                    Internet Protocol, Version 6 (IPv6)
                              Specification

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   This document specifies version 6 of the Internet Protocol (IPv6),
   also sometimes referred to as IP Next Generation or IPng.

Table of Contents

1. Introduction

      IP version 6 (IPv6) is a new version of the Internet Protocol,
      designed as the successor to IP version 4 (IPv4) [RFC-791]. The
      changes from IPv4 to IPv6 fall primarily into the following
      categories:

         o Expanded Addressing Capabilities

            IPv6 increases the IP address size from 32 bits to 128 bits, to
            support more levels of addressing hierarchy, a much greater
            number of addressable nodes, and simpler auto-configuration of
            addresses. 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, used to send
            a packet to any one of a group of nodes.

         o Header Format Simplification

            Some IPv4 header fields have been dropped or made optional, to
            reduce the common-case processing cost of packet handling and
            to limit the bandwidth cost of the IPv6 header.

         o Improved Support for Extensions and Options

            Changes in the way IP header options are encoded allows for
            more efficient forwarding, less stringent limits on the length
            of options, and greater flexibility for introducing new options
            in the future.

         o Flow Labeling Capability

            A new capability is added to enable the labeling of packets
            belonging to particular traffic "flows" for which the sender
            requests special handling, such as non-default quality of
            service or "real-time" service.

o Authentication and Privacy Capabilities

Extensions to support authentication, data integrity, and
(optional) data confidentiality are specified for IPv6.

This document specifies the basic IPv6 header and the initially-
defined IPv6 extension headers and options. It also discusses packet
size issues, the semantics of flow labels and traffic classes, and
the effects of IPv6 on upper-layer protocols. The format and
semantics of IPv6 addresses are specified separately in [ADDRARCH].
The IPv6 version of ICMP, which all IPv6 implementations are required
to include, is specified in [ICMPv6].
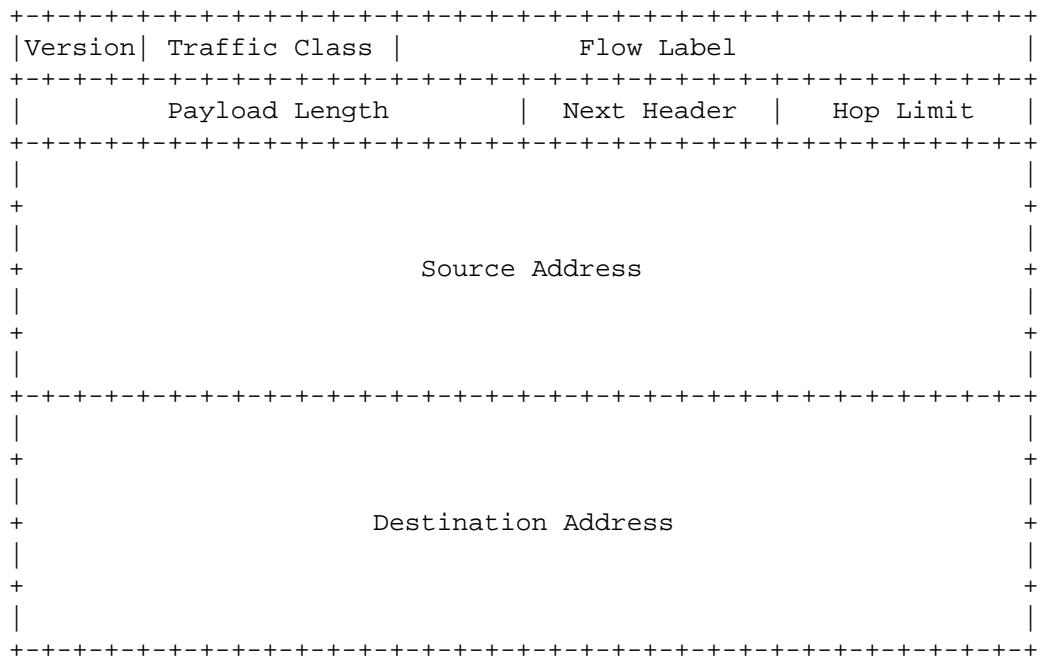
2. Terminology

node   - a device that implements IPv6.

router   - a node that forwards IPv6 packets not explicitly
                addressed to itself. [See Note below].

host   - any node that is not a router. [See Note below].

upper layer   - a protocol layer immediately above IPv6. Examples are
                transport protocols such as TCP and UDP, control
                protocols such as ICMP, routing protocols such as OSPF,
                and internet or lower-layer protocols being "tunneled"
                over (i.e., encapsulated in) IPv6 such as IPX,
                AppleTalk, or IPv6 itself.

link   - a communication facility or medium over which nodes can
                communicate at the link layer, i.e., the layer
                immediately below IPv6. Examples are Ethernets (simple
                or bridged); PPP links; X.25, Frame Relay, or ATM
                networks; and internet (or higher) layer "tunnels",
                such as tunnels over IPv4 or IPv6 itself.

neighbors   - nodes attached to the same link.

interface   - a node's attachment to a link.

address   - an IPv6-layer identifier for an interface or a set of
                interfaces.

packet   - an IPv6 header plus payload.

link MTU   - the maximum transmission unit, i.e., maximum packet
                size in octets, that can be conveyed over a link.

       path MTU   - the minimum link MTU of all the links in a path between
                   a source node and a destination node.

    Note: it is possible, though unusual, for a device with multiple
    interfaces to be configured to forward non-self-destined packets
    arriving from some set (fewer than all) of its interfaces, and to
    discard non-self-destined packets arriving from its other interfaces.
    Such a device must obey the protocol requirements for routers when
    receiving packets from, and interacting with neighbors over, the
    former (forwarding) interfaces. It must obey the protocol
    requirements for hosts when receiving packets from, and interacting
    with neighbors over, the latter (non-forwarding) interfaces.

3. IPv6 Header Format

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Version| Traffic Class |           Flow Label                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Payload Length        |  Next Header  |   Hop Limit   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    |                                                               |
    +                         Source Address                        +
    |                                                               |
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    |                                                               |
    +                      Destination Address                      +
    |                                                               |
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

    Version   4-bit Internet Protocol version number = 6.

    Traffic Class   8-bit traffic class field. See section 7.

    Flow Label   20-bit flow label. See section 6.
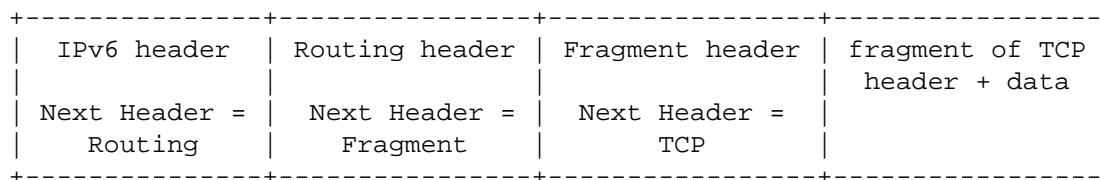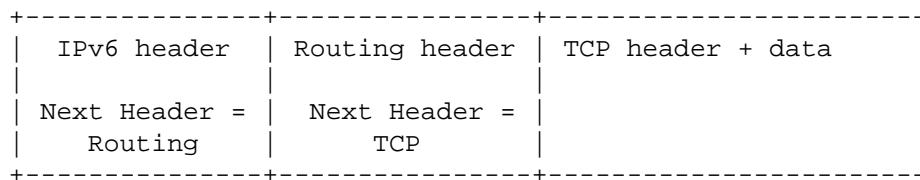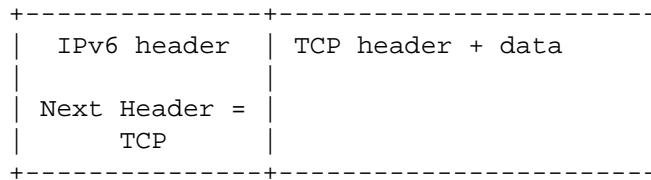
    Payload Length   16-bit unsigned integer. Length of the IPv6
                        payload, i.e., the rest of the packet following
                        this IPv6 header, in octets. (Note that any

extension headers [section 4] present are
considered part of the payload, i.e., included
in the length count.)

Next Header   8-bit selector. Identifies the type of header
                     immediately following the IPv6 header. Uses the
                     same values as the IPv4 Protocol field [RFC-1700
                     et seq.].

Hop Limit   8-bit unsigned integer. Decremented by 1 by
                     each node that forwards the packet. The packet
                     is discarded if Hop Limit is decremented to
                     zero.

Source Address   128-bit address of the originator of the packet.
                     See [ADDRARCH].

Destination Address 128-bit address of the intended recipient of the
                     packet (possibly not the ultimate recipient, if
                     a Routing header is present). See [ADDRARCH]
                     and section 4.4.

4. IPv6 Extension Headers

   In IPv6, optional internet-layer information is encoded in separate
   headers that may be placed between the IPv6 header and the upper-
   layer header in a packet. There are a small number of such extension
   headers, each identified by a distinct Next Header value. As
   illustrated in these examples, an IPv6 packet may carry zero, one, or
   more extension headers, each identified by the Next Header field of
   the preceding header:

```
   +---------------+------------------------
   |  IPv6 header  | TCP header + data
   |               |
   | Next Header = |
   |      TCP      |
   +---------------+------------------------


   +---------------+----------------+------------------------
   |  IPv6 header  | Routing header | TCP header + data
   |               |                |
   | Next Header = |  Next Header = |
   |    Routing    |       TCP      |
   +---------------+----------------+------------------------


   +---------------+----------------+-----------------+-----------------
   |  IPv6 header  | Routing header | Fragment header | fragment of TCP
   |               |                |                 |  header + data
   | Next Header = |  Next Header = |  Next Header =  |
   |    Routing    |    Fragment    |       TCP       |
   +---------------+----------------+-----------------+-----------------
```

   With one exception, extension headers are not examined or processed
   by any node along a packet's delivery path, until the packet reaches
   the node (or each of the set of nodes, in the case of multicast)
   identified in the Destination Address field of the IPv6 header.
   There, normal demultiplexing on the Next Header field of the IPv6
   header invokes the module to process the first extension header, or
   the upper-layer header if no extension header is present. The
   contents and semantics of each extension header determine whether or
   not to proceed to the next header. Therefore, extension headers must
   be processed strictly in the order they appear in the packet; a
   receiver must not, for example, scan through a packet looking for a
   particular kind of extension header and process that header prior to
   processing all preceding ones.

The exception referred to in the preceding paragraph is the Hop-by-
Hop Options header, which carries information that must be examined
and processed by every node along a packet's delivery path, including
the source and destination nodes. The Hop-by-Hop Options header,
when present, must immediately follow the IPv6 header. Its presence
is indicated by the value zero in the Next Header field of the IPv6
header.

If, as a result of processing a header, a node is required to proceed
to the next header but the Next Header value in the current header is
unrecognized by the node, it should discard the packet and send an
ICMP Parameter Problem message to the source of the packet, with an
ICMP Code value of 1 ("unrecognized Next Header type encountered")
and the ICMP Pointer field containing the offset of the unrecognized
value within the original packet. The same action should be taken if
a node encounters a Next Header value of zero in any header other
than an IPv6 header.

Each extension header is an integer multiple of 8 octets long, in
order to retain 8-octet alignment for subsequent headers. Multi-
octet fields within each extension header are aligned on their
natural boundaries, i.e., fields of width n octets are placed at an
integer multiple of n octets from the start of the header, for n = 1,
2, 4, or 8.

A full implementation of IPv6 includes implementation of the
following extension headers:

        Hop-by-Hop Options
        Routing (Type 0)
        Fragment
        Destination Options
        Authentication
        Encapsulating Security Payload

The first four are specified in this document; the last two are
specified in [RFC-2402] and [RFC-2406], respectively.

4.1 Extension Header Order

When more than one extension header is used in the same packet, it is
recommended that those headers appear in the following order:

        IPv6 header
        Hop-by-Hop Options header
        Destination Options header (note 1)
        Routing header
        Fragment header

Authentication header (note 2)
Encapsulating Security Payload header (note 2)
Destination Options header (note 3)
upper-layer header

note 1: 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.

note 2: additional recommendations regarding the relative
order of the Authentication and Encapsulating
Security Payload headers are given in [RFC-2406].

note 3: for options to be processed only by the final
destination of the packet.

Each extension header should occur at most once, except for the
Destination Options header which should occur at most twice (once
before a Routing header and once before the upper-layer header).

If the upper-layer header is another IPv6 header (in the case of IPv6
being tunneled over or encapsulated in IPv6), it may be followed by
its own extension headers, which are separately subject to the same
ordering recommendations.

If and when other extension headers are defined, their ordering
constraints relative to the above listed headers must be specified.

IPv6 nodes must accept and attempt to process extension headers in
any order and occurring any number of times in the same packet,
except for the Hop-by-Hop Options header which is restricted to
appear immediately after an IPv6 header only. Nonetheless, it is
strongly advised that sources of IPv6 packets adhere to the above
recommended order until and unless subsequent specifications revise
that recommendation.

4.2 Options

   Two of the currently-defined extension headers -- the Hop-by-Hop
   Options header and the Destination Options header -- carry a variable
   number of type-length-value (TLV) encoded "options", of the following
   format:

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
    |  Option Type  |  Opt Data Len |  Option Data
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
```

       Option Type   8-bit identifier of the type of option.

       Opt Data Len   8-bit unsigned integer. Length of the Option
                           Data field of this option, in octets.

       Option Data   Variable-length field. Option-Type-specific
                           data.

   The sequence of options within a header must be processed strictly in
   the order they appear in the header; a receiver must not, for
   example, scan through the header looking for a particular kind of
   option and process that option prior to processing all preceding
   ones.

   The Option Type identifiers are internally encoded such that their
   highest-order two bits specify the action that must be taken if the
   processing IPv6 node does not recognize the Option Type:

      00 - skip over this option and continue processing the header.

      01 - discard the packet.

      10 - discard the packet and, regardless of whether or not the
           packet's Destination Address was a multicast address, send an
           ICMP Parameter Problem, Code 2, message to the packet's
           Source Address, pointing to the unrecognized Option Type.

      11 - discard the packet and, only if the packet's Destination
           Address was not a multicast address, send an ICMP Parameter
           Problem, Code 2, message to the packet's Source Address,
           pointing to the unrecognized Option Type.

   The third-highest-order bit of the Option Type specifies whether or
   not the Option Data of that option can change en-route to the
   packet's final destination. When an Authentication header is present

   in the packet, for any option whose data may change en-route, its
   entire Option Data field must be treated as zero-valued octets when
   computing or verifying the packet's authenticating value.

      0 - Option Data does not change en-route

      1 - Option Data may change en-route

   The three high-order bits described above are to be treated as part
   of the Option Type, not independent of the Option Type. That is, a
   particular option is identified by a full 8-bit Option Type, not just
   the low-order 5 bits of an Option Type.

   The same Option Type numbering space is used for both the Hop-by-Hop
   Options header and the Destination Options header. However, the
   specification of a particular option may restrict its use to only one
   of those two headers.

   Individual options may have specific alignment requirements, to
   ensure that multi-octet values within Option Data fields fall on
   natural boundaries. The alignment requirement of an option is
   specified using the notation xn+y, meaning the Option Type must
   appear at an integer multiple of x octets from the start of the
   header, plus y octets. For example:

      2n   means any 2-octet offset from the start of the header.
      8n+2   means any 8-octet offset from the start of the header,
             plus 2 octets.

   There are two padding options which are used when necessary to align
   subsequent options and to pad out the containing header to a multiple
   of 8 octets in length. These padding options must be recognized by
   all IPv6 implementations:

   Pad1 option (alignment requirement: none)

      +-+-+-+-+-+-+-+-+
      | 0 |
      +-+-+-+-+-+-+-+-+

      NOTE! the format of the Pad1 option is a special case -- it does
            not have length and value fields.

      The Pad1 option is used to insert one octet of padding into the
      Options area of a header. If more than one octet of padding is
      required, the PadN option, described next, should be used, rather
      than multiple Pad1 options.

PadN option (alignment requirement: none)

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
   |       1       | Opt Data Len |  Option Data
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
```

The PadN option is used to insert two or more octets of padding
into the Options area of a header. For N octets of padding, the
Opt Data Len field contains the value N-2, and the Option Data
consists of N-2 zero-valued octets.

Appendix B contains formatting guidelines for designing new options.

4.3 Hop-by-Hop Options Header

The Hop-by-Hop Options header is used to carry optional information
that must be examined by every node along a packet's delivery path.
The Hop-by-Hop Options header is identified by a Next Header value of
0 in the IPv6 header, and has the following format:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Next Header  |  Hdr Ext Len  |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   .                                                               .
   .                        Options                                .
   .                                                               .
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Header   8-bit selector. Identifies the type of header
                     immediately following the Hop-by-Hop Options
                     header. Uses the same values as the IPv4
                     Protocol field [RFC-1700 et seq.].

Hdr Ext Len   8-bit unsigned integer. Length of the Hop-by-
                     Hop Options header in 8-octet units, not
                     including the first 8 octets.

Options   Variable-length field, of length such that the
                     complete Hop-by-Hop Options header is an integer
                     multiple of 8 octets long. Contains one or more
                     TLV-encoded options, as described in section
                     4.2.

The only hop-by-hop options defined in this document are the Pad1 and
PadN options specified in section 4.2.

4.4 Routing Header

   The Routing header is used by an IPv6 source to list one or more
   intermediate nodes to be "visited" on the way to a packet's
   destination. This function is very similar to IPv4's Loose Source
   and Record Route option. The Routing header is identified by a Next
   Header value of 43 in the immediately preceding header, and has the
   following format:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Next Header  |  Hdr Ext Len  |  Routing Type | Segments Left |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   .                                                               .
   .                       type-specific data                     .
   .                                                               .
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Next Header   8-bit selector. Identifies the type of header
                        immediately following the Routing header. Uses
                        the same values as the IPv4 Protocol field
                        [RFC-1700 et seq.].

   Hdr Ext Len   8-bit unsigned integer. Length of the Routing
                        header in 8-octet units, not including the first
                        8 octets.

   Routing Type   8-bit identifier of a particular Routing header
                        variant.

   Segments Left   8-bit unsigned integer. Number of route
                        segments remaining, i.e., number of explicitly
                        listed intermediate nodes still to be visited
                        before reaching the final destination.

   type-specific data   Variable-length field, of format determined by
                        the Routing Type, and of length such that the
                        complete Routing header is an integer multiple
                        of 8 octets long.
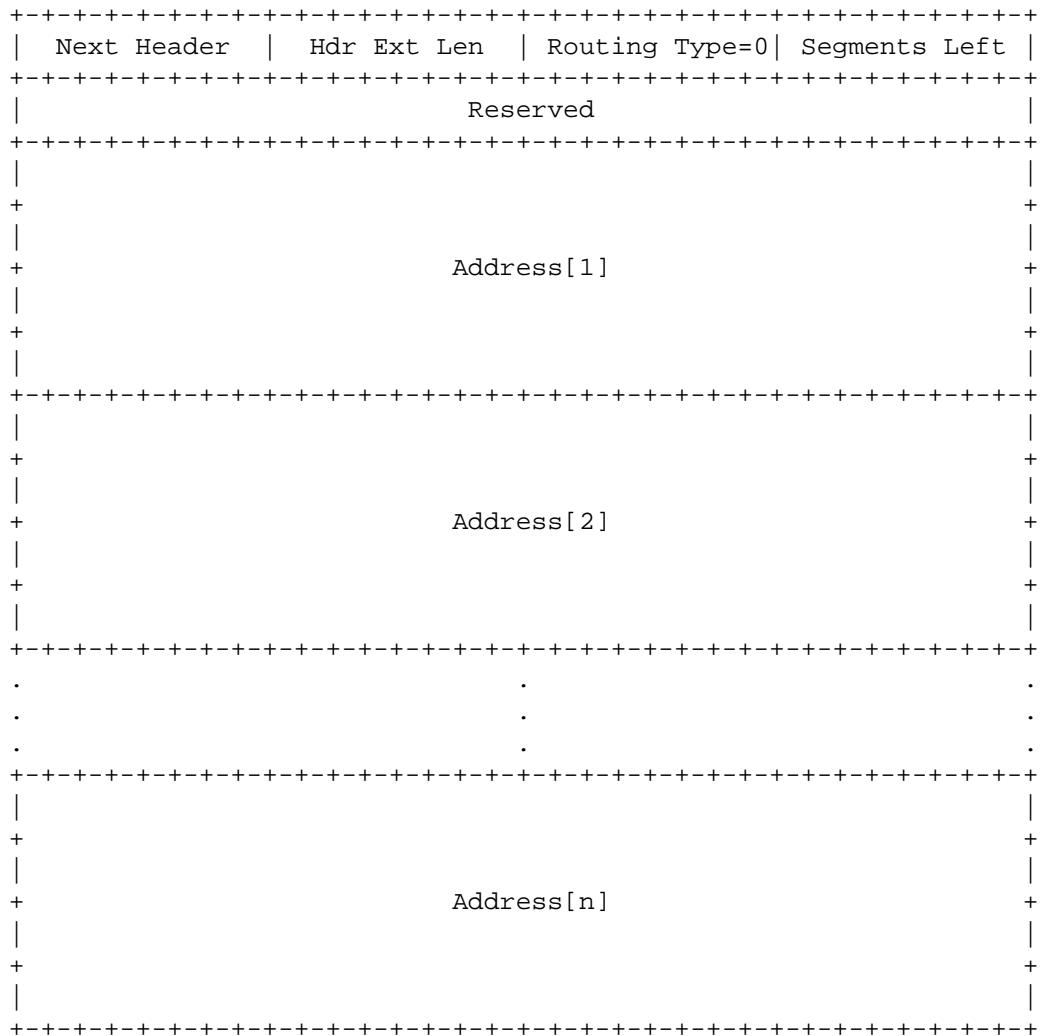
   If, while processing a received packet, a node encounters a Routing
   header with an unrecognized Routing Type value, the required behavior
   of the node depends on the value of the Segments Left field, as
   follows:

   If Segments Left is zero, the node must ignore the Routing header
   and proceed to process the next header in the packet, whose type
   is identified by the Next Header field in the Routing header.

   If Segments Left is non-zero, the node must discard the packet and
   send an ICMP Parameter Problem, Code 0, message to the packet's
   Source Address, pointing to the unrecognized Routing Type.

If, after processing a Routing header of a received packet, an
intermediate node determines that the packet is to be forwarded onto
a link whose link MTU is less than the size of the packet, the node
must discard the packet and send an ICMP Packet Too Big message to
the packet's Source Address.

   The Type 0 Routing header has the following format:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Next Header  | Hdr Ext Len  | Routing Type=0| Segments Left |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            Reserved                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                             |
   +                                                             +
   |                                                             |
   +                          Address[1]                         +
   |                                                             |
   +                                                             +
   |                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                             |
   +                                                             +
   |                                                             |
   +                          Address[2]                         +
   |                                                             |
   +                                                             +
   |                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   .                               .                             .
   .                               .                             .
   .                               .                             .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                             |
   +                                                             +
   |                                                             |
   +                          Address[n]                         +
   |                                                             |
   +                                                             +
   |                                                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Next Header   8-bit selector. Identifies the type of header
                       immediately following the Routing header. Uses
                       the same values as the IPv4 Protocol field
                       [RFC-1700 et seq.].

   Hdr Ext Len   8-bit unsigned integer. Length of the Routing
                       header in 8-octet units, not including the first
                       8 octets. For the Type 0 Routing header, Hdr
                       Ext Len is equal to two times the number of
                       addresses in the header.

   Routing Type   0.

Segments Left   8-bit unsigned integer. Number of route
                       segments remaining, i.e., number of explicitly
                       listed intermediate nodes still to be visited
                       before reaching the final destination.

Reserved   32-bit reserved field. Initialized to zero for
                       transmission; ignored on reception.

Address[1..n]   Vector of 128-bit addresses, numbered 1 to n.

Multicast addresses must not appear in a Routing header of Type 0, or
in the IPv6 Destination Address field of a packet carrying a Routing
header of Type 0.

A Routing header is not examined or processed until it reaches the
node identified in the Destination Address field of the IPv6 header.
In that node, dispatching on the Next Header field of the immediately
preceding header causes the Routing header module to be invoked,
which, in the case of Routing Type 0, performs the following
algorithm:

```
   if Segments Left = 0 {
      proceed to process the next header in the packet, whose type is
      identified by the Next Header field in the Routing header
   }
   else if Hdr Ext Len is odd {
         send an ICMP Parameter Problem, Code 0, message to the Source
         Address, pointing to the Hdr Ext Len field, and discard the
         packet
   }
   else {
      compute n, the number of addresses in the Routing header, by
      dividing Hdr Ext Len by 2

      if Segments Left is greater than n {
         send an ICMP Parameter Problem, Code 0, message to the Source
         Address, pointing to the Segments Left field, and discard the
         packet
      }
      else {
         decrement Segments Left by 1;
         compute i, the index of the next address to be visited in
         the address vector, by subtracting Segments Left from n

         if Address [i] or the IPv6 Destination Address is multicast {
            discard the packet
         }
         else {
            swap the IPv6 Destination Address and Address[i]

            if the IPv6 Hop Limit is less than or equal to 1 {
               send an ICMP Time Exceeded -- Hop Limit Exceeded in
               Transit message to the Source Address and discard the
               packet
            }
            else {
               decrement the Hop Limit by 1

               resubmit the packet to the IPv6 module for transmission
               to the new destination
            }
         }
      }
   }
```

As an example of the effects of the above algorithm, consider the
case of a source node S sending a packet to destination node D, using
a Routing header to cause the packet to be routed via intermediate
nodes I1, I2, and I3. The values of the relevant IPv6 header and
Routing header fields on each segment of the delivery path would be
as follows:

As the packet travels from S to I1:

```
    Source Address = S                  Hdr Ext Len = 6
    Destination Address = I1            Segments Left = 3
                                        Address[1] = I2
                                        Address[2] = I3
                                        Address[3] = D
```

As the packet travels from I1 to I2:

```
    Source Address = S                  Hdr Ext Len = 6
    Destination Address = I2            Segments Left = 2
                                        Address[1] = I1
                                        Address[2] = I3
                                        Address[3] = D
```

As the packet travels from I2 to I3:
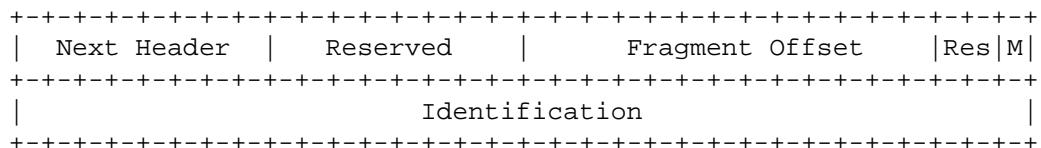
```
    Source Address = S                  Hdr Ext Len = 6
    Destination Address = I3            Segments Left = 1
                                        Address[1] = I1
                                        Address[2] = I2
                                        Address[3] = D
```

As the packet travels from I3 to D:

```
    Source Address = S                  Hdr Ext Len = 6
    Destination Address = D             Segments Left = 0
                                        Address[1] = I1
                                        Address[2] = I2
                                        Address[3] = I3
```

4.5 Fragment Header

   The Fragment header is used by an IPv6 source to send a packet larger
   than would fit in the path MTU to its destination. (Note: unlike
   IPv4, fragmentation in IPv6 is performed only by source nodes, not by
   routers along a packet's delivery path -- see section 5.) The
   Fragment header is identified by a Next Header value of 44 in the
   immediately preceding header, and has the following format:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Next Header  |   Reserved    |      Fragment Offset    |Res|M|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         Identification                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Next Header   8-bit selector. Identifies the initial header
                        type of the Fragmentable Part of the original
                        packet (defined below). Uses the same values as
                        the IPv4 Protocol field [RFC-1700 et seq.].

   Reserved   8-bit reserved field. Initialized to zero for
                        transmission; ignored on reception.

   Fragment Offset   13-bit unsigned integer. The offset, in 8-octet
                        units, of the data following this header,
                        relative to the start of the Fragmentable Part
                        of the original packet.

   Res   2-bit reserved field. Initialized to zero for
                        transmission; ignored on reception.

   M flag   1 = more fragments; 0 = last fragment.

   Identification   32 bits. See description below.
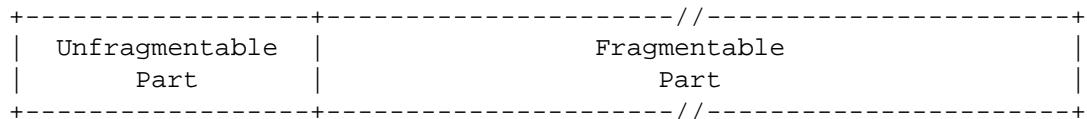
   In order to send a packet that is too large to fit in the MTU of the
   path to its destination, a source node may divide the packet into
   fragments and send each fragment as a separate packet, to be
   reassembled at the receiver.

   For every packet that is to be fragmented, the source node generates
   an Identification value. The Identification must be different than
   that of any other fragmented packet sent recently* with the same
   Source Address and Destination Address. If a Routing header is
   present, the Destination Address of concern is that of the final
   destination.

     * "recently" means within the maximum likely lifetime of a packet,
       including transit time from source to destination and time spent
       awaiting reassembly with other fragments of the same packet.
       However, it is not required that a source node know the maximum
       packet lifetime. Rather, it is assumed that the requirement can
       be met by maintaining the Identification value as a simple, 32-
       bit, "wrap-around" counter, incremented each time a packet must
       be fragmented. It is an implementation choice whether to
       maintain a single counter for the node or multiple counters,
       e.g., one for each of the node's possible source addresses, or
       one for each active (source address, destination address)
       combination.

   The initial, large, unfragmented packet is referred to as the
   "original packet", and it is considered to consist of two parts, as
   illustrated:

   original packet:

   +------------------+----------------------//-----------------------+
   | Unfragmentable | Fragmentable |
   | Part | Part |
   +------------------+----------------------//-----------------------+
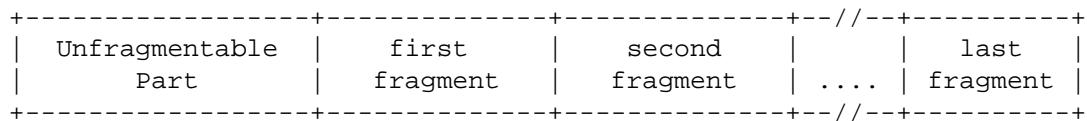
      The Unfragmentable Part consists of the IPv6 header plus any
      extension headers that must be processed by nodes en route to the
      destination, that is, all headers up to and including the Routing
      header if present, else the Hop-by-Hop Options header if present,
      else no extension headers.
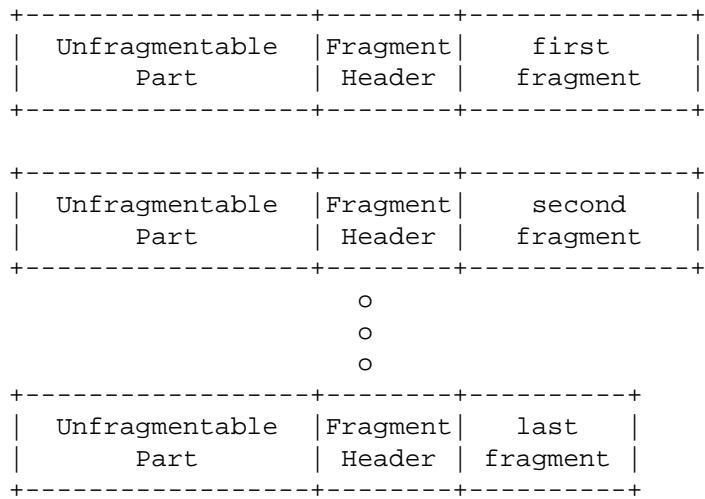
      The Fragmentable Part consists of the rest of the packet, that is,
      any extension headers that need be processed only by the final
      destination node(s), plus the upper-layer header and data.

   The Fragmentable Part of the original packet is divided into
   fragments, each, except possibly the last ("rightmost") one, being an
   integer multiple of 8 octets long. The fragments are transmitted in
   separate "fragment packets" as illustrated:

   original packet:

   +------------------+--------------+--------------+--//--+----------+
   | Unfragmentable | first | second | | last |
   | Part | fragment | fragment | .... | fragment |
   +------------------+--------------+--------------+--//--+----------+

fragment packets:

```
+------------------+--------+--------------+
|  Unfragmentable  |Fragment|    first     |
|       Part       | Header |   fragment   |
+------------------+--------+--------------+


+------------------+--------+--------------+
|  Unfragmentable  |Fragment|    second    |
|       Part       | Header |   fragment   |
+------------------+--------+--------------+
                       o
                       o
                       o
+------------------+--------+----------+
|  Unfragmentable  |Fragment|   last   |
|       Part       | Header | fragment |
+------------------+--------+----------+
```

Each fragment packet is composed of:

     (1) The Unfragmentable Part of the original packet, with the
         Payload Length of the original IPv6 header changed to contain
         the length of this fragment packet only (excluding the length
         of the IPv6 header itself), and the Next Header field of the
         last header of the Unfragmentable Part changed to 44.

     (2) A Fragment header containing:

             The Next Header value that identifies the first header of
             the Fragmentable Part of the original packet.

             A Fragment Offset containing the offset of the fragment,
             in 8-octet units, relative to the start of the
             Fragmentable Part of the original packet. The Fragment
             Offset of the first ("leftmost") fragment is 0.

             An M flag value of 0 if the fragment is the last
             ("rightmost") one, else an M flag value of 1.

             The Identification value generated for the original
             packet.

     (3) The fragment itself.

The lengths of the fragments must be chosen such that the resulting
fragment packets fit within the MTU of the path to the packets'
destination(s).

At the destination, fragment packets are reassembled into their
original, unfragmented form, as illustrated:

reassembled original packet:

```
+------------------+----------------------//-----------------------+
|  Unfragmentable  |                 Fragmentable                  |
|       Part       |                     Part                      |
+------------------+----------------------//-----------------------+
```

The following rules govern reassembly:

   An original packet is reassembled only from fragment packets that
   have the same Source Address, Destination Address, and Fragment
   Identification.

   The Unfragmentable Part of the reassembled packet consists of all
   headers up to, but not including, the Fragment header of the first
   fragment packet (that is, the packet whose Fragment Offset is
   zero), with the following two changes:

      The Next Header field of the last header of the Unfragmentable
      Part is obtained from the Next Header field of the first
      fragment's Fragment header.

      The Payload Length of the reassembled packet is computed from
      the length of the Unfragmentable Part and the length and offset
      of the last fragment. For example, a formula for computing the
      Payload Length of the reassembled original packet is:

        PL.orig = PL.first - FL.first - 8 + (8 * FO.last) + FL.last

        where
        PL.orig = Payload Length field of reassembled packet.
        PL.first = Payload Length field of first fragment packet.
        FL.first = length of fragment following Fragment header of
                    first fragment packet.
        FO.last = Fragment Offset field of Fragment header of
                    last fragment packet.
        FL.last = length of fragment following Fragment header of
                    last fragment packet.

   The Fragmentable Part of the reassembled packet is constructed
   from the fragments following the Fragment headers in each of the
   fragment packets. The length of each fragment is computed by
   subtracting from the packet's Payload Length the length of the

headers between the IPv6 header and fragment itself; its relative
position in Fragmentable Part is computed from its Fragment Offset
value.

The Fragment header is not present in the final, reassembled
packet.

The following error conditions may arise when reassembling fragmented
packets:

If insufficient fragments are received to complete reassembly of a
packet within 60 seconds of the reception of the first-arriving
fragment of that packet, reassembly of that packet must be
abandoned and all the fragments that have been received for that
packet must be discarded. If the first fragment (i.e., the one
with a Fragment Offset of zero) has been received, an ICMP Time
Exceeded -- Fragment Reassembly Time Exceeded message should be
sent to the source of that fragment.

If the length of a fragment, as derived from the fragment packet's
Payload Length field, is not a multiple of 8 octets and the M flag
of that fragment is 1, then that fragment must be discarded and an
ICMP Parameter Problem, Code 0, message should be sent to the
source of the fragment, pointing to the Payload Length field of
the fragment packet.

If the length and offset of a fragment are such that the Payload
Length of the packet reassembled from that fragment would exceed
65,535 octets, then that fragment must be discarded and an ICMP
Parameter Problem, Code 0, message should be sent to the source of
the fragment, pointing to the Fragment Offset field of the
fragment packet.
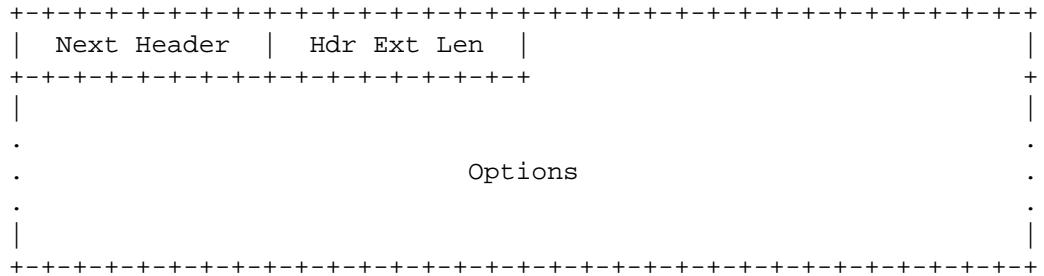
The following conditions are not expected to occur, but are not
considered errors if they do:

The number and content of the headers preceding the Fragment
header of different fragments of the same original packet may
differ. Whatever headers are present, preceding the Fragment
header in each fragment packet, are processed when the packets
arrive, prior to queueing the fragments for reassembly. Only
those headers in the Offset zero fragment packet are retained in
the reassembled packet.

The Next Header values in the Fragment headers of different
fragments of the same original packet may differ. Only the value
from the Offset zero fragment packet is used for reassembly.

4.6 Destination Options Header

   The Destination Options header is used to carry optional information
   that need be examined only by a packet's destination node(s). The
   Destination Options header is identified by a Next Header value of 60
   in the immediately preceding header, and has the following format:

```
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Next Header  |  Hdr Ext Len  |                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
     |                                                               |
     .                                                               .
     .                            Options                            .
     .                                                               .
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Next Header   8-bit selector. Identifies the type of header
                          immediately following the Destination Options
                          header. Uses the same values as the IPv4
                          Protocol field [RFC-1700 et seq.].

   Hdr Ext Len   8-bit unsigned integer. Length of the
                          Destination Options header in 8-octet units, not
                          including the first 8 octets.

   Options   Variable-length field, of length such that the
                          complete Destination Options header is an
                          integer multiple of 8 octets long. Contains one
                          or more TLV-encoded options, as described in
                          section 4.2.

   The only destination options defined in this document are the Pad1
   and PadN options specified in section 4.2.

   Note that there are two possible ways to encode optional destination
   information in an IPv6 packet: either as an option in the Destination
   Options header, or as a separate extension header. The Fragment
   header and the Authentication header are examples of the latter
   approach. Which approach can be used depends on what action is
   desired of a destination node that does not understand the optional
   information:

      o If the desired action is for the destination node to discard
         the packet and, only if the packet's Destination Address is not
         a multicast address, send an ICMP Unrecognized Type message to
         the packet's Source Address, then the information may be
         encoded either as a separate header or as an option in the

Destination Options header whose Option Type has the value 11
in its highest-order two bits. The choice may depend on such
factors as which takes fewer octets, or which yields better
alignment or more efficient parsing.

o If any other action is desired, the information must be encoded
   as an option in the Destination Options header whose Option
   Type has the value 00, 01, or 10 in its highest-order two bits,
   specifying the desired action (see section 4.2).

4.7 No Next Header

The value 59 in the Next Header field of an IPv6 header or any
extension header indicates that there is nothing following that
header. If the Payload Length field of the IPv6 header indicates the
presence of octets past the end of a header whose Next Header field
contains 59, those octets must be ignored, and passed on unchanged if
the packet is forwarded.

5. Packet Size Issues

IPv6 requires that every link in the internet have an MTU of 1280
octets or greater. On any link that cannot convey a 1280-octet
packet in one piece, link-specific fragmentation and reassembly must
be provided at a layer below IPv6.

Links that have a configurable MTU (for example, PPP links [RFC-
1661]) must be configured to have an MTU of at least 1280 octets; it
is recommended that they be configured with an MTU of 1500 octets or
greater, to accommodate possible encapsulations (i.e., tunneling)
without incurring IPv6-layer fragmentation.

From each link to which a node is directly attached, the node must be
able to accept packets as large as that link's MTU.

It is strongly recommended that IPv6 nodes implement Path MTU
Discovery [RFC-1981], in order to discover and take advantage of path
MTUs greater than 1280 octets. However, a minimal IPv6
implementation (e.g., in a boot ROM) may simply restrict itself to
sending packets no larger than 1280 octets, and omit implementation
of Path MTU Discovery.

In order to send a packet larger than a path's MTU, a node may use
the IPv6 Fragment header to fragment the packet at the source and
have it reassembled at the destination(s). However, the use of such
fragmentation is discouraged in any application that is able to
adjust its packets to fit the measured path MTU (i.e., down to 1280
octets).

A node must be able to accept a fragmented packet that, after
reassembly, is as large as 1500 octets. A node is permitted to
accept fragmented packets that reassemble to more than 1500 octets.
An upper-layer protocol or application that depends on IPv6
fragmentation to send packets larger than the MTU of a path should
not send packets larger than 1500 octets unless it has assurance that
the destination is capable of reassembling packets of that larger
size.

In response to an IPv6 packet that is sent to an IPv4 destination
(i.e., a packet that undergoes translation from IPv6 to IPv4), the
originating IPv6 node may receive an ICMP Packet Too Big message
reporting a Next-Hop MTU less than 1280. In that case, the IPv6 node
is not required to reduce the size of subsequent packets to less than
1280, but must include a Fragment header in those packets so that the
IPv6-to-IPv4 translating router can obtain a suitable Identification
value to use in resulting IPv4 fragments. Note that this means the
payload may have to be reduced to 1232 octets (1280 minus 40 for the
IPv6 header and 8 for the Fragment header), and smaller still if
additional extension headers are used.

6. Flow Labels

The 20-bit Flow Label field in the IPv6 header 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. This aspect of IPv6 is, at the time of
writing, still experimental and subject to change as the requirements
for flow support in the Internet become clearer. Hosts or routers
that do not support the functions of the Flow Label field are
required to set the field to zero when originating a packet, pass the
field on unchanged when forwarding a packet, and ignore the field
when receiving a packet.

Appendix A describes the current intended semantics and usage of the
Flow Label field.

7. Traffic Classes

The 8-bit Traffic Class field in the IPv6 header is available for use
by originating nodes and/or forwarding routers to identify and
distinguish between different classes or priorities of IPv6 packets.
At the point in time at which this specification is being written,
there are a number of experiments underway in the use of the IPv4
Type of Service and/or Precedence bits to provide various forms of
"differentiated service" for IP packets, other than through the use
of explicit flow set-up. The Traffic Class field in the IPv6 header
is intended to allow similar functionality to be supported in IPv6.

It is hoped that those experiments will eventually lead to agreement
on what sorts of traffic classifications are most useful for IP
packets. Detailed definitions of the syntax and semantics of all or
some of the IPv6 Traffic Class bits, whether experimental or intended
for eventual standardization, are to be provided in separate
documents.
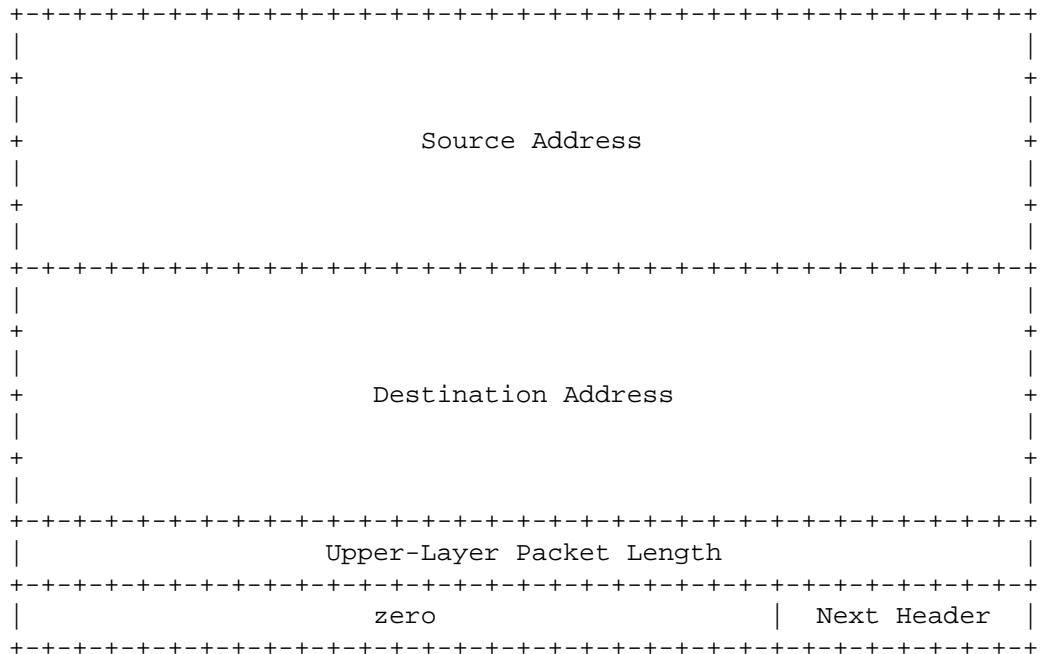
The following general requirements apply to the Traffic Class field:

   o The service interface to the IPv6 service within a node must
      provide a means for an upper-layer protocol to supply the value
      of the Traffic Class bits in packets originated by that upper-
      layer protocol. The default value must be zero for all 8 bits.

   o Nodes that support a specific (experimental or eventual
      standard) use of some or all of the Traffic Class bits are
      permitted to change the value of those bits in packets that
      they originate, forward, or receive, as required for that
      specific use. Nodes should ignore and leave unchanged any bits
      of the Traffic Class field for which they do not support a
      specific use.

   o An upper-layer protocol must not assume that the value of the
      Traffic Class bits in a received packet are the same as the
      value sent by the packet's source.

8. Upper-Layer Protocol Issues

8.1 Upper-Layer Checksums

   Any transport or other upper-layer protocol that includes the
   addresses from the IP header in its checksum computation must be
   modified for use over IPv6, to include the 128-bit IPv6 addresses
   instead of 32-bit IPv4 addresses. In particular, the following
   illustration shows the TCP and UDP "pseudo-header" for IPv6:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                                                               |
   +                         Source Address                        +
   |                                                               |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                                                               +
   |                                                               |
   +                      Destination Address                      +
   |                                                               |
   +                                                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Upper-Layer Packet Length                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      zero                     |  Next Header  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

      o If the IPv6 packet contains a Routing header, the Destination
         Address used in the pseudo-header is that of the final
         destination. At the originating node, that address will be in
         the last element of the Routing header; at the recipient(s),
         that address will be in the Destination Address field of the
         IPv6 header.

      o The Next Header value in the pseudo-header identifies the
         upper-layer protocol (e.g., 6 for TCP, or 17 for UDP). It will
         differ from the Next Header value in the IPv6 header if there
         are extension headers between the IPv6 header and the upper-
         layer header.

      o The Upper-Layer Packet Length in the pseudo-header is the
         length of the upper-layer header and data (e.g., TCP header
         plus TCP data). Some upper-layer protocols carry their own

length information (e.g., the Length field in the UDP header);
for such protocols, that is the length used in the pseudo-
header. Other protocols (such as TCP) do not carry their own
length information, in which case the length used in the
pseudo-header is the Payload Length from the IPv6 header, minus
the length of any extension headers present between the IPv6
header and the upper-layer header.

o Unlike IPv4, when UDP packets are originated by an IPv6 node,
the UDP checksum is not optional. That is, whenever
originating a UDP packet, an IPv6 node must compute a UDP
checksum over the packet and the pseudo-header, and, if that
computation yields a result of zero, it must be changed to hex
FFFF for placement in the UDP header. IPv6 receivers must
discard UDP packets containing a zero checksum, and should log
the error.

The IPv6 version of ICMP [ICMPv6] includes the above pseudo-header in
its checksum computation; this is a change from the IPv4 version of
ICMP, which does not include a pseudo-header in its checksum. The
reason for the change is to protect ICMP from misdelivery or
corruption of those fields of the IPv6 header on which it depends,
which, unlike IPv4, are not covered by an internet-layer checksum.
The Next Header field in the pseudo-header for ICMP contains the
value 58, which identifies the IPv6 version of ICMP.

8.2 Maximum Packet Lifetime

Unlike IPv4, IPv6 nodes are not required to enforce maximum packet
lifetime. That is the reason the IPv4 "Time to Live" field was
renamed "Hop Limit" in IPv6. In practice, very few, if any, IPv4
implementations conform to the requirement that they limit packet
lifetime, so this is not a change in practice. Any upper-layer
protocol that relies on the internet layer (whether IPv4 or IPv6) to
limit packet lifetime ought to be upgraded to provide its own
mechanisms for detecting and discarding obsolete packets.

8.3 Maximum Upper-Layer Payload Size

When computing the maximum payload size available for upper-layer
data, an upper-layer protocol must take into account the larger size
of the IPv6 header relative to the IPv4 header. For example, in
IPv4, TCP's MSS option is computed as the maximum packet size (a
default value or a value learned through Path MTU Discovery) minus 40
octets (20 octets for the minimum-length IPv4 header and 20 octets
for the minimum-length TCP header). When using TCP over IPv6, the
MSS must be computed as the maximum packet size minus 60 octets,

because the minimum-length IPv6 header (i.e., an IPv6 header with no
extension headers) is 20 octets longer than a minimum-length IPv4
header.

8.4 Responding to Packets Carrying Routing Headers

When an upper-layer protocol sends one or more packets in response to
a received packet that included a Routing header, the response
packet(s) must not include a Routing header that was automatically
derived by "reversing" the received Routing header UNLESS the
integrity and authenticity of the received Source Address and Routing
header have been verified (e.g., via the use of an Authentication
header in the received packet). In other words, only the following
kinds of packets are permitted in response to a received packet
bearing a Routing header:

     o Response packets that do not carry Routing headers.

     o Response packets that carry Routing headers that were NOT
        derived by reversing the Routing header of the received packet
        (for example, a Routing header supplied by local
        configuration).

     o Response packets that carry Routing headers that were derived
        by reversing the Routing header of the received packet IF AND
        ONLY IF the integrity and authenticity of the Source Address
        and Routing header from the received packet have been verified
        by the responder.

Appendix A. Semantics and Usage of the Flow Label Field

   A flow is a sequence of packets sent from a particular source to a
   particular (unicast or multicast) destination for which the source
   desires special handling by the intervening routers. The nature of
   that special handling might be conveyed to the routers by a control
   protocol, such as a resource reservation protocol, or by information
   within the flow's packets themselves, e.g., in a hop-by-hop option.
   The details of such control protocols or options are beyond the scope
   of this document.

   There may be multiple active flows from a source to a destination, as
   well as traffic that is not associated with any flow. A flow is
   uniquely identified by the combination of a source address and a
   non-zero flow label. Packets that do not belong to a flow carry a
   flow label of zero.

   A flow label is assigned to a flow by the flow's source node. New
   flow labels must be chosen (pseudo-)randomly and uniformly from the
   range 1 to FFFFF hex. The purpose of the random allocation is to
   make any set of bits within the Flow Label field suitable for use as
   a hash key by routers, for looking up the state associated with the
   flow.

   All packets belonging to the same flow must be sent with the same
   source address, destination address, and flow label. If any of those
   packets includes a Hop-by-Hop Options header, then they all must be
   originated with the same Hop-by-Hop Options header contents
   (excluding the Next Header field of the Hop-by-Hop Options header).
   If any of those packets includes a Routing header, then they all must
   be originated with the same contents in all extension headers up to
   and including the Routing header (excluding the Next Header field in
   the Routing header). The routers or destinations are permitted, but
   not required, to verify that these conditions are satisfied. If a
   violation is detected, it should be reported to the source by an ICMP
   Parameter Problem message, Code 0, pointing to the high-order octet
   of the Flow Label field (i.e., offset 1 within the IPv6 packet).

   The maximum lifetime of any flow-handling state established along a
   flow's path must be specified as part of the description of the
   state-establishment mechanism, e.g., the resource reservation
   protocol or the flow-setup hop-by-hop option. A source must not re-
   use a flow label for a new flow within the maximum lifetime of any
   flow-handling state that might have been established for the prior
   use of that flow label.

When a node stops and restarts (e.g., as a result of a "crash"), it
must be careful not to use a flow label that it might have used for
an earlier flow whose lifetime may not have expired yet. This may be
accomplished by recording flow label usage on stable storage so that
it can be remembered across crashes, or by refraining from using any
flow labels until the maximum lifetime of any possible previously
established flows has expired. If the minimum time for rebooting the
node is known, that time can be deducted from the necessary waiting
period before starting to allocate flow labels.

There is no requirement that all, or even most, packets belong to
flows, i.e., carry non-zero flow labels. This observation is placed
here to remind protocol designers and implementors not to assume
otherwise. For example, it would be unwise to design a router whose
performance would be adequate only if most packets belonged to flows,
or to design a header compression scheme that only worked on packets
that belonged to flows.

Appendix B. Formatting Guidelines for Options
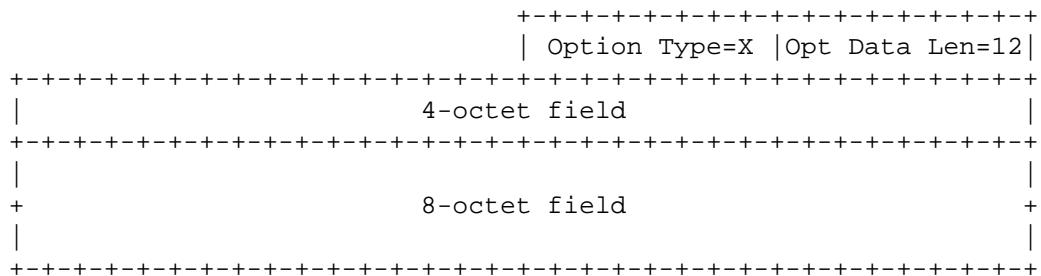
   This appendix gives some advice on how to lay out the fields when
   designing new options to be used in the Hop-by-Hop Options header or
   the Destination Options header, as described in section 4.2. These
   guidelines are based on the following assumptions:

      o One desirable feature is that any multi-octet fields within the
         Option Data area of an option be aligned on their natural
         boundaries, i.e., fields of width n octets should be placed at
         an integer multiple of n octets from the start of the Hop-by-
         Hop or Destination Options header, for n = 1, 2, 4, or 8.

      o Another desirable feature is that the Hop-by-Hop or Destination
         Options header take up as little space as possible, subject to
         the requirement that the header be an integer multiple of 8
         octets long.

      o It may be assumed that, when either of the option-bearing
         headers are present, they carry a very small number of options,
         usually only one.
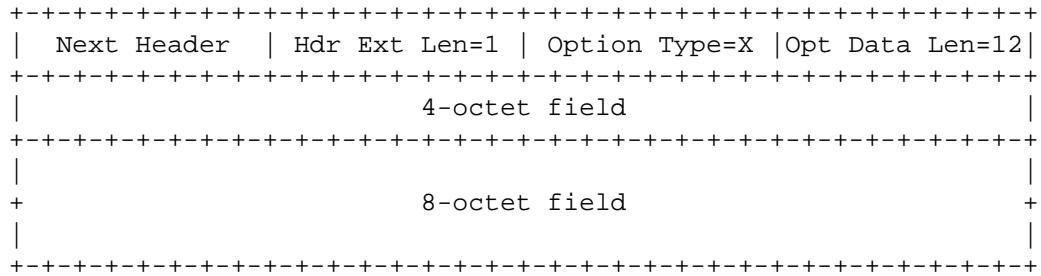
   These assumptions suggest the following approach to laying out the
   fields of an option: order the fields from smallest to largest, with
   no interior padding, then derive the alignment requirement for the
   entire option based on the alignment requirement of the largest field
   (up to a maximum alignment of 8 octets). This approach is
   illustrated in the following examples:

   Example 1

   If an option X required two data fields, one of length 8 octets and
   one of length 4 octets, it would be laid out as follows:

                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                | Option Type=X |Opt Data Len=12|
      +-+
      | 4-octet field |
      +-+
      | |
      +   8-octet field   +
      | |
      +-+
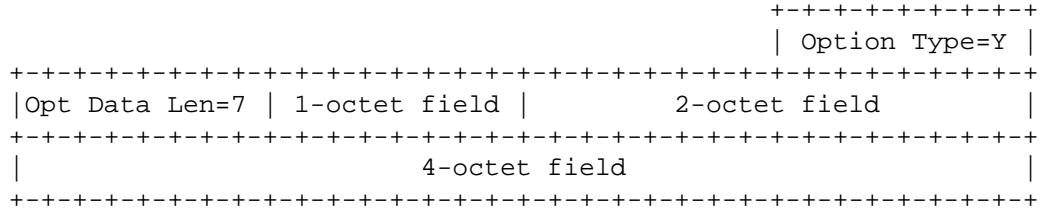
Its alignment requirement is 8n+2, to ensure that the 8-octet field
starts at a multiple-of-8 offset from the start of the enclosing
header. A complete Hop-by-Hop or Destination Options header
containing this one option would look as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=1 | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
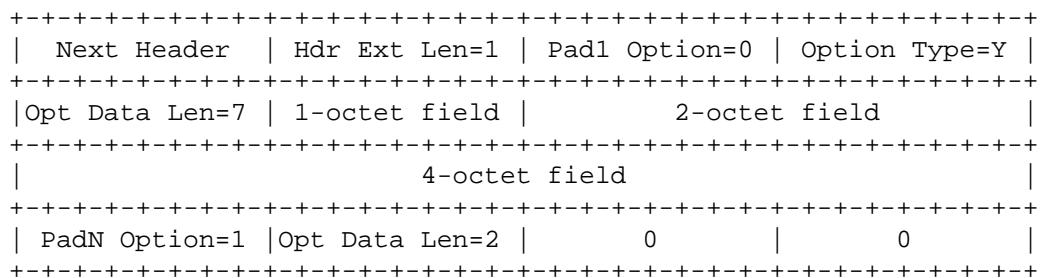
Example 2

If an option Y required three data fields, one of length 4 octets,
one of length 2 octets, and one of length 1 octet, it would be laid
out as follows:

```
                                               +-+-+-+-+-+-+-+-+
                                               | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Its alignment requirement is 4n+3, to ensure that the 4-octet field
starts at a multiple-of-4 offset from the start of the enclosing
header. A complete Hop-by-Hop or Destination Options header
containing this one option would look as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=1 | Pad1 Option=0 | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=2 |       0       |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example 3

A Hop-by-Hop or Destination Options header containing both options X
and Y from Examples 1 and 2 would have one of the two following
formats, depending on which option appeared first:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=3 | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=1 |       0       | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=2 |       0       |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  | Hdr Ext Len=3 | Pad1 Option=0 | Option Type=Y |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Opt Data Len=7 | 1-octet field |         2-octet field         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PadN Option=1 |Opt Data Len=4 |       0       |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       0       |       0       | Option Type=X |Opt Data Len=12|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4-octet field                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         8-octet field                         +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Security Considerations

   The security features of IPv6 are described in the Security
   Architecture for the Internet Protocol [RFC-2401].

Acknowledgments

   The authors gratefully acknowledge the many helpful suggestions of
   the members of the IPng working group, the End-to-End Protocols
   research group, and the Internet Community At Large.

Authors' Addresses

   Stephen E. Deering
   Cisco Systems, Inc.
   170 West Tasman Drive
   San Jose, CA 95134-1706
   USA

   Phone: +1 408 527 8213
   Fax: +1 408 527 8254
   EMail: deering@cisco.com

   Robert M. Hinden
   Nokia
   232 Java Drive
   Sunnyvale, CA 94089
   USA

   Phone: +1 408 990-2004
   Fax: +1 408 743-5677
   EMail: hinden@iprg.nokia.com

References

   [RFC-2401]   Kent, S. and R. Atkinson, "Security Architecture for the
                Internet Protocol", RFC 2401, November 1998.

   [RFC-2402]   Kent, S. and R. Atkinson, "IP Authentication Header",
                RFC 2402, November 1998.

   [RFC-2406]   Kent, S. and R. Atkinson, "IP Encapsulating Security
                Protocol (ESP)", RFC 2406, November 1998.

   [ICMPv6]   Conta, A. and S. Deering, "ICMP for the Internet
                Protocol Version 6 (IPv6)", RFC 2463, December 1998.

   [ADDRARCH]   Hinden, R. and S. Deering, "IP Version 6 Addressing
                 Architecture", RFC 2373, July 1998.

   [RFC-1981]   McCann, J., Mogul, J. and S. Deering, "Path MTU
                 Discovery for IP version 6", RFC 1981, August 1996.

   [RFC-791]   Postel, J., "Internet Protocol", STD 5, RFC 791,
                 September 1981.

   [RFC-1700]   Reynolds, J. and J. Postel, "Assigned Numbers", STD 2,
                 RFC 1700, October 1994. See also:
                 http://www.iana.org/numbers.html

   [RFC-1661]   Simpson, W., "The Point-to-Point Protocol (PPP)", STD
                 51, RFC 1661, July 1994.

CHANGES SINCE RFC-1883

   This memo has the following changes from RFC-1883. Numbers identify
   the Internet-Draft version in which the change was made.

     02) Removed all references to jumbograms and the Jumbo Payload
         option (moved to a separate document).

     02) Moved most of Flow Label description from section 6 to (new)
         Appendix A.

     02) In Flow Label description, now in Appendix A, corrected maximum
         Flow Label value from FFFFFF to FFFFF (i.e., one less "F") due
         to reduction of size of Flow Label field from 24 bits to 20
         bits.

     02) Renumbered (relettered?) the previous Appendix A to be Appendix
         B.

     02) Changed the wording of the Security Considerations section to
         avoid dependency loop between this spec and the IPsec specs.

     02) Updated R. Hinden's email address and company affiliation.

         ----

     01) In section 3, changed field name "Class" to "Traffic Class" and
         increased its size from 4 to 8 bits. Decreased size of Flow
         Label field from 24 to 20 bits to compensate for increase in
         Traffic Class field.

01) In section 4.1, restored the order of the Authentication Header
    and the ESP header, which were mistakenly swapped in the 00
    version of this memo.

01) In section 4.4, deleted the Strict/Loose Bit Map field and the
    strict routing functionality from the Type 0 Routing header, and
    removed the restriction on number of addresses that may be
    carried in the Type 0 Routing header (was limited to 23
    addresses, because of the size of the strict/loose bit map).

01) In section 5, changed the minimum IPv6 MTU from 576 to 1280
    octets, and added a recommendation that links with configurable
    MTU (e.g., PPP links) be configured to have an MTU of at least
    1500 octets.

01) In section 5, deleted the requirement that a node must not send
    fragmented packets that reassemble to more than 1500 octets
    without knowledge of the destination reassembly buffer size, and
    replaced it with a recommendation that upper-layer protocols or
    applications should not do that.

01) Replaced reference to the IPv4 Path MTU Discovery spec (RFC-
    1191) with reference to the IPv6 Path MTU Discovery spec (RFC-
    1981), and deleted the Notes at the end of section 5 regarding
    Path MTU Discovery, since those details are now covered by RFC-
    1981.

01) In section 6, deleted specification of "opportunistic" flow
    set-up, and removed all references to the 6-second maximum
    lifetime for opportunistically established flow state.

01) In section 7, deleted the provisional description of the
    internal structure and semantics of the Traffic Class field, and
    specified that such descriptions be provided in separate
    documents.

        ----

00) In section 4, corrected the Code value to indicate "unrecognized
    Next Header type encountered" in an ICMP Parameter Problem
    message (changed from 2 to 1).

00) In the description of the Payload Length field in section 3, and
    of the Jumbo Payload Length field in section 4.3, made it
    clearer that extension headers are included in the payload
    length count.

00) In section 4.1, swapped the order of the Authentication header
    and the ESP header. (NOTE: this was a mistake, and the change
    was undone in version 01.)

00) In section 4.2, made it clearer that options are identified by
    the full 8-bit Option Type, not by the low-order 5 bits of an
    Option Type. Also specified that the same Option Type numbering
    space is used for both Hop-by-Hop Options and Destination
    Options headers.

00) In section 4.4, added a sentence requiring that nodes processing
    a Routing header must send an ICMP Packet Too Big message in
    response to a packet that is too big to fit in the next hop link
    (rather than, say, performing fragmentation).

00) Changed the name of the IPv6 Priority field to "Class", and
    replaced the previous description of Priority in section 7 with
    a description of the Class field. Also, excluded this field
    from the set of fields that must remain the same for all packets
    in the same flow, as specified in section 6.

00) In the pseudo-header in section 8.1, changed the name of the
    "Payload Length" field to "Upper-Layer Packet Length". Also
    clarified that, in the case of protocols that carry their own
    length info (like non-jumbogram UDP), it is the upper-layer-
    derived length, not the IP-layer-derived length, that is used in
    the pseudo-header.

00) Added section 8.4, specifying that upper-layer protocols, when
    responding to a received packet that carried a Routing header,
    must not include the reverse of the Routing header in the
    response packet(s) unless the received Routing header was
    authenticated.

00) Fixed some typos and grammatical errors.

00) Authors' contact info updated.