This means that if a user's NAT router cannot be upgraded to support tunneling mechanisms, **no tunnels can be sourced from the customer site**. Teredo tunnel was developed to solve this problem. It provides address assignment and host-to-host automatic tunneling for unicast IPv6 connectivity when hosts are located behind IPv4 NATs. Teredo tunnels carry the IPv6 data encapsulated in IPv4 UDP datagrams (port 3544).

A Teredo client, located behind NAT, knows that the IPv4 globally unique address of a Teredo server. The client initiates the tunnel by contacting the server, which in turn signals the setup process to a Teredo Relay router connected to the IPv6 domain that has to be reached. The server is a stateless device and just proxies the tunnel setup process. The Teredo tunnel is described in detail in the latest revision of Internet Draft *draft-huitema-v6ops-teredo*.

7.2. Automatically Established Tunnels

Automatic tunneling permits two IPv4/IPv6 hosts to communicate with each other by using the IPv4 network without pre-configuring tunnels. Within an IPv6 unicast environment, there are several mechanisms for implementing automatic tunnels; among them are *ISATAP* and *6to4*. The *automatic IPv4-compatible tunnel* mechanism is generally considered to be superseded by other tunnel mechanisms and is not addressed here.

7.2.1 ISATAP

The Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) specified in RFC 4214 was designed to provide a scalable tunneling mechanism within a **privately addressed** or globally addressed IPv4 site. Similar to other tunneling mechanisms, ISATAP encapsulates *IPv6 in IPv4* using ***ip-protocol 41***, thus ISATAP **will not work through NAT**. ISATAP treats the underlying IPv4 infrastructure as an NBMA link layer. This mechanism is automatic after the router has been configured.

Any client or router within the site that is aware of the existence of the ISATAP router can establish a tunnel to it. ISATAP is standardized in the latest version of *draft-ietf-ngtrans-isatap*. It should be noted that because of the automatic nature of the tunnel, access control has to be implemented to make sure that not all nodes that learn about an ISATAP router can establish a tunnel with it.

The key concept of ISATAP is that of the ISATAP format interface ID. Assuming that the IPv4 infrastructure is the link layer on which IPv6 is running; then in the case of ISATAP the first 32-

bits of the interface ID are 0000:5EFE and the other 32 are the IPv4 address of the interface as shown below in Example 1.

IPv4 Address 200.15.15.1
IPv4 Address in hexadecimal format C80F:0F01
ISATAP-format Interface ID **0000:5EFE:C80F:0F01**

Example 1 Building an ISATAP Format Interface ID

This interface ID can be appended to a unique-local, a link-local or global unicast prefix. Hosts configured with an IPv4 address and enabled for ISATAP will automatically build a link-local address based on the mechanism described in Figure 10. The hosts can perform a name service lookup for the service ISATAP and download the list of IPv4 addresses of the routers supporting ISATAP in that domain. This will enable the configured hosts to establish as ISATAP tunnel to one of the advertised routers.

The client sends a Router Solicitation and with the suppression of RAs being explicitly disabled on the tunnel interface of the router the client will receive the necessary information to autoconfigure. Figure 11 shows a configuration example for an access router terminating an ISATAP tunnel for a host.

The interface ID used for the link-local, unique local and/or global IPv6 addresses is generated from the IPv4 address. The ISATAP tunnels can be used between the routers within a site. The router configuration will be the same as the one presented in Example 1.

ISATAP provides a practical method of providing IPv6 access to host and to interconnected IPv6 locations that does not have to deal with NAT devices and this makes ISATAP especially attractive for connecting hosts that are within an enterprise or provider's network. This mechanism is not particularly well suited for broadband users due do the use of NAT, however, where tunnels can be created from the CPEs that have IP addresses routable within the provider's network, this mechanism can be made to work.

ISATAP is well suited for providing IPv6 access to individual, isolated dual-stack nodes within an IPv4 site. ISATAP mode of operation also lends itself well to ad-hoc networking and this proves useful for 3GPP (3rd Generation Partner Project), networks. 3GPP networks use GPRS (General Packet Radio Service) which is the technology in use by most mobile phone systems today.

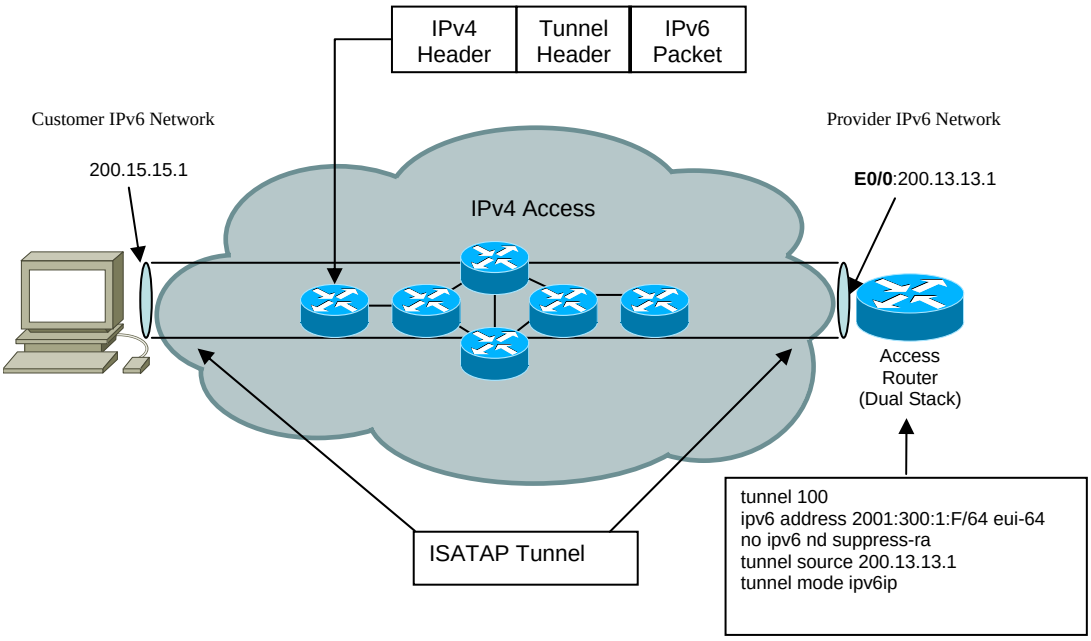


Figure 11 ISATAP Tunnel

7.2.2 IPv6 over IPv4 Tunnels

When the backbone network does not support IPv6, some tunneling mechanism is required. IPv6 in IPv4 tunnel encapsulates IPv6 traffic into IPv4 packets to traverse the IPv4 backbone. Many tunneling mechanisms are available for deploying IPv6. These mechanisms differ by the location of the tunnel endpoints and the manner in which these endpoints are determined. Each of these tunneling mechanisms requires that the endpoints be dual-stacked. Two (2) tunnel mechanisms have started to emerge in the context of an IPv4-only backbone: MCTs (Manually Configured Tunnels), among which *IPv6 over GRE tunnel* is one; and automatic tunnels, in particular *6to4 tunnels*.

7.2.3 RFC 3056 "6to4" Tunnel

6ot4 is an automatic tunneling mechanism, specified in RFC 3056. 6to4 enables isolated IPv6 networks, using the 6to4 addressing plan, to interconnect over an IPv4-only backbone at minimum configuration costs. The tunnel destination is not explicitly configured as in other tunneling mechanisms, but obtained dynamically from the IPv4 address embedded in the destination IPv6 address of the packet. Figure 13 illustrates a 6to4 network configuration.

Two (2) deployment scenarios are possible. In the simpler case, two (2) or more IPv6 sites need to interconnect over an IPv4 backbone. Each site is configured with a 2002:V4ADDR::/48 prefix, where V4ADDR is a unique IPv4 address for the site.

In figure 13, Network A is allocated the prefix 2002:**C80F:F01**::/48 and Network B is allocated 2002:C89B:B01::/48. The embedded IPv4 address C80F:f01 (200.15.15.1) is the IPv4 address for Network A. IPv4 prefixes (200.15.15/24) are distributed by an IPv4 routing protocol running in the IPv4 backbone. When Host A sends traffic to Host B (destination 2002:C80B:B01:100::1),

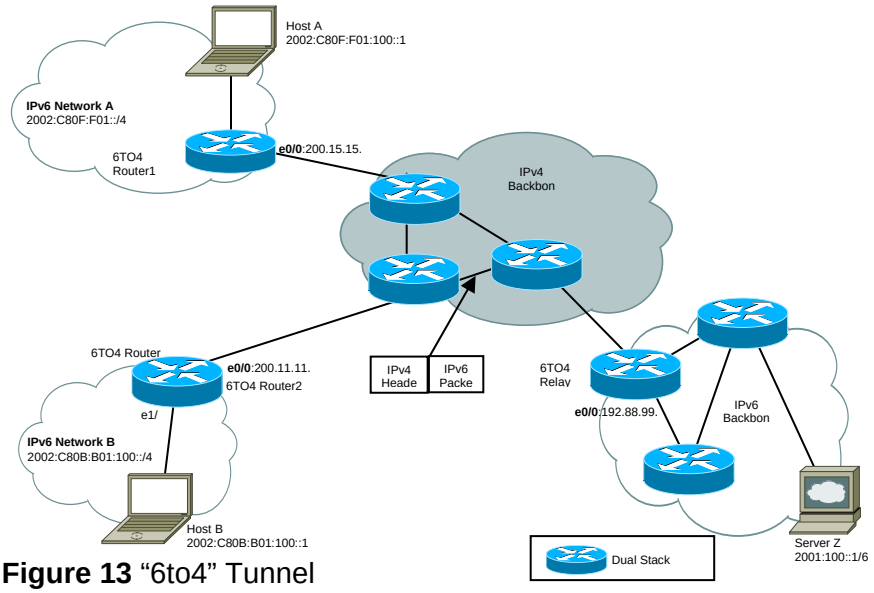


Figure 13 "6to4" Tunnel

it is routed via . This router has a 6to4 tunnel configured, with a tunnel source (200.15.15.1) but no tunnel destination. The tunnel destination is computed on-the-fly by extracting the embedded IPV4ADDR from the destination address 2002:**C80B:B01**:100::1/64, and used to encapsulate the IPv6 packet into IPv4 (source 200.15.15.1/destination 200.11.11.1). The reverse path is symmetrical.

The more complex scenario is that of a 6to4 site that requires access to global native IPv6 resources (2001::/16 Internet resources or 3FFE::/16 6bone resources). Outbound, from Network A to the IPv6 backbone, traffic is routed into the 6to4 tunnel using the default route ipv6 route::/0 2002:C058:6301::1. This route provides the IPv4 tunnel destination 192.88.99.1. A dual-stack router on the boundary between the IPv4 network and the native IPv6 domain, specifically the 6to4 relay, removes the IPv4 header and forwards the IPv6 packet natively. In-bound, the 6to4 relay uses Host A, 2002:C80F:F01:100::1, to determine the tunnel endpoint and attach the IPv4 header.

The 6to4 transition mechanism has the following advantages over manually configured tunnels:

- IPv4 tunnels does not have to be advertised as the IPv6 sites can get the address of the endpoints via a name-to-address lookup by using DNS (Domain Name Server)
- Tunnels are stateless, and therefore consume fewer resources
- Once the 6to4 relays are created, no additional configuration is required to provide 6to4 sites with connectivity to a wide-area IPv6 routing infrastructure.

The 6to4 transition mechanism also has disadvantages such as:

- When hosts within the 6to4 site have multiple addresses (6to4 and native) to select from, they must select their 6to4 address; otherwise, the 6to4 relay cannot dynamically figure out the endpoint of the tunnel.
- The underlying IPv4 address determines the enterprise 6to4 IPv6 address prefix, so the migration to native IPv6 requires the renumbering of the network.
- Limited to static or BGP4+ routing.

8 Protocol Translation

The transition mechanisms discussed thus far enabled end-to-end IPv6 connectivity. They are based on native IPv6 integration in a network, also called dual-stack, or IPv6 encapsulation over IPv4. There is a potential scenario where IPv6-only devices need to communicate with IPv4-only nodes. Such a requirement drives the need for a translation mechanism that enables communication between equipment that supports only one of the IP versions.

Even when hosts can be migrated to a dual-stack at the networking layer, some applications will remain IPv4-only, just because their implementation relies too much on IPv4 addresses.

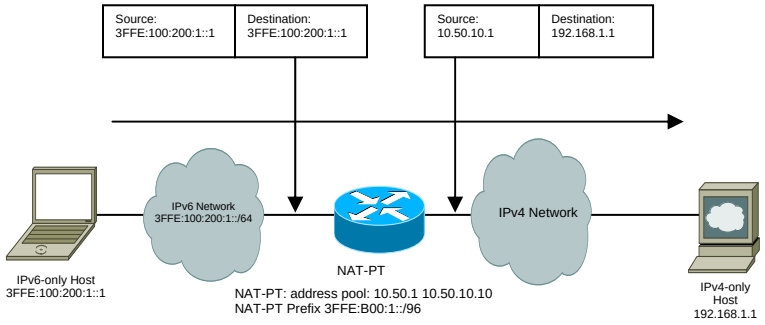


Figure 14 NAT-PT

In such environments, you can use a different set of mechanisms, based on protocol translation rather than tunneling. These mechanisms include the following:

- Application Layer Gateway (ALG)
- Bump-in-the-stack (BIS), specified in RFC 2767
- SOCKS-based gateway, specified in RFC 3089 and RFC 1928
- TCP-UDP Relay, specified in RFC 3142
- Dual-Stack Translation Mechanism (DSTM)

- Network Address Translation-Protocol Translation (NAT-PT), specified in RFC 2765 and RFC 2766

These mechanisms fall into two (2) categories: those that have an impact on hosts and those that does not. BIS and SOCKS belong to the first category and TCP-UDP Relay and NAT-PT belong to the second. The following sections provide additional information regarding basic protocol translation mechanisms that could be employed.[18]

8.1 Independent Application Layer Gateway (ALG)

The term ALG generically describes a software component that can interpret the packets of a particular protocol for a particular application. Currently, ALGs are typically employed within stateful firewalls to examine packet structures, fields, data contents, packet types, and packet sequences for several common applications. It is also possible to establish proxy servers, mail gateways, or web gateways with Application Layer Gateways (ALGs) to translate between IPv4 and IPv6 protocol packets. Since establishing ALGs independent of other established IPv6 mechanisms can involve funding requirements beyond those involved for migration, the reader is cautioned to carefully examine any cost impacts prior to undertaking any meaningful effort to use independent ALGs as part of an organization-wide solution.

8.2 RFC 2767 Bump in the Stack (BIS)

RFC 2767 Bump in the Stack (BIS) is intended to work on dual-stack hosts only. Installed on the endpoint and intended to translate IPv4 application requests within the network stack. BIS translates packets using the stateless IP/ICMP translation algorithm (SIIT) algorithm.

8.2 RFC 3089 SOCKS-Based IPv6/IPv4 Gateway

RFC 3089 SOCKS-Based IPv6/IPv4 Gateway leverages the RFC 1928 SOCKS proxy protocol. The endpoint is configured as a SOCKS client who communicates to a SOCKS server with dual-stack support. The primary advantage is that the SOCKS client may be configured as an IPv4 single-stack node.

8.4 RFC 3142 TCP-UDP Relay

RFC 3142 TCP-UDP Relay provides translation of OSI Transport Layer between IPv6-only and IPv4-only networks using a Transport Relay Translator (TRT). The primary advantages of TCP-UDP relay is that (a) modification of endpoint hosts are not required, and (b) MTU and fragmentation is not a problem.

8.5 Dual-Stack Translation Mechanism (DSTM)

Dual-Stack Translation Mechanism (DSTM) connects IPv6-only networks to IPv4-only remote endpoints, with IPv4 packets encapsulated within IPv6 packets. This mechanism requires a DSTM Server within the IPv6 domain. The major advantage is that it is transparent to the application, but DSTM does not support asymmetric traffic paths.[6]

8.6 RFC 3089 SOCKS-Based IPv6/IPv4 Gateway

RFC 3089 SOCKS-Based IPv6/IPv4 Gateway leverages the RFC 1928 SOCKS proxy protocol. The endpoint is configured as a SOCKS client who communicates to a SOCKS server with dual-

[6] For additional information see http://www.ipv6-es.com/02/docs/pedro_ruiz_2.pdf.

stack support. The primary advantage is that the SOCKS client may be configured as an IPv4 single-stack node.

8.7 RFC 2766 NAT-PT

NAT-PT enables IPv6-only hosts to communicate with IPv4-only hosts and applications. It is described in RFC 2766 and uses the IP translation referred as stateless IP/ICMP translation algorithm (SIIT), specified in RFC 2765. Tables 2 and 3 illustrate the header translation rules, from IPv4 to IPv6 and IPv6 to IPv4.

Any IPv4 option is ignored during the protocol-translation process. IPv4 fragmented packets are translated using the IPv6 fragment header. Most ICMPv4 control messages are discarded during translation, except echo requests and replies which are mapped into IPv6 with the proper code, 128 or 129. Error messages are translated whenever possible. The reverse translation is done pretty much the same way. The IPv4 checksum computation is performed after the header has been translated.

The manner in which source and destination addresses get translated is configuration controlled, or dynamic. In the simple case, explicit mapping is provided at the NAT-PT for both source and destination in both directions.

IPv6 Header Fields	Values
Version	6
Traffic Class	TOS
Flow Label	0
Payload length	Total length – header length
Next Header	Protocol
Hop Limit	TTL
Source Address	NAT-PT
Destination Address	NAT-PT

Table 2 Header Translation IPv4 to IPv6

IPv4 Header Fields	Values
Version	4
IHL	5 (no options)
TOS	Traffic Class
Total Length	Payload length + length of the IPv4 header
Identification	0
Flags	More Fragments = 0 Don’t Fragment = 1

Fragment Offset	0
TTL	Hop Limit
Protocol	Next Header Field
Checksum	Computed
Source Address	NAT-PT
Destination Address	NAT-PT
Options	Not Translated

Table 3 Header Translation IPv6 to IPv4

NAT-PT behaves in much the same manner as NAT in that it operates in various modes: static, where one IPv4 address is used to communicate with the IPv4-only host; and dynamic, where a pool of addresses is available, and dynamically bound to active flows; and PAT (Port Address Translation), where a single IPv4 address is used, and a TCP/UDP port number serves as a discriminator.

```
interface Ethernet0/0
 ipv6 address 3ffe:100:200:1::2/64
 ipv6 enable
 ipv6 nat
!
interface Ethernet1/0
 ip address 192.168.1.2 255.255.255.0
 ipv6 nat
!Entry for static mapping of v4 source to v6
ipv6 nat v4v6 source 192.168.1.1 3ffe:b00:1::1
!Entry for mapping of v6 source to dynamic v4
ipv6 nat v6v4 source list pt1 pool v4pool
ipv6 nat v6v4 pool v4pool 10.59.10.1 10.50.10.10 prefix-length 24
ipv6 nat translation udp-timeout 600
ipv6 nat prefix 3ffe:b00:1::/96
!
ipv6 access-list pt1
 permit ipv6 3ffe:100:200:1::/64 any
```

Example 3 Sample NAT-PT Router Configuration for Figure 7

NAT-PT can also be used as the DNS application-level gateway to translate the DNS transaction. The DNS query for an AAAA record gets translated into a query for an A record. Upon receiving a response with an IPv4 address, the NAT-PT builds an IPv6 address by prefixing it with the global NAT-PT prefix and sends the response to the IPv6 host. At the same time, it creates a dynamic entry for translating the destination. For the destination, when communications is initiated from the IPv6 host, the IPv6 destination can embed the IPv4 address. Figure 14 illustrates dynamic NAT-PT using a pool of address: 10.50.10.1 – 10.50.10.10. Because NAT-PT cannot support communications between IPv6-only networks via an IPv4 backbone and vice versa; NAT-PT cannot be viewed as a **preferred** mechanism for transitioning. However, there maybe scenarios where NAT-PT **maybe** feasible and therefore is offered here as an option.

9 Other IPv6 Transition Considerations

In addition to the basic IPv6 infrastructure, there are numerous miscellaneous design considerations to address.

9.1 IPv4 Firewalls and ACLs

If any legacy equipment exists and security policy permits, protocol number 41 entries may be necessary to temporarily allow IPv6 traffic to bypass non-IPv6-capable firewalls and access control lists (ACLs). An example might look like:

permit 41 host a.b.c.d host w.x.y.z

Additionally, IPv6 firewalls may need to permit ICMP messages that were less important in IPv4. For example, allowing the ICMPv6 Packet Too Big message to traverse the firewall is the primary means of discovering an MTU issue along the backbone path. In IPv6, only the source node can fragment; intermediate routers do not. By default, traffic that violates the path MTU is dropped by intermediate routers.

9.2 MTU and Bandwidth

The minimum IPv6 MTU length is 1280 bytes, whereas IPv4's minimum MTU length is 68 bytes. The maximum MTU and runt size is still based upon the network medium (e.g. 1500 and 64 bytes respectively for Ethernet). As with any encapsulation technique, any use of IPv6 over IPv4 tunneling or similar techniques will further lower the current network's maximum MTU threshold. This can affect applications sensitive to delay or packet fragmentation. Overall bandwidth requirements are generally unnoticeable, but could be a consideration for extremely low-bandwidth applications or on networks with a large number of low-bandwidth nodes. Careful prior planning in the Performance and Traffic Engineering arenas is recommended prior to any deployment.[7]

9.3 NAT

Significant challenges exist with NAT as currently deployed within the average enterprise:

- In addition to a 128-bit IP address field, IPv6 uses IP protocol number 41 which typically breaks NAT.

9.4 Multicast

The significant differences between IPv6 Multicast and IPv4 Multicast are summarized as follows:[8]

- IPv4 IGMP is replaced with IPv6 MLD
- IPv4 MSDP has no IPv6 counterpart. Must be replaced with Embedded RP, SSM, or some other option.
- IPv6 Multicast does not support Dense Mode multicast.

[7] For additional details, see http://www.tcpipguide.com/free/t_IPv6DatagramSizeMaximumTransmissionUnitMTUFragment.htm.
[8] Source: Cisco Systems, "Cisco IOS IPv6 Multicast Introduction," http://www.cisco.com/en/US/products/ps6553/products_white_paper09186a0080203e90.shtml

- FF:FF is the multicast MAC layer prefix for Ethernet.

10 Conclusion

In this paper we have discussed various mechanisms that are available to assist in the transition from IPv4 to IPv6. We have looked into the advantages and disadvantages of each mechanism how each mechanism works to achieve its goal.

This paper has also given an overview of the problems with IPv4, which evolves around address depletion and quality of service (QoS). IPv6 addresses these problems through the use of its expanded address space and improved header format, where the extension headers are the key improvement. The use of extension headers makes the processing time shorter as it is a part of the payload and not the header itself. The management of IPv6 addresses will also be more efficient when compared to the system in IPv4 with class A, B and C addresses.

The most important feature, and the primary reason for this paper, is the ability of IPv6 to coexist with IPv4. The IETF provided for mechanisms to make this coexistence possible.

The three (3) basic mechanisms available to support the coexistence between IPv4 and IPv6 are mainly divided into:

- Dual-stack mechanisms
- Tunneling mechanisms
- Translation mechanisms

These mechanisms, while not perfect, provide the tools by which a transition from IPv4 to IPv6 can be achieved.

IPv6 will be able to be integrated into existing infrastructures in various ways. It is expected that dual-stack networks will become the standard approach, but various transitioning mechanisms are available today to lower the cost of the first steps toward IPv4 and IPv6 coexistence. This paper has summarized those mechanisms to enable the reader to evaluate their use in the different deployment scenarios that will also confront those that will be implementing the transition.

Finally, the development of this paper was heavily dependent on RFCs from the IETF and several books which is listed in Appendix A and Reference sections respectively.

Appendix A *IPv6-Related RFCs*

RFC 1809: Using the Flow Label Field in IPv6

RFC 1881: IPv6 Address Allocation Management

RFC 1883: Internet Protocol, Version 6 (IPv6) Specification

RFC 1884: IP Version 6 Addressing Architecture

RFC 1885: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)

RFC 1886: DNS Extensions to support IP version 6

RFC 1887: An Architecture for IPv6 Unicast Address Allocation

RFC 1888: OSI NSAPs and IPv6

RFC 1897: IPv6 Testing Address Allocation

RFC 1924: A Compact Representation of IPv6 Addresses

RFC 1933: Transition Mechanisms for IPv6 Hosts and Routers

RFC 1970: Neighbor Discovery for IP Version 6 (IPv6)

RFC 1971: IPv6 Stateless Address Autoconfiguration

RFC 1972: A Method for the Transmission of IPv6 Packets over Ethernet Networks

RFC 1981: Path MTU Discovery for IP version 6

RFC 2019: Transmission of IPv6 Packets over FDDI

RFC 2023: IP Version 6 over PPP

RFC 2030: Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI

RFC 2073: An IPv6 Provider-Based Unicast Address Format

RFC 2080: RIPng for IPv6

RFC 2133: Basic Socket Interface Extensions for IPv6

RFC 2147: TCP and UDP over IPv6 Jumbograms

RFC 2185: Routing Aspects of IPv6 Transition

RFC 2292: Advanced Sockets API for IPv6

RFC 2373: IP Version 6 Addressing Architecture

RFC 2374: An IPv6 Aggregatable Global Unicast Address Format

RFC 2375: IPv6 Multicast Address Assignments

RFC 2428: FTP Extensions for IPv6 and NATs

RFC 2452: IP Version 6 Management Information Base for the Transmission Control Protocol

RFC 2454: IP Version 6 Management Information Base for the User Datagram Protocol

RFC 2460: Internet Protocol, Version 6 (IPv6) Specification

RFC 2461: Neighbor Discovery for IP Version 6 (IPv6)

RFC 2462: IPv6 Stateless Address Autoconfiguration

RFC 2463: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification

RFC 2464: Transmission of IPv6 Packets over Ethernet Networks

RFC 2465: Management Information Base for IP Version 6: Textual Conventions and General Group

RFC 2466: Management Information Base for IP Version 6: ICMPv6 Group

RFC 2467: Transmission of IPv6 Packets over FDDI Networks

RFC 2470: Transmission of IPv6 Packets over Token Ring Networks

RFC 2471: IPv6 Testing Address Allocation

RFC 2472: IP Version 6 over PPP

RFC 2473: Generic Packet Tunneling in IPv6 Specification

RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers

RFC 2491: IPv6 over Non-Broadcast Multiple Access (NBMA) networks

RFC 2492: IPv6 over ATM Networks

RFC 2497: Transmission of IPv6 Packets over ARCnet Networks

RFC 2526: Reserved IPv6 Subnet Anycast Addresses

RFC 2529: Transmission of IPv6 over IPv4 Domains without Explicit Tunnels

RFC 2545: Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing

RFC 2553: Basic Socket Interface Extensions for IPv6

RFC 2590: Transmission of IPv6 Packets over Frame Relay Networks Specification

RFC 2675: IPv6 Jumbograms

RFC 2710: Multicast Listener Discovery (MLD) for IPv6

RFC 2711: IPv6 Router Alert Option

RFC 2732: Format for Literal IPv6 Addresses in URL's

RFC 2740: OSPF for IPv6

RFC 2874: DNS Extensions to Support IPv6 Address Aggregation and Renumbering

RFC 2893: Transition Mechanisms for IPv6 Hosts and Routers

RFC 2894: Router Renumbering for IPv6

RFC 2928: Initial IPv6 Sub-TLA ID Assignments

RFC 3019: IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol

RFC 3041: Privacy Extensions for Stateless Address Autoconfiguration in IPv6

RFC 3053: IPv6 Tunnel Broker

RFC 3056: Connection of IPv6 Domains via IPv4 Clouds

RFC 3089: A SOCKS-based IPv6/IPv4 Gateway Mechanism

RFC 3111: Service Location Protocol Modifications for IPv6

RFC 3122: Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification

RFC 3142: An IPv6-to-IPv4 Transport Relay Translator

RFC 3146: Transmission of IPv6 Packets over IEEE 1394 Networks

RFC 3162: RADIUS and IPv6

RFC 3175: Aggregation of RSVP for IPv4 and IPv6 Reservations

RFC 3177: IAB/IESG Recommendations on IPv6 Address

RFC 3178: IPv6 Multihoming Support at Site Exit Routers

RFC 3226: DNSSEC and IPv6 A6 aware server/resolver message size requirements

RFC 3266: Support for IPv6 in Session Description Protocol (SDP)

RFC 3306: Unicast-Prefix-based IPv6 Multicast Addresses

RFC 3307: Allocation Guidelines for IPv6 Multicast Addresses

RFC 3314: Recommendations for IPv6 in Third Generation Partnership Project (3GPP) Standards

RFC 3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

RFC 3316: Internet Protocol Version 6 (IPv6) for Some Second and Third Generation Cellular Hosts

RFC 3363: Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)

RFC 3364: Tradeoffs in Domain Name System (DNS) Support for Internet Protocol version 6 (IPv6)

RFC 3484: Default Address Selection for Internet Protocol version 6 (IPv6)

RFC 3493: Basic Socket Interface Extensions for IPv6

RFC 3513: Internet Protocol Version 6 (IPv6) Addressing Architecture

RFC 3531: A Flexible Method for Managing the Assignment of Bits of an IPv6 Address Block

RFC 3542: Advanced Sockets Application Program Interface (API) for IPv6

RFC 3572: Internet Protocol Version 6 over MAPOS (Multiple Access Protocol over SONET/SDH)

RFC 3582: Goals for IPv6 Site-Multihoming Architectures

RFC 3587: IPv6 Global Unicast Address Format

References

[1] Federal Government Transition Internet Protocol Version 4 (IPv4) to Internet Protocol Version 6 (IPv6) Frequently Asked Questions 2/15/2006
[2] R. Hinden, *"IPv6 overview", May* 14, 1995, http://playground.sun.com/pub/ipng/html/ipng-main.html
[3] General IPv6 information homepage, http://www.ipv6.org
[4] S. Deering, R. Hinden, *"Internet Protocol, Version 6 (IPv6) Specification"*, RFC2460, December 1998 (Draft)
[5] S.Hagen, "*IPv6 Essentials"*, first edition, O'Reilly & Associates, Inc., July 2002.
[6] S.Hagen, "*IPv6 Essentials"*, first edition, O'Reilly & Associates, Inc., July 2002.
[7] S. Deering, R. Hinden, *"Internet Protocol, Version 6 (IPv6) Specification"*, RFC2460, December 1998 (Draft)
[8] S.Hagen,"*IPv6 Essentials"* , first edition, O'Reilly & Associates, Inc., July 2002
[9] G. H. Anthes, Computerworld, *"Internet Protocol Version 6"*, January 20, 2003, URL: http://www.computerworld.com/printthis/2003/0,4814,77627,00.html

[10] Larry L. Peterson, Bruce S. Davie, "*Computer Networks: A Systems Approach"*, second edition, Morgan Kaufmann publishers, 2000
[11] S.Hagen,"*IPv6 Essentials"* , first edition, O'Reilly & Associates, Inc., 2002
[12] R. Hinden, S. Deering, R. Fink, T. Hain, "*Initial IPv6 Sub-TLA ID Assignments"*, RFC 2928, September 2000
[13] Thomson, S., and C. T. Nartan, "IPv6 Stateless Address Autoconfiguration, Work in Progress, December 1995
[14] Bound, J., "Dynamic Host Configuration Protocol for IPv6 for IPv6 (DHCP)", Work in Progress, November 1995
[15] Droms, R., "Dynamic Host Configuration Protocol", RFC 1541, October 1993.
[16] Croft, W., and J. Gilmore, "Bootstrap Protocol", RFC 951, September 1985.
[17] Finlayson, R., Mann, T., Mogul, J., and M. Theimer, "Reverse Address Resolution Protocol", RFC 903, June 1984.
[18] Desmeules, Regis, "Cisco Self-Study: Implementing Cisco IPv6 Networks (IPv6)," Cisco Press, Indianapolis, IN, May 2003.