Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

Wikipedia
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

Languages
日本語
Русский
Edit links

IPv6 packet

From Wikipedia, the free encyclopedia

An **IPv6 packet** is the smallest message entity exchanged using Internet Protocol version 6 (IPv6).

Packets consist of control information for addressing and routing and a payload of user data. The control information in IPv6 packets is subdivided into a mandatory fixed header and optional extension headers. The payload of an IPv6 packet is typically a datagram or segment of the higher-level transport layer protocol, but may be data for an internet layer (e.g., ICMPv6) or link layer (e.g., OSPF) instead.

IPv6 packets are typically transmitted over the link layer (i.e., over Ethernet or Wi-Fi), which encapsulates each packet in a frame. Packets may also be transported over a higher-layer tunneling protocol, such as IPv4 when using 6to4 or Teredo transition technologies.

In contrast to IPv4, routers do not fragment IPv6 packets larger than the maximum transmission unit (MTU), it is the sole responsibility of the originating node. A minimum MTU of 1,280 octets is mandated by IPv6, but hosts are "strongly recommended" to use Path MTU Discovery to take advantage of MTUs greater than the minimum.[1]

Since July 2017, the Internet Assigned Numbers Authority (IANA) is responsible for registering all IPv6 parameters that are used in IPv6 packet headers.[1]

Contents [hide]

Fixed header [edit]

The fixed header starts an IPv6 packet and has a size of 40 octets (320 bits).[1]

Fixed header format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Version*				*Traffic class*								*Flow label*																			
4	32	*Payload length*																*Next header*								*Hop limit*							
8	64	*Source address*																															
12	96																																
16	128																																
20	160																																
24	192	*Destination address*																															
28	224																																
32	256																																
36	288																																

Version **(4 bits)**
The constant 6 (bit sequence `0110`).

Traffic Class **(6+2 bits)**
The bits of this field hold two values. The six most-significant bits hold the differentiated services field (DS field), which is used to classify packets.[2][3] Currently, all standard DS fields end with a '0' bit. Any DS field that ends with two '1' bits is intended for local or experimental use.[4]
The remaining two bits are used for Explicit Congestion Notification (ECN);[5] priority values subdivide into ranges: traffic where the source provides congestion control and non-congestion control traffic.

Flow Label **(20 bits)**
A high-entropy identifier of a flow of packets between a source and destination. A flow is a group of packets, e.g., a TCP session or a media stream. The special flow label 0 means the packet does not belong to any flow (using this scheme). An older scheme identifies flow by source address and port, destination address and port, protocol (value of the last *Next Header* field).[6] It has further been suggested that the flow label be used to help detect spoofed packets.[7]

Payload Length **(16 bits)**
The size of the payload in octets, including any extension headers. The length is set to zero when a *Hop-by-Hop* extension header carries a Jumbo Payload option.[8]

Next Header **(8 bits)**
Specifies the type of the next header. This field usually specifies the transport layer protocol used by a packet's payload. When extension headers are present in the packet this field indicates which extension header follows. The values are shared with those used for the IPv4 protocol field, as both fields have the same function (see List of IP protocol numbers).

Hop Limit **(8 bits)**
Replaces the time to live field in IPv4. This value is decremented by one at each forwarding node and the packet is discarded if it becomes 0. However, the destination node should process the packet normally even if received with a hop limit of 0.

Source Address **(128 bits)**
The unicast IPv6 address of the sending node.

Destination Address **(128 bits)**
The IPv6 unicast or multicast address of the destination node(s).

In order to increase performance, and since current link layer technology and transport layer protocols are assumed to provide sufficient error detection,[9] the header has no checksum to protect it.[1]

Extension headers [edit]

Extension headers carry optional internet layer information and are placed between the fixed header and the upper-layer protocol header.[1] Extension headers form a chain, using the *Next Header* fields. The *Next Header* field in the fixed header indicates the type of the first extension header; the *Next Header* field of the last extension header indicates the type of the upper-layer protocol header in the payload of the packet. All extension headers are a multiple of 8 octets in size; some extension headers require internal padding to meet this requirement.

There are several extension headers defined, and new extension headers may be defined in the future. Most extension headers are examined and processed at the packet's destination. *Hop-by-Hop Options* can be processed and modified by intermediate nodes and, if present, must be the first extension. All extension headers are optional and should only appear at most once, except for the *Destination Options* header extension, which may appear twice.[1]

If a node does not recognize a specific extension header, it should discard the packet and send a *Parameter Problem* message (ICMPv6 type 4, code 1).[1]

The defined extension headers below are listed in the preferred order for the case where there is more than one extension header following the fixed header.

Extension header	Type	Description
Hop-by-Hop Options	0	Options that need to be examined by all devices on the path
Routing	43	Methods to specify the route for a datagram (used with Mobile IPv6)
Fragment	44	Contains parameters for fragmentation of datagrams
Authentication Header (AH)	51	Contains information used to verify the authenticity of most parts of the packet
Encapsulating Security Payload (ESP)	50	Carries encrypted data for secure communication
Destination Options (before upper-layer header)	60	Options that need to be examined only by the destination of the packet
Mobility (currently without upper-layer header)	135	Parameters used with Mobile IPv6
Host Identity Protocol	139	Used for Host Identity Protocol version 2 (HIPv2)[10]
Shim6 Protocol	140	Used for Shim6[11]
Reserved	253	Used for experimentation and testing[12][4]
Reserved	254	Used for experimentation and testing[12][4]

Value 59 (No Next Header) in the Next Header field indicates that there is no next header *whatsoever* following this one, not even a header of an upper-layer protocol. It means that, from the header's point of view, the IPv6 packet ends right after it: the payload should be empty.[1] There could, however, still be data in the payload if the payload length in the first header of the packet is greater than the length of all extension headers in the packet. This data should be ignored by hosts, but passed unaltered by routers.

Hop-by-hop options and destination options [edit]

The *Hop-by-Hop Options* extension header may be examined and altered by all nodes on the packet's path, including sending and receiving nodes. (For authentication, option values that may change along the path are ignored.) The *Destination Options* extension header need to be examined by the destination node(s) only. The extension headers are both at least 8 octets in size; if more options are present than will fit in that space, blocks of 8 octets are added to the header repeatedly—containing options and padding—until all options are represented.

***Hop-by-Hop Options* and *Destination Options* extension header format**

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Next header*								*Header extension length*								*Options and padding*															
4	32	*Options and padding*																															
8	64	Optional: more *Options and padding*																															
12	96																																

Next Header **(8 bits)**

Specifies the type of the next header.

Header extension length **(8 bits)**

Length of this header in 8-octet units, not including the first 8 octets.

Options **(variable)**

Contains one or more options, and optional padding fields to align options and to make the total header length a multiple of 8 octets. Options are TLV-coded.

Routing [edit]

The *Routing* extension header is used to direct a packet to one or more intermediate nodes before being sent to its destination. The header is at least 8 octets in size; if more *Type-specific Data* is needed than will fit in 4 octets, blocks of 8 octets are added to the header repeatedly, until all *Type-specific Data* is placed.[1]

***Routing* extension header format**

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Next header*								*Header extension length*								*Routing type*								*Segments left*							
4	32	*Type-specific data*																															
8	64	Optional: more *type-specific data*...																															
12	96																																

Next header **(8 bits)**

Indicates the type of the next header.

Header extension length **(8 bits)**

The length of this header, in multiples of 8 octets, not including the first 8 octets.

Routing type **(8 bits)**

A value between 0 and 255, as assigned by IANA.[13]

Type	Status	Comment
0	Deprecated	Due to the fact that with Routing Header type 0 a simple but effective[14] denial-of-service attack could be launched, this header is deprecated since 2007[15] and host and routers are required to ignore these headers.
1	Deprecated	Used for the Nimrod[16] project funded by DARPA. It is deprecated since 2009.
2	Allowed	A limited version of type 0 and is used for Mobile IPv6, where it can hold the Home Address of the Mobile Node.
3	Allowed	RPL Source Route Header[17] for Low-Power and Lossy Networks.

253	Private use	May be used for testing, not for actual implementations. *RFC3692-style Experiment 1*.[12]
254	PrivateuUse	May be used for testing, not for actual implementations. *RFC3692-style Experiment 2*.[12]

***Segments Left* (8 bits)**

Number of nodes this packet still has to visit before reaching its final destination.

***Type-specific Data* (variable)**

Data that belongs to this type of routing header.

Fragment [edit]

In order to send a packet that is larger than the path MTU, the sending node splits the packet into fragments. The *Fragment* extension header carries the information necessary to reassemble the original (unfragmented) packet.[1]

***Fragment* extension header format**

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Next header*								*Reserved*								*Fragment offset*													*Res*		*M*
4	32	*Identification*																															

***Next header* (8 bits)**

Identifies the type of the next header.

***Reserved* (8 bits)**

Initialized to all zeroes.

***Fragment offset* (13 bits)**

Offset, in 8-octet units, relative to the start of the fragmentable part of the original packet.

***Res* (2 bits)**

Reserved; initialized to zeroes.

***M Flag* (1 bit)**

1 means more fragments follow; 0 means last fragment.

***Identification* (32 bits)**

Packet identification value, generated by the source node. Needed for reassembly of the original packet.

Authentication Header (AH) and Encapsulating Security Payload (ESP) [edit]

The *Authentication Header* and the *Encapsulating Security Payload* are part of IPsec and are used identically in IPv6 and in IPv4.[18][19]

Payload [edit]

The fixed and optional IPv6 headers are followed with the *upper-layer payload*, the data provided by the transport layer, for example a TCP segment or a UDP datagram. The *Next Header* field of the last IPv6 header indicates what type of payload is contained in this packet.

Standard payload length [edit]

The payload length field of IPv6 (and IPv4) has a size of 16 bits, capable of specifying a maximum length of 65 535 octets for the payload. In practice, hosts determine the maximum usable payload length using Path MTU Discovery (yielding the minimum MTU along the path from sender to receiver), to avoid having to fragment packets. Most Link Layer protocols have MTUs considerably smaller than 65 535 octets.

Jumbogram [edit]

An optional feature of IPv6, the *jumbo payload* option in a *Hop-By-Hop Options* extension header,[8] allows the exchange of packets with payloads of up to one octet less than 4 GB (2^{32} − 1 = 4 294 967 295 octets), by making use of a 32-bit length field. Packets with such payloads are called jumbograms.

Since both TCP and UDP include fields limited to 16 bits (length, urgent data pointer), support for IPv6 jumbograms requires modifications to the Transport Layer protocol implementation.[8] Jumbograms are only relevant for links that have a MTU larger than 65 583 octets (more than 65 535 octets for the payload, plus 40 octets for the fixed header, plus 8 octets for the *Hop-by-Hop* extension header). Only few Link Layer protocols can process packets larger than 65 535 octets.[*citation needed*]

Fragmentation [edit]

Unlike in IPv4, IPv6 routers (intermediate nodes) never fragment IPv6 packets. Packets exceeding the size of the maximum transmission unit (MTU) of the destination link are dropped and this condition is signaled by a *Packet too Big* ICMPv6 type 2 message to the originating node, similarly to the IPv4 method when the *Don't Fragment* bit is set.[1] End nodes in IPv6 are expected to perform Path MTU Discovery to determine the maximum size of packets to send, and the upper-layer protocol is expected to limit the payload size.

However, if the upper-layer protocol is unable to do so, the sending host may use the *Fragment* extension header instead. Any data link layer conveying IPv6 data must be capable of transmitting an IP packet containing up to 1,280 bytes, thus the sending endpoint may limit its packets to 1,280 bytes and avoid any need for fragmentation or Path MTU Discovery.

Fragmenting [edit]

A packet containing the first fragment of an original (larger) packet consists of five parts: the per-fragment headers (the crucial original headers that are repeatedly used in each fragment), followed by the *Fragment* extension header containing a zero Offset, then all the remaining original extension headers, then the original upper-layer header (alternatively the ESP header), and a piece of the original payload.[1]

Each subsequent packet consists of three parts: the per-fragment headers, followed by the *Fragment* extension header, and by a part of the original payload as identified by a Fragment Offset.

The per-fragment headers are determined based on whether the original contains *Routing* or *Hop-by-Hop* extension header:

- neither exists: the per-fragment part is just the fixed header
- *Routing* extension header exists: the per-fragment headers include the fixed header and all the extension headers up to and including the *Routing* one
- *Hop-by-Hop* extension header exists: the per-fragment headers consist of only the fixed header and the *Hop-by-Hop* extension header.

In any case, the last header of the per-fragment part has its *Next Header* value set to `44` to indicate that a *Fragment* extension header follows.

Each *Fragment* extension header has its *M* flag set to `1` (indicating more fragments follow), except the last, whose flag is set to `0`.

Each fragment's length is a multiple of 8 octets, except the last fragment.

The per-fragment headers were historically called the "unfragmentable part", referring to pre-2014 possibility of fragmenting the rest of headers. Now no headers are actually fragmentable.[20]

Reassembly [edit]

The original packet is reassembled by the receiving node by collecting all fragments and placing each fragment at the right offset and discarding the *Fragment* extension headers of the packets that carried them. Packets containing fragments need not arrive in sequence; they will be rearranged by the receiving node.

If not all fragments are received within 60 seconds after receiving the first packet with a fragment, reassembly of the original packet is abandoned and all fragments are discarded. If the *first* fragment was received (which contains the fixed header), a *Time Exceeded* message (ICMPv6 type 3, code 1) is returned to the node originating the fragmented packet, if the packet was discarded for this reason.

When reassembling node detects a fragment that overlaps with another fragment, the reassembly of the original packet aborts and all fragments must be silently dropped.[1] The only possible exception is that a node may optionally ignore the exact duplicates of a fragment instead of treating exact duplicates as overlapping each other.[1]

Receiving hosts must make a best-effort attempt to reassemble fragmented IP datagrams that, after reassembly, contain up to 1500 bytes. Hosts are permitted to make an attempt to reassemble fragmented datagrams larger than 1,500 bytes, but they are also permitted to silently discard any datagram after it becomes apparent that the reassembled packet would be larger than 1,500 bytes. Therefore, senders should avoid sending fragmented IP datagrams with a total reassembled size larger than 1,500 bytes, unless they have previous assurance that the receiver is capable of reassembling such large datagrams.

Security [edit]

Research has shown that the use of fragmentation can be leveraged to evade network security controls. As a result, in 2014 the earlier allowance for overflowing the IPv6 header chain beyond the first fragment became forbidden in order to avoid some very pathological fragmentation cases.[20] Additionally, as a result of research on the evasion of Router Advertisement Guard,[21] the use of fragmentation with neighbor discovery is deprecated, and the use of fragmentation with Secure Neighbor Discovery (SEND) is discouraged.[22]

References [edit]

1. ^ a b c d e f g h i j k l m n S. Deering; R. Hinden (July 2017). *Internet Protocol, Version 6 (IPv6) Specification*. Internet Engineering Task Force (IETF). doi:10.17487/RFC8200. RFC 8200. Obsoletes RFC 2460.
2. ^ K. Nickols; S. Blake; F. Baker; D. Black (December 1998). *Definition of the Differentiated Service Field (DS Field) in the IPv4 and IPv6 Headers*. doi:10.17487/RFC2474. RFC 2474.
3. ^ D. Grossman (April 2002). *New Terminology and Clarifications for DiffServ*. doi:10.17487/RFC3260. RFC 3260.
4. ^ a b c B. Fenner (November 2006). *Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers*. Network Working Group. doi:10.17487/RFC4727. RFC 4727.
5. ^ K. Ramakrishnan; S. Floyd; D. Black (September 2001). *The Addition of Explicit Congestion Notification (ECN) to IP*. doi:10.17487/RFC3168. RFC 3168.
6. ^ S. Amante; B. Carpenter; S. Jiang; J. Rajahalme (November 2011). *IPv6 Flow Label Specification*. doi:10.17487/RFC6437. RFC 6437.
7. ^ Use of the IPv6 Flow Label as a Transport-Layer Nonce to Defend Against Off-Path Spoofing Attacks
8. ^ a b c D. Borman; S. Deering; R. Hinden (August 1999). *IPv6 Jumbograms*. doi:10.17487/RFC2675. RFC 2675.
9. ^ C. Partridge; F. Kastenholz (December 1994). *Technical Criteria for Choosing IP The Next Generation (IPng)*. sec. 6.2. doi:10.17487/RFC1726. RFC 1726.
10. ^ T. Heer; P. Jokela; T. Henderson (April 2015). R. Moskowitz (ed.). *Host Identity Protocol Version 2 (HIPv2)*. Internet Engineering Task Force (IETF). doi:10.17487/RFC7401. ISSN 2070-1721. RFC 7401.
11. ^ E. Nordmark; M. Bagnulo (June 2009). *Shim6: Level 3 Multihoming Shim Protocol for IPv6*. Networking Working Group. doi:10.17487/RFC5533. RFC 5533.
12. ^ a b c d T. Narten (January 2004). *Assigning Experimental and Testing Numbers Considered Useful*. Network Working Group. doi:10.17487/RFC3692. RFC 3692. BCP 82. Updates RFC 2434.
13. ^ "Internet Protocol Version 6 (IPv6) Parameters: Routing Types". IANA. Retrieved 2017-11-17.
14. ^ Philippe Biondi, Arnoud Ebalard (April 2007). "IPv6 Routing Header Security" (PDF). EADS. Retrieved 3 December 2010. "Type 0: the evil mechanism..."
15. ^ J. Abley; P. Savola; G. Neville-Neil (December 2007). *Deprecation of Type 0 Routing Headers in IPv6*. doi:10.17487/RFC5095. RFC 5095.
16. ^ I. Castineyra; N. Chiappa; M. Steenstrup (August 1996). *The Nimrod Routing Architecture*. doi:10.17487/RFC1992. RFC 1992.
17. ^ J. Hui; JP. Vasseur; D. Culler; V. Manral (March 2012). *An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)*. Internet Engineering Task Force (IETF). doi:10.17487/RFC6554. RFC 6554.
18. ^ S. Kent (December 2005). *IP Authentication Header*. doi:10.17487/RFC4302. RFC 4302.
19. ^ S. Kent (December 2005). *IP Encapsulating Security Payload*. doi:10.17487/RFC4302. RFC 4302.
20. ^ a b F. Gont; V. Manral; R. Bonica (January 2014). *Implications of Oversized IPv6 Header Chains*. doi:10.17487/RFC7112. RFC 7112.
21. ^ F. Gont (February 2014). *Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)*. doi:10.17487/RFC7113. RFC 7113.
22. ^ F. Gont (August 2013). *Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery*. doi:10.17487/RFC6980. RFC 6980.

v · t · e **Internet Protocol version 6** [hide]

General	IPv6 · IPv6 address · **IPv6 packet** · Mobile IPv6
Deployment	IPv6 deployment · World IPv6 Day and World IPv6 Launch Day · Comparison of IPv6 support in operating systems · Comparison of IPv6 support in common applications · List of IPv6 tunnel brokers
IPv4 to IPv6 topics	IPv4 address exhaustion · IPv6 transition mechanism
Related protocols	DHCPv6 · ICMPv6 (Neighbor Discovery Protocol · Multicast Listener Discovery · Secure Neighbor Discovery · Multicast router discovery) · Site Multihoming by IPv6 Intermediation

Categories: Packets (information technology) | IPv6