Application Note

Internet Protocol Version 6 Implementation on the C-Port C-5 Network Processor

Overview

Until now it has been possible to regard the Internet Protocol Version 6 (IPv6) as tomorrow's problem. However, the Asia-Pacific region is already running out of IPv4 addresses, and the same thing will happen in Europe shortly thereafter. New applications, particularly for 3G mobile phones and other 'always on' appliances such as wireless PDAs will deplete the remaining pool of IPv4 addresses even faster. Large scale IPv6 deployment is predicted to begin in Japan (the world leader in IPv6) by 2004, regardless of whether 3G wireless deployment happens on schedule.

This Application Note examines the features of IPv6 which are new or changed from IPv4, and which have implications for C-5 NP supporting the protocol. This information provides a short summary of major IPv6 features and their impact on C-5 NP applications.

This information is based on relevant IETF RFCs, IETF drafts, and reference text. See References at the end of this document for a list of sources.

The new or changed features of IPv6 are:

- New and simplified IP header
- New and expanded addressing architecture
- Improved support for IP options
- Integrated Security mechanisms
- Flow Labelling
- Neighbor Discovery and Auto configuration

This document describes all of the above features except Flow Labelling, because it is not yet defined, and probably will not be in the foreseeable future. You will also find:

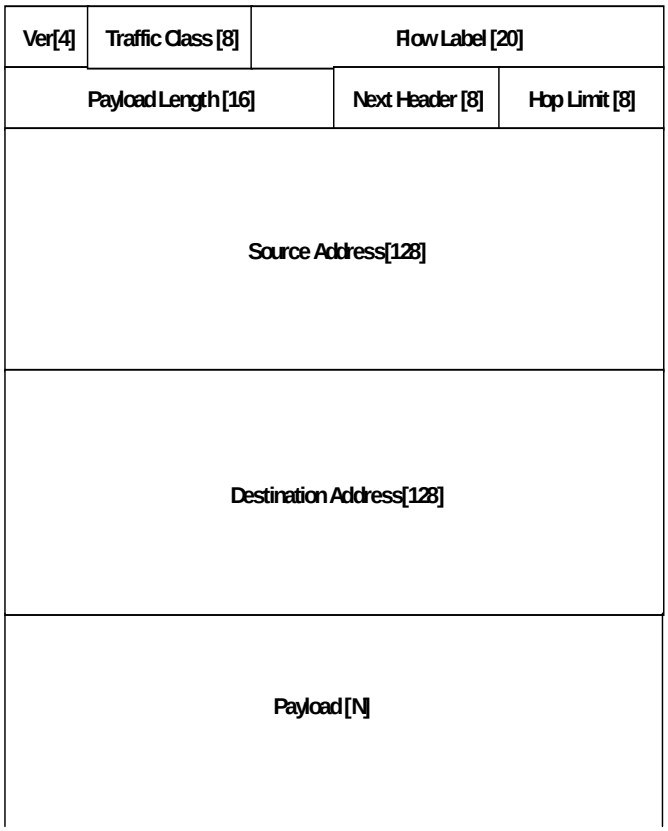
- Transition strategies from IPv4 to IPv6
- A very high-level design for an IPv6 application module

Contents

IPv6 Header

The IPv6 header defined in [IPv6] is as shown in Figure 1.

Figure 1 IPv6 Header



The fields of the address are as follows:

Field Name	Bit Length	Description
Flow Label.	20	20 bits used to label an IPv6 flow (a sequence of packets with the same characteristics). There is no consensus as yet on how this field will be used.
Traffic Class	8	An 8 bit field equivalent to the DS field (Differentiated Services - previously known as the TOS byte) in an IPv4 header. Used to determine per-hop behavior in a Diffserv enabled router.
Ver	4	The same as the Version field in an IPv4 header. Has the value 6 to indicate an IPv6 packet header
Hop Limit	8	An 8 bit field equivalent to the TTL (time to live) field of the IPv4 header. Decremented by 1 at each hop. The packet is discarded if the hop count becomes zero. As in IPv4 this is used to guard against packet forwarding loops, and to implement a trace route function.
Next Header	8	An 8 bit field equivalent to the Protocol field in an IPv4 header. Indicates the protocol of the header immediately following the IPv6 header. This can be set to indicate a UDP or a TCP header, or to indicate an IPv6 extension header (see table 3 in section 4.1).
Payload Length	16	16 bits indicating the length in bytes of additional headers, if any, and the packet data itself.
Source Address	128	The 128 bit address of the source of the packet.
Destination Address	128	The 128 bit address of the destination of the packet. The format of the Source and Destination address fields are described in the next section.
Payload	[N]	Additional headers and packet data. Length determined by the Payload Length field, up to a maximum of 65,535 bytes

The main differences between the IPv6 and IPv4 headers are:

- There is no IP Options field, and therefore no need for a header length field. In IPv6 IP options are defined in extension headers rather than in a variable length field in the header itself. IPv4 headers can be anywhere between 20 and 60 bytes in length, but all IPv6 headers are 40 bytes long. This change is intended to simplify header processing in IPv6.
- There are no fields dealing with packet fragmentation (Datagram ID, Flags, Fragment Offset in IPv4). Packet fragmentation by intermediate nodes is not permitted in IPv6, although end to end fragmentation between the source and destination nodes can be supported via an extension header.
- There is no header checksum. Again, the intention is to simplify header processing, by removing the need for checksum recalculation when the header is modified by an intermediate node (by decrementing the Hop Limit or setting the Traffic Class byte).

IPv6 Addresses

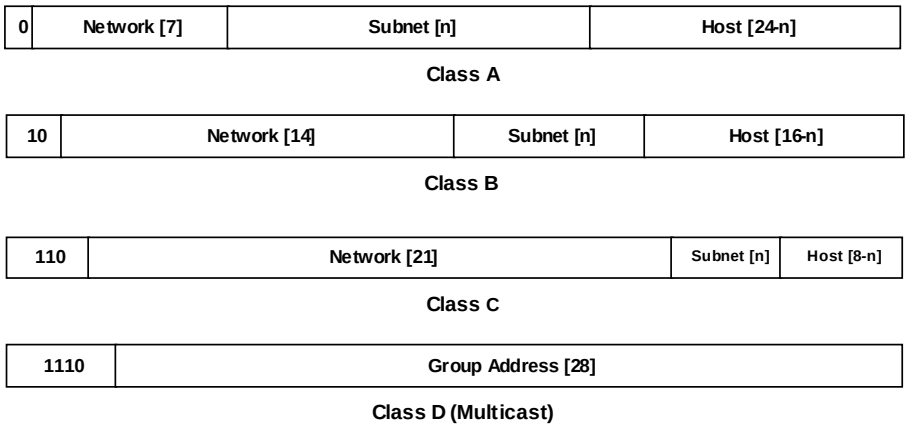
This section highlights the foundation of IPv4 that IPv6 addressing is built upon and discusses specific IPv6 address implementation.

IPv4 Addresses Reviewed

IPv6 addressing architecture builds on the architecture that has evolved over time for IPv4, and particularly the use of 'supernetting' otherwise known as CIDR (Classless Inter-Domain Routing).

IPv4 distinguishes four classes of address, three for unicast and one for multicast, as shown in Figure 2:

Figure 2 IPv4 Address Formats



IPv4 unicast addresses are hierarchical. The Internet is subdivided into Networks, each network may be further subdivided into sub-networks (subnets), and each subnet connects to a number of hosts (workstations, servers etc.). Each network is an independent administrative domain, and the network administrator is responsible for determining how many bits are allocated to identify hosts on a subnet, and thus how many bits remain to specify subnets. In most networks, 8 bits are used for the Host field, although the number is entirely left to the administrator's discretion.

Unicast address classes all start with a network number whose length is determined by the first 1 to 3 bits, a variable length subnet number (which may be omitted if the network is configured as a single subnet), and a host identifier, which specifies a particular node on the subnet. Initially, network numbers were handed out to organizations such as corporations and government departments, according to the projected size of their network. This allocation scheme was wasteful of address space, since class A assignees rarely needed to address 2^{24} hosts. All class A and B network numbers were assigned by 1994, and Class C network numbers are now in very short supply.

The original scheme has been extended by the introduction of Classless Inter-Domain Routing (CIDR), sometimes known as supernetting, which permits a block of 2^x consecutive Class C addresses to be supernetted, and treated as a

format with 21 - x bits used as the network number and x + 8 bits used for the subnet/host.

Another change has been the move from address ownership to address leasing. In the past, an organization was assigned an IP network number for all time. Now address blocks are assigned to Tier 1 service providers (typically carriers with national long haul networks and Internet exchange operators), who in turn lease sub-blocks of addresses to their customers, who in turn may lease address blocks to their customer, and so on. This concept of address leasing rather than address ownership plays a central role in IPv6 addressing architecture.

IPv4 unicast addresses can be viewed by anyone other than address administrators as consisting of an n bit network number, an m bit subnet number and a 32 - n - m bit host number. Routers can view the address even more simply as consisting of an n + m bit prefix and a 32 - n - m bit host number. Routes are represented in forwarding tables by a 32-bit address and a subnet prefix length. Addresses are looked up in the forwarding table by using a longest prefix match algorithm, and no knowledge of the internal structure of the address is required.

IPv6 Address Principles

IPv6 distinguishes three types of address:

- Unicast
- Anycast
- Multicast.

Anycast addresses are identical in format to Unicast addresses, and permit a number of hosts in a particular region to be assigned the same address. Routers within the region are required to maintain route table entries for all instances of the address and to pick the "nearest" one as the destination for packets sent to the Anycast address, so from the point of view of packet forwarding there is no difference between the two types of address.

All addresses are 128 bits long, and the first few bits of the address define the type and format of the address. These are known as the Format Prefix (FP). The following table lists the Format Prefixes currently assigned. Additional prefixes are defined but no use has yet been assigned to them. See [ARCH] for details

Table 1 Assigned IPv6 Format Prefixes

Prefix (binary)	Allocation
0000 0000	Reserved (used for IPv4 compliant IPv6 addresses)
0000 001	Reserved for NASP allocation
001	Aggregatable Global Unicast Addresses
1111 1110 10	Link-Local Unicast Addresses
1111 1110 11	Site-Local Unicast Addresses
1111 1111	Multicast Addresses

Unicast address formats are described in detail in [UNI]. The objective of these formats are to minimize the size of routing tables in core Routers, and to enable

the same longest prefix match forwarding table lookup algorithm as for IPv4, although with a larger subnet prefix (up to 64 bits vs. 32 bits for IPv4). As in the case of IPv4, routers need not be aware of the internal structure of IPv6 addresses.

IPv6 Address Representation

IPv4 addresses are normally written out as four decimal numbers, one for each byte of the address, separated by dots, for example

24.112.48.254

63.85.179.85

192.168.0.45

Similarly, IPv6 addresses are written as strings of digits with delimiters, but in this case each group of 16 bits of the address is written as four hex digits separated by colons. Leading zeros in any group of hex digits need not be written. There are eight groups of digits, so that:

3A6C:52:137F:1:FF12:48AA:12D3:1A2B

FEC0:0:0:0:19A7:28F1:1234:DE

0:0:0:0:0:0:1870:30FE

are all valid IPv6 addresses. Two rules are introduced to simplify writing addresses

1. Long sequences of zeros often occur in IPv6 addresses, as in the second and third example. In this case the string of consecutive zeros can be written with the gap separator :: representing the elided string of zeros. The second example can therefore be written more compactly as:
   FEC0: :19A7:28F1:1234:DE
   (Only one gap is allowed in an address).
2. Where the IPv6 address incorporates an IPv4 address, they can be written in standard IPv4 decimal dot notation, so the third example (which is an IPv4 compatible format IPv6 address) can be written:
   0:0:0:0:0:0:24.112.48.254
   or using both simplification rules, as
   : :24.112.48.254

For routing purposes, addresses are written with a separate decimal value indicating the length of the subnet prefix, in a similar way to CIDR addresses in IPv4. So

3A6C:52:1300::/48

Indicates that the first 48 bits are to be used as a subnet prefix for routing purposes. The length of the subnet prefix depends on where in the network hierarchy it is held. In the backbone networks of the Internet it can be as short as 16 bits or as long as 48 bits due to route aggregation. In a local ISPs network it can be as long as 64 bits.

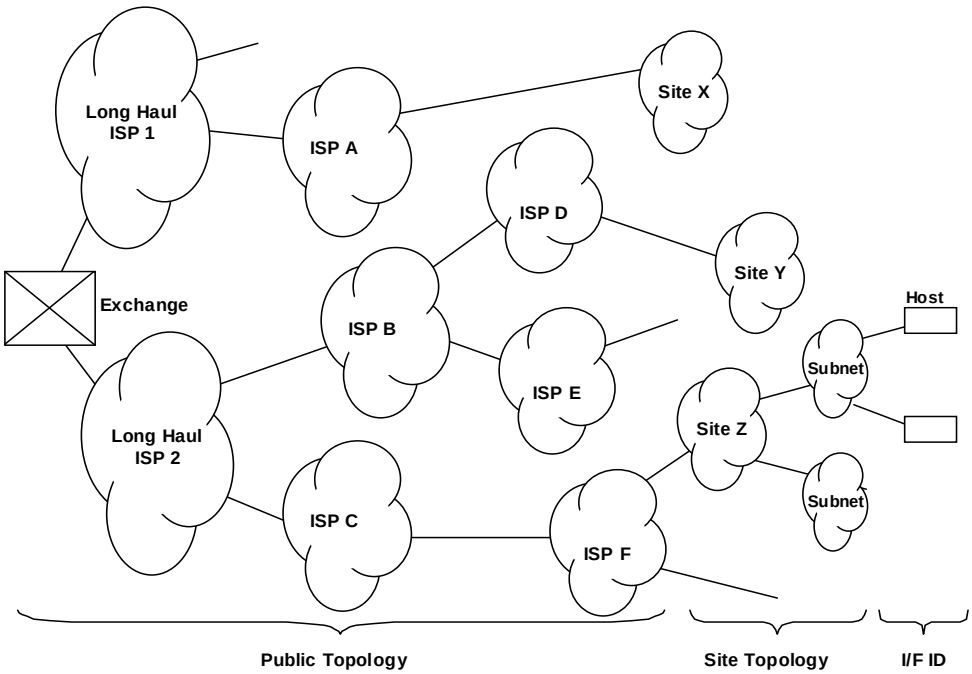
Aggregatable Global Unicast

Aggregatable Global Unicast addresses support a three level addressing hierarchy:

- Public Topology
- Site Topology
- Interface Identifier.

The network topology assumed is shown in the following figure. Public Topology is the network of Public long haul ISPs, the Internet Exchanges that interconnect them through peering arrangements, and regional and local ISPs. The first two form the backbone of the Internet.

Figure 3 Internet Topology



The regional/local ISPs receive traffic from subscriber sites and pass it up, possibly through one or more connected local/regional ISPs to the internet backbone, where it is forwarded to the destination ISP and sent down to the destination subscriber. The format of an aggregatable global unicast address is shown in Figure 4.

Figure 4 Aggregatable Global Unicast Format

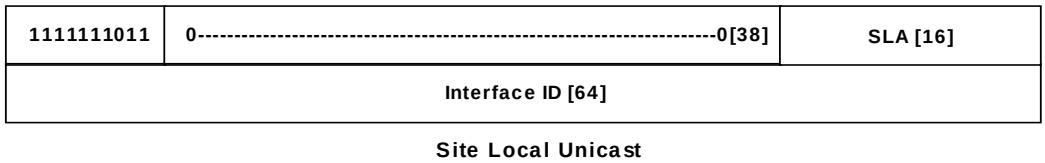
001	TLA [13]	00000000	NLA [24]	SLA [16]
Interface ID [64]				

Field Name	Bit Length	Description
SLA	16	Site-Level Aggregation Identifier. The TLA and NLA fields are used to route data through the public Internet to a specific site. The SLA is used by the organization operating the site to define its own routing hierarchy. Such an organization might choose to use the SLA to define a flat space of 2^{16} subnets, or can be subdivided into fields to provide a multi level hierarchy just like the NLA (for example a topology with 128 subnets each having up to 512 sub-subnets).
NLA	24	Next-Level Aggregation Identifier. The NLA is administered by assignees of TLA numbers and may be divided arbitrarily into sub fields to identify their own subnets and the networks of the hierarchy of ISP networks that connect to them. For example, the first n bits could be used to number the subnets of the assignees own topology and the next m bits could be used to number the local ISPs connected to it. The next 24 - n - m bits could then be used by the local ISPs to number their own subnets and identify the sites they serve. See [UNI] for a more detailed discussion of NLA usage
Reserved	8	8-bit field is reserved for future use, either as an extension of the TLA or NLA, or both, as needed
TLA	13	Top-Level Aggregation Identifier. The 13-bit TLA identifies the network of a long haul ISP or Internet Exchange. TLAs are assigned to top level networks by the Internet Numbering Authority
Format Prefix	3	Format Prefix 001
Interface ID	64	Identifies an interface on a subnet. The Interface ID is required to be an EUI-64 format address. In the case where the interface is on an Ethernet subnet, the Interface ID is formed from the 48-bit Mac Address (in EUI-48 format) as described in Appendix A of [ARCH]. Other techniques are used for non-Ethernet subnets as described in a number of "IPv6 over xxx" RFCs. These RFCs are collected in [SALUS].

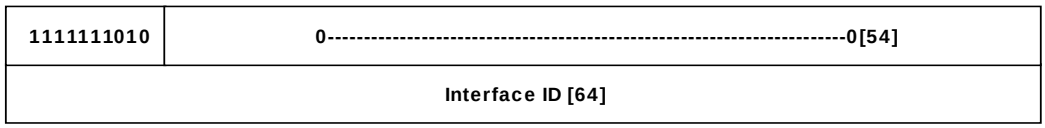
Link and Site Local Unicast

These addresses are the equivalent of IPv4 Network 10 addresses, and can be used by organizations that do not require connection to the public internet. The formats are shown in Figure 5.

Figure 5 Link and Site Local Unicast Formats



Site Local Unicast



Link Local Unicast

The **Site local** address is only valid within a single site. Packets must not be forwarded from the site onto the public Internet. The fields are:

Field Name	Bit Length	Description
SLA	16	Subnet used to define site topology in the same way as the SLA field in the aggregatable global address format
	38	38 zeros
Prefix	10	Format Prefix
Interface ID	64	Interface ID in the same IEEE EUI-64 format as described earlier.

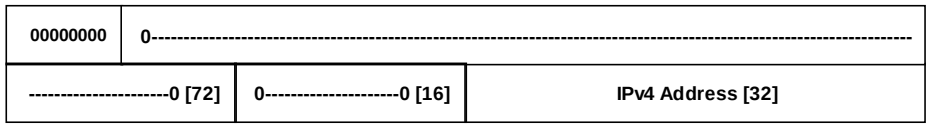
The **Link Local** address is only valid within a single subnet. Link local addressed packets must never be forwarded beyond the home subnet (and therefore can be ignored by routers). The fields of the address are:

Field Name	Bit Length	Description
SLA	16	SLA = 0
		38 zeros
Prefix	10	Format Prefix
Interface ID	64	Interface ID as above.

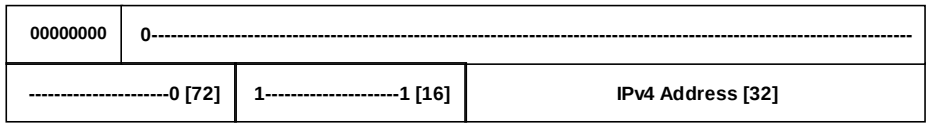
Embedded IPv4 Addresses

The remaining two formats are designed to ease the coexistence of IPv4 and IPv6 networks. The consensus is that IPv6 will initially be introduced at isolated sites, that need to use the existing IPv4 infrastructure to communicate. See the section on IPv4 to IPv6 Transition for a discussion of transition issues.

Figure 6 Embedded IPv4 Address Formats



IPv4 Compatible



IPv4 Mapped

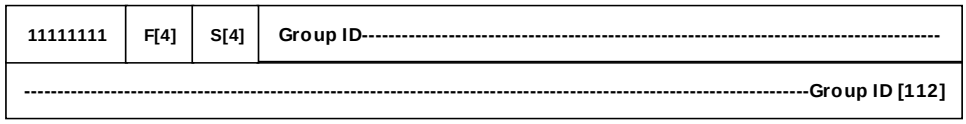
Both formats start with Format Prefix = 00000000, followed by a further 72 0s. In both cases the bottom 32 bits are an IPv4 address. The IPv4 compatible format then has the middle 16 bits set to 0, for an address prefix of ::/96, while the IPv4 mapped format has these bits set to 1, for an address prefix of ::FFFF/96.

The IPv4 compatible format is used to facilitate the tunnelling of IPv6 packets over an IPv4 network. The IPv4 mapped format is used by IPv6 only nodes to address IPv4 only nodes.

Multicast

The multicast format, shown in Figure 7, is used in the same way as the multicast address format in IPv4. A packet with a multicast address received by a router may be replicated and forwarded to neighboring routers or hosts that have subscribed to the multicast.

Figure 7 Multicast Format.



Field Name	Bit Length	Description
Group ID	112	This identifies the multicast group. If the group ID has global scope, it must be chosen to be globally unique. [MCAST2] suggests a way of doing this by basing the Group ID on the Unicast address of the originator. The GroupID must be unique for any combination of the F and S fields, but can be used for different groups for different F and S values
S	4	The four-bit field defines the **scope** of the multicast address. Table 2 shows the currently assigned field values. The remaining values are currently unassigned and should not be used.
F		Flags. Only the last bit is so far assigned. If set to 0 it means the address is well known. If set to 1 it means the address is transient (temporary). Well-known addresses are assigned by INA (for example, DHCP servers have been assigned the Group ID 19).
Prefix	8	Format Prefix

Table 2 Assigned Scope Field Definitions

Scope (Hex)	Meaning
0	Reserved
1	Node-Local Scope
2	Link-Local Scope
5	Site-Local Scope
8	Organization-Local Scope
E	Global Scope
F	Reserved

Special Addresses

Only two addresses are reserved for special use in IPv6. They are:

1. 0: : (all zeros) which is used for an unspecified address (used as a source address by nodes that have not been configured during Autoconfiguration)
2. 0: :1, the loopback address, used for testing purposes

Extension Headers

A number of IP extension headers have been defined in [IPv6] which replace the IP options and some other fields of an IPv4 header. The first extension header appears immediately following the IPv6 header itself. Each extension header includes a next header field to indicate what type of header, if any, follows it. The headers defined so far are:

- The Hop-by-Hop header (HH), which contains information which must be examined and action taken on at each hop in the packet path
- The Routing header (RH), which contains source routing information, either partially or completely defining the path the packet must take between source and destination.

- The Fragment header (FH), which is used by a source node to indicate to the destination node that it has fragmented a packet which was larger than the smallest MTU in the path from source to destination. The destination node uses this information to reassemble the packet. This header can be ignored by intermediate nodes (routers).
- The Destination Options header (DOH), which indicates any special processing that should be applied to the packet at the destination node.
- The Authentication header (AH), used as part of the integrated IPv6 security mechanisms
- The Encapsulating Security Payload header (ESP). Also used as part of the IPv6 security mechanisms.

Options immediately follow the IPv6 header and precede any higher level protocol header (for example, TCP/UDP). If present they have to occur in a specific order, that is:

1. Hop-by-Hop Options Header
2. Destination Options Header (for options to be applied at the destination indicated by the IPv6 destination address and at other intermediate points defined by a routing header)
3. Routing Header
4. Fragment Header
5. Authentication Header
6. Encapsulating Security Payload Header
7. Destinations Options Header (For options to be applied only at the final destination of the packet)

The AH and ESP are discussed in the section on security. The remaining headers are described in more detail below.

Next Header Field Values

The IPv6 header and all IPv6 extension headers include an 8 bit Next Header field, which indicates the higher level protocol of the next header, or the type of the next extension header. Some of the possible values of this field are shown in Table 3. For a complete list of assigned numbers, see [AN]

Table 3 Next Header Field Definitions

Next Header Value	Next Header Description	Notes
0	Hop-by-Hop Header	Extension header
4	IPv4 Header	Used to tunnel IPv4 packets
6	TCP header	Upper Layer Header
17	UDP header	Upper Layer Header
41	IPv6 header	Used to tunnel IPv6 packets

Table 3 Next Header Field Definitions (Continued)

Next Header Value	Next Header Description	Notes
43	Routing Header	Extension header
44	Fragmentation Header	Extension header
50	Encapsulated Security Payload	Extension header
51	Authentication Header	Extension header
58	ICMPv6	
59	No Next Header	
60	Destination Options Header	Extension header

Hop-by-Hop Header

Only one Hop-by-Hop option has so far been specified. This is the Jumbo Payload option [JUMBO] which is used to permit the transmission of packets with payloads of up to $2^{32} - 1$ bytes, rather than the maximum of 64 K bytes allowed for in the standard IPv6 header. The format of the header is shown in Figure 8.

Figure 8 Jumbo Payload Header.

Next Header [8]	Hdr Ext Len = 0	Option Type = 1100 0010	Data Length = 4
Jumbo Payload Length [32]			

The fields of the header are:

Field Name	Bit Length	Description
Data Length	4	The Length of the option data in bytes, in this case set to 4
Option Type	8	The top three bits are used to indicate what the destination node should do with an option field it does not recognize (if set to 11 as in this case it indicates that the packet should be discarded and an ICMPv6 message sent to the source address, provided the destination address not a multicast address). The third bit indicates whether the option data can change in transit (= 1) or cannot be changed (= 0, as in this case). The remaining 5 bits are used to further identify the option, set to 2 in this case
Hdr Ext Len	8	Header Extension Length. An 8 bit field set to the total length of the header in bytes, minus 8. In this case it is set to 0, indicating an 8 byte header.
Next Header	8	An 8 bit field with the same meaning as the Next header field in the IPv6 header.
Jumbo Payload Length	32	The final 32 bit field is the length of the Jumbo Payload in bytes.

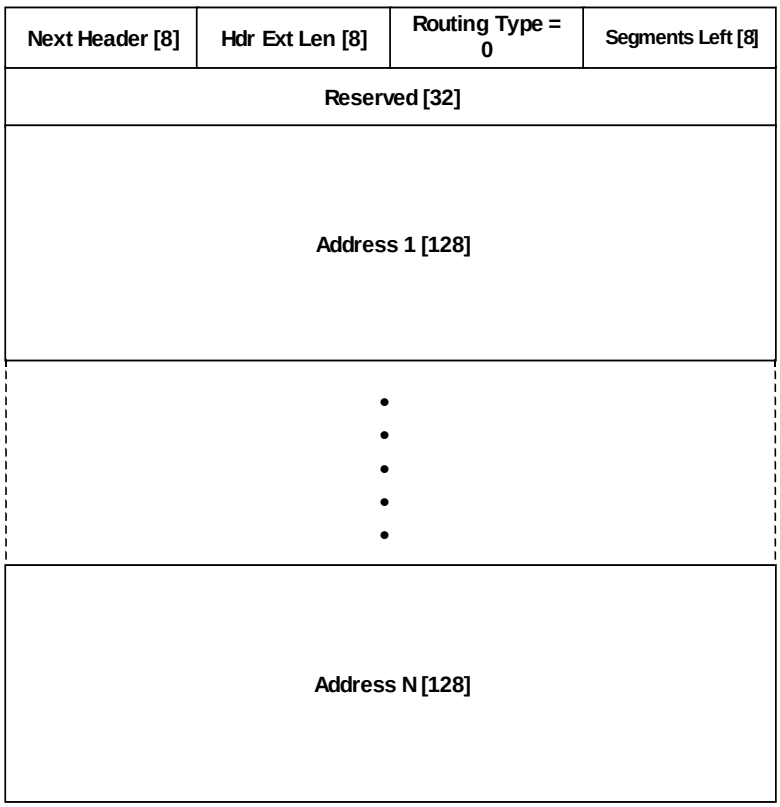
If this option header is present, the Payload Length field of the IPv6 header is not used, and must be set to 0.

Other Hop-by-Hop options may be defined in the future, and will be distinguished by different Option Type values.

Routing Header

The routing header, as shown in Figure 9, is used for loose or strict source routing.

Figure 9 Type 0 Routing Header.



Only one routing header has so far been defined, the Type 0 header. This contains a list of the IP addresses of up to 255 nodes the packet must transit between source and destination. The format of the header is as shown in the figure above.

The fields in the Header are as follows:

Field Name	Bit Length	Description
Segments Left	8	8 bits set by the source to indicate how many intermediate nodes are to be transited by the packet, and decremented by 1 at every transit node. See below for an explanation.
Routing Type.	8	8 bits set to 0. If other types of routing options are introduced they will be assigned different values for this field.
Hdr Ext Len.	8	As for Hop-by-Hop header
Next Header	8	As for Hop-by-Hop header
	32	Reserved.
Address 1	128	Address 1...AddressN. A list of addresses of N intermediate nodes to be transited
...	...	_
Address N	128	Address N

The way this header is processed is as follows: Suppose a packet is sent from source S to destination D and must go through intermediate nodes I1, I2, and I3.

1. The packet is sent from S with the destination address in the main IPv6 header set to I1, the address list set to I2, I3, D, and the Segments Left field set to 3.
2. At I1, the first address in the list (I2) is swapped with the destination address in the main header (I1) and the Segments Left field is decremented by one.
3. The packet is forwarded to the new destination address I2. At I2 the destination address (I2) is swapped with the second address in the list (I3), and Segments Left field is again decremented.
4. The process is repeated at I3, when the destination address I3 is swapped with the third and final entry in the address list (the final destination D), and the Segments Left field is decremented to 0.

On arrival at D, the address list will have been set to I1, I2, I3, and Segments Left will be 0, indicating that there are no more nodes to be visited.

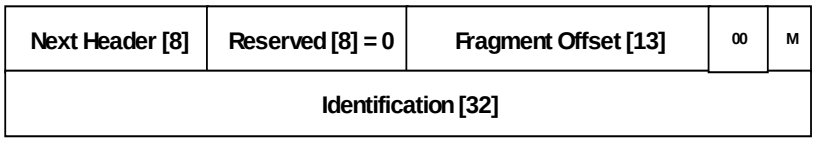
Fragment Header

The fragment header replaces the fields in the IPv4 header used to control fragmentation. The difference is that in IPv6 intermediate nodes do not need to process the header and are not permitted to fragment packets. At intermediate routers, If the MTU for the next hop is less than the size of the packet, the packet is discarded and an ICMP 'Packet Too Big' message is sent back to the source.

IPv6 requires that MTU (Maximum Transfer Unit) of any link must be at least 1280 bytes. If attempting to send packets longer that 1280 bytes, sending nodes are recommended to use path MTU discovery to determine the smallest MTU along the path to the destination. If the packet is longer than this value, it can be

fragmented by the source node, and the Fragment Extension is then used by the destination node to control the reassembly of the packet.

Figure 10 Fragment Header



Field Name	Bit Length	Description
M	1	A one bit flag set to 1 to indicate that there are more fragments to come.
	2	The next field is unused and must be set to 0.
Fragment Offset	13	A 13 bit field indicating, in units of 8 bytes, where this fragment starts in relation to the start of the packet. If set to N, it indicates that the fragment starts at byte 8*N of the packet
Reserved.	8	8 bits currently unused. Must be set to 0.
Next Header	8	As before
Identification	32	A 32 bit label used by the source to identify the packet that the fragment is part of, and used by the destination to reassemble the packet.

Destination Options Header

The destination options header is intended to convey optional information which is processed at the destination of the packet. So far, no Destination Options have been defined in RFCs, but some have been proposed in Internet Drafts.

Security

The term "*security*" covers three related issues:

1. Authentication (determination that data was sent by the node that claims to have sent it)
2. Integrity (determination that data has been received exactly as it was sent, and has not been modified in transit)
3. Confidentiality (prevention of unauthorized receivers of data from being able to read and use it).

The AH and ESP are used in IPv6 to meet these goals. The AH can be used by a sender to digitally sign a packet so that on reception the packet sender can be authenticated (Authentication and Integrity). The ESP does the same thing, and also permits the packet data to be encrypted so that only the intended receiver can decrypt and use it (Confidentiality).

IPv6 security follows the IPSec architecture also used for IPv4 defined by [IPSEC]. Although routers do not have to process these headers unless they are the destination of an IPv6 packet which includes them (unlikely), they are included here for completeness.

Authentication Header

The AH described in [AUTH] is used to permit an Integrity Check Value to be transmitted with a packet. The ICV is calculated by the sender and checked by the recipient to verify that the packet was transmitted by the sender, and has been received is as it was transmitted, without modification. The header format is as shown in Figure 11.

Figure 11 Authentication Header.

Next Header [8]	Payload Length[8]	Reserved [16] = 0
Security Parameters Index [32]		
Sequence Number [32]		
Authentication Data [N*32]		

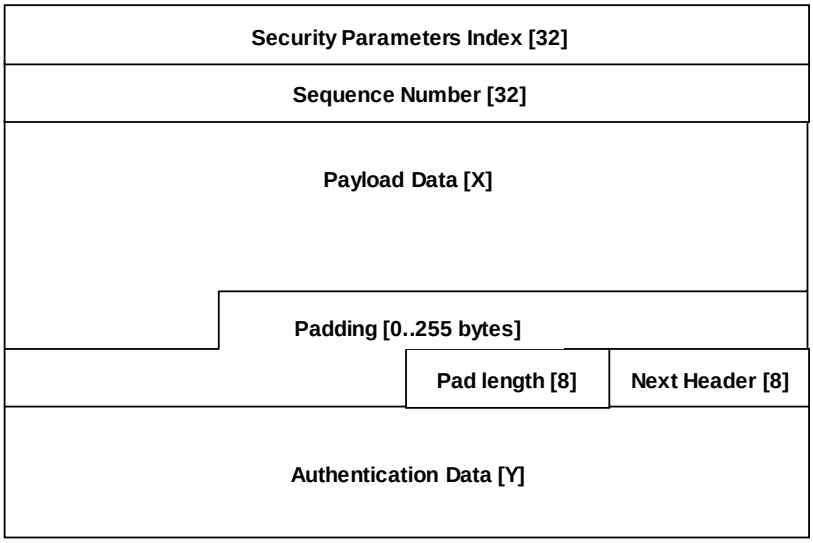
The fields are:

Field Name	Bit Length	Description
Reserved	16	16 bits are reserved and must be set to 0.
Payload Length	8	The length of the header in 32 bit words, minus 2
Next Header	8	as before
SPI	32	Security Parameters Index Identifies the security association to be used for authentication
Sequence Number	32	A counter incremented by the sender for every packet sent to the same destination. The destination should discard any packet received with a sequence number it has already processed.
Authentication data.	N*32	The Integrity Check data used to authenticate the packet. Used as determined by the SPI to authenticate the packet

Encapsulating Security Payload Header

The ESPH described in [ESPH] is used to send and received packets with encrypted payloads. In addition, it provides the same authentication capabilities as the AH. The format of the header is as shown in Figure 12.

Figure 12 Encapsulating Security Payload Header



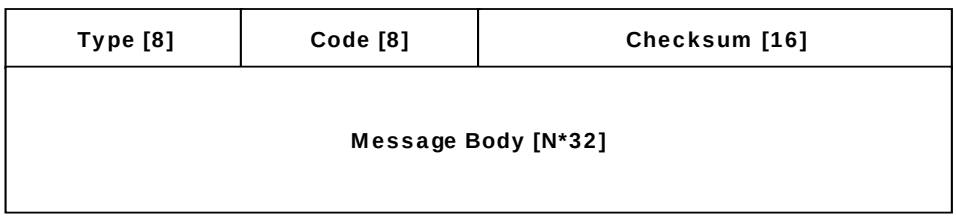
The fields are:

Field Name	Bit Length	Description
Security Parameters Index	32	Used as for the AH.
Sequence Number	32	Used as for the AH
Payload Data.	X	The encrypted payload along with any data required for decryption (e.g. initialization data)
Padding.	0-256	Added to the header to ensure that the payload data ends on the appropriate byte boundary
Pad length	8	The number of padding bytes added
Next header	8	As for all other options headers
Authentication Data	Y	Used as for the AH.

ICMPv6

In the same way as ICMP for IPv4, ICMPv6 is used by IPv6 nodes to report errors and convey information between nodes. An ICMPv6 message has an IPv6 header with a Next Header field value of 58. The general format of an ICMPv6 message is as follows:

Figure 13 ICMPv6 Message Format



Field Name	Bit Length	Description
Checksum	16	The checksum is a 16 bit CRC check calculated over the entire ICMPv6 message, including an IPv6 "Pseudo Header" as defined in Section 8.1 of [IPv6].
Code	8	The code field further defines the message content
Type	8	The Type field identifies the message contents. ICMPv6 messages are divided into two classes; Error messages and Informational messages. The Type values of Error messages go from 0 to 127. Informational messages have type values from 128 to 255.
Message Body	N*32	The message body depends on the message type. Every ICMPv6 error message includes as much of the offending packet as can be accommodated without exceeding the minimum IPv6 MTU (1280 bytes).

ICMPv6 messages received by a node are sent to control plane software for processing. Informational messages are normally formulated and sent by control plane software. Error messages are normally formulated and sent by data plane (C-5 NP) software.

Error Messages

ICMPv6 error messages are formulated and sent when any of the errors listed in the table below are detected.

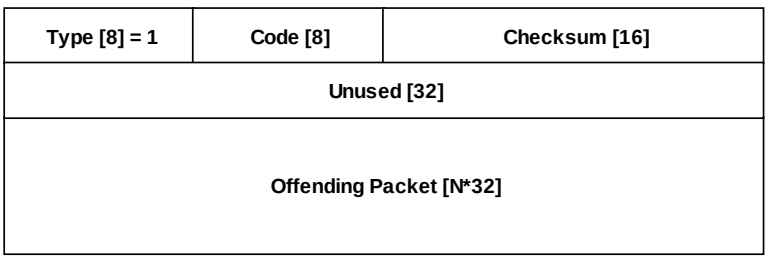
Table 4 ICMPv6 Error Message Types.

Type	Meaning
1	Destination Unreachable
2	Packet too Big
3	Time exceeded
4	Parameter Problem

Destination Unreachable

A Destination Unreachable message is sent to the source address of a packet that cannot be delivered. The format of the Destination Unreachable message is as follows:

Figure 14 Destination Unreachable Message



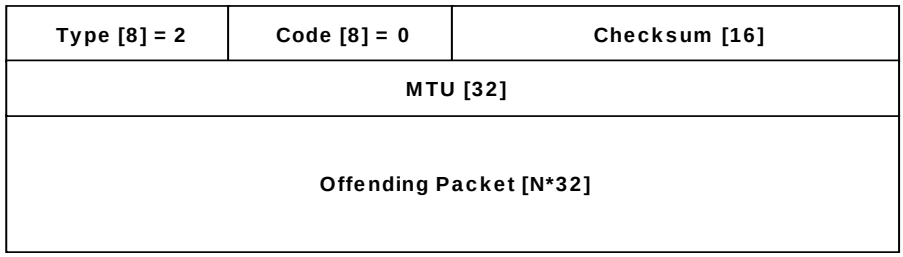
The code field indicates the reason the message was sent

Code	Meaning
0	No route to destination
1	Communication with destination prohibited
3	Address unreachable
4	Port unreachable

Packet Too Big

This message is sent to the source address of a packet that is larger than the MTU of the next hop link. The message format is:

Figure 15 Packet Too Big Message.

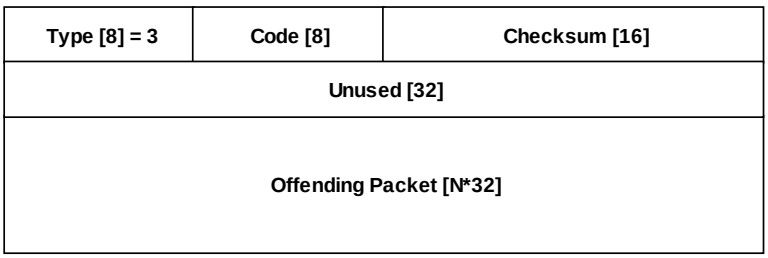


The MTU field is set to the MTU of the next hop link.

Time Exceeded

This message is sent to the source of a packet whose Hop Limit has gone to zero. The format of the message is:

Figure 16 Time Exceeded Message

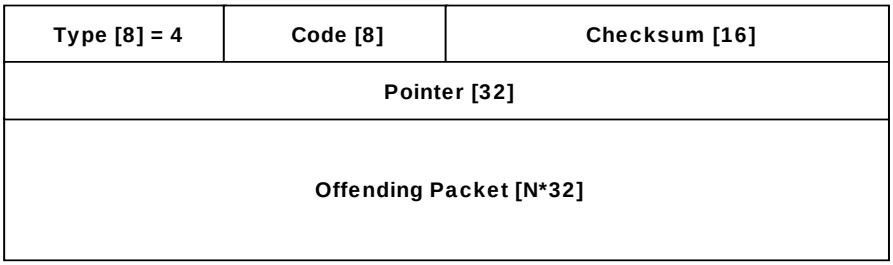


The code field is set to 0 if the hop limit of a received packet is zero, or the hop limit is decremented to zero. The Code field is set to 1 if the fragment reassembly time was exceeded.

Parameter Problem

This message is sent to the source of a packet which contained an error in its header fields. The message format is:

Figure 17 Parameter Problem Message.



- Code indicates the specific problem with the packet, as defined in Table 6.
- Pointer is the offset within the packet where the error occurred.

Table 5 Diagnostic Messages

Code	Meaning
0	Erroneous Header Field
1	Unrecognized Next Header type
2	Unrecognized IPv6 Option

Informational messages are used by control plane software for diagnostic and network configuration purposes. The Diagnostic messages are:

Table 6 ICMPv6 Diagnostic Message Types

Type	Meaning
128	Echo Request
129	Echo Reply

The Echo Request message may be sent by a network node to another node to request that it send back an Echo Reply message. The message body of an echo request contains data that must be returned unchanged in the echo reply message. See [ICMP] for more details. These messages can be used for network diagnostic purposes, as the basis for a PING utility for IPv6.

Neighbor Discovery

Neighbor Discovery messages are used by hosts and routers to solicit and advertise IP Prefixes and Link Layer addresses. Neighbor Discovery replaces IPv4's ARP and RARP and also provides support for:

- Router Discovery - used by hosts to identify local routers.
- Prefix Discovery - used by hosts to determine their Unicast address prefix.
- Configuration Discovery - used by hosts to determine parameters such as the local link MTU
- Next-Hop Determination - used by hosts to determine whether a destination address is on the local link or needs to be sent via a router (in which case router discovery can be used to determine the router's IP and Link layer address)

- Reach ability Testing - used to determine whether a node is still reachable
- Duplicate Address detection - used to determine whether the address a node wishes to use is already in use on the local link.
- Redirection - used if a router being sent a packet determines that it is not the best router to be used to reach the destination address.

The Neighbor Discovery messages are listed in the table below. These messages are used for Stateless Autoconfiguration of nodes as described in [AUTO]. The formats of Neighbor Discovery messages are described in [ND].

Table 7 ICMPv6 Neighbor Discovery Message Types.

Type	Meaning
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect

Router Solicitation and Advertisement

Routers are required to periodically broadcast router advertisement messages containing

- Their address
- Network address prefix
- Suggested hop limit value
- Link MTU

Hosts may broadcast router solicitation messages to prompt routers on the local link to immediately reply with a router advertisement message.

Neighbor Solicitation and Advertisement

Neighbor Solicitation messages may be sent to a node on the same local link to

- Request the link layer address of a neighbor
- Verify that a neighbor is still reachable using a previously determined link layer address
- Determine whether the link layer address they intend to use is unique on the local link.

Neighbor Advertisement messages are sent be nodes either in response to a neighbor solicitation message or when their link layer address changes.

Redirect

Redirect messages are sent by routers to notify hosts that they are not the best router to use to reach a particular destination.

IPv4 to IPv6 Transition

It is clear that there will not be a flash cut of the Internet from IPv4 to IPv6. For a very long time after IPv6 is introduced into the Internet, IPv4 and IPv6 hosts and networks will have to coexist and work together.

The consensus view is that IPv6 will initially be introduced in isolated sites, which will be interconnected by the IPv4 based public internet. [TRAN] lays out a number of techniques to permit IPv4/IPv6 coexistence. They are

1. Dual Stack. Supporting both IPv4 and IPv6 stacks on the same host or router. Dual stack routers can be used to bridge between IPv6 and IPv4 domains. Dual stack hosts acting as web servers are able to provide content to both IPv4 and IPv6 clients.
2. 6 over 4 configured tunnelling of IPv6 over IPv4. Encapsulating IPv6 packets in IPv4 headers to carry then over IPv4 infrastructure between IPv6 islands. Relies on Dual stack nodes to perform encapsulation and de-encapsulation at each end of the tunnel
3. 6 over 4 automatic tunnelling. Uses IPv4 compatible addresses to automatically tunnel IPv6 packets over IPv4 infrastructure.

A different tunnelling mechanism, known as 6 to 4, is described in [6TO4].

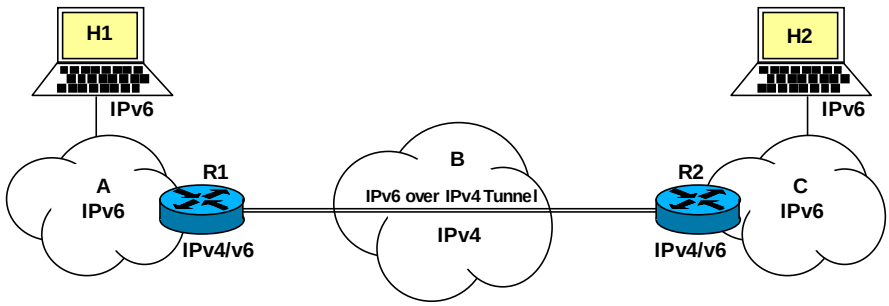
Dual Stack Nodes

A dual stack router or host is one that can send and receive both IPv4 and IPv6 packets. Received packets are sent up through either the IPv4 stack or the IPv6 stack depending on the value of the Version field (the first byte) of the packet header. Applications bind to either the TCP/IPv4 stack or the TCP/IPv6 stack (or both) to transmit packets.

Tunnelling

Tunnels are used to transport IPv6 packets over an existing IPv4 network infrastructure. The tunnels are created between dual stack nodes at the edge of the IPv4 network. The figure below illustrates Router to Router tunnelling.

Figure 18 Router to Router Tunnelling.

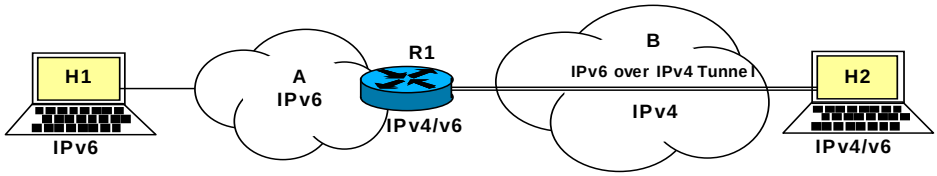


Networks A and C are IPv6 only islands connected by IPv4 only network B. Routers R1 and R2 at the edge of networks A and C are dual stack nodes. They are connected by a permanently configured IPv4 tunnel across network B. An IPv6 packet from H1 to H2 travels through network A to router R1. At R1 it is

encapsulated with an IPv4 header with Protocol 41, source address R1 and destination address R2, and forwarded through network B as an IPv4 packet. At R2, the encapsulating IPv4 header is discarded and the original IPv6 packet is forwarded across network C to H2. The tunnel between R1 and R2 is known as an IPv6 over IPv4 tunnel.

The next figure illustrates Router - Host tunnelling

Figure 19 Router - Host Tunnelling.



In this case H2 is a dual stack host on an IPv4 network. Here IPv6 packets from H1 are encapsulated at R1 and de-encapsulated at H2. Traffic flowing in the reverse direction, from H2 to H1, is encapsulated by H2 and de-encapsulated at R1

Other configurations are possible. Host to Host tunnelling could be used between dual stack hosts to communicate over an IPv4 only infrastructure, although why anyone would want to do such a thing (other than for the purpose of testing IPv6 stacks and dual stack node operation) is less than obvious.

6over4 Configured Tunnels

At the start of the tunnel, the IPv4 address of the end of the tunnel must be obtained. The source address is the IPv4 address of the node at the start of the tunnel. The destination address in the encapsulating IPv4 header is the pre-configured address of the end of the tunnel. In Figure 18, the IPv4 header would have source address R1, destination address R2. In the case of Figure 19, the source would be R1 and the destination would be H2.

The source address is always the address of the start of the tunnel. The destination address may be determined by configuration data; this is known as configured tunnelling. Because of the administrative overhead of configuring IPv6 tunnels, the concept of a Tunnel Broker [BROKER] has been introduced. This is a network node that can (semi)automatically configure tunnels between isolated IPv6 networks.

6over4 configured tunnelling is used by the 6Bone [6BONE], a test network used to connect isolate IPv6 sites (mostly universities) across the existing IPv4 Internet infrastructure.

6over4 Automatic Tunnels

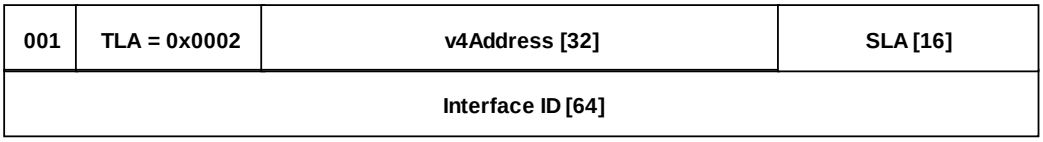
If the tunnel end is a host, it is possible to automatically derive the IPv4 destination address from the IPv6 destination address (since they are both addresses for the same node) provided the IPv6 address of the destination host node is in IPv4 compatible format (see section "Embedded IPv4 Addresses"); this is known as automatic tunnelling.

To use automatic tunnelling, in the example of Figure 19, host H2 would set its IPv6 address equal to its 32 bit IPv4 address prefixed by 32 zeros. Router R1 would automatically tunnel the packet to H2 by encapsulating it in a header with source address = R1, destination address = the low order 32 bits of the IPv4 compatible IPv6 destination address.

6to4 Tunnels

6to4 uses a different form of IPv6 address which includes an embedded IPv4 address. The address format is a variant of the Aggregatable Global Unicast address format (see section "Aggregatable Global Unicast"), with the TLA set to the reserved value 0x0002, and the NLA set to a 32 bit IPv4 address, as shown in Figure 20

Figure 20 6to4 Address Format.



Referring to Figure 18, the v4address field for network A would be the IPv4 address of the dual stack router R1. If this address is a.b.c.d, all IPv6 nodes on network A use the IPv6 address prefix 2002:a.b.c.d::/48. Similarly, if the IPv4 address of R2 were x.y.z.q, all the IPv6 nodes on network C would use the IPv6 address prefix 2002:x.y.z.q::/48.

Tunnelling across network B is done as follows. H1 sends a packet to H2 which arrives at R1. R1 detects that the destination address is a 6to4 address by matching the address prefix 2002::/16. It then creates an IPv4 header by using the v4address field in the IPv6 destination address as the IPv4 destination address, and its own IPv4 address as the source address. At R2 the header is stripped off and the IPv6 packet is delivered to H2.

C-5 NP Implementation of IPv6

Until IPv6 is more widely deployed than it has been to date, there are many things that affect application performance and memory requirements that remain unknown such as:

- Number of forwarding table entries and size of the forwarding table. The intention of IPv6 is to reduce the size of forwarding tables, particularly in the core of the network, but table sizes can only be established by practical experience with IPv6 networks.
- Distribution of address prefixes. In the case of IPv4 it is known that 80% of prefixes are between 16 and 24 bits long. No such data is available for IPv6, although it is expected that most prefixes will be between 16 and 48 bits long, due to the format of Aggregatable Global Unicast addresses.
- Number of extension headers per packet, and number of headers that require hop-by-hop processing

- Future uses of the Flow Label. Although there is no agreement on usage so far, it may be used in the future to provide QoS on a per-flow basis. This is a hot topic in IETF, although there is a body of opinion that says that the Flow Label should be reserved for future use (as happened with TOS bits in IPv4 before Diffserv came along).

The bottom line is that we must allow the greatest possible flexibility for setting engineering parameters and implementing packet processing. Here the C-5 NP should have an advantage over competing NPs with more hard coded functionality.

IPv6 Header Processing

In the early stages of IPv6 introduction and for many years to come, most IPv6 routers will be dual stack nodes. This means that they will have to support both IPv4 and IPv6 protocol processing, implying two sets of header processing algorithms and two sets of data tables (forwarding, packet filtering etc.). RxSDP byte processor micro-code must be extended to test the Version field of the packet header and then apply either current IPv4 header parsing (Version = 4) or IPv6 header parsing (Version = 6). A packet with any other version number will be dropped.

The design of IPv6 header processing SDP micro-code and CPRC software is very similar to the design for IPv4 header processing (see [IPPROC] for a description of current IPv4 header processing), although some of the operations required in IPv4 processing (e.g. header checksum recalculation, header length calculation) are not required for IPv6. This section, provides a high level design view of the SDP, CPRC and TLU operations required for processing IPv6 packets.

RxSDP Processing

The Receive Serial Data Processor (RxSDP) performs bit- and byte-level interpretation and parsing of the incoming data stream. The RxSDP processes the layer 2 headers to determine the type of packet. If the packet is IP, the RxSDP continues to parse the frame as an IP datagram. The RxSDP then checks the value of the Version field. If the Version is not equal to 4 or 6, the packet is dropped. For an IPv6 packet (Version = 6), the RxSDP performs the following processing on the IP header:

- Saves the following IP header fields to extract space:
  - IP version
  - Traffic Class
  - Flow Label
  - Payload Length
  - Next Header
  - Hop Limit
  - Source Address

  - Destination Address

- Determines whether the destination address is Unicast, multicast, or embedded IPv4 and launches the appropriate lookup on the destination address (see IPv6 Address Lookup below).
- Writes the packet header and payload to DMEM for DMA to a DRAM buffer

Ingress CPRC Processing

The ingress CPRC reads the information in extract space and the result of the lookup launched by the RxSDP and performs the following operations:

- If the Hop Limit has expired, composes an ICMPv6 Time Exceeded message and sends it to the source address, as defined in [ICMP].
- If the routing table lookup on the Destination Address failed, composes an ICMPv6 Destination Unreachable message and sends it to the source address.
- If the routing entry returned by the table lookup for the Destination Address specifies an outbound interface that is the same as the interface the packet was received on, composes an ICMPv6 Redirect message and sends it to the source address.
- If none of the above tests result in an ICMP error message, then the CPRC forwards the packet.
  - Unicast packets are forwarded by enqueuing a descriptor to the queue number specified in the routing entry returned from the Destination Address table lookup.
  - IP Multicast packets are forwarded by enqueuing to each of the queues specified in a queue vector specified in the routing entry returned from the Destination Address table lookup.

Egress CPRC Processing

The egress CPRC dequeues the packet descriptor constructed by the ingress CPRC and performs the following operations:

- If the Packet Length is greater than the MTU on the egress interface, constructs an ICMPv6 “Packet Too Big” message and sends it to the source address.
- Places Layer 2 and 3 header information in merge space for processing by the TxSDP.

TxSDP Processing

The Transmit Serial Data Processor (TxSDP) performs bit- and byte-level transmission of the outgoing layer 2 stream. The IP Forwarding Module presents IP packets to the layer 2 microcode. The layer 2 microcode encapsulates the packet for transmission, using the information from merge space, and then

transmits the layer 3 header and the packet payload, which are retrieved from BMU SDRAM by DMA to DMEM

Extension Header Processing

The ingress CP can normally ignore options headers, if present. However, under two conditions it will need to process them. These conditions are:

1. If it needs to launch a TLU operation for packet classification using a key including TCP/UDP port numbers (e.g. for Diffserv marking). In this case, the list of option headers must be scanned by reading successive Next Header values and incrementing a pointer by the corresponding option header length until either a the TCP/UDP header is found, indicated by a Next Header value of 6 or 17 in the preceding option header, or the list of option headers terminate, indicated by a Next Header value of 59 (no next header) or 60 (ICMPv6 message) in the last option header.
2. If the Next Header value in the IP header is 0, indicating a Hop-by-Hop header containing a Jumbo Payload length. In this case the Jumbo Payload length must be extracted from the first option header and used instead of the Payload Length from the IPv6 header, which is placed in extract space by the RxSDP.

Preferably, the RxSDP should perform these operations. However, if there is inadequate SDP IMEM available, the ingress CPRC will be required to perform this processing.

In all other circumstances, options headers can be ignored. Either they are not required to be processed by intermediate nodes, or the node will be the destination of the packet (either the final destination or an intermediate destination if a routing header is present), in which case the packet should be sent to the control plane for processing.

IPv6 Address Lookup

The lookup algorithm for IPv6 packets depends on the packet format, which is determined by the RxSDP, by examining the format prefix of the destination address (the first 3 to 10 bits of the address)

- Format prefixes 001 and 1111 1110 11 indicate a Unicast address lookup
- Format prefix 1111 1111 indicates a multicast address lookup
- Format prefix 0000 0000 indicates an embedded IPv4 address lookup
- All other prefixes are illegal. No lookup will be launched, the packet payload can be discarded and the problem signalled to the CPRC via extract space. The CPRC will construct a Parameter Problem ICMP message and send it to the source address.

Unicast Address Lookup

IPv6 Unicast addresses are looked up in the Unicast route table. This will be an Index/VP Trie/Data table similar to the IPv4 Unicast route table, but using a 96 bit rather than a 32 bit key.

An LPM lookup is required, using the first 64 bits of the address padded out to form the 96 bit key. For indirect routes (routes to remote hosts) this lookup will return all the data needed for packet forwarding.

However, for direct routes (routes to locally connected hosts), a lookup of the full 128 bits of the address is required. In this case, a second lookup of a Hash/Trie/ Data table of local hosts is required.

The initial lookup of the Unicast route table would return a result including a route type and a 32 bit subnet identifier field. If the route type indicates a direct route, the subnet identifier would then concatenated with the bottom 64 bits of the address (the Interface ID field) to form a 96 bit key. An exact match lookup of the local hosts table would then be performed using this 96 bit key. The second lookup would return the data required to forward the packet.

Multicast Address Lookup

.For multicast addresses, a 120 bit exact match lookup of the 112 bit GroupID, 4 bit Flags and 4 bit Scope fields is required.

Unlike IPv4, there is no need to use the source address as part of the lookup key, as the GroupID is must be globally unique if the address has global scope (Scope filed field = 0xE). Since the lookup key exceeds 112 bits, this will require a two stage exact match lookup or the use of an external TLE. Long Key Lookups are discussed below.

Embedded IPv4 Address Lookup

For IPv4 compatible and IPv4 mapped addresses an LPM lookup on the bottom 32 bits of the address (the IPv4 address) is required. For dual stack nodes, the existing IPv4 table structures can be used (a 16 bit Indexed Pointer table followed by a 16 bit VP Trie). For IPv6 only nodes, an LPM lookup on the last 64 bits can be used, using the same tables as for IPv6 Unicast lookups

Packet Classification

Layer 3/4 classification keys comprise a subset of the fields of the IP and UDP/ TCP header, and may include

- Source IP Address (128 bits)
- Destination IP Address (128 bits)
- Next Header (8 bits)
- Traffic Class (8 bits)
- Flow label (20 bits)
- UDP/TCP source port (16 bits)
- UDP/TCP destination port (16 bits)

Key lengths can range from 148 bits if only the Flow Label and Source Address are used to 324 bits if all fields are used. To implement this a Long Key lookup will be required, as described in the next section.

Lookups with Long Keys

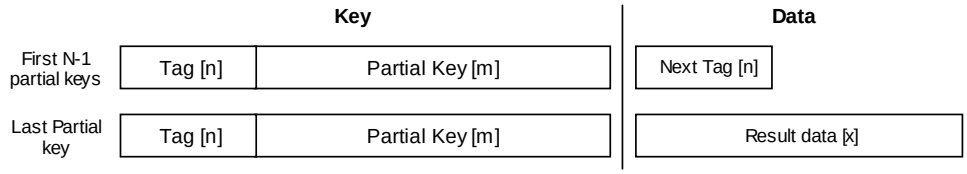
As seen earlier, implementing IPv6 requires performing exact match lookups with keys of longer than 112 bits. This section provides an algorithm for performing lookups with long keys using the TLU.

Basic Algorithm

In essence, the approach is to divide the long key K into N partial keys ki, so that K = k1∩k2 ∩....∩kN. N lookups are then performed, each using one of the partial keys concatenated with an n bit tag ti which is the data returned from the previous lookup (if successful).

The table entries are as shown in the figure below:

Figure 21 Table Entries for Long Key Lookups



The Key for each entry is an n bit tag concatenated with an m bit partial key. The data for each entry is the tag to be used in the next search, or the data associated with the entire key K if this is the last partial key. The length in bits m of the partial keys is chosen such that n+m < 112. Long keys of up to ~300 bits can be accommodated with two or three part lookups.

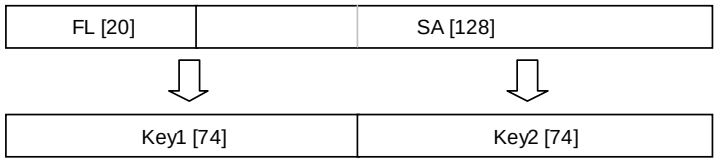
The lookup algorithm for a three part key is as follows:

1. Perform a lookup using the search key 0∩k1. If the search succeeds, read the next tag value t1 = tag(0∩k1) associated with this entry. If the search fails, exit.
2. Perform an entry lookup using the search key t1∩k2. If the search succeeds, read the next tag value t2 = tag(t1k∩2) associated with this entry. If the search fails, exit.
3. Perform an entry lookup using the search key t2∩k3. If the search succeeds, read and return the result data associated with the key K. If the search fails, exit.

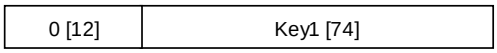
Example

To take a real example of a possible IPv6 QoS classification based on Flow Labels, suppose a 148 bit lookup is required on the 20 bit Flow Label and the 128 bit Source Address. To apply the algorithm, we divide this long key into two partial keys each of 74 bits, as shown below:

Figure 22 Example



If we assume the use of a 12 bit tag (see below for a discussion on tag length), the first lookup is performed using an 86 bit key composed of a tag of 12 zeros concatenated with Key1:



If the first lookup succeeds, the data returned is a new 12 bit Tag value which is concatenated with Key2 to form the next 86 bit lookup key:

Finally, if the second lookup succeeds, the data returned will be the data associated with the original 148 bit key.

Rationale

At first sight, it might be thought that one could simply divide a long key into a number of shorter keys, look them up one by one and use the result returned by the last lookup (assuming that the previous lookups did not fail). However, this simple scheme will not work. For example, suppose that a key is divided into two partial keys A and B. If the lookup on A succeeds all it tells you is that for some long key, its first part is A. Similarly if the lookup on B succeeds it tells you that for some long key the second part is B. However, consider the case where that long keys are A∩C and D∩B. Then a lookup on A∩B will succeed when it should in fact fail.

The use of Tag values prevents these false matches from occurring. In the example above, the entry for B would be prefixed by the tag value returned by a lookup on D. If we call this Tag(D), the key of the entry for B will be Tag(D) ∩B. If the lookup on A returns the value Tag(A), then the second lookup will use the search key Tag(A) ∩B which will not match the key of the entry for B, so the lookup will fail, as it should.

The Tag also copes with the case where a second part key is associated with 2 or more first part keys. For example, consider the case where two long keys are X∩Y and Z∩Y. The second part key Y is common to both long keys, but must return different data depending on whether the first part of the key was X or Z. The solution is to have two table entries for the second part Y, one tagged with the value returned by the lookup on X, giving the key Tag(X) ∩Y for the entry, and the other tagged with the value returned by an initial lookup on Z, giving Tag(Z) ∩Y.

Evidently the tag values returned by a lookup must all be unique. A tag value of 0 is associated with all entries that should match a first sub key lookup. Non-zero tags are used to label entries for the second, third, etc. parts of a long key. So if two long keys are A ∩B and C ∩A, there will be two table entries for A, one labelled with a zero tag (key 0 ∩A) and the other with the tag returned by a lookup on C (key Tag(C) ∩A)

Tag Length

The length of the tag is chosen according to the number of keys expected in the table. If there are a total of L long keys each divided into N sub keys, there will be a maximum of N x L table entries. The tag bit length n must be chosen such that $2^n >= (N - 1) \times L + 1$ so that every table entry can return a unique next tag value.

This formula ensures that there are enough bits to assign a unique non-zero next tag value to be returned by lookups on all entries for sub keys except the last (which does not have an associated next tag value).

The maximum tag length ever required is 20 bits, since no table can have more than 2^{20} entries.

Table Maintenance

With this algorithm, a difficulty arises with table maintenance. Again assume that two long keys are A∩C and D∩B. Suppose the long key A ∩C is deleted from the table. The partial key C should be deleted but the partial key A should not, since the key A∩B is still valid. However, if the long key A∩B is subsequently deleted, both the entries for the sub key B and the sub key A should be deleted.

This can be achieved by keeping a usage count in the data associated with each partial key entry except the last. For all but the last partial key, on deletion the usage count is decremented, but the entry is only erased if the count becomes zero, indicating that there is now no long key that includes this sub key. The last partial key entry will always be deleted.

For example, suppose that a long key K is divided into three partial keys k1∩k2 ∩k3. To delete this long key, the following steps are performed:

First, check that k1∩k2 ∩k3 is a valid long key value

1. Perform a lookup using the search key 0∩k1. If the search succeeds, read the tag value t1 = tag(0∩k1) for this entry. If the search fails, exit with an error indication.
2. Perform a lookup using the search key t1∩k2. If the search succeeds, read the tag value t2 = tag(t1∩k2) for this entry. If the search fails, exit with an error indication.
3. Perform an entry lookup using the search key t2∩k3. If the search fails, exit with an error indication.
4. We now know that k1∩k2 ∩k3 is a valid key. To delete it the following steps are performed
5. Read the count value c1 = count(0∩k1). Decrement c1. If the result is 0, delete the table entry for 0∩k1 and deallocate the tag value t1.
6. Read the count value c2 = count(t1∩k2). Decrement c2. If the result is 0, delete the table entry for t1∩k2 and deallocate the tag value t2.
7. Delete the table entry for t2∩k3.

The last table entry does not need to be tested for reuse before being deallocated. Similarly, when a new long key is added to a table, for all but the last partial key, if it already exists, its count is incremented. A new table entry is only added if the partial key did not already exist in the table. An entry is always added for the last partial key.

Using the same example, the process of inserting a new long key K divided into three partial keys k1∩k2 ∩k3 is as follows.

First, check whether k1∩k2 ∩k3 is already in the table:

1. Perform a lookup using the search key 0∩k1. If the search succeeds, read the tag value t1 = tag(0∩k1) for this entry. If the search fails, go to step 4.
2. Perform a lookup using the search key t1∩k2. If the search succeeds, read the tag value t2 = tag(t1∩k2) for this entry. If the search fails, go to step 4
3. Perform an entry lookup using the search key t2∩k3. If the succeeds, the key is already present, so exit with an error indication.

   We now know that k1∩k2 ∩k3 is not present in the table. To add it the following steps are performed
4. If partial key entry 0∩k1 is already in the table, read the count value c1 = count(0∩k1) and increment c1. Otherwise, add a new table entry with key 0∩k1, count c1 = 0, and a newly allocated tag value t1.
5. If partial key entry t1∩k2 is already in the table, read the count value c2 = count(t1∩k2) and increment c2. Otherwise, add a new table entry with key t1∩k2, count c2 = 0, and a newly allocated tag value t2.
6. Add a new table entry with key t2∩k3 and the data associated with the long key K.

The only remaining issue is the allocation and deallocation of tag values. A possible way of handling this is to use a busy/idle bitmap.

References

[AN]	RFC 1700 - Assigned Numbers
[MGT]	RFC 1881 - IPv6 Address Allocation Management
[ARCH]	RFC 2373 - IP Version 6 Addressing Architecture
[UNI]	RFC 2374 - An IPv6 Aggregatable Global Unicast Address Format
[MULTI]	RFC 2375 - IPv6 Multicast Address Assignments
[IPSEC]	RFC 2401 - Security Architecture for the Internet Protocol
[AUTH]	RFC 2402 - IP Authentication Header
[ESPH]	RFC 2406 - IP Encapsulating Security Protocol
[ADMIN]	RFC 2450 - Proposed TLA and NLA Assignment Rules
[IPv6]	RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification
[ND]	RFC 2461 - Neighbor Discovery for IPv6
[AUTO]	RFC 2462 - IPv6 Stateless Address Autoconfiguration
[ICMP]	RFC 2463 -Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification
[ETHER]	RFC 2464 - Transmission of IPv6 Packets over Ethernet Networks
[ANY]	RFC 2526 - Reserved IPv6 Subnet Anycast Addresses

[JUMBO] RFC 2675 - IPv6 Jumbograms

[TRAN] RFC 2893 - Transition Mechanisms for IPv6 Hosts and Routers

[BROKER] RFC 3053 - IPv6 Tunnel Broker

[6TO4] RFC 3056 - Connection of IPv6 Domains over IPv4 Clouds

[SEL] Draft-ietf-ipngwg-default-addr-select-03

[SCOP] Draft-ietf-ipngwg-scoping-arch-02

[MCAST2] Draft-ietf-ipngwg-uni-based-mcast-01 (extension of RFC 2375)

[ARCH2] Draft-ietf-ipngwg-addr-arch-v3-05 (replacement for RFC 2373)

[SALUS] IPv6 Addressing RFCs. Peter H. Salus, published by Morgan Kaufman, 2000

[LOSHIN] IPv6 Clearly Explained. Pete Loshin, published by Morgan Kaufmann, 1999

[6BONE] http://www.6bone.net

[IP PROC] C-Ware Reference Library Guide for CST 1.7, "IP Forwarding Module"

MOTOROLA