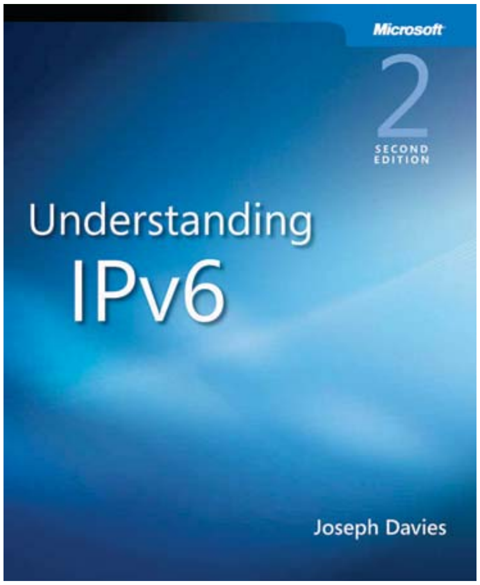

Understanding IPv6, Second Edition

Joseph Davies

To learn more about this book, visit Microsoft Learning at
http://www.microsoft.com/MSPress/books/11607.aspx

978-0-7356-2446-7

Table of Contents

---

5 ICMPv6

6 Neighbor Discovery

14 Teredo

15 IPv6 Security Considerations

16 Deploying IPv6

Appendix A

Link-Layer Support for IPv6

Appendix B

Windows Sockets Changes for IPv6

Appendix C: IPv6 RFC Index

Appendix D: Testing for Understanding Answers

Appendix E: Setting Up an IPv6 Test Lab

Appendix F: Mobile IPv6

Chapter 4

The IPv6 Header

At the end of this chapter, you should be able to do the following:

- Describe the structure of an IPv6 packet.
- List and describe the fields in the IPv4 header.
- List and describe the fields in the IPv6 header.
- Compare and contrast the fields in the IPv4 header with the fields in the IPv6 header.
- List and describe each IPv6 extension header.
- Describe the IPv6 maximum transmission unit (MTU).
- Describe the new pseudo-header used for upper-layer checksums.

Structure of an IPv6 Packet

An Internet Protocol version 6 (IPv6) packet consists of an IPv6 header, extension headers, and an upper-layer protocol data unit. Figure 4-1 shows the structure of an IPv6 packet.

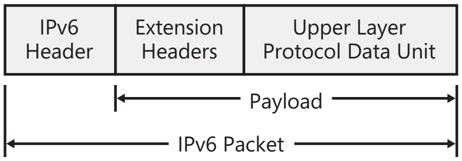

Figure 4-1 The structure of an IPv6 packet

The components of an IPv6 packet are the following:

IPv6 Header

- The IPv6 header is always present and is a fixed size of 40 bytes. The fields in the IPv6 header are described in the "IPv6 Header" section in this chapter.

Extension Headers

- Zero or more extension headers can be present and are of varying lengths. If extension headers are present, a Next Header field in the IPv6 header indicates the first extension header. Within each extension header is another Next Header field, indicating the next extension header. The last extension header indicates the header for the upper-layer protocol—such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or Internet Control Message Protocol for version 6 (ICMPv6)—contained within the upper-layer protocol data unit.

  The IPv6 header and extension headers replace the existing IPv4 header and its options. The new extension header format allows IPv6 to be enhanced to support future needs and capabilities. Unlike options in the IPv4 header, IPv6 extension headers have no maximum size and can expand to accommodate all the extension data needed for IPv6 communication. IPv6 extension headers are described in the "IPv6 Extension Headers" section in this chapter.

Upper-Layer Protocol Data Unit

- The upper-layer protocol data unit (PDU) typically consists of an upper-layer protocol header and its payload (for example, an ICMPv6 message, a TCP segment, or a UDP message).

  The IPv6 packet payload is the combination of the IPv6 extension headers and the upper-layer PDU. Normally, it can be up to 65,535 bytes long. IPv6 packets with payloads larger than 65,535 bytes in length, known as *jumbograms*, can also be sent.

IPv4 Header

Before examining the IPv6 header, you might find it helpful, for contrasting purposes, to review the IPv4 header shown in Figure 4-2.

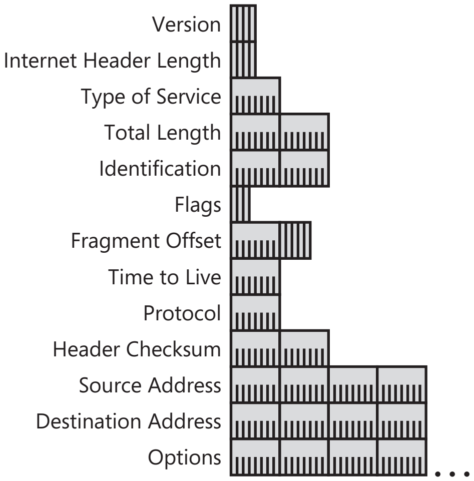

Figure 4-2 The structure of the IPv4 header

Following is a list of the fields in the IPv4 header:

Version

- The Version field indicates the version of IP and is set to 4. The size of this field is 4 bits.

Internet Header Length

- The Internet Header Length (IHL) field indicates the number of 4-byte blocks in the IPv4 header. The size of this field is 4 bits. Because an IPv4 header is a minimum of 20 bytes in size, the smallest value of the IHL field is 5. IPv4 options can extend the minimum IPv4 header size in increments of 4 bytes. If an IPv4 option is not an integral multiple of 4 bytes in length, the remaining bytes are padded with padding options, making the entire IPv4 header an integral multiple of 4 bytes. With a maximum IHL value of 0xF, the maximum size of the IPv4 header, including options, is 60 bytes (15 × 4).

Type of Service

- The Type of Service field indicates the desired service expected by this packet for delivery through routers across the IPv4 internetwork. The size of this field is 8 bits, including bits originally defined in RFC 791 for precedence, delay, throughput, reliability, and cost characteristics. RFC 2474 provides the modern definition of the high-order 6 bits of the Type of Service field as a Differentiated Services (DS) field. The DS field allows devices in a network to mark, unmark, and classify packets for forwarding. This is usually done based on the needs of an application. For example, Voice over IP and other real-time packets take precedence over e-mail in congested areas of the network. This is commonly referred to as Quality of Service (QoS). The low-order 2 bits of the Type of Service field are used for Explicit Congestion Notification (ECN), as defined in RFC 3168.

Total Length

- The Total Length field indicates the total length of the IPv4 packet (IPv4 header + IPv4 payload) and does not include link-layer framing. The size of this field is 16 bits, which can indicate an IPv4 packet that is up to 65,535 bytes long.

Identification

- The Identification field identifies this specific IPv4 packet. The size of this field is 16 bits. The Identification field is selected by the source node of the IPv4 packet. If the IPv4 packet is fragmented, all the fragments retain the Identification field value so that the destination node can group the fragments for reassembly.

Flags

- The Flags field identifies flags for the fragmentation process. The size of this field is 3 bits; however, only 2 bits are defined for current use. There are two flags—one to indicate whether the IPv4 packet can be fragmented and another to indicate whether more fragments follow the current fragment.

Fragment Offset

- The Fragment Offset field indicates the position of the fragment relative to the beginning of the original IPv4 payload. The size of this field is 13 bits.

Time-to-Live

- The Time-to-Live (TTL) field indicates the maximum number of links on which an IPv4 packet can travel before being discarded. The size of this field is 8 bits. The TTL field was originally defined as a time count for the number of seconds the packet could exist on the network. An IPv4 router determined the length of time required (in seconds) to forward the IPv4 packet and decremented the TTL accordingly. Modern routers almost always forward an IPv4 packet in less than a second, and they are required by RFC 791 to decrement the TTL by at least one. Therefore, the TTL becomes a maximum link count with the value set by the sending node. When the TTL equals 0, an ICMPv4 Time Exceeded-Time to Live Exceeded in Transit message is sent to the source of the packet and the packet is discarded.

Protocol

- The Protocol field identifies the upper-layer protocol. The size of this field is 8 bits. For example, a value of 6 in this field identifies TCP as the upper-layer protocol, a decimal value of 17 identifies UDP, and a value of 1 identifies ICMPv4. The Protocol field is used to identify the upper-layer protocol that is to receive the IPv4 packet payload.

Header Checksum

- The Header Checksum field provides a checksum on the IPv4 header only. The size of this field is 16 bits. The IPv4 payload is not included in the checksum calculation, as the IPv4 payload usually contains its own checksum. Each IPv4 node that receives IPv4 packets verifies the IPv4 header checksum and silently discards the IPv4 packet if checksum verification fails. When a router forwards an IPv4 packet, it must decrement the TTL. Therefore, the Header Checksum value is recomputed at each hop between source and destination.

Source Address

- The Source Address field stores the IPv4 address of the originating host. The size of this field is 32 bits.

Destination Address

- The Destination Address field stores the IPv4 address of an intermediate destination (in the case of source routing) or the destination host. The size of this field is 32 bits.

Options

- The Options field stores one or more IPv4 options. The size of this field is a multiple of 32 bits (4 bytes). If an IPv4 option does not use all 32 bits, padding options must be added so that the IPv4 header is an integral number of 4-byte blocks that can be indicated by the IHL field.

IPv6 Header

The IPv6 header is a streamlined version of the IPv4 header. It eliminates fields that are either unneeded or rarely used, and it adds a field that provides better support for real-time traffic. Figure 4-3 shows the structure of the IPv6 header as described in RFC 2460.

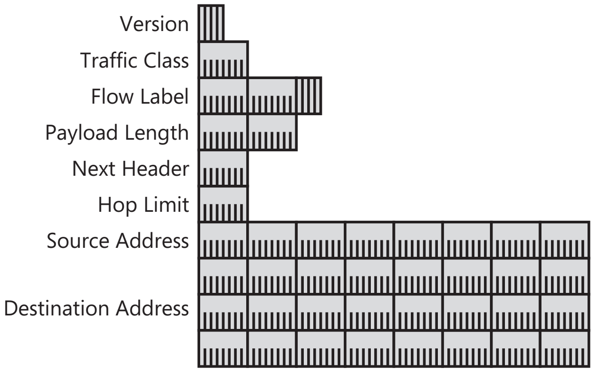

Figure 4-3 The structure of the IPv6 header

The following list describes the fields in the IPv6 header:

Version

- The Version field indicates the version of IP and is set to 6. The size of this field is 4 bits. While the purpose of the Version field is defined in the same way for both IPv4 and IPv6, its value is not used to pass the packet to an IPv4 or IPv6 protocol layer. This identification is done through a protocol identification field in the link-layer header. For example, a common link-layer encapsulation for Ethernet, called Ethernet II, uses a 16-bit EtherType field to identify the Ethernet frame payload. For IPv4 packets, the EtherType field is set to 0x800. For IPv6 packets, the EtherType field is set to 0x86DD. Thus, the determination of the protocol of the Ethernet payload occurs before the packet is passed to the appropriate protocol layer.

Traffic Class

- The Traffic Class field indicates the IPv6 packet's class or priority. The size of this field is 8 bits. This field provides functionality similar to the IPv4 Type of Service field. Like the Type of Service field in the IPv4 header, the first 6 bits of the Traffic Class field are the DS field as defined in RFC 2474 and the last 2 bits are used for ECN as defined in RFC 3168.

Flow Label

- The Flow Label field indicates that this packet belongs to a specific sequence of packets between a source and destination, requiring special handling by intermediate IPv6 routers. The size of this field is 20 bits. The flow label is used for prioritized delivery, such as delivery needed by real-time data (voice and video). For default router handling, the Flow Label field is set to 0. To distinguish a given flow, an intermediate router can use the packet's source address, destination address, and flow label. Therefore, there can be multiple flows between a source and destination, as distinguished by separate non-zero flow labels. The details of the use of the Flow Label field are described in RFC 3697.

Payload Length

- The Payload Length field indicates the length of the IPv6 payload. The size of this field is 16 bits. The Payload Length field includes the extension headers and the upper-layer PDU. With 16 bits, an IPv6 payload of up to 65,535 bytes can be indicated. For payload lengths greater than 65,535 bytes, the Payload Length field is set to 0 and the Jumbo Payload option is used in the Hop-by-Hop Options extension header, which is described in the "Hop-by-Hop Options Header" section in this chapter.

Next Header

- The Next Header field indicates either the type of the first extension header (if present) or the protocol in the upper-layer PDU (such as TCP, UDP, or ICMPv6). The size of this field is 8 bits. When indicating an upper-layer protocol, the Next Header field uses the same values that are used in the IPv4 Protocol field.

Hop Limit

- The Hop Limit field indicates the maximum number of links over which the IPv6 packet can travel before being discarded. The size of this field is 8 bits. The Hop Limit field is similar to the IPv4 TTL field, except that there is no historical relation to the amount of time (in seconds) that the packet is queued at the router. When Hop Limit equals 0 at a router, the router sends an ICMPv6 Time Exceeded-Hop Limit Exceeded in Transit message to the source and discards the packet.

Source Address

- The Source Address field indicates the IPv6 address of the originating host. The size of this field is 128 bits.

Destination Address

- The Destination Address field indicates the IPv6 address of the current destination node. The size of this field is 128 bits. In most cases, the Destination Address field is set to the final destination address. However, if a Routing extension header is present, the Destination Address field might be set to the address of the next intermediate destination.

Network Monitor CaptureHere is an example of an IPv6 header, as displayed by Network Monitor 3.1 (capture 04_01 in the \NetworkMonitorCaptures folder on the companion CD-ROM):

```
  Frame:
+ Ethernet: Etype = IPv6
- Ipv6: Next Protocol = ICMPv6, Payload Length = 40
  - Versions: IPv6, Internet Protocol, DSCP 0
     Version:  (0110............................) IPv6, Internet Protocol, 6(0x6)
     DSCP:     (....000000......................) Differentiated services codepoint 0
     ECT:      (..........0.....................) ECN-Capable Transport not set
     CE:       (...........0....................) ECN-CE not set
     FlowLabel: (............00000000000000000000) 0
    PayloadLength: 40 (0x28)
    NextProtocol: ICMPv6, 58(0x3a)
    HopLimit: 128 (0x80)
    SourceAddress: FE80:0:0:0:260:97FF:FE02:6E8F
    DestinationAddress: FE80:0:0:0:260:97FF:FE02:6D3D
+ Icmpv6: Echo request, ID = 0x0, Seq = 0x18
```

This ICMPv6 Echo Request packet uses the default Traffic Class and Flow Label and a Hop Limit of 128, and it is sent between two hosts using link-local addresses.

Values of the Next Header Field

Table 4-1 lists typical values of the Next Header field for an IPv6 header or an IPv6 extension header. Each of the IPv6 extension headers is covered later in the chapter.

Table 4-1 Typical Values of the Next Header Field

Value (Decimal)	Header
0	Hop-by-Hop Options header
6	TCP
17	UDP
41	Encapsulated IPv6 header
43	Routing header
44	Fragment header
50	Encapsulating Security Payload header
51	Authentication header
58	ICMPv6
59	No next header
60	Destination Options header

For the most current list of the reserved values for the IPv4 Protocol and IPv6 Next Header fields, see *http://www.iana.org/assignments/protocol-numbers*.

In looking at the value of the Next Header field to indicate no next header, it would seem to make more sense to set its value to 0, rather than 59. However, the designers of IPv6 wanted to optimize the processing of IPv6 packets at intermediate routers. The only extension header that must be processed at every intermediate router is the Hop-by-Hop Options header. To optimize the test of whether the Hop-by-Hop Options header is present, its Next Header value is set to 0. In router hardware, it is easier to test for a value of 0 than to test for a value of 59.

Comparing the IPv4 and IPv6 Headers

In comparing the IPv4 and IPv6 headers, you can see the following:

- The number of fields has dropped from 12 (including options) in the IPv4 header to 8 in the IPv6 header.
- The number of fields that must be processed by an intermediate router has dropped from 6 to 4, making the forwarding of normal IPv6 packets more efficient.
- Seldom-used fields such as fields supporting fragmentation and options in the IPv4 header have been moved to extension headers in the IPv6 header.
- The size of the IPv6 header has doubled from 20 bytes for a minimum-sized IPv4 header to 40 bytes. However, the new IPv6 header contains source and destination addresses that are four times longer than IPv4 source and destination addresses.

Table 4-2 lists the individual differences between the IPv4 and IPv6 header fields.

Table 4-2 IPv4 Header Fields and Corresponding IPv6 Equivalents

IPv4 Header Field	IPv6 Header Field
Version	Same field but with a different version number.
Internet Header Length	Removed in IPv6. IPv6 does not include a Header Length field because the IPv6 header is always a fixed length of 40 bytes. Each extension header is either a fixed length or indicates its own length.
Type of Service	Replaced by the IPv6 Traffic Class field.
Total Length	Replaced by the IPv6 Payload Length field, which indicates only the size of the payload.
Identification Flags Fragment Offset	Removed in IPv6. Fragmentation information is not included in the IPv6 header. It is contained in a Fragment extension header.
Time-to-Live	Replaced by the IPv6 Hop Limit field.
Protocol	Replaced by the IPv6 Next Header field.
Header Checksum	Removed in IPv6. The link layer has a checksum that performs bit-level error detection for the entire IPv6 packet.
Source Address	The field is the same except that IPv6 addresses are 128 bits in length.
Destination Address	The field is the same except that IPv6 addresses are 128 bits in length.
Options	Removed in IPv6. IPv6 extension headers replace IPv4 options.

The one new field in the IPv6 header that is not included in the IPv4 header is the Flow Label field.

The result of the new IPv6 header is a reduction in the critical router loop, the set of instructions that must be executed to determine how to forward a packet. To forward a normal IPv4 packet, a router typically performs the following in its critical router loop:

1. Verify the Header Checksum field by performing its own checksum calculation and comparing its result with the result stored in the IPv4 header. Although this step is required by RFC 1812, modern high-speed routers commonly skip it.
2. Verify the value of the Version field. Although this step is not required by RFC 791 or 1812, performing this step saves network bandwidth, as a packet containing an invalid version number is not propagated across the IPv4 internetwork only to be discarded by the destination node.
3. Decrement the value of the TTL field. If its new value is less than 1, send an ICMPv4 Time Exceeded-Time to Live Exceeded in Transit message to the source of the packet and then discard the packet. If not, place the new value in the TTL field.
4. Check for the presence of IPv4 header options. If present, process them.
5. Use the value of the Destination Address field and the contents of the local routing table to determine a forwarding interface and a next-hop IPv4 address. If a route is not found, send an ICMPv4 Destination Unreachable-Host Unreachable message to the source of the packet and discard the packet.
6. If the IPv4 MTU of the forwarding interface is less than the value of the Total Length field and the Don't Fragment (DF) flag is set to 0, perform IPv4 fragmentation. If the MTU of the forwarding interface is less than the value of the Total Length field and the DF flag is set to 1, send an ICMPv4 Destination Unreachable-Fragmentation Needed And DF Set message to the source of the packet and discard the packet.
7. Recalculate the new header checksum, and place its new value in the Header Checksum field.
8. Forward the packet by using the appropriate forwarding interface.

Note This critical router loop for IPv4 routers is a simplified list of items that an IPv4 router typically performs when forwarding. This list is not meant to imply any specific implementation nor an optimized order in which to process IPv4 packets for forwarding.

To forward a normal IPv6 packet, a router typically performs the following steps in its critical router loop:

1. Verify the value of the Version field. Although this step is not required by RFC 2460, performing it saves network bandwidth, because a packet containing an invalid version number is not propagated across the IPv6 internetwork only to be discarded by the destination node.
2. Decrement the value of the Hop Limit field. If its new value is less than 1, send an ICMPv6 Time Exceeded-Hop Limit Exceeded in Transit message to the source of the packet and discard the packet. If not, place the new value in the Hop Limit field.
3. Check the Next Header field for a value of 0. If it is 0, process the Hop-by-Hop Options header.
4. Use the value of the Destination Address field and the contents of the local routing table to determine a forwarding interface and a next-hop IPv6 address. If a route is not found, send an ICMPv6 Destination Unreachable-No Route To Destination message to the source of the packet and then discard the packet.
5. If the link MTU of the forwarding interface is less than 40 plus the value of the Payload Length field, send an ICMPv6 Packet Too Big message to the source of the packet and discard the packet.
6. Forward the packet by using the appropriate forwarding interface.

Note This critical router loop for IPv6 routers is a simplified list of items that an IPv6 router typically performs when forwarding. This list is not meant to imply any specific implementation nor an optimized order in which to process packets for forwarding.

As you can see, the process to forward an IPv6 packet is much simpler than for an IPv4 packet, as it does not have to verify and recalculate a header checksum, perform fragmentation, or process options not intended for the router.

IPv6 Extension Headers

The IPv4 header includes all options. Therefore, each intermediate router must check for their existence and process them when present. This can cause performance degradation in the forwarding of IPv4 packets. With IPv6, delivery and forwarding options are moved to extension headers. The only extension header that must be processed at each intermediate router is the Hop-by-Hop Options extension header. This increases IPv6 header processing speed and improves the performance of forwarding IPv6 packets.

RFC 2460 specifies that the following IPv6 extension headers must be supported by all IPv6 nodes:

- Hop-by-Hop Options header
- Destination Options header
- Routing header
- Fragment header
- Authentication header
- Encapsulating Security Payload header

With the exception of the Authentication header and Encapsulating Security Payload header, all the IPv6 extension headers just listed are defined in RFC 2460.

In a typical IPv6 packet, no extension headers are present. If special handling is required by either intermediate routers or the destination, the sending host adds one or more extension headers.

Each extension header must fall on a 64-bit (8-byte) boundary. Extension headers of a fixed size must be an integral multiple of 8 bytes. Extension headers of variable size contain a Header Extension Length field and must use padding as needed to ensure that their size is an integral multiple of 8 bytes.

The Next Header field in the IPv6 header and zero or more extension headers form a chain of pointers. Each pointer indicates the type of header that comes after the immediate header until the upper-layer protocol is ultimately identified. Figure 4-4 shows the chain of pointers formed by the Next Header field for various IPv6 packets.

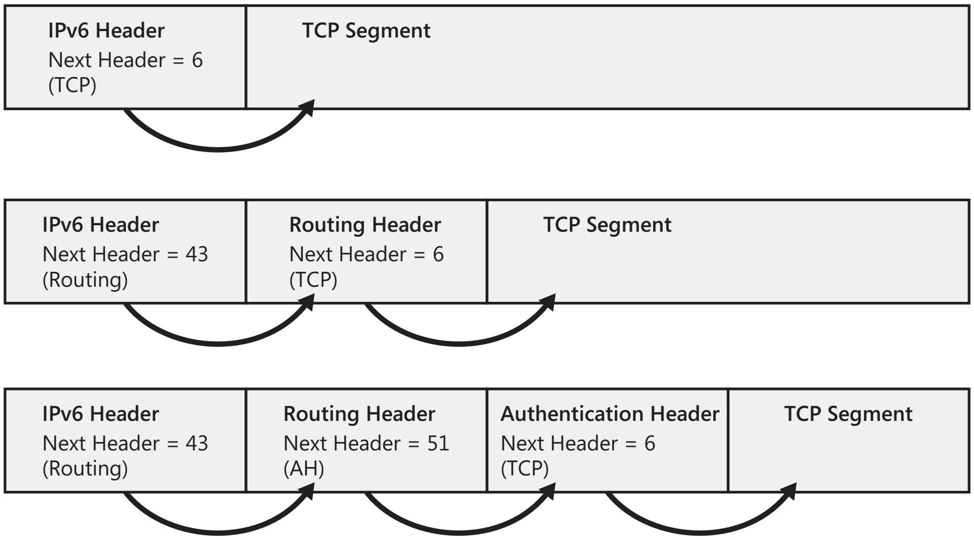

Figure 4-4 The chain of pointers formed by the Next Header field

Extension Headers Order

Extension headers are processed in the order in which they are present. Because the only extension header that is processed by every node on the path is the Hop-by-Hop Options header, it must be first. There are similar rules for other extension headers. In RFC 2460, it is recommended that extension headers be placed after the IPv6 header in the following order:

1. Hop-by-Hop Options header
2. Destination Options header (for intermediate destinations when the Routing header is present)
3. Routing header
4. Fragment header
5. Authentication header
6. Encapsulating Security Payload header
7. Destination Options header (for the final destination)

Hop-by-Hop Options Header

The Hop-by-Hop Options header is used to specify delivery parameters at each hop on the path to the destination. It is identified by the value of 0 in the IPv6 header's Next Header field. Figure 4-5 shows the structure of the Hop-by-Hop Options header.

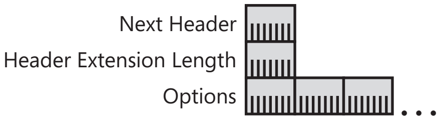

Figure 4-5 The structure of the Hop-by-Hop Options header

The Hop-by-Hop Options header consists of a Next Header field, a Header Extension Length field, and an Options field that contains one or more options. The value of the Header Extension Length field is the number of 8-byte blocks in the Hop-by-Hop Options extension header, not including the first 8 bytes. Therefore, for an 8-byte Hop-by-Hop Options header, the value of the Header Extension Length field is 0. Padding options are used to ensure 8-byte boundaries.

An IPv6 Router Optimization

The interpretation of the Header Extension Length field in the Hop-by-Hop Options header is another example of how the designers of IPv6 wanted to optimize processing of IPv6 packets at intermediate routers. For packets with a Hop-by-Hop Options header, one of the first operations is to determine the size of the header. If the Header Extension Length field were defined to be the number of 8-byte blocks in the header, its minimum value would be 1 (the minimum-sized Hop-by-Hop Options header is 8 bytes long). To ensure robustness in an IPv6 forwarding implementation, a field whose valid values begin at 1 has to be checked for the invalid value of 0 before additional processing can be done.

With the current definition of the Header Extension Length field, 0 is a valid value and no testing of invalid values needs to be done. The number of bytes in the Hop-by-Hop Options header is calculated from the following formula: (header extension length + 1) × 8.

An option is a set of fields that either describes a specific characteristic of the packet delivery or provides padding. Options are sent in the Hop-by-Hop Options header and Destination Options header (described later in this chapter). Each option is encoded in the type-length-value (TLV) format that is commonly used in TCP/IP protocols. Figure 4-6 shows the structure of an option.

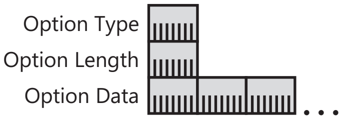

Figure 4-6 The structure of an option

The Option Type field both identifies the option and determines the way it is handled by the processing node. The Option Length field indicates the number of bytes in the option, not including the Option Type and Option Length fields. The option data is the specific data associated with the option.

An option might have an alignment requirement to ensure that specific fields within the option fall on desired boundaries. For example, it is easier to process an IPv6 address if it falls on an 8-byte boundary. Alignment requirements are expressed by using the notation $xn + y$, indicating that the option must begin at a byte boundary equal to an integral

---

multiple of *x* bytes plus *y* bytes from the start of the header. For example, the alignment requirement 4n + 2 indicates that the option must begin at a byte boundary of (an integral multiple of 4 bytes) + 2 bytes. In other words, the option must begin at the byte boundary of 6, 10, 14, and so on, relative to the start of the Hop-by-Hop Options or Destination Options headers. To accommodate alignment requirements, padding typically appears before an option and between each option when multiple options are present.

Option Type Field

Within the Option Type field, the two high-order bits indicate how the option is handled when the node processing the option does not recognize the option type. Table 4-3 lists the defined values of these two bits and their purpose.

Table 4-3 Values of the Two High-Order Bits in the Option Type Field

Value (Binary)	Action Taken
00	Skip the option.
01	Silently discard the packet.
10	Discard the packet, and send an ICMPv6 Parameter Problem message to the sender if the Destination Address field in the IPv6 header is a unicast or multicast address.
11	Discard the packet, and send an ICMPv6 Parameter Problem message to the sender if the Destination Address field in the IPv6 header is not a multicast address.

The third-highest-order bit of the Option Type indicates whether the option data can change (= 1) or not change (= 0) in the path to the destination.

Pad1 Option

The Pad1 option is defined in RFC 2460. It is used to insert a single byte of padding so that the Hop-by-Hop Options or Destination Options headers fall on 8-byte boundaries and to accommodate the alignment requirements of options. The Pad1 option has no alignment requirements. Figure 4-7 shows the Pad1 option.

Option Type
= 0

Figure 4-7 The structure of the Pad1 option

The Pad1 option consists of a single byte; Option Type is set to 0, and it has no length or value fields. With Option Type set to 0, the option is skipped if not recognized and it is not allowed to change in transit.

PadN Option

The PadN option is defined in RFC 2460. It is used to insert two or more bytes of padding so that the Hop-by-Hop Options or Destination Options headers fall on 8-byte boundaries and to accommodate the alignment requirements of options. The PadN option has no alignment requirements. Figure 4-8 shows the PadN option.

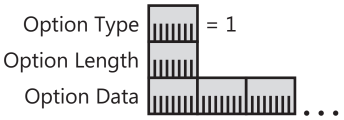

Figure 4-8 The structure of the PadN option

The PadN option consists of the Option Type field (set to 1), the Length field (set to the number of padding bytes present), and 0 or more bytes of padding. With the Option Type field set to 1, the option is skipped if not recognized and it is not allowed to change in transit.

Jumbo Payload Option

The Jumbo Payload option is defined in RFC 2675. It is used to indicate a payload size that is greater than 65,535 bytes. The Jumbo Payload option has the alignment requirement of 4n + 2. Figure 4-9 shows the Jumbo Payload option.

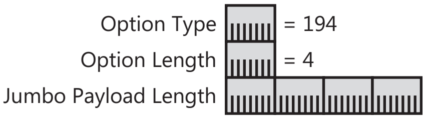

Figure 4-9 The structure of the Jumbo Payload option

With the Jumbo Payload option, the Payload Length field in the IPv6 header no longer indicates the size of the IPv6 packet payload. Instead, the Jumbo Payload Length field in the Jumbo Payload option indicates the size, in bytes, of the IPv6 packet payload. With a 32-bit Jumbo Payload Length field, payload sizes of up to 4,294,967,295 bytes can be indicated. An IPv6 packet with a payload size greater than 65,535 bytes is known as a *jumbogram*. With the Option Type field set to 194 (0xC2 hexadecimal, binary 11000010), the packet is discarded and an ICMPv6 Parameter Problem message is sent if the option is not recognized and the destination address is not a multicast address, and the option is not allowed to change in transit.

The IPv6 protocol in Windows Vista, Windows Server 2008, and Windows XP with Service Pack 2 supports incoming jumbograms at the IPv6 layer. However, there is no support in UDP or TCP for sending or receiving jumbograms.

Router Alert Option

The Router Alert option (Option Type 5) is defined in RFC 2711 and is used to indicate to a router that the contents of the packet require additional processing. The Router Alert option has the alignment requirement of 2n + 0. Figure 4-10 shows the structure of the Router Alert option.

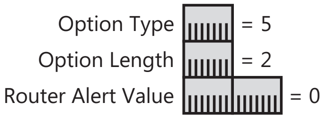

Figure 4-10 The structure of the Router Alert option

The Router Alert option is used for Multicast Listener Discovery (MLD) and the Resource ReSerVation Protocol (RSVP). With the Option Type field set to 5, the option is skipped if not recognized and it is not allowed to change in transit.

Network Monitor CaptureHere is an example of a Hop-by-Hop Options header as displayed by Network Monitor 3.1 (capture 04_02 in the \NetworkMonitorCaptures folder on the companion CD-ROM):

```
  Frame:
+ Ethernet: Etype = IPv6
- Ipv6: Next Protocol = ICMPv6, Payload Length = 32
  + Versions: IPv6, Internet Protocol, DSCP 0
    PayloadLength: 32 (0x20)
    NextProtocol: HOPOPT, IPv6 Hop-by-Hop Option, 0(0)
    HopLimit: 1 (0x1)
    SourceAddress: FE80:0:0:0:2B0:D0FF:FEE9:4143
    DestinationAddress: FF02:0:0:0:0:1:FFE9:4143
  - HopbyHopHeader:
     NextHeader: ICMPv6
     ExtHdrLen: 0(8 bytes)
   - OptionRouterAlert:
    - OptionType: Router Alert
       Action:     (00......) Skip over this option
       C:          (..0.....) Option Data does not change en-route
       OptionType: (...00101) Router Alert
      OptDataLen: 2 bytes
      Value: Datagram contains a Multicast Listener Discovery message, 0 (0x0)
   - OptionPadN:
    - OptionType: PadN
       Action:     (00......) Skip over this option
       C:          (..0.....) Option Data does not change en-route
       OptionType: (...00001) PadN
      OptDataLen: 0 bytes
      OptionData: 0 bytes
+ Icmpv6: Multicast Listener Report
```

Notice the use of the Router Alert option (option type 5) and the PadN option (option type 1) to pad the entire Hop-by-Hop Options header to 8 bytes (1-byte Next Header field + 1-byte Option Length field + 4-byte Router Alert option + 2-byte PadN option).

Destination Options Header

The Destination Options header is used to specify packet delivery parameters for either intermediate destinations or the final destination. This header is identified by the value of 60 in the previous header's Next Header field. The Destination Options header has the same structure as the Hop-by-Hop Options header, as shown in Figure 4-11.

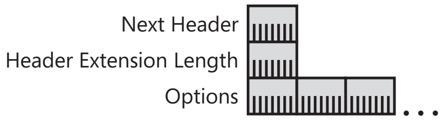

Figure 4-11 The structure of the Destination Options header

The Destination Options header is used in two ways:

1. If a Routing header is present, it specifies delivery or processing options at each intermediate destination. In this case, the Destination Options header occurs before the Routing header.
2. If no Routing header is present, or if this header occurs after the Routing header, this header specifies delivery or processing options at the final destination.

An example of a destination option is the Home Address destination option for Mobile IPv6.

Home Address Option

The Home Address destination option (Option Type 201) is defined in RFC 3775 and is used to indicate the home address of the mobile node. The home address is an address assigned to the mobile node when it is attached to the home link and through which the mobile node is always reachable, regardless of its location on an IPv6 network. For information about when the Home Address option is sent, see Appendix F, "Mobile IPv6." The Home Address option has the alignment requirement of 8n + 6. Figure 4-12 shows the structure of the Home Address option.

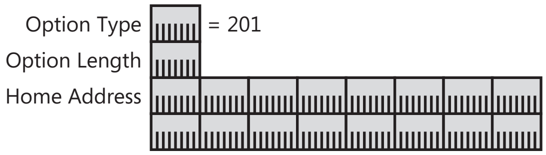

Figure 4-12 The structure of the Home Address option

The following list describes the fields in the Home Address option:

Option Type

- With the Option Type field set to 201 (0xC9 hexadecimal, 11001001 binary), the packet is discarded and an ICMPv6 Parameter Problem message is sent if the option is not recognized and the destination address is not a multicast address, and the option is not allowed to change in transit.

Option Length

- The Option Length field indicates the length of the option in bytes, not including the Option Type and Option Length fields. Because the only field past the Option Length field is the Home Address field to store an IPv6 address, the Option Length field is set to 16.

Home Address

- The Home Address field indicates the home address of the mobile node. The size of this field is 128 bits.

For an example of the Home Address option in the Destination Options header, see the Network Monitor Capture 04_03 in the \NetworkMonitorCaptures folder on the companion CD-ROM.

Summary of Option Types

Table 4-4 lists the different option types for options in Hop-by-Hop Options and Destination Options headers.

Table 4-4 Option Types

Option Type	Option and Where It Is Used	Alignment Requirement
0	Pad1 option: Hop-by-Hop and Destination Options headers	None
1	PadN option: Hop-by-Hop and Destination Options headers	None
194 (0xC2)	Jumbo Payload option: Hop-by-Hop Options header	4n + 2
5	Router Alert option: Hop-by-Hop Options header	2n + 0
201 (0xC9)	Home Address option: Destination Options header	8n + 6

Routing Header

IPv4 defines strict source routing, in which each intermediate destination must be only one hop away, and loose source routing, in which each intermediate destination can be one or more hops away. IPv6 source nodes can use the Routing header to specify a source route, which is a list of intermediate destinations for the packet to travel to on its path to the final destination. The Routing header is identified by the value of 43 in the previous header's Next Header field. Figure 4-13 shows the structure of the Routing header.

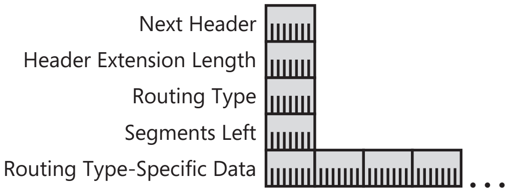

Figure 4-13 The structure of the Routing header

The Routing header consists of a Next Header field, a Header Extension Length field (defined in the same way as the Hop-by-Hop Options extension header), a Routing Type field, a Segments Left field that indicates the number of intermediate destinations that are still to be visited, and routing type-specific data.

RFC 2460 also defines Routing Type 0, used for loose source routing. Figure 4-14 shows the structure of the Routing Type 0 header.

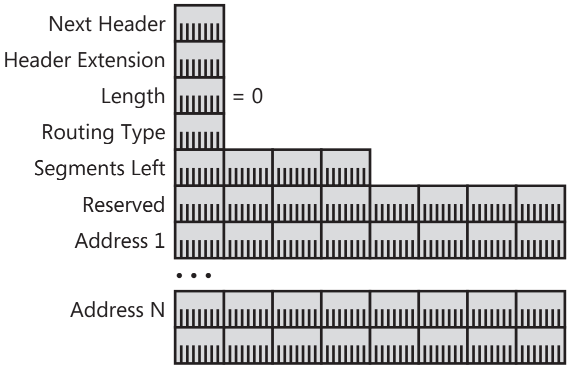

Figure 4-14 The structure of the Routing Type 0 header

For Routing Type 0, the routing type-specific data consists of a 32-bit Reserved field and a list of intermediate destination addresses, including the final destination address. When the packet is initially sent, the destination address is set to the first intermediate destination, and the routing type-specific data is the list of additional intermediate destinations and the final destination. The Segments Left field is set to the total number of addresses included in the routing type-specific data.

When the IPv6 packet reaches an intermediate destination, the Routing header is processed and the following actions are taken:

1. The current destination address and the address in the (N – Segments Left + 1) position in the list of addresses are swapped, where *N* is the total number of addresses in the Routing header.
2. The Segments Left field is decremented.
3. The packet is forwarded.

By the time the packet arrives at the final destination, the Segments Left field has been set to 0 and the list of intermediate addresses visited in the path to the destination is recorded in the Routing header.

IPv6 in Windows Vista will accept and process an incoming packet with a Routing Type 0 header. Because of security concerns, the Internet Engineering Task Force (IETF) is deprecating support for the Routing Type 0 header. IPv6 in Windows Server 2008 will silently discard an incoming packet with a Routing Type 0 header.

Note Mobile IPv6 uses a Type 2 Routing header. For more information, see Appendix F, "Mobile IPv6."

Here is an example of the Routing header as displayed by Network Monitor 3.1 (capture 04_04 in the \NetworkMonitorCaptures folder on the companion CD-ROM):

```
  Frame:
+ Ethernet: Etype = IPv6
- Ipv6: Next Protocol = ICMPv6, Payload Length = 64
  + Versions: IPv6, Internet Protocol, DSCP 0
    PayloadLength: 64 (0x40)
    NextProtocol: IPv6 Routing header, 43(0x2b)
    HopLimit: 127 (0x7F)
    SourceAddress: FEC0:0:0:2:2B0:D0FF:FEE9:4143
    DestinationAddress: FEC0:0:0:2:260:97FF:FE02:6E8F
  - RoutingHeader:
     NextHeader: ICMPv6
     ExtHdrLen: 2(24 bytes)
     RoutingType: 0 (0x0)
     SegmentsLeft: 1 (0x1)
     Reserved: 0 (0x0)
     RouteAddress: FEC0:0:0:1:260:8FF:FE52:F9D8
+ Icmpv6: Echo request, ID = 0x0, Seq = 0x3d1a
```

In this simple example of the Routing header, an ICMPv6 Echo Request message is sent from the source FEC0::2:2B0:D0FF:FEE9:4143 to the destination FEC0::1:260:8FF:FE52:F9D8 using the intermediate destination of FEC0::2:260:97FF:FE02:6E8F.

Fragment Header

The Fragment header is used for IPv6 fragmentation and reassembly services. This header is identified by the value of 44 in the previous header's Next Header field. Figure 4-15 shows the structure of the Fragment header.

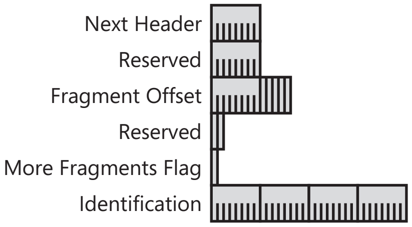

Figure 4-15 The structure of the Fragment header

The Fragment header includes a Next Header field, a 13-bit Fragment Offset field, a More Fragments flag, and a 32-bit Identification field. The Fragment Offset, More Fragments flag, and Identification fields are used in the same way as the corresponding fields in the IPv4 header. Because the use of the Fragment Offset field is defined for 8-byte fragment blocks, the Fragment header cannot be used for jumbograms. The maximum number that can be expressed with the 13-bit Fragment Offset field is 8191. Therefore, Fragment Offset can be used to indicate only a fragment data starting position of up to 8191 × 8, or 65,528.

In IPv6, only source nodes can fragment payloads. If the payload submitted by the upper-layer protocol is larger than the link or path MTU, IPv6 fragments the payload at the source and uses the Fragment header to provide reassembly information. An IPv6 router will never fragment an IPv6 packet being forwarded.

Because the IPv6 internetwork will not transparently fragment payloads, data sent from applications that do not have an awareness of the destination path MTU will not be able to sense when data needing fragmentation by the source is discarded by IPv6 routers. This can be a problem for unicast or multicast traffic sent as a UDP message or other types of message streams that do not use TCP.

Differences in Fragmentation Fields

There are some subtle differences between the fragmentation fields in IPv4 and IPv6. In IPv4, the fragmentation flags are the three high-order bits of the 16-bit quantity composed of the combination of the fragmentation flags and the Fragment Offset field. In IPv6, the bits used for fragmentation flags are the three low-order bits of the 16-bit quantity composed of the combination of the fragmentation flags and the Fragment Offset field. In IPv4, the Identification field is 16 bits rather than 32 bits in IPv6, and in IPv6 there is no Don’t Fragment flag. Because IPv6 routers never perform fragmentation, the Don’t Fragment flag is always be set to 1 for all IPv6 packets and therefore does not need to be included.

IPv6 Fragmentation Process

When an IPv6 packet is fragmented, it is initially divided into unfragmentable and fragmentable parts:

- The unfragmentable part of the original IPv6 packet must be processed by intermediate nodes between the fragmenting node and the destination. This part consists of the IPv6 header, the Hop-by-Hop Options header, the Destination Options header for intermediate destinations, and the Routing header.
- The fragmentable part of the original IPv6 packet must be processed only at the final destination node. This part consists of the Authentication header, the Encapsulating Security Payload header, the Destination Options header for the final destination, and the upper-layer PDU.

Next, the IPv6 fragment packets are formed. Each fragment packet consists of the unfragmentable part, a fragment header, and a portion of the fragmentable part. Figure 4-16 shows the IPv6 fragmentation process for a packet fragmented into three fragments.

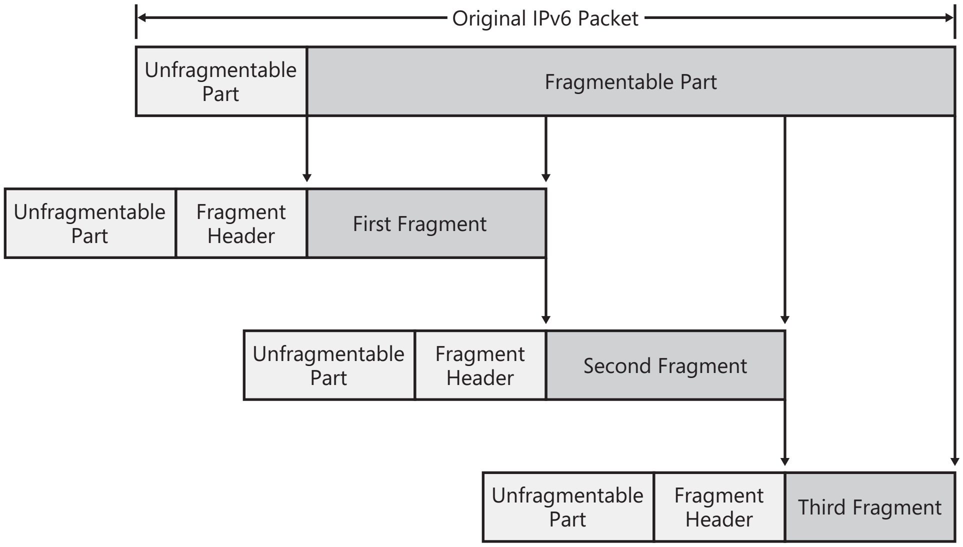

Figure 4-16 The IPv6 fragmentation process

In each fragment, the Next Header field in the Fragment header indicates the first header or the upper-layer protocol in the original fragmentable part. The Fragment Offset field in the Fragment header indicates the offset, in 8-byte units known as *fragment blocks*, of this fragment relative to the original payload. The More Fragments flag is set on all fragment packets except the last fragment packet. All fragment packets created from the same IPv6 packet must contain the same Identification field value.

Fragmentation of IPv6 packets can occur when the upper-layer protocol of the sending host submits a packet to IPv6 that is larger than the path MTU to the destination. Examples of IPv6 fragmentation are when a UDP application that is not aware of a path MTU sends large packets to a destination, or when a TCP application sends a packet before it is made aware of a path MTU update that lowers the path MTU. In this latter case, IPv6 is aware of the new path MTU, but TCP is not. TCP submits the TCP segment by using the old, larger value of the path MTU, and IPv6 fragments the TCP segment to fit the new, lower path MTU value. Once TCP is made aware of the new path MTU, subsequent TCP segments are not fragmented.

IPv6 packets sent to IPv4 destinations that undergo IPv6-to-IPv4 header translation might receive a path MTU update of less than 1280. In this case, the sending host sends IPv6 packets with a Fragment header in which the Fragment Offset field is set to 0 and the More Fragments flag is not set, and with a smaller payload size of 1272 bytes. The Fragment header is included so that the IPv6-to-IPv4 translator can use the Identification field in the Fragment header to perform IPv4 fragmentation to reach the IPv4 destination.

Here is an example of a Fragment header as displayed by Network Monitor 3.1 (frame 3 of capture 04_05 in the \NetworkMonitorCaptures folder on the companion CD-ROM):

```
  Frame:

+ Ethernet: Etype = IPv6

- Ipv6: Next Protocol = ICMPv6, Payload Length = 1456

  + Versions: IPv6, Internet Protocol, DSCP 0
```

```
    PayloadLength: 1456 (0x5B0)
    NextProtocol: IPv6 Fragment header, 44(0x2c)
    HopLimit: 128 (0x80)
    SourceAddress: FE80:0:0:0:210:5AFF:FEAA:20A2
    DestinationAddress: FE80:0:0:0:250:DAFF:FED8:C153
  - FragmentHeader:
     NextHeader: ICMPv6
     Reserved: 0 (0x0)
   - FragmentInfor:
      FragmentOffset: 2896(0XB50)
      Reserved: (.............00.)
      M:        (...............1) More fragments
      Identification: 5 (0x5)
     FragmentData: Binary Large Object (1448 Bytes)
```

This is a fragment of a payload that uses the identification number of 5 and starts in byte position 2896 relative to the fragmentable portion of the original IPv6 payload.

IPv6 Reassembly Process

The fragment packets are forwarded by the intermediate IPv6 router or routers to the destination IPv6 address. The fragment packets can take different paths to the destination and arrive in a different order in which they were sent. To reassemble the fragment packets into the original payload, IPv6 uses the Source Address and Destination Address fields in the IPv6 header and the Identification field in the Fragment header to group the fragments. Figure 4-17 shows the IPv6 reassembly process.

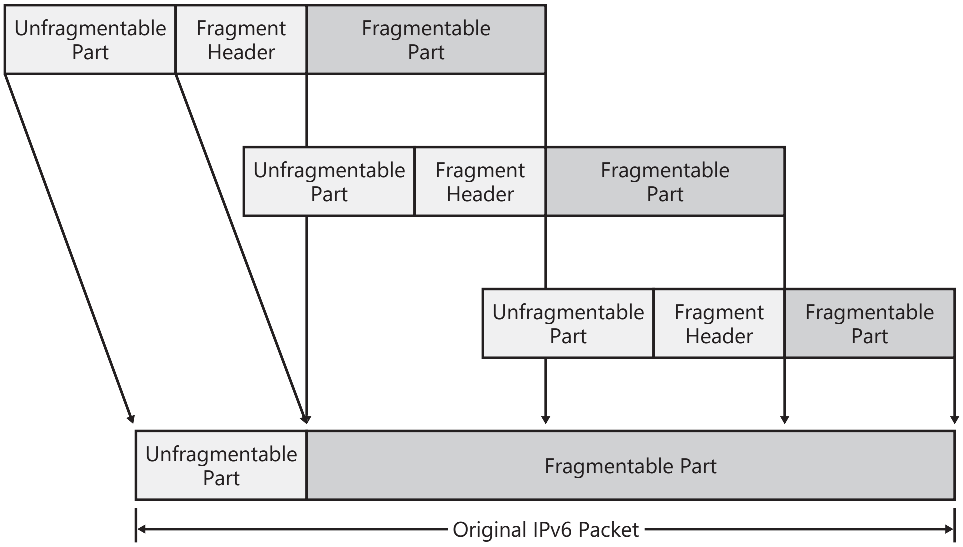

Figure 4-17 The IPv6 reassembly process

After all the fragments arrive, the original payload length is calculated and the Payload Length field in the IPv6 header for the reassembled packet is updated. Additionally, the Next Header field of the last header of the unfragmentable part is set to the Next Header field of the Fragment header of the first fragment.

RFC 2460 recommends a reassembly time of 60 seconds before abandoning reassembly and discarding the partially reassembled packet. If the first fragment has arrived and reassembly has not completed, the reassembling host sends an ICMPv6 Time Exceeded-Fragment Reassembly Time Exceeded message to the source of the fragment.

Authentication Header

The Authentication header provides data authentication (verification of the node that sent the packet), data integrity (verification that the data was not modified in transit), and antireplay protection (assurance that captured packets cannot be retransmitted and accepted as valid data) for the IPv6 packet including the fields in the IPv6 header that do not change in transit across an IPv6 internetwork. The Authentication header, described in RFC 2402, is part of the security architecture for IP, as defined in RFC 2401. The Authentication header is identified by the value of 51 in the previous header's Next Header field. Figure 4-18 shows the structure of the Authentication header.

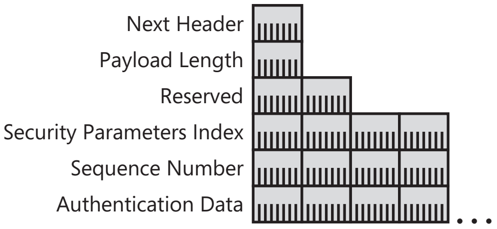

Figure 4-18 The structure of the Authentication header

The Authentication header contains a Next Header field, a Payload Length field (the number of 4-byte blocks in the Authentication header, not counting the first two), a Reserved field, a Security Parameters Index (SPI) field that helps identify a specific IP Security (IPSec) security association (SA), a Sequence Number field that provides antireplay protection, and an Authentication Data field that contains an integrity value check (ICV). The ICV provides data authentication and data integrity.

The Authentication header does not provide data confidentiality services for the upper-layer PDU by encrypting the data so that it cannot be viewed without the encryption key. To obtain data authentication and data integrity for the entire IPv6 packet and data confidentiality for the upper-layer PDU, you can use both the Authentication header and the Encapsulating Security Payload header and trailer.

Encapsulating Security Payload Header and Trailer

The Encapsulating Security Payload (ESP) header and trailer, described in RFC 2406, provide data confidentiality, data authentication, data integrity, and replay protection services to the encapsulated payload. The ESP header provides no security services for the IPv6 header or extension headers that occur before the ESP header. The ESP header and

---

trailer are identified by the value of 50 in the previous header's Next Header field. Figure 4-19 shows the structure of the ESP header and trailer.

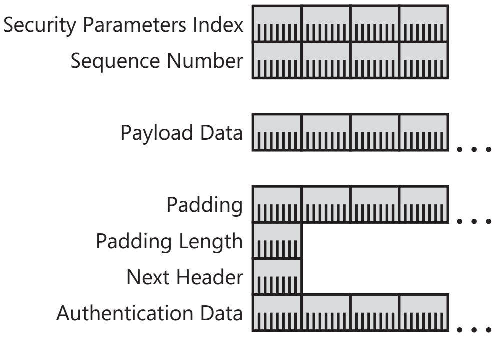

Figure 4-19 The structure of the Encapsulating Security Payload header and trailer

The ESP header contains an SPI field that helps identify the IPSec SA, and a Sequence Number field that provides antireplay protection. The ESP trailer contains the Padding, Padding Length, Next Header, and Authentication Data fields. The Padding field is used to ensure 4-byte boundaries for the ESP payload and appropriate data-block boundaries for encryption algorithms. The Padding Length field indicates the size of the Padding field in bytes. The Authentication Data field contains the ICV.

Details about how the ESP header and trailer provide data confidentiality, authentication, and integrity through cryptographic techniques are beyond the scope of this book.

IPv6 MTU

IPv6 requires that the link layer support a minimum MTU size of 1280 bytes. Link layers that do not support this MTU size must provide a link-layer fragmentation and reassembly scheme that is transparent to IPv6. For link layers that can support a configurable MTU size, RFC 2460 recommends that they be configured with an MTU size of at least 1500 bytes (the IPv6 MTU for Ethernet II encapsulation). An example of a configurable MTU is the Maximum Receive Unit (MRU) of a Point-to-Point Protocol (PPP) link.

Like IPv4, IPv6 provides a Path MTU Discovery process that uses the ICMPv6 Packet Too Big message described in the "Path MTU Discovery" section of Chapter 5, "ICMPv6." Path MTU Discovery allows the transmission of IPv6 packets that are larger than 1280 bytes.

IPv6 source hosts can fragment payloads of upper-layer protocols that are larger than the path MTU by using the process and Fragment header previously described. However, the use of IPv6 fragmentation is highly discouraged. An IPv6 node must be able to reassemble a fragmented packet that is at least 1500 bytes in size.

Table 4-5 lists commonly used local area network (LAN) and wide area network (WAN) technologies and their defined IPv6 MTUs.

Table 4-5 IPv6 MTUs for Common LAN and WAN Technologies

LAN or WAN Technology	IPv6 MTU
Ethernet (Ethernet II encapsulation)	1500
Ethernet (IEEE 802.3 SubNetwork Access Protocol [SNAP] encapsulation)	1492
IEEE 802.11	2312
Token Ring	Varies
FDDI	4352
Attached Resource Computer Network (ARCNet)	9072
PPP	1500
X.25	1280
Frame Relay	1592
Asynchronous Transfer Mode (ATM) (Null or SNAP encapsulation)	9180

For more information about LAN and WAN encapsulations for IPv6 packets, see Appendix A, “Link-Layer Support for IPv6.”

Upper-Layer Checksums

The current implementation of TCP, UDP, and ICMP for IPv4 incorporates into their checksum calculation a pseudo-header that includes both the IPv4 Source Address and Destination Address fields. This checksum calculation must be modified for TCP, UDP, and ICMPv6 traffic sent over IPv6 to include IPv6 addresses. Figure 4-20 shows the structure of the new IPv6 pseudo-header that must be used by TCP, UDP, and ICMPv6 checksum calculations. IPv6 uses the same algorithm as IPv4 for computing the checksum value.

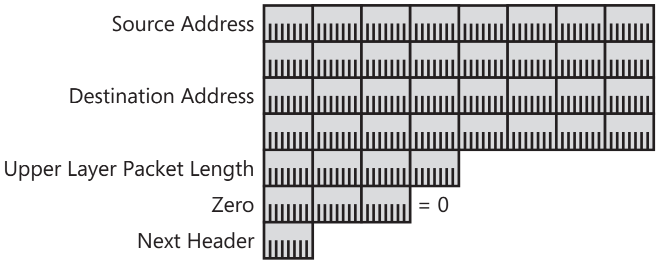

Figure 4-20 The structure of the new IPv6 pseudo-header

The IPv6 pseudo-header includes the Source Address field, the Destination Address field, an Upper Layer Packet Length field that indicates the length of the upper-layer PDU, and a Next Header field that indicates the upper-layer protocol for which the checksum is being calculated.

References

The following references were cited in this chapter:

- RFC 791 — "Internet Protocol"
- RFC 1812 — "Requirements for IP Version 4 Routers"
- RFC 2401 — "Security Architecture for the Internet Protocol"
- RFC 2402 — "IP Authentication Header"
- RFC 2406 — "IP Encapsulating Security Payload (ESP)"
- RFC 2460 — "Internet Protocol, Version 6 (IPv6)"
- RFC 2474 — "Definition of the Differentiated Services Field (DS Field)"
- RFC 2675 — "IPv6 Jumbograms"
- RFC 2711 — "IPv6 Router Alert Option"
- RFC 3168 — "The Addition of Explicit Congestion Notification (ECN) to IP"
- RFC 3697 — "IPv6 Flow Label Specification"
- RFC 3775 — "Mobility Support in IPv6"

You can obtain these RFCs from *http://www.ietf.org/rfc.html*.

Testing for Understanding

To test your understanding of the IPv6 header, answer the following questions. See Appendix D, "Testing for Understanding Answers," to check your answers.

1. Why does the IPv6 header not include a checksum?
2. What is the IPv6 equivalent to the IHL field in the IPv4 header?
3. How does the combination of the Traffic Class and Flow Label fields provide better support for prioritized traffic delivery?
4. Which extension headers are fragmentable and why? Which extension headers are not fragmentable and why?
5. Describe a situation that results in an IPv6 packet that contains a Fragment Header in which the Fragment Offset field is set to 0 and the More Fragments flag is not set.
6. Describe how the new upper-layer checksum calculation affects transport layer protocols such as TCP and UDP.

If the minimum MTU for IPv6 packets is 1280 bytes, how are 1280-byte packets sent on a link that supports only 512-byte frames?