Network Working Group
Internet-Draft
Obsoletes: 2460 (if approved)
Intended status: Standards Track
Expires: March 31, 2016

S. Deering
Retired
R. Hinden
Check Point Software

September 28, 2015

Internet Protocol, Version 6 (IPv6) Specification
draft-hinden-6man-rfc2460bis-07

Abstract

This document specifies version 6 of the Internet Protocol (IPv6), also sometimes referred to as IP Next Generation or IPng. It obsoletes RFC2460

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on March 31, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction 3
2. Terminology 4
3. IPv6 Header Format 5
4. IPv6 Extension Headers 6
4.1. Extension Header Order 9
4.2. Options 10
4.3. Hop-by-Hop Options Header 12
4.4. Routing Header 13
4.5. Fragment Header 14
4.6. Destination Options Header 21
4.7. No Next Header 22
4.8. Defining New Extention Headers and Options 22
5. Packet Size Issues 23
6. Flow Labels 24
7. Traffic Classes 25
8. Upper-Layer Protocol Issues 25
8.1. Upper-Layer Checksums 25
8.2. Maximum Packet Lifetime 27
8.3. Maximum Upper-Layer Payload Size 27
8.4. Responding to Packets Carrying Routing Headers 27
9. IANA Considerations 28
10. Security Considerations 28
11. Acknowledgments 28
12. References 28
12.1. Normative References 28
12.2. Informative References 29
Appendix A. Formatting Guidelines for Options 30
Appendix B. CHANGES SINCE RFC2460 33
Authors' Addresses 36

1. Introduction

IP version 6 (IPv6) is a new version of the Internet Protocol, designed as the successor to IP version 4 (IPv4) [RFC0791]. The changes from IPv4 to IPv6 fall primarily into the following categories:

- 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.

- 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.

- 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.

- Flow Labeling Capability

  A new capability is added to enable the labeling of sequences of packets for which the sender requests to be treated in the network as a single flow.

- 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

[I-D.hinden-6man-rfc4291bis]. The IPv6 version of ICMP, which all IPv6 implementations are required to include, is specified in [RFC4443]

Note: As this document obsoletes [RFC2460], any document referenced in this document that includes pointers to RFC2460, should be interpreted as referencing this document.

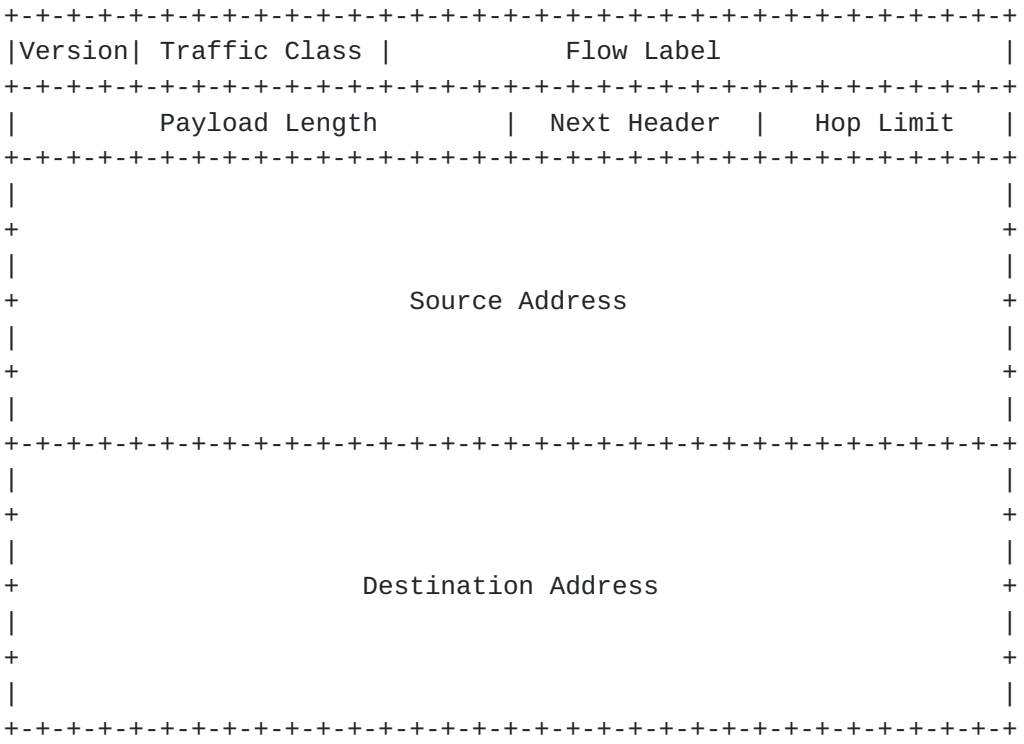
2. Terminology

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

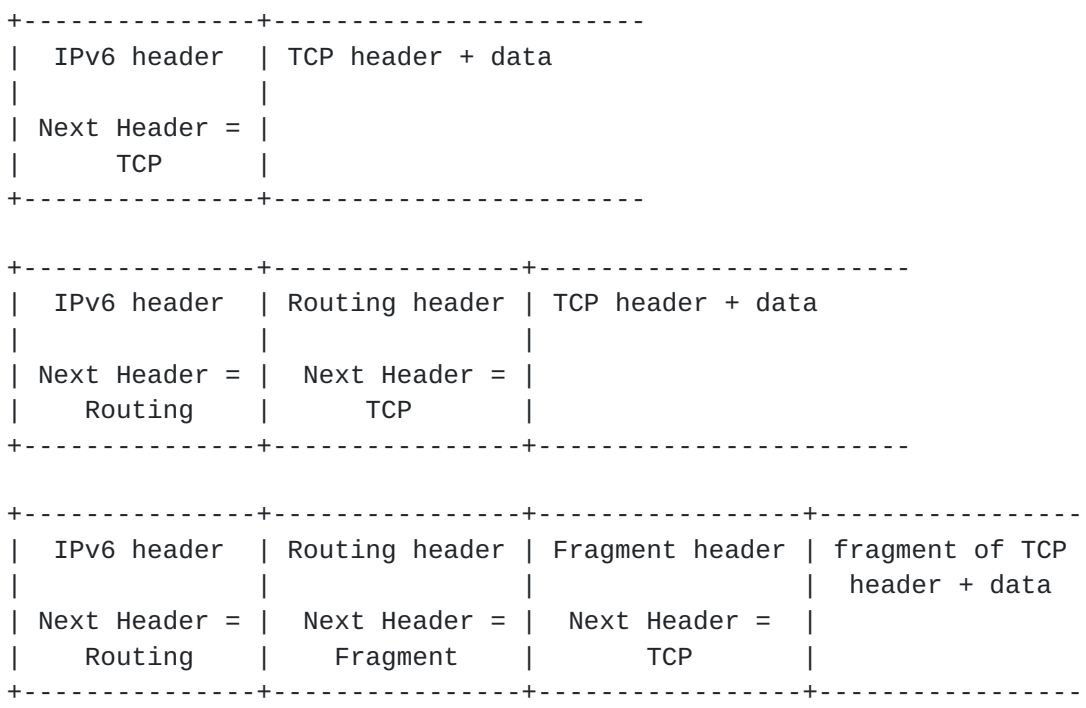


Version	4-bit Internet Protocol version number = 6.
Traffic Class	8-bit traffic class field. See section 7.
Flow Label	20-bit flow label. See section 6.
Payload Length	16-bit unsigned integer. Length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets. (Note that any 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 [IANA-PN].
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, or is received with a zero Hop Limit.
Source Address	128-bit address of the originator of the packet. See [I-D.hinden-6man-rfc4291bis].
Destination Address	128-bit address of the intended recipient of the packet (possibly not the ultimate recipient, if a Routing header is present). See [I-D.hinden-6man-rfc4291bis] 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:



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.

With one exception, extension headers are not 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. Note: If an intermediate forwarding node examines an extension header for any reason, it must do so in accordance with the provisions of [RFC7045]. At the Destination node, 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 should 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.

It should be noted that due to performance restrictions nodes may ignore the Hop-by-Hop Option header, drop packets containing a hop-by-hop option header, or assign packets containing a hop-by-hop option header to a slow processing path. Designers planning to use a hop-by-hop option need to be aware of this likely behaviour.

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
- Fragment
- Destination Options
- Authentication
- Encapsulating Security Payload

The first three are specified in this document; the last two are specified in [RFC4302] and [RFC4303], 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 [RFC4303].

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).

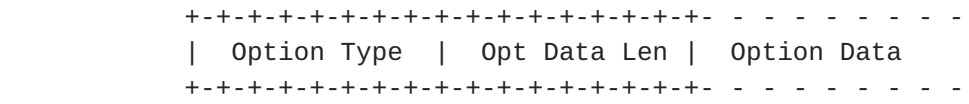
If the upper-layer header is another IPv6 header (in the case of IPv6 being tunneled over or encapsulated in IPv6), it may be followed by its own extension headers, which are separately subject to the same ordering recommendations.

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	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.

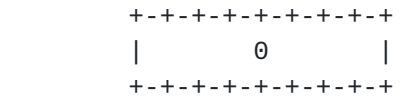
The same Option Type numbering space is used for both the Hop-by-Hop Options header and the Destination Options header. However, the specification of a particular option may restrict its use to only one of those two headers.

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:

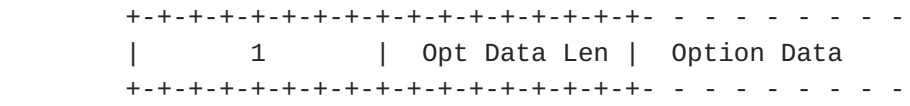
Pad1 option (alignment requirement: none)



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)

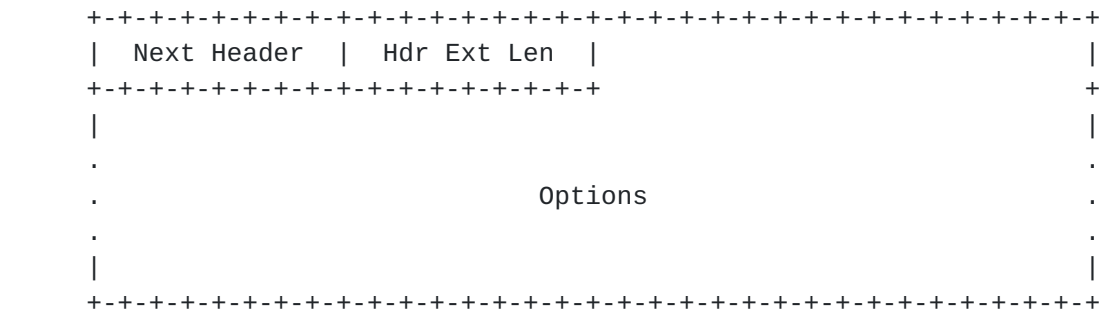


The PadN option is used to insert two or more octets of padding into the Options area of a header. For N octets of padding, the Opt Data Len field contains the value N-2, and the Option Data consists of N-2 zero-valued octets.

Appendix A 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:



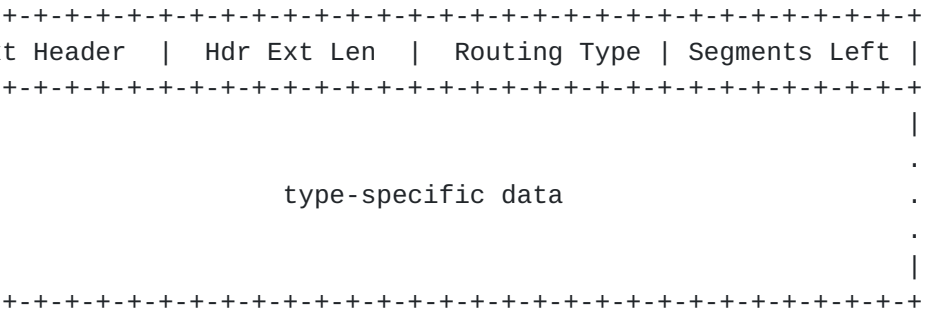
Next Header 8-bit selector. Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].

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	8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 Protocol field [IANA-PN].
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.

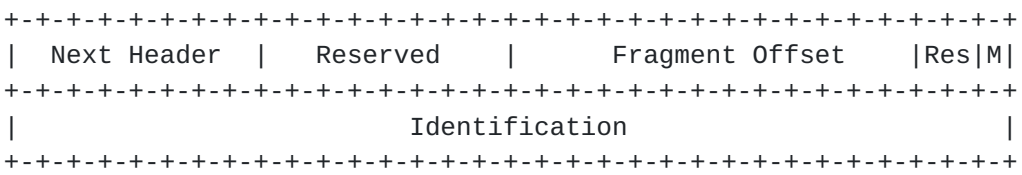
If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

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 currently defined IPv6 Routing Headers and their status can be found at [IANA-RH]. Allocation guidelines for IPv6 Routing Headers can be found in [RFC5871].

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	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 [IANA-PN].
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.

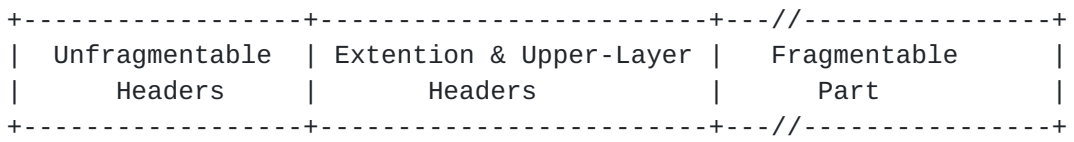
In order to send a packet that is too large to fit in the MTU of the path to its destination, a source node may divide the packet into fragments and send each fragment as a separate packet, to be reassembled at the receiver.

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 three parts, as illustrated:

original packet:



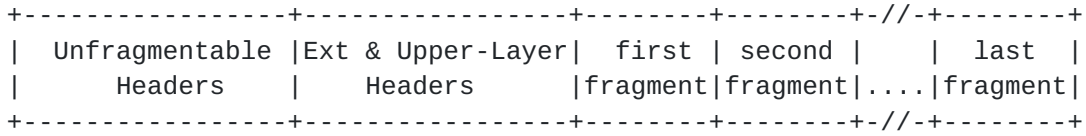
The Unfragmentable Headers consists of the IPv6 header plus any extension headers that must be processed by nodes en route to the destination, that is, all headers up to and including the Routing header if present, else the Hop-by-Hop Options header if present, else no extension headers.

The Ext Hdrs are all other extension headers that are not included in the Unfragmentable headers part of the packet. For this purpose, the IP Authentication Header (AH) and the Encapsulating Security Payload (ESP) are not considered extension headers. The Upper-Layer Header is the first upper-layer header that is not an IPv6 extension header. Examples of upper-layer headers include TCP, UDP, IPv4, IPv6, ICMPv6, and as noted AH and ESP.

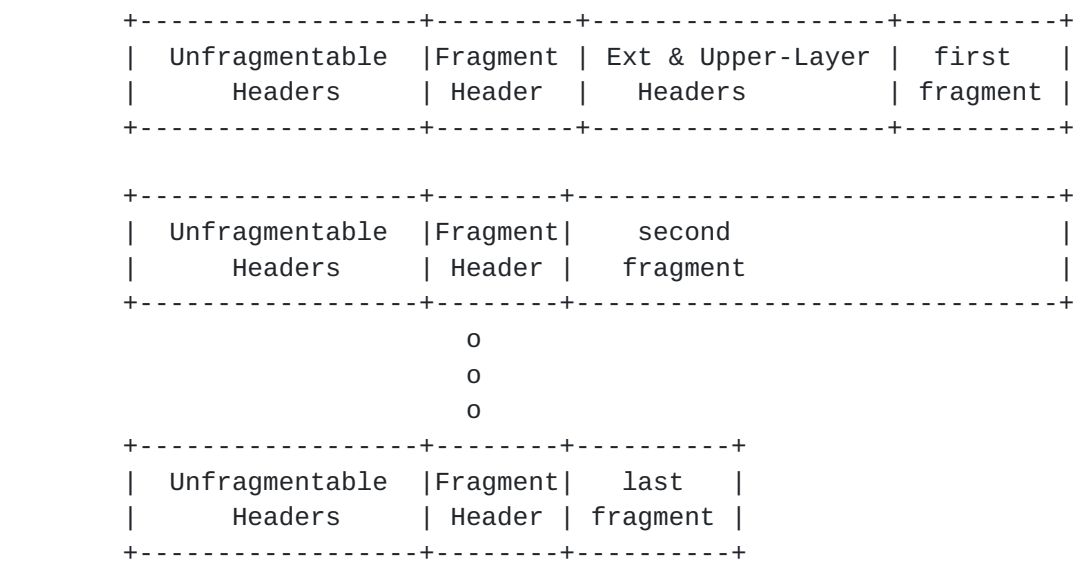
The Fragmentable Part consists of the rest of the packet after the upper-layer header.

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:



fragment packets:



The first fragment packet is composed of:

(1) The Unfragmentable Headers 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 Headers changed to 44.

(2) A Fragment header containing:

The Next Header value that identifies the first header after the Unfragmentable Headers 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 1 as this is the first fragment.

The Identification value generated for the original packet.

(3) Extension Headers, if any, and the Upper-Layer header. These headers must be in the first fragment.

(4) The first fragment.

The subsequent fragment packets are composed of:

(1) The Unfragmentable Headers 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 Headers changed to 44.

(2) A Fragment header containing:

The Next Header value that identifies the first header after the Unfragmentable Headers 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.

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.

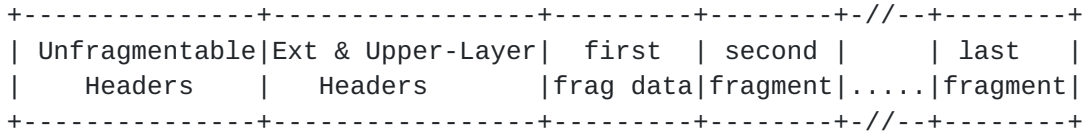
(3) The fragment itself.

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).

Fragments must not be created that overlap with any other fragments created from the original packet.

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

reassembled original packet:



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 Headers 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 Headers 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 Headers 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.

If any of the fragments being reassembled overlaps with any other fragments being reassembled for the same packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded.

If the fragment is a whole datagram (that is, both the Fragment Offset field and the M flag are zero), then it does not need any further reassembly and should be processed as a fully reassembled packet (i.e., updating Next Header, adjust Payload Length, removing the Fragmentation Header, etc.). Any other fragments that match this packet (i.e., the same IPv6 Source Address, IPv6 Destination Address, and Fragment Identification) should be processed independently.

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.

If the first fragment does not include all headers through an Upper-Layer header, then that fragment should be discarded and an ICMP Parameter Problem, Code 3, message should be sent to the source of the fragment, with the Pointer field set to zero.

The following conditions are not expected to occur, but are not considered errors if they do:

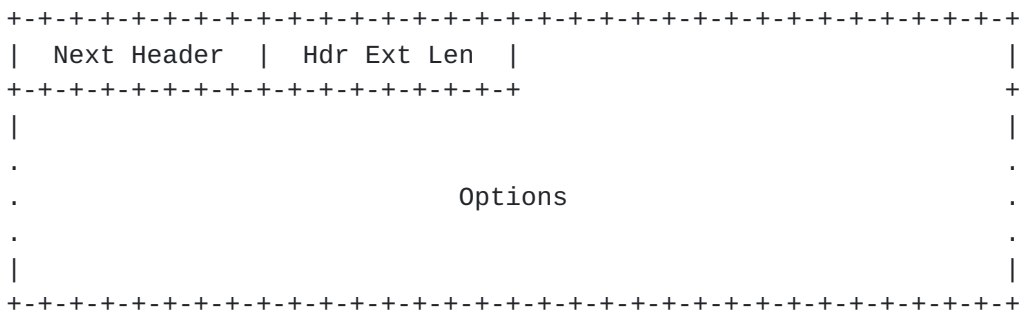
The number and content of the headers preceding the Fragment header of different fragments of the same original packet may differ. Whatever headers are present, preceding the Fragment header in each fragment packet, are processed when the packets arrive, prior to queueing the fragments for reassembly. 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	8-bit selector. Identifies the type of header immediately following the Destination Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].
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:

- 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.

- 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.

4.8. Defining New Extention Headers and Options

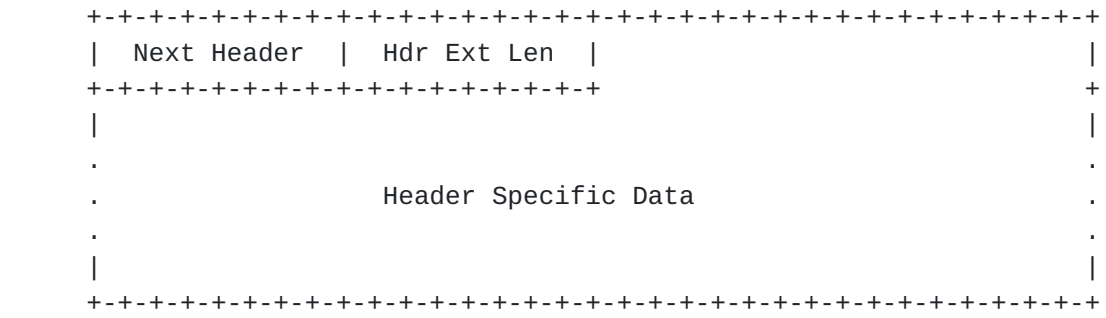
No new extension headers that require hop-by-hop behavior should be defined.

New hop-by-hop options are not recommended because, due to performance restrictions, nodes may ignore the Hop-by-Hop Option header, drop packets containing a hop-by-hop header, or assign packets containing a hop-by-hop header to a slow processing path. Designers considering defining new hop-by-hop options need to be aware of this likely behaviour. There has to a very clear justification why any new hop-by-hop option is needed before it is standardized.

Instead of defining new Extension Headers, it is recommended that the Destination Options header is used to carry optional information that need be examined only by a packet's destination node(s), because they provide better handling and backward compatibility. Defining new

IPv6 extension headers is not recommended. There has to a very clear justification why any new extension header is needed before it is standardized.

If new Extension Headers are defined, they need to use the following format:



Next Header	8-bit selector. Identifies the type of header immediately following the extension header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.
Header Specific Data	Variable-length field, Fields specific to the extension header.

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 [RFC1661]) 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 [RFC1981], 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 is used by a source to label sequences of packets to be treated in the network as a single flow.

The current definition of the IPv6 Flow Label can be found in [RFC6437].

7. Traffic Classes

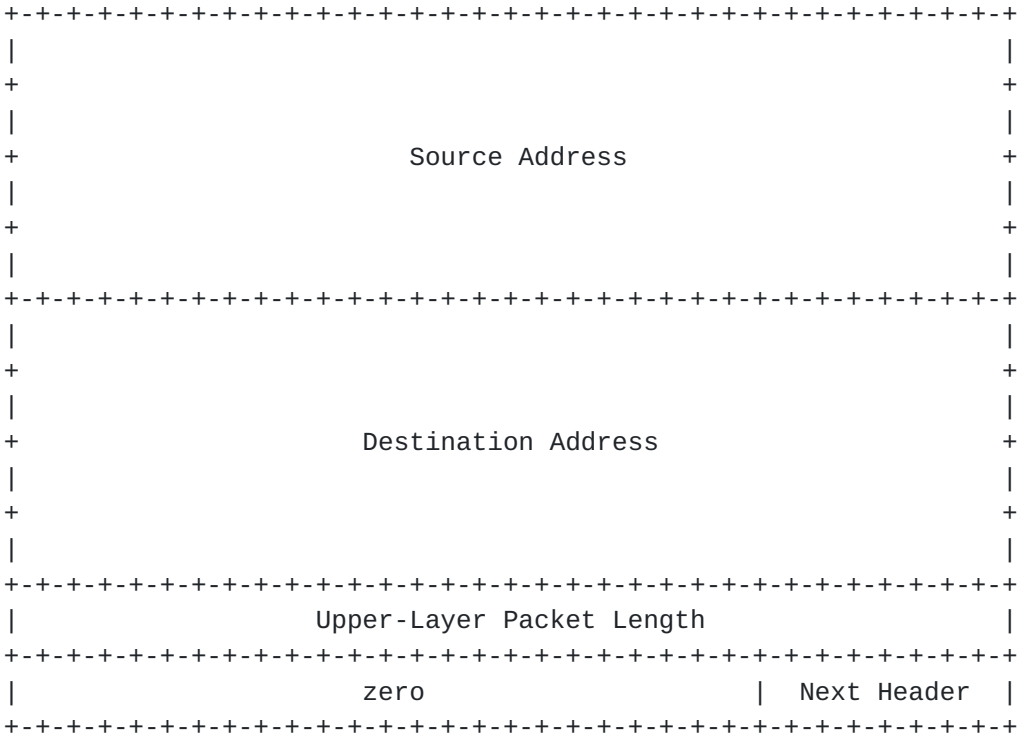
The 8-bit Traffic Class field in the IPv6 header is used by the network for traffic management. The value of the Traffic Class bits in a received packet might be different from the value sent by the packet's source.

The current use of the Traffic Class field for Differentiated Services and Explicit Congestion Notification is specified in [RFC2474] and [RFC3168].

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:



- 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.

- 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.

- 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.

- Unlike IPv4, the default behavior when UDP packets are originated by an IPv6 node, is that 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.

- As an exception to the default behaviour, protocols that use UDP as a tunnel encapsulation may enable zero-checksum mode for a specific port (or set of ports) for sending and/or receiving. Any node implementing zero-checksum mode must follow the requirements specified in "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [RFC6936].

The IPv6 version of ICMP [RFC4443] 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:

- Response packets that do not carry Routing headers.
- 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).
- 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.

9. IANA Considerations

None.

10. Security Considerations

The security features of IPv6 are described in the Security Architecture for the Internet Protocol [RFC4301].

11. 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.

12. References

12.1. Normative References

[I-D.hinden-6man-rfc4291bis] Hinden, B. and S. Deering, "IP Version 6 Addressing Architecture", draft-hinden-6man-rfc4291bis-00 (work in progress), September 2015.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <http://www.rfc-editor.org/info/rfc2474>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <http://www.rfc-editor.org/info/rfc6437>.

12.2. Informative References

[IANA-PN] "Assigned Internet Protocol Numbers", <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.

[IANA-RH] "IANA Routing Types Parameter Registry", <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-3>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, DOI 10.17487/RFC1661, July 1994, <http://www.rfc-editor.org/info/rfc1661>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[RFC5871] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the IPv6 Routing Header", RFC 5871, DOI 10.17487/RFC5871, May 2010, <http://www.rfc-editor.org/info/rfc5871>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <http://www.rfc-editor.org/info/rfc6936>.

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <http://www.rfc-editor.org/info/rfc7045>.

Appendix A. 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:

- 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.

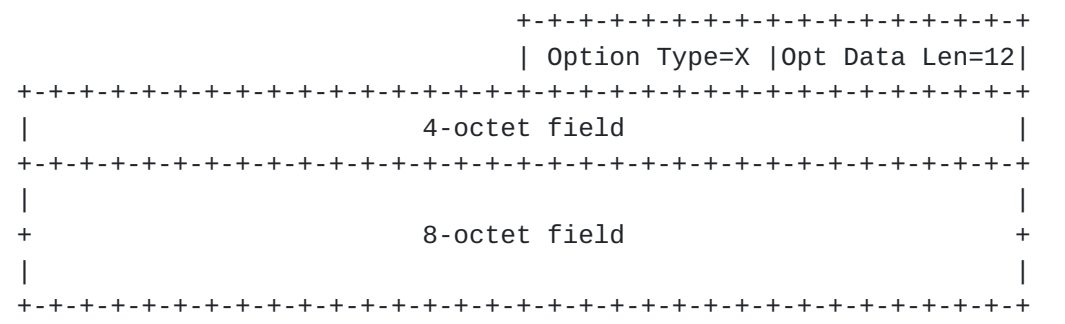
- Another desirable feature is that the Hop-by-Hop or Destination Options header take up as little space as possible, subject to the requirement that the header be an integer multiple of 8 octets long.

- 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.

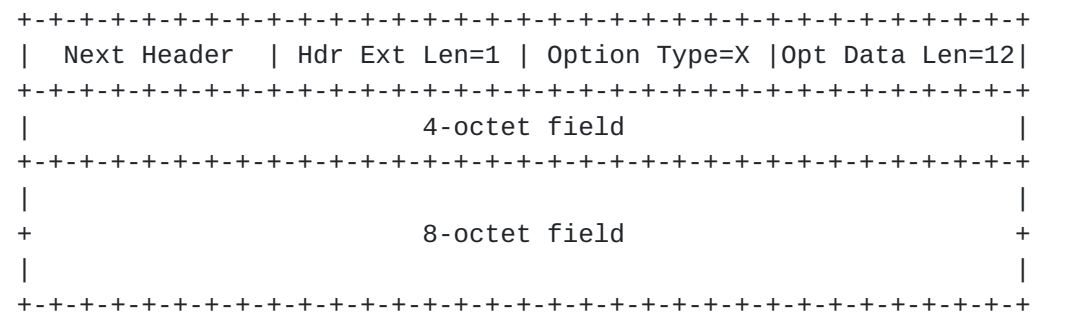
These assumptions suggest the following approach to laying out the fields of an option: order the fields from smallest to largest, with no interior padding, then derive the alignment requirement for the entire option based on the alignment requirement of the largest field (up to a maximum alignment of 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:

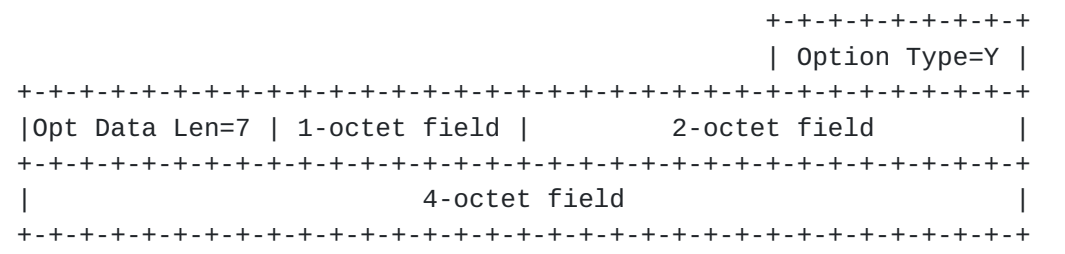


Its alignment requirement is 8n+2, to ensure that the 8-octet field starts at a multiple-of-8 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

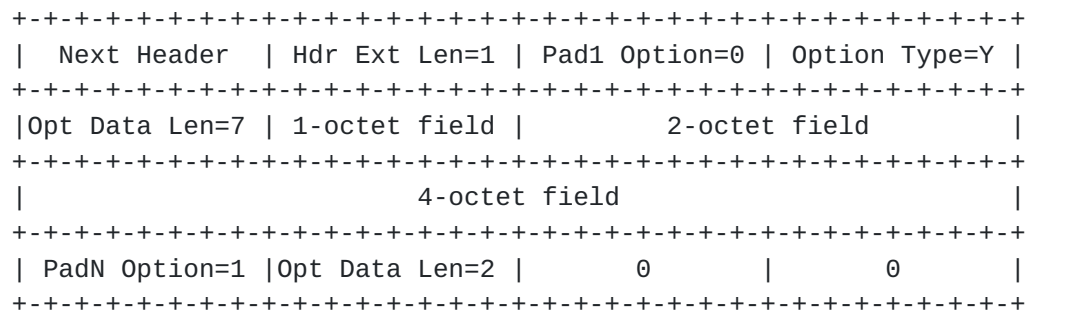


Example 2

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:

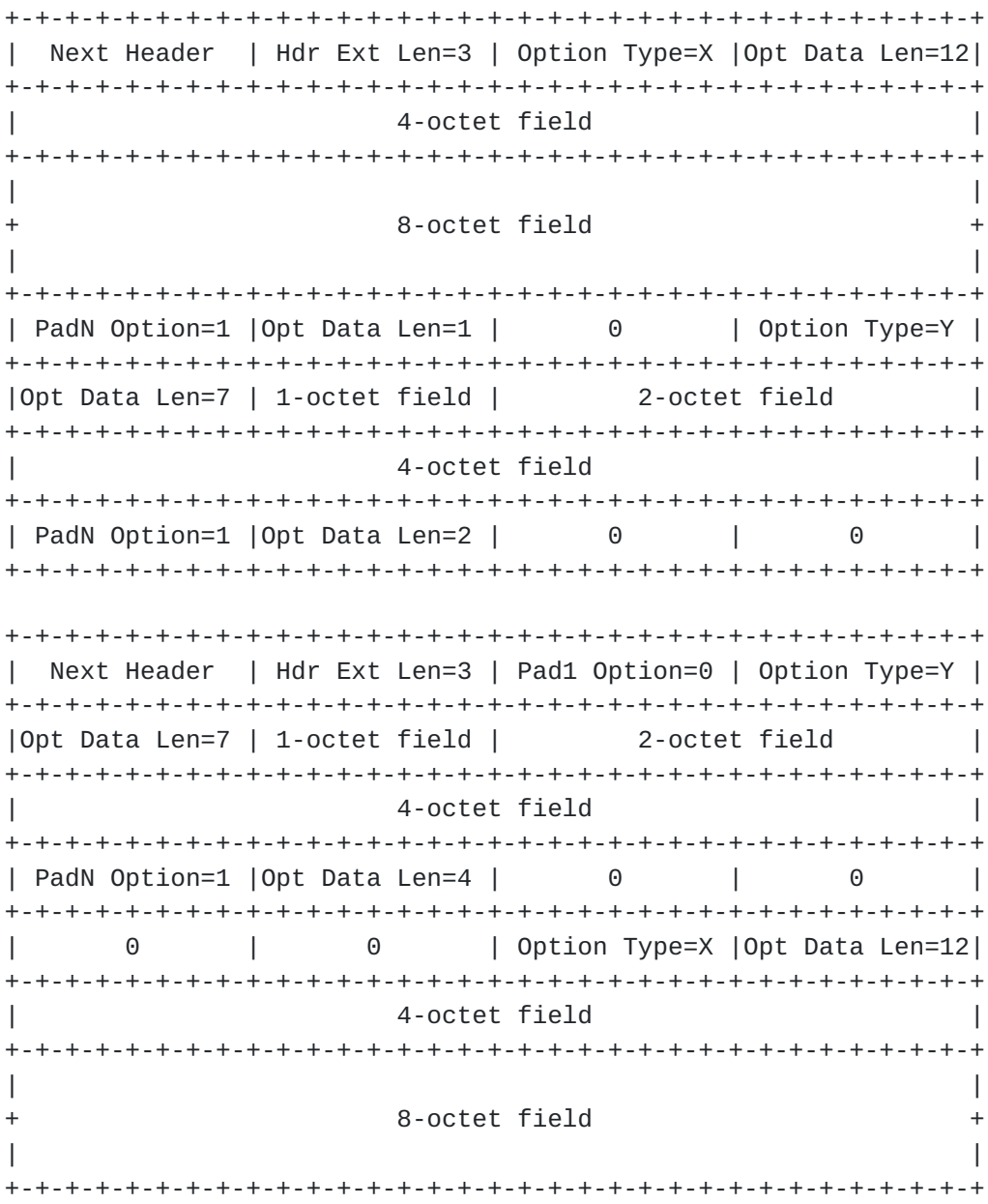


Its alignment requirement is 4n+3, to ensure that the 4-octet field starts at a multiple-of-4 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:



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:



Appendix B. CHANGES SINCE RFC2460

This memo has the following changes from RFC2460. Numbers identify the Internet-Draft version in which the change was made.

07) The purpose of this draft is to update references to current versions and assign references to normative and informative.

07) Editorial changes.

06) The purpose of this draft is to incorporate the updates dealing with Extension headers as defined in RFC6564, RFC7045, and RFC7112. The changes include:

RFC6564: Added new Section 4.8 that describe recommendations for defining new Extension headers and options

RFC7045: The changes were to add a reference to RFC7045, change the requirement for processing the hop-by-hop option to a should, and added a note that due to performance restrictions some nodes won't process the Hop-by-Hop Option header.

RFC7112: The changes were to revise the Fragmentation Section to require that all headers through the first Upper-Layer Header are in the first fragment. This changed the text describing how packets are fragmented and reassembled and added a new error case.

06) Editorial changes.

05) The purpose of this draft is to incorporate the updates dealing with fragmentation as defined in RFC5722 and RFC6946. Note: The issue relating to the handling of exact duplicate fragments identified on the mailing list is left open.

05) Fix text in the end of Section 4.0 to correct the number of extension headers defined in this document.

05) Editorial changes.

04) The purpose of this draft is to update the document to incorporate the update made by RFC6935 "UDP Checksums for Tunneled Packets".

04) Remove Routing (Type 0) header from the list of required extension headers.

04) Editorial changes.

03) The purpose of this draft is to update the document for the deprecation of the RH0 Routing Header as specified in RFC5095 and the allocations guidelines for routing headers as specified in RFC5871. Both of these RFCs updated RFC2460.

02) The purpose of this version of the draft is to update the document to resolve the open Errata on RFC2460.

Errata ID: 2541: This errata notes that RFC2460 didn't update RFC2205 when the length of the Flow Label was changed from 24 to 20 bits from RFC1883. This issue was resolved in RFC6437 where the Flow Label is defined. This draft now references RFC6437. No change is required.

Errata ID: 4279: This errata noted that the specification doesn't handle the case of a forwarding node receiving a packet with a zero Hop Limit. This is fixed in Section 3.0 of this draft. Note: No change was made regarding host behaviour.

Errata ID: 2843: This errata is marked rejected. No change is required.

02) Editorial changes to the Flow Label and Traffic Class text.

01) The purpose of this version of the draft is to update the document to point to the current specifications of the IPv6 Flow Label field as defined in [RFC6437] and the Traffic Class as defined in [RFC2474] and [RFC3168].

00) The purpose of this version is to establish a baseline from RFC2460. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC and Internet Draft, fixing a few ID Nits, and updates to the authors information. There should not be any content changes to the specification.

Authors' Addresses

Stephen E. Deering
Retired
Vancouver, British Columbia
Canada

Robert M. Hinden
Check Point Software
959 Skyway Road
San Carlos, CA 94070
USA

Email: bob.hinden@gmail.com