CHAPTER 4

Extra Materials for Chapter 4

In this document, we discuss four topics that were briefly mentioned in Chapter 4 of the textbook: IPv4 options, ICMPv4 packets, IPv6 extension headers, and ICMPv6 packets. These materials may be useful for those reader who need to work with IPv4 or IPv6.

4.1 IPv4 OPTIONS

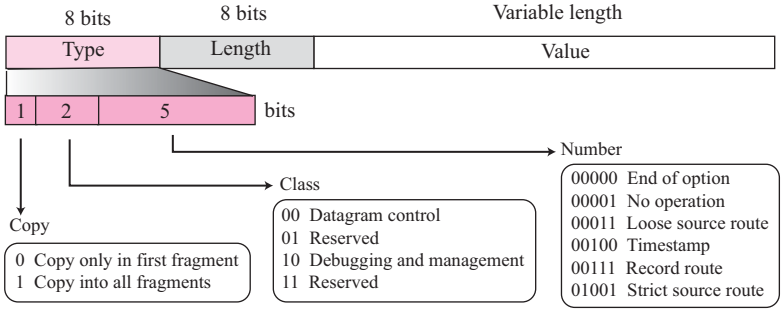
The header of the IP datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes (as mentioned in the text). The variable part comprises the options, which can be a maximum of 40 bytes.

Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging. Although options are not a required part of the IP header, option processing is required of the IP software. This means that all implementations must be able to handle options if they are present in the header.

Format

Figure 4.1 shows the format of an option. It is composed of a 1-byte type field, a 1-byte length field, and a variable-sized value field. The three fields are often referred to as type-length-value or TLV.

Figure 4.1 *Option format*



Type

The **type field** is 8 bits long and contains three subfields: copy, class, and number.

- **Copy.** This 1-bit subfield controls the presence of the option in fragmentation. When its value is 0, it means that the option must be copied only to the first fragment. If its value is 1, it means the option must be copied to all fragments.
- **Class.** This 2-bit subfield defines the general purpose of the option. When its value is 00, it means that the option is used for datagram control. When its value is 10, it means that the option is used for debugging and management. The other two possible values (01 and 11) have not yet been defined.
- **Number.** This 5-bit subfield defines the type of option. Although 5 bits can define up to 32 different types, currently only 6 types are in use.

Length

The **length field** defines the total length of the option including the type field and the length field itself. This field is not present in all of the option types.

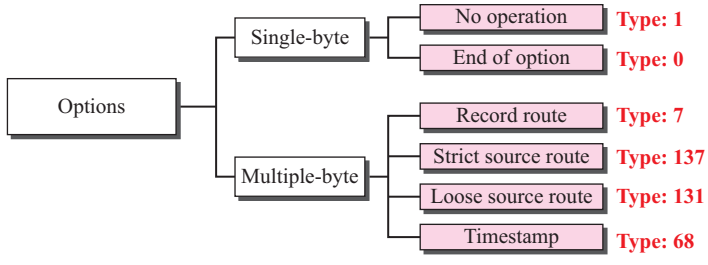
Value

The **value field** contains the data that specific options require. Like the length field, this field is also not present in all option types.

Option Types

As mentioned previously, only six options are currently being used. Two of these are 1-byte options, and they do not require the length or the data fields. Four of them are multiple-byte options; they require the length and the data fields (see Figure 4.2).

Figure 4.2 *Categories of options*



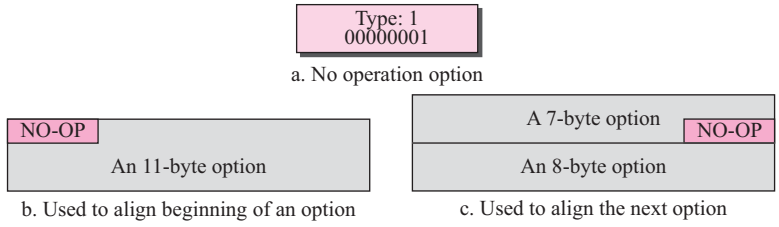
No-Operation Option

A **no-operation option** is a 1-byte option used as a filler between options. For example, it can be used to align the next option on a 16-bit or 32-bit boundary (see Figure 4.3).

End-of-Option Option

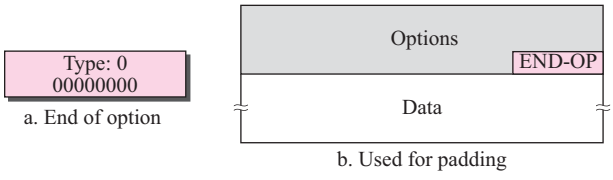
An **end-of-option option** is also a 1-byte option used for padding at the end of the option field. It, however, can only be used as the last option. Only one end-of-option option can be used. After this option, the receiver looks for the payload data. This

Figure 4.3 *No operation option*



means that if more than 1 byte is needed to align the option field, some no-operation options must be used, followed by an end-of-option option (see Figure 4.4).

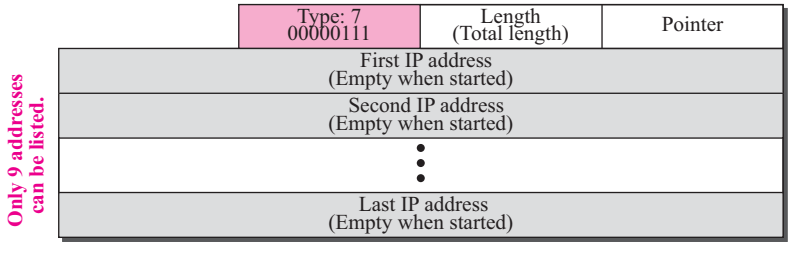
Figure 4.4 *End-of-option option*



Record-Route Option

A **record-route option** is used to record the Internet routers that handle the datagram. It can list up to nine router IP addresses since the maximum size of the header is 60 bytes, which must include 20 bytes for the base header. This implies that only 40 bytes are left over for the option part. The source creates placeholder fields in the option to be filled by the visited routers. Figure 4.5 shows the format of the record route option.

Figure 4.5 *Record-route option*

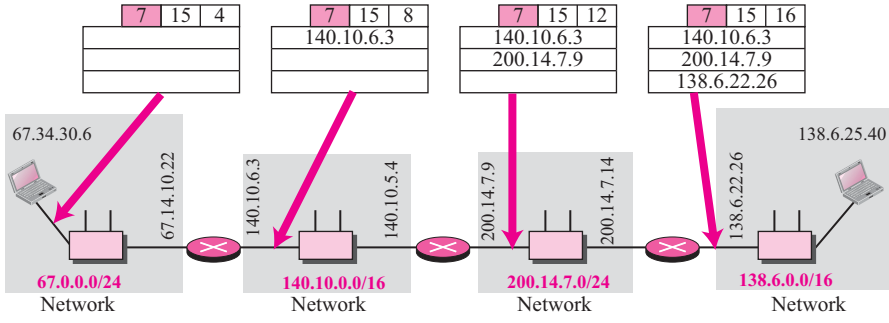


Both the code and length fields have been described above. The **pointer field** is an offset integer field containing the byte number of the first empty entry. In other words, it points to the first available entry.

The source creates empty fields for the IP addresses in the data field of the option. When the datagram leaves the source, all of the fields are empty. The pointer field has a value of 4, pointing to the first empty field.

When the datagram is traveling, each router that processes the datagram compares the value of the pointer with the value of the length. If the value of the pointer is greater than the value of the length, the option is full and no changes are made. However, if the value of the pointer is not greater than the value of the length, the router inserts its outgoing IP address in the next empty field (remember that a router has more than one IP address). In this case, the router adds the IP address of its interface from which the datagram is leaving. The router then increments the value of the pointer by 4. Figure 4.6 shows the entries as the datagram travels left to right from router to router.

Figure 4.6 *Record-route concept*



Strict-Source-Route Option

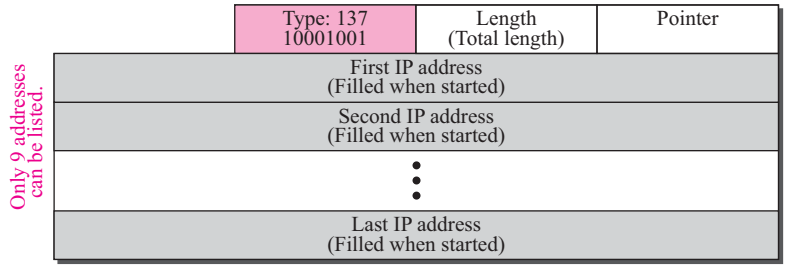
A **strict-source-route option** is used by the source to predetermine a route for the datagram as it travels through the Internet. Dictation of a route by the source can be useful for several purposes. The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput. Alternatively, it may choose a route that is safer or more reliable for the sender's purpose. For example, a sender can choose a route so that its datagram does not travel through a competitor's network.

If a datagram specifies a strict source route, all of the routers defined in the option must be visited by the datagram. A router must not be visited if its IP address is not listed in the datagram. If the datagram visits a router that is not on the list, the datagram is discarded and an error message is issued. If the datagram arrives at the destination and some of the entries were not visited, it will also be discarded and an error message issued.

Regular users of the Internet, however, are not usually aware of the physical topology of the Internet. Consequently, strict source routing is not the choice of most users. Figure 4.7 shows the format of the strict source route option.

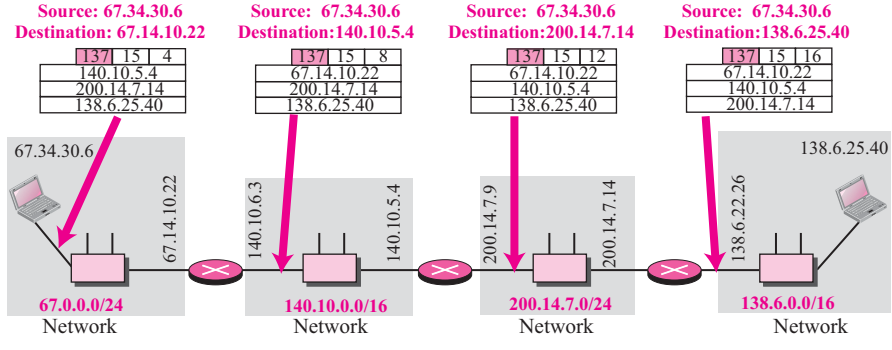
The format is similar to the record route option with the exception that all of the IP addresses are entered by the sender. When the datagram is traveling, each router that processes the datagram compares the value of the pointer with the value of the length. If the value of the pointer is greater than the value of the length, the datagram has vis-

Figure 4.7 *Strict-source-route option*



ited all of the predefined routers. The datagram cannot travel anymore; it is discarded and an error message is created. If the value of the pointer is not greater than the value of the length, the router compares the destination IP address with its incoming IP address: If they are equal, it processes the datagram, swaps the IP address pointed by the pointer with the destination address, increments the pointer value by 4, and forwards the datagram. If they are not equal, it discards the datagram and issues an error message. Figure 4.8 shows the actions taken by each router as a datagram travels from source to destination.

Figure 4.8 *Strict-source-route concept*



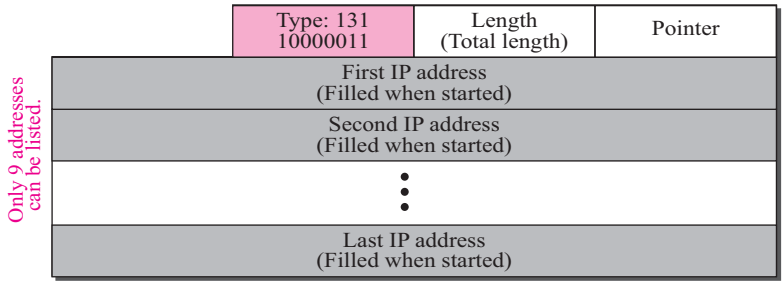
Loose-Source-Route Option

A **loose-source-route option** is similar to the strict source route, but it is more relaxed. Each router in the list must be visited, but the datagram can visit other routers as well. Figure 4.9 shows the format of the loose source route option.

Timestamp

A **timestamp option** is used to record the time of datagram processing by a router. The time is expressed in milliseconds from midnight, Universal Time. Knowing the time a datagram is processed can help users and managers track the behavior of the routers in

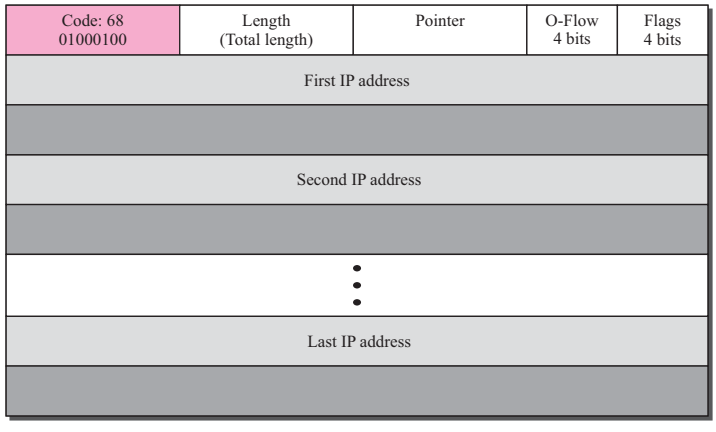
Figure 4.9 *Loose-source-route option*



the Internet. We can estimate the time it takes for a datagram to go from one router to another. We say *estimate* because, although all routers may use Universal Time, their local clocks may not be synchronized.

However, non-privileged users of the Internet are not usually aware of the physical topology of the Internet. Consequently, a timestamp option is not a choice for most users. Figure 4.10 shows the format of the timestamp option.

Figure 4.10 *Timestamp option*



In this figure, the definitions of the code and length fields are the same as before. The overflow field records the number of routers that could not add their timestamp because no more fields were available. The flags field specifies the visited router responsibilities. If the flag value is 0, each router adds only the timestamp in the provided field. If the flag value is 1, each router must add its outgoing IP address and the timestamp. If the value is 3, the IP addresses are given, and each router must check the given IP address with its own incoming IP address. If there is a match, the router overwrites the IP address with its outgoing IP address and adds the timestamp (see Figure 4.11).

Figure 4.11 *Use of flag in timestamp*

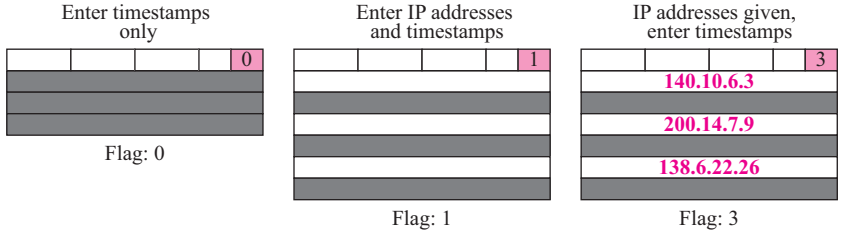
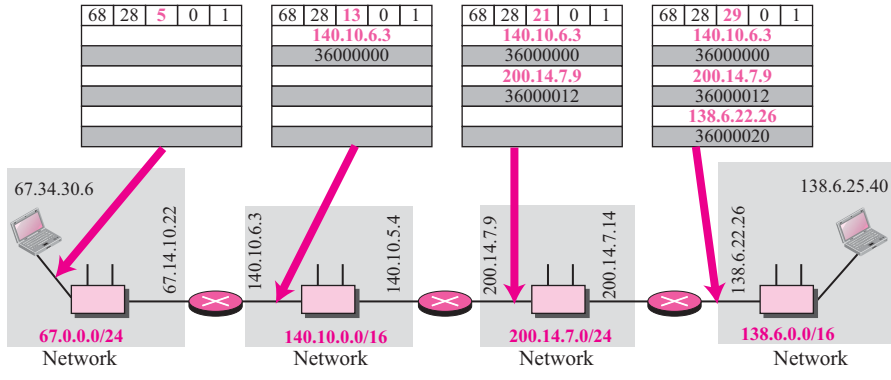


Figure 4.12 shows the actions taken by each router when a datagram travels from source to destination. The figure assumes a flag value of 1.

Figure 4.12 *Timestamp concept*



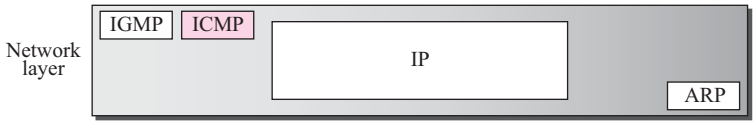
4.2 ICMPv4 PACKETS

We briefly discussed the role of the ICMPv4 in the textbook. The IP protocol has no error-reporting or error-correcting mechanism. What happens if something goes wrong? What happens if a router must discard a datagram because it cannot find a router to the final destination, or because the time-to-live field has a zero value? What happens if the final destination host must discard all fragments of a datagram because it has not received all fragments within a predetermined time limit? These are examples of situations where an error has occurred and the IP protocol has no built-in mechanism to notify the original host.

The IP protocol also lacks a mechanism for host and management queries. A host sometimes needs to determine if a router or another host is alive. And sometimes a network manager needs information from another host or router.

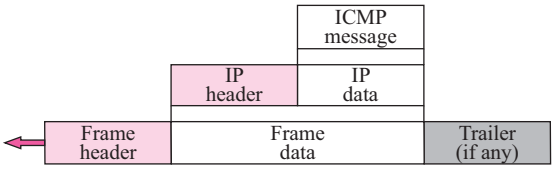
The **Internet Control Message Protocol (ICMP)** has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol. Figure 4.13 shows the position of ICMP in relation to IP and other protocols in the network layer.

Figure 4.13 *Position of ICMP in the network layer*



ICMP itself is a network layer protocol. However, its messages are not passed directly to the data link layer as would be expected. Instead, the messages are first encapsulated inside IP datagrams before going to the lower layer (see Figure 4.14).

Figure 4.14 *ICMP encapsulation*



The value of the protocol field in the IP datagram is 1 to indicate that the IP data is an ICMP message.

Messages

ICMP messages are divided into two broad categories: **error-reporting messages** and **query messages.** The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet. The query messages, which occur in pairs, help a host or a network manager get specific information from a router or another host. For example, nodes can discover their neighbors. Also, hosts can discover and learn about routers on their network and routers can help a node redirect its messages. Table 4.1 lists the ICMP messages in each category.

Table 4.1 *ICMP messages*

Category	*Type*	*Message*
Error-reporting messages	3	Destination unreachable
	4	Source quench
	11	Time exceeded
	12	Parameter problem
	5	Redirection

Table 4.1 *ICMP messages (continued)*

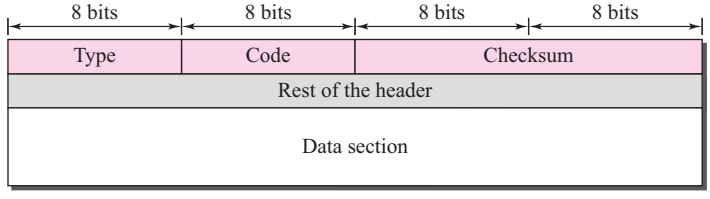
Category	*Type*	*Message*
Query messages	8 or 0	Echo request or reply
	13 or 14	Timestamp request or reply

Message Format

An ICMP message has an 8-byte header and a variable-size data section. Although the general format of the header is different for each message type, the first 4 bytes are common to all. As Figure 4.15 shows, the first field, ICMP type, defines the type of the message. The code field specifies the reason for the particular message type. The last common field is the checksum field. The rest of the header is specific for each message type.

The data section in error messages carries information for finding the original packet that had the error. In query messages, the data section carries extra information based on the type of the query.

Figure 4.15 *General format of ICMP messages*



Error Reporting Messages

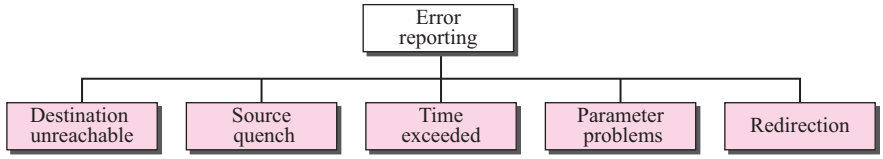
One of the main responsibilities of ICMP is to report errors. Although technology has produced increasingly reliable transmission media, errors still exist and must be handled. IP, as discussed in Chapter 4 in the textbook, is an unreliable protocol. This means that error checking and error control are not a concern of IP. ICMP was designed, in part, to compensate for this shortcoming. However, ICMP does not correct errors, it simply reports them. Error correction is left to the higher-level protocols. Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses. ICMP uses the source IP address to send the error message to the source (originator) of the datagram.

Five types of errors are handled: destination unreachable, source quench, time exceeded, parameter problems, and redirection (see Figure 4.16).

The following are important points about ICMP error messages:

- No ICMP error message will be generated in response to a datagram carrying an ICMP error message.
- No ICMP error message will be generated for a fragmented datagram that is not the first fragment.

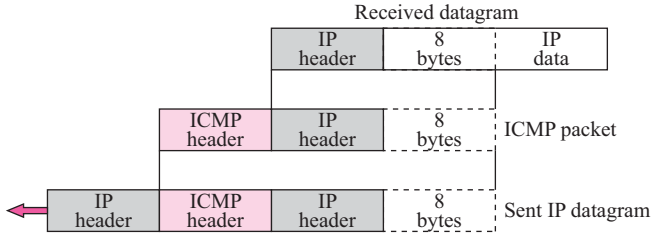
Figure 4.16 *Error-reporting messages*



- No ICMP error message will be generated for a datagram having a multicast address.
- No ICMP error message will be generated for a datagram having a special address such as 127.0.0.0 or 0.0.0.0.

Note that all error messages contain a data section that includes the IP header of the original datagram plus the first 8 bytes of data in that datagram. The original datagram header is added to give the original source, which receives the error message, information about the datagram itself. The 8 bytes of data are included because, as we will saw in Chapters 3 of the textbook, on UDP and TCP protocols, the first 8 bytes provide information about the port numbers (UDP and TCP) and sequence number (TCP). This information is needed so the source can inform the protocols (TCP or UDP) about the error. ICMP forms an error packet, which is then encapsulated in an IP datagram (see Figure 4.17).

Figure 4.17 *Contents of data field for the error messages*

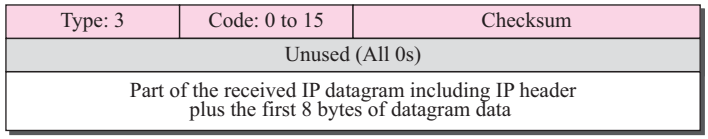


Destination Unreachable

When a router cannot route a datagram or a host cannot deliver a datagram, the datagram is discarded and the router or the host sends a **destination-unreachable message** back to the source host that initiated the datagram. Figure 4.18 shows the format of the destination-unreachable message. The code field for this type specifies the reason for discarding the datagram:

- **Code 0.** The network is unreachable, possibly due to hardware failure.
- **Code 1.** The host is unreachable. This can also be due to hardware failure.

Figure 4.18 *Destination-unreachable format*



- **Code 2.** The protocol is unreachable. An IP datagram can carry data belonging to higher-level protocols such as UDP, TCP, and OSPF. If the destination host receives a datagram that must be delivered, for example, to the TCP protocol, but the TCP protocol is not running at the moment, a code 2 message is sent.
- **Code 3.** The port is unreachable. The application program (process) that the datagram is destined for is not running at the moment.
- **Code 4.** Fragmentation is required, but the DF (do not fragment) field of the datagram has been set. In other words, the sender of the datagram has specified that the datagram not be fragmented, but routing is impossible without fragmentation.
- **Code 5.** Source routing cannot be accomplished. In other words, one or more routers defined in the source routing option cannot be visited.
- **Code 6.** The destination network is unknown. This is different from code 0. In code 0, the router knows that the destination network exists, but it is unreachable at the moment. For code 6, the router has no information about the destination network.
- **Code 7.** The destination host is unknown. This is different from code 1. In code 1, the router knows that the destination host exists, but it is unreachable at the moment. For code 7, the router is unaware of the existence of the destination host.
- **Code 8.** The source host is isolated.
- **Code 9.** Communication with the destination network is administratively prohibited.
- **Code 10.** Communication with the destination host is administratively prohibited.
- **Code 11.** The network is unreachable for the specified type of service. This is different from code 0. Here the router can route the datagram if the source had requested an available type of service.
- **Code 12.** The host is unreachable for the specified type of service. This is different from code 1. Here the router can route the datagram if the source had requested an available type of service.
- **Code 13.** The host is unreachable because the administrator has put a filter on it.
- **Code 14.** The host is unreachable because the host precedence is violated. The message is sent by a router to indicate that the requested precedence is not permitted for the destination.
- **Code 15.** The host is unreachable because its precedence was cut off. This message is generated when the network operators have imposed a minimum level of prece-

dence for the operation of the network, but the datagram was sent with a precedence below this level.

Note that destination-unreachable messages can be created either by a router or the destination host. Code 2 and code 3 messages can only be created by the destination host; the messages of the remaining codes can only be created by routers.

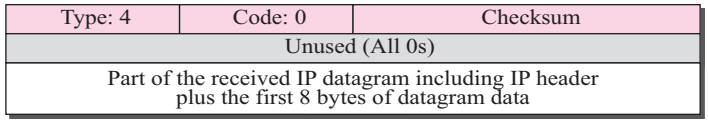
Note that even if a router does not report a destination-unreachable message, it does not necessarily mean that the datagram has been delivered. For example, if a datagram is traveling through an Ethernet network, there is no way that a router knows that the datagram has been delivered to the destination host or the next router because Ethernet does not provide any acknowledgment mechanism.

Source Quench

The IP protocol is a connectionless protocol. There is no communication between the source host, which produces the datagram, the routers, which forward it, and the destination host, which processes it. One of the ramifications of this absence of communication is the lack of *flow control* and *congestion control.*

The **source-quench message** in ICMP was designed to add a kind of flow control and congestion control to the IP. When a router or host discards a datagram due to congestion, it sends a source-quench message to the sender of the datagram. This message has two purposes. First, it informs the source that the datagram has been discarded. Second, it warns the source that there is congestion somewhere in the path and that the source should slow down (quench) the sending process. The source-quench format is shown in Figure 4.19.

Figure 4.19 *Source-quench format*



There are some points that deserve more explanation. First, the router or destination host that has experienced the congestion sends one source-quench message for each discarded datagram to the source host. Second, there is no mechanism to tell the source that the congestion has been relieved and the source can resume sending datagrams at its previous rate. The source continues to lower the rate until no more source-quench messages are received. Third, the congestion can be created either by a one-to-one or many-to-one communication. In a one-to-one communication, a single high-speed host could create datagrams faster than a router or the destination host can handle. In this case, source-quench messages can be helpful. They tell the source to slow down. In a many-to-one communication, many sources create datagrams that must be handled by a router or the destination host. In this case, each source can be sending datagrams at different speeds, some of them at a low rate, others at a high rate. Here, the source-quench message may not be very useful. The router or the destination host has no clue which source is responsible for the congestion. It may drop

a datagram from a very slow source instead of dropping the datagram from the source that has actually created the congestion.

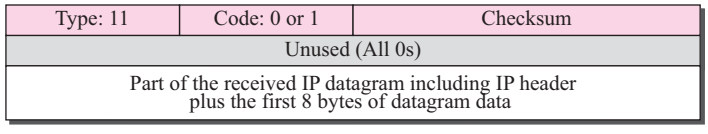
Time Exceeded

The **time-exceeded message** is generated in two cases:

- First, as we saw in Chapter 4 of the textbook, routers use routing tables to find the next hop (next router) that must receive the packet. If there are errors in one or more routing tables, a packet can travel in a loop or a cycle, going from one router to the next or visiting a series of routers endlessly. As we saw in Chapter 4 of the book, each datagram contains a field called *time to live* that controls this situation. When a datagram visits a router, the value of this field is decremented by 1. When the time-to-live value reaches 0, after decrementing, the router discards the datagram. However, when the datagram is discarded, a time-exceeded message must be sent by the router to the original source.
- Second, a time-exceeded message is also generated when all fragments that make up a message do not arrive at the destination host within a certain time limit. When the first fragment arrives, the destination host starts a timer. If all the fragments have not arrived when the time expires, the destination discards all the fragments and sends a time-exceeded message to the original sender.

Figure 4.20 shows the format of the time-exceeded message. Code 0 is used when the datagram is discarded by the router due to a time-to-live field value of zero. Code 1 is used when arrived fragments of a datagram are discarded because some fragments have not arrived within the time limit.

Figure 4.20 *Time-exceeded message format*



Parameter Problem

Any ambiguity in the header part of a datagram can create serious problems as the datagram travels through the Internet. If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter-problem message back to the source.

A parameter-problem message can be created by a router or the destination host.

Figure 4.21 shows the format of the **parameter-problem message.** The code field in this case specifies the reason for discarding the datagram:

- **Code 0.** There is an error or ambiguity in one of the header fields. In this case, the value in the pointer field points to the byte with the problem. For example, if the value is zero, then the first byte is not a valid field.
- **Code 1.** The required part of an option is missing. In this case, the pointer is not used.

Figure 4.21 *Parameter-problem message format*

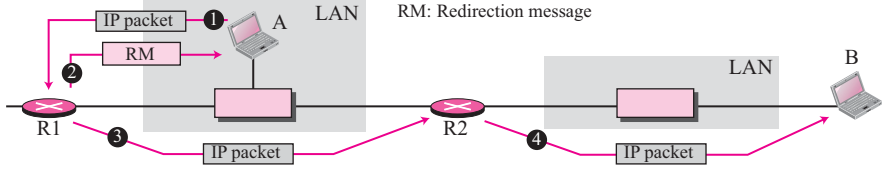
Type: 12	Code: 0 or 1	Checksum
Pointer	Unused (All 0s)	
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Redirection

When a router needs to send a packet destined for another network, it must know the IP address of the next appropriate router. The same is true if the sender is a host. Both routers and hosts then must have a routing table to find the address of the router or the next router. Routers take part in the routing update process and are supposed to be updated constantly. Routing is dynamic.

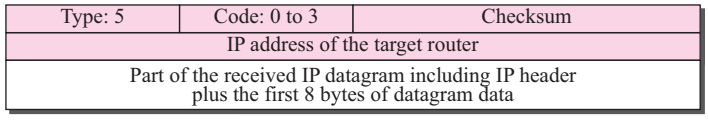
However, for efficiency, hosts do not take part in the routing update process because there are many more hosts in an internet than routers. Updating the routing tables of hosts dynamically produces unacceptable traffic. The hosts usually use static routing. When a host comes up, its routing table has a limited number of entries. It usually knows only the IP address of one router, the default router. For this reason, the host may send a datagram, which is destined for another network, to the wrong router. In this case, the router that receives the datagram will forward the datagram to the correct router. However, to update the routing table of the host, it sends a redirection message to the host. This concept of redirection is shown in Figure 4.22. Host A wants to send a datagram to host B. Router R2 is obviously the most efficient routing choice, but host A did not choose router R2. The datagram goes to R1 instead. R1, after consulting its table, finds that the packet should have gone to R2. It sends the packet to R2 and, at the same time, sends a redirection message to host A. Host A's routing table can now be updated.

Figure 4.22 *Redirection concept*



The format of the **redirection message** is shown in Figure 4.23. Note that the IP address of the appropriate target is given in the second row.

Figure 4.23 *Redirection message format*



Although the redirection message is considered an error-reporting message, it is different from other error messages. The router does not discard the datagram in this case; it is sent to the appropriate router. The code field for the redirection message narrows down the redirection:

- **Code 0.** Redirection for a network-specific route.
- **Code 1.** Redirection for a host-specific route.
- **Code 2.** Redirection for a network-specific route based on a specified type of service.
- **Code 3.** Redirection for a host-specific route based on a specified type of service.

Query Messages

In addition to error reporting, ICMP can also diagnose some network problems. This is accomplished through the query messages. A group of five different pairs of messages have been designed for this purpose, but three of these pairs are deprecated today, as we discuss later in the section. Only two pairs are used today: echo request and replay and timestamp request and replay. In this type of ICMP message, a node sends a message that is answered in a specific format by the destination node.

Echo Request and Reply

The **echo-request** and **echo-reply messages** are designed for diagnostic purposes. Network managers and users utilize this pair of messages to identify network problems. The combination of echo-request and echo-reply messages determines whether two systems (hosts or routers) can communicate with each other.

A host or router can send an echo-request message to another host or router. The host or router that receives an echo-request message creates an echo-reply message and returns it to the original sender.

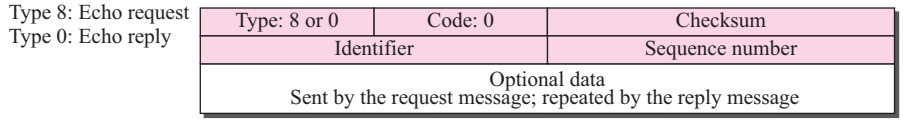
The echo-request and echo-reply messages can be used to determine if there is communication at the IP level. Because ICMP messages are encapsulated in IP datagrams, the receipt of an echo-reply message by the machine that sent the echo request is proof that the IP protocols in the sender and receiver are communicating with each other using the IP datagram. Also, it is proof that the intermediate routers are receiving, processing, and forwarding IP datagrams.

The echo-request and echo-reply messages can also be used by a host to see if another host is reachable. At the user level, this is done by invoking the packet Internet

groper (ping) command. Today, most systems provide a version of the *ping* command that can create a series (instead of just one) of echo-request and echo-reply messages, providing statistical information.

Echo request, together with echo reply, can determine whether or not a node is functioning properly. The node to be tested is sent an echo-request message. The optional data field contains a message that must be repeated exactly by the responding node in its echo-reply message. Figure 4.24 shows the format of the echo-reply and echo-request message. The identifier and sequence number fields are not formally defined by the protocol and can be used arbitrarily by the sender. The identifier is often the same as the process ID.

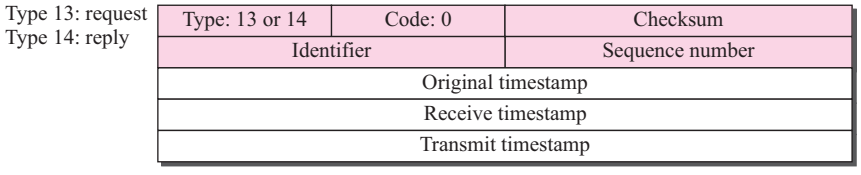
Figure 4.24 *Echo-request and echo-reply messages*



Timestamp Request and Reply

Two machines (hosts or routers) can use the **timestamp-request** and **timestamp-reply messages** to determine the round-trip time needed for an IP datagram to travel between them. It can also be used to synchronize the clocks in two machines. The format of these two messages is shown in Figure 4.25.

Figure 4.25 *Timestamp-request and timestamp-reply message format*



The three timestamp fields are each 32 bits long. Each field can hold a number representing time measured in milliseconds from midnight in Universal Time (formerly called Greenwich Mean Time). (Note that 32 bits can represent a number between 0 and 4,294,967,295, but a timestamp in this case cannot exceed 86,400,000 = 24 × 60 × 60 × 1000.)

The source creates a timestamp-request message. The source fills the *original timestamp* field with the Universal Time shown by its clock at departure time. The other two timestamp fields are filled with zeros.

The destination creates the timestamp-reply message. The destination copies the original timestamp value from the request message into the same field in its reply message. It then fills the *receive timestamp* field with the Universal Time shown by its

clock at the time the request was received. Finally, it fills the *transmit timestamp* field with the Universal Time shown by its clock at the time the reply message departs.

The timestamp-request and timestamp-reply messages can be used to compute the one-way or round-trip time required for a datagram to go from a source to a destination and then back again. The formulas are

sending time = receive timestamp − original timestamp
receiving time = returned time − transmit timestamp
round-trip time = sending time + receiving time

Note that the sending and receiving time calculations are accurate only if the two clocks in the source and destination machines are synchronized. However, the round-trip calculation is correct even if the two clocks are not synchronized because each clock contributes twice to the round-trip calculation, thus canceling any difference in synchronization.

For example, given the following information:

original timestamp: 46	**receive timestamp: 59**
transmit timestamp: 60	**return time: 67**

We can calculate the round-trip time to be 20 milliseconds:

sending time = 59 − 46 = 13 milliseconds
receiving time = 67 − 60 = 7 milliseconds
round-trip time = 13 + 7 = 20 milliseconds

Given the actual one-way time, the timestamp-request and timestamp-reply messages can also be used to synchronize the clocks in two machines using the following formula:

Time difference = receive timestamp − (original timestamp field + one-way time duration)

The one-way time duration can be obtained either by dividing the round-trip time duration by two (if we are sure that the sending time is the same as the receiving time) or by other means. For example, we can tell that the two clocks in the previous example are 3 milliseconds out of synchronization because

Time difference = 59 − (46 + 10) = 3

Deprecated Messages

Three pairs of messages are declared obsolete by IETF:

1. *Information request and replay* messages are not used today because their duties are done by Address Resolution Protocol (ARP) discussed in Chapter 5 in the textbook.

2. *Address mask request and reply* messages are not used today because their duties are done by Dynamic Host Configuration Protocol (DHCP), discussed in Chapter 4 in the textbook.
3. *Router solicitation and advertisement* messages are not used today because their duties are done by Dynamic Host Configuration Protocol (DHCP) as discussed in Chapter 4 of the book.

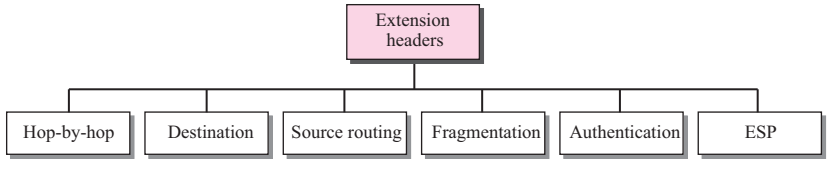
4.3 IPv6 EXTENSION HEADERS

We discussed IPv6 in Chapter 4 of the textbook. An IPv6 packet is made of a base header and some extension headers (Figures 4.101 and 4.101 in the textbook). The length of the base header is fixed at 40 bytes. However, to give more functionality to the IP datagram, the base header can be followed by up to six **extension headers.** Many of these headers are options in IPv4.

Types of Extension Headers

Six types of extension headers have been defined. These are hop-by-hop option, source routing, fragmentation, authentication, encrypted security payload, and destination option (see Figure 4.26).

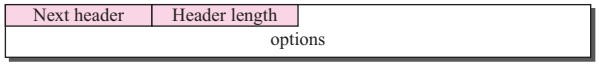
Figure 4.26 *Extension header types*



Hop-by-Hop Option

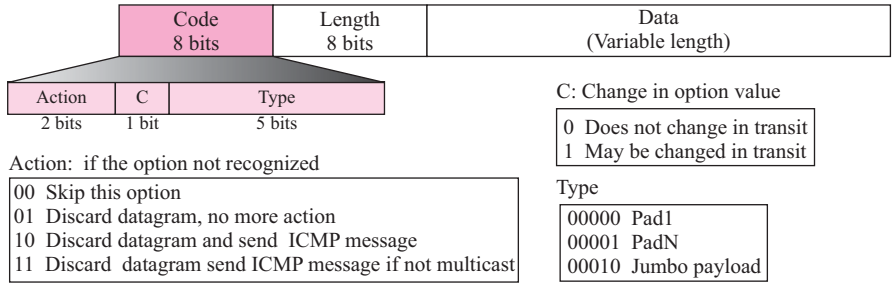
The **hop-by-hop option** is used when the source needs to pass information to all routers visited by the datagram. For example, perhaps routers must be informed about certain management, debugging, or control functions. Or, if the length of the datagram is more than the usual 65,535 bytes, routers must have this information. Figure 4.27 shows the format of the hop-by-hop option header. The first field defines the next header in the chain of headers. The header length defines the number of bytes in the header (including the next header field). The rest of the header contains different options.

Figure 4.27 *Hop-by-hop option header format*



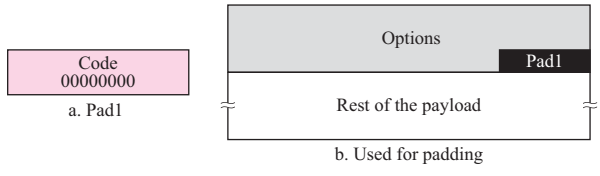
So far, only three hop-by-hop options have been defined: **Pad1, PadN,** and **jumbo payload.** Figure 4.28 shows the general format of the option.

Figure 4.28 *The format of options in a hop-by-hop option header*



- **Pad1.** This option is 1 byte long and is designed for alignment purposes. Some options need to start at a specific bit of the 32-bit word (see the jumbo payload description to come). If an option falls short of this requirement by exactly one byte, Pad1 is added to make up the difference. Pad1 contains neither the option length field nor the option data field. It consists solely of the option code field with all bits set to 0 (action is 00, the change bit is 0, and type is 00000). Pad1 can be inserted anywhere in the hop-by-hop option header (see Figure 4.29).

Figure 4.29 *Pad1*



- **PadN.** PadN is similar in concept to Pad1. The difference is that PadN is used when 2 or more bytes are needed for alignment. This option consists of 1 byte of option code, 1 byte of the option length, and a variable number of zero padding bytes. The value of the option code is 1 (action is 00, the change bit is 0, and type is 00001). The option length contains the number of padding bytes. See Figure 4.30.
- **Jumbo payload.** Recall that the length of the payload in the IP datagram can be a maximum of 65,535 bytes. However, if for any reason a longer payload is required, we can use the jumbo payload option to define this longer length. The jumbo payload option must always start at a multiple of 4 bytes plus 2 from the beginning of the extension headers. The jumbo payload option starts at the $(4n + 2)$ byte, where n is a small integer. See Figure 4.31.

Figure 4.30 *PadN*

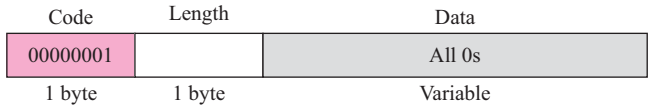
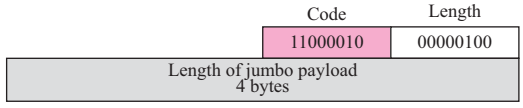


Figure 4.31 *Jumbo payload*



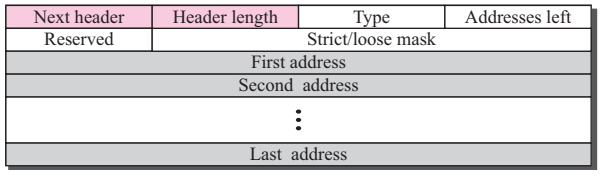
Destination Option

The **destination option** is used when the source needs to pass information to the destination only. Intermediate routers are not permitted access to this information. The format of the destination option is the same as the hop-by-hop option. So far, only the Pad1 and PadN options have been defined.

Source Routing

The source routing extension header combines the concepts of the strict source route and the loose source route options of IPv4. The source routing header contains a minimum of seven fields (see Figure 4.32). The first two fields, next header and header length, are identical to that of the hop-by-hop extension header. The type field defines loose or strict routing. The addresses left field indicates the number of hops still needed to reach the destination. The strict/loose mask field determines the rigidity of routing. If set to strict, routing must follow exactly as indicated by the source. If, instead, the mask is loose, other routers may be visited in addition to those in the header.

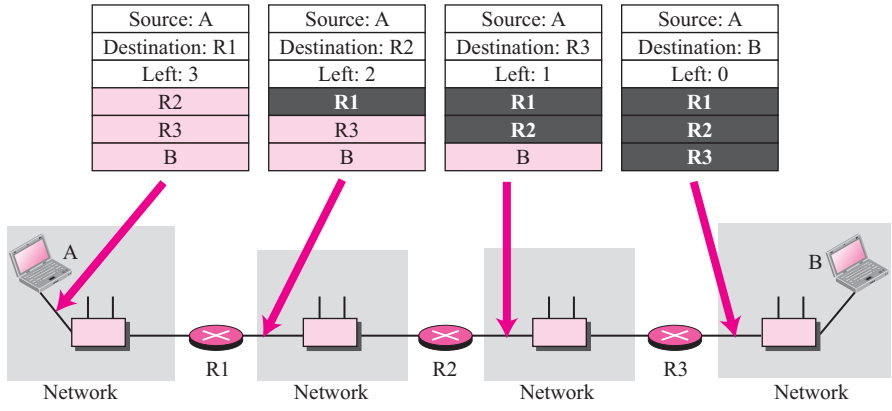
Figure 4.32 *Source routing*



The destination address in source routing does not conform to our previous definition (the final destination of the datagram). Instead, it changes from router to router. For example, in Figure 4.33, Host A wants to send a datagram to Host B using a specific route: A to R1 to R2 to R3 to B. Notice the destination address in the base headers.

It is not constant as you might expect. Instead, it changes at each router. The addresses in the extension headers also change from router to router.

Figure 4.33 *Source routing example*

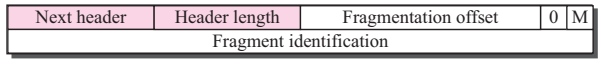


Fragmentation

The concept of **fragmentation** in IPv6 is the same as that in IPv4. However, the place where fragmentation occurs differs. In IPv4, the source or a router is required to fragment if the size of the datagram is larger than the MTU of the network over which the datagram travels. In IPv6, only the original source can fragment. A source must use a **Path MTU Discovery technique** to find the smallest MTU supported by any network on the path. The source then fragments using this knowledge.

If the source does not use a Path MTU Discovery technique, it fragments the datagram to a size of 1,280 bytes or smaller. This is the minimum size of MTU required for each network connected to the Internet. Figure 4.34 shows the format of the fragmentation extension header.

Figure 4.34 *Fragmentation*

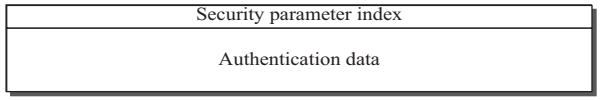


Authentication

The **authentication** extension header has a dual purpose: it validates the message sender and ensures the integrity of data. The former is needed so the receiver can be sure that a message is from the genuine sender and not from an imposter. The latter is needed to check that the data is not altered in transition by some hacker.

The format of the authentication extension header is shown in Figure 4.35. The security parameter index field defines the algorithm used for authentication. The authentication data field contains the actual data generated by the algorithm.

Figure 4.35 *Authentication*



Many different algorithms can be used for authentication. Figure 4.36 outlines the method for calculating the authentication data field.

Figure 4.36 *Calculation of authentication data*

The sender passes a 128-bit security key, the entire IP datagram, and the 128-bit security key again to the algorithm. Those fields in the datagram with values that change during transmission (for example, hop count) are set to zero. The datagram passed to the algorithm includes the authentication header extension, with the authentication data field set to zero. The algorithm creates authentication data which is inserted into the extension header prior to datagram transmission.

The receiver functions in a similar manner. It takes the secret key and the received datagram (again, with changeable fields set to zero) and passes them to the authentication algorithm. If the result matches that in the authentication data field, the IP datagram is authentic; otherwise, the datagram is discarded.

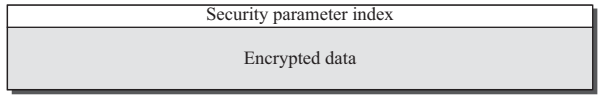
Encrypted Security Payload

The **encrypted security payload (ESP)** is an extension that provides confidentiality and guards against eavesdropping. Figure 4.37 shows the format. The security parameter index field is a 32-bit word that defines the type of encryption/decryption used. The other field contains the encrypted data along with any extra parameters needed by the algorithm. **Encryption** can be implemented in two ways: transport mode or tunnel mode, which we discussed in the textbook when we discuss IPSec.

Comparison of Options between IPv4 and IPv6

The following shows a quick comparison between the options used in IPv4 and the options used in IPv6 (as extension headers).

Figure 4.37 *Encrypted security payload*

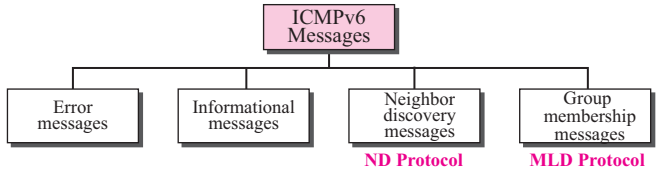


- The no-operation and end-of-option options in IPv4 are replaced by Pad1 and PadN options in IPv6.
- The record route option is not implemented in IPv6 because it was not used.
- The timestamp option is not implemented because it was not used.
- The source route option is called the source route extension header in IPv6.
- The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6.
- The authentication extension header is new in IPv6.
- The encrypted security payload extension header is new in IPv6.

4.4 ICMPv6 PACKETS

ICMPv6 is the counterpart of ICMPv4 designed for IPv6. However, ICMPv6 is more complex. It provides four categories of messages as shown in Figure 4.38.

Figure 4.38 *Categories of ICMPv6 messages*

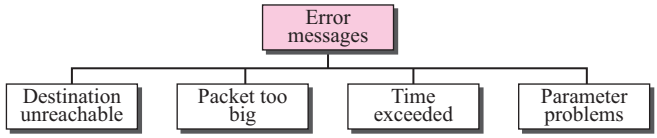


Error-Reporting Messages

As we saw in our discussion of version 4, one of the main responsibilities of ICMPv6 is to report errors. Four types of errors are handled: destination unreachable, packet too big, time exceeded, and parameter problems (see Figure 4.39). Note that the source-quenched message, which is used to control congestion in version 4, is eliminated in this version because the priority and flow label fields in IPv6 are supposed to take care of congestion. The redirection message has moved from the error-reporting category to the neighbor-discovery category, so we discuss it as part of the neighbor-discovery messages.

ICMPv6 forms an error packet, which is then encapsulated in an IPv6 datagram. This is delivered to the original source of the failed datagram.

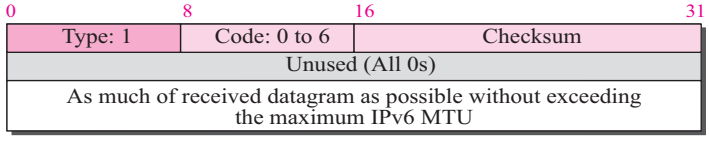
Figure 4.39 *Error-reporting messages*



Destination-Unreachable Message

The concept of the destination unreachable message is the same as described for ICMPv4. When a router cannot forward a datagram or a host cannot deliver the content of the datagram to the upper layer protocol, the router or the host discards the datagram and sends a *destination-unreachable* error message to the source host. Figure 4.40 shows the format of the destination-unreachable message.

Figure 4.40 *Destination-unreachable message*



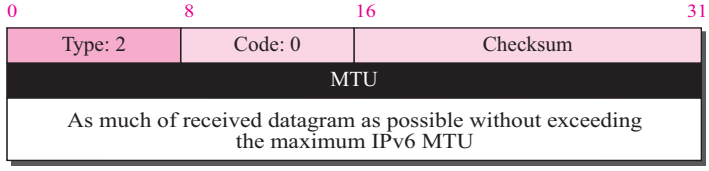
The code field for this type specifies the reason for discarding the datagram and explains exactly what has failed:

- **Code 0.** No path to destination.
- **Code 1.** Communication with the destination is administratively prohibited.
- **Code 2.** Beyond the scope of source address.
- **Code 3.** Destination address is unreachable.
- **Code 4.** Port unreachable.
- **Code 5.** Source address failed (filtering policy).
- **Code 6.** Reject route to destination.

Packet-Too-Big Message

This is a new type of message added to version 6. Since IPv6 does not fragment at the router, if a router receives a datagram that is larger than the maximum transmission unit (MTU) size of the network through which the datagram should pass, two things happen. First, the router discards the datagram. Second, an ICMP error packet—a **packet-too-big message**—is sent to the source. Figure 4.41 shows the format of the packet. Note that there is only one code (0) and that the MTU field informs the sender of the maximum size packet accepted by the network.

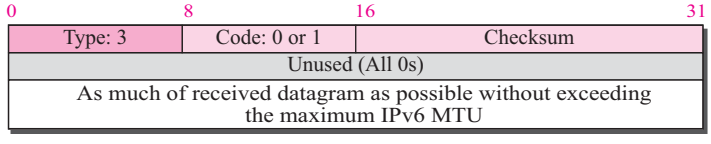
Figure 4.41 *Packet-too-big message*



Time-Exceeded Message

A *time-exceeded* error message is generated in two cases: when the *time to live* value becomes zero and when not all fragments of a datagram have arrived in the time limit. The format of the *time-exceeded* message in version 6 is similar to the one in version 4. The only difference is that the type value has changed to 3. Figure 4.42 shows the format of the time-exceeded message.

Figure 4.42 *Time-exceeded message*

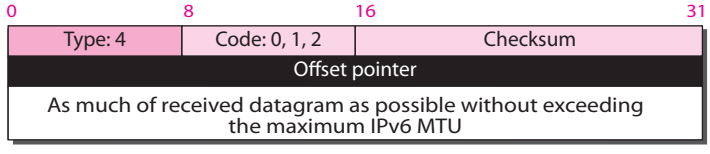


As in version 4, code 0 is used when the datagram is discarded by the router due to a hop-limit field value of zero. Code 1 is used when fragments of a datagram are discarded because other fragments have not arrived within the time limit.

Parameter-Problem Message

Any ambiguity in the header of the datagram can create serious problems as the datagram travels through the Internet. If a router or the destination host discovers any ambiguous or missing value in any field, it discards the datagram and sends a parameter-problem message to the source. The message in ICMPv6 is similar to its version 4 counterpart. However, the type value has been changed to 4 and the size of the offset pointer field has been increased to 4 bytes. There are also three different codes instead of two. Figure 4.43 shows the format of the parameter problem message.

Figure 4.43 *Parameter-problem message*



The code field specifies the reason for discarding the datagram and the cause of failure:

- **Code 0.** Erroneous header field.
- **Code 1.** Unrecognized next header type.
- **Code 2.** Unrecognized IPv6 option.

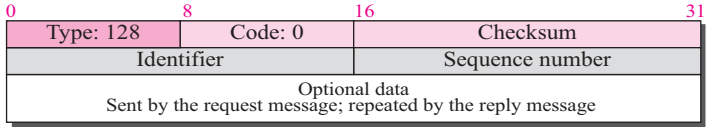
Informational Messages

Two of the ICMPv6 messages can be categorized as informational messages: echo request and echo reply messages. The echo request and echo response messages are designed to check if two devices in the Internet can communicate with each other. A host or router can send an echo request message to another host; the receiving computer or router can reply using the echo response message.

Echo-Request Message

The idea and format of the echo-request message is the same as the one in version 4. The only difference is the value for the type as shown in Figure 4.44.

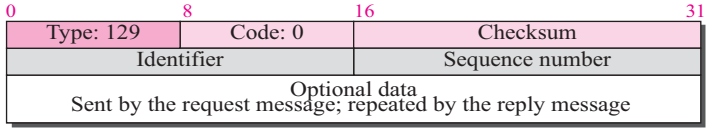
Figure 4.44 *Echo-request message*



Echo-Reply Message

The idea and format of the echo-reply message is the same as the one in version 4. The only difference is the value for the type as shown in Figure 4.45.

Figure 4.45 *Echo-reply messages*



Neighbor-Discovery Messages

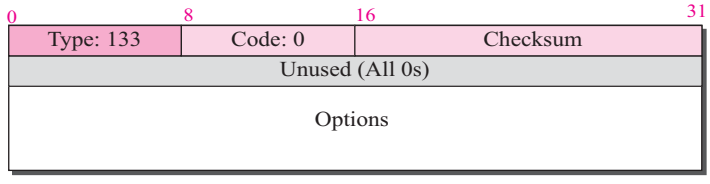
Several messages in the ICMPv6 have been redefined in ICMPv6 to handle the issue of neighbor discovery. Some new messages have also been added to provide extension. The most important issue is the definition of two new protocols that clearly define the functionality of these group messages: the **Neighbor-Discovery (ND) protocol** and the **Inverse-Neighbor-Discovery (IND) protocol.** These two protocols are used by nodes (hosts or routers) on the same link (network) for three main purposes:

1. Hosts use the ND protocol to find routers in the neighborhood that will forward packets for them.
2. Nodes use the ND protocol to find the link layer addresses of neighbors (nodes attached to the same network).
3. Nodes use the IND protocol to find the IPv6 addresses of the neighbor.

Router-Solicitation Message

The idea behind the *router-solicitation* message is the same as in version 4. A host uses the router-solicitation message to find a router in the network that can forward an IPv6 datagram for the host. The only option that is so far defined for this message is the inclusion of physical (data link layer) address of the host to make the response easier for the router. The format of the message is shown in Figure 4.46. The type of this message is 133.

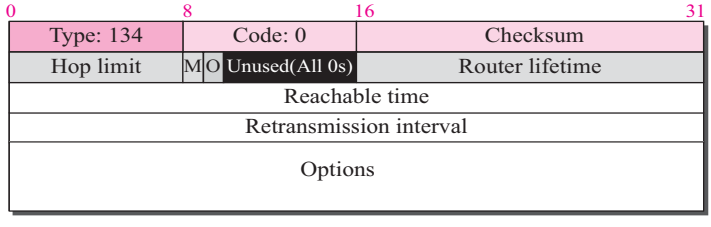
Figure 4.46 *Router-solicitation message*



Router-Advertisement Message

The *router-advertisement* message is sent by a router in response to a router solicitation message. Figure 4.47 shows the format of the router-advertisement message.

Figure 4.47 *Router-advertisement message*



The fields are explained below:

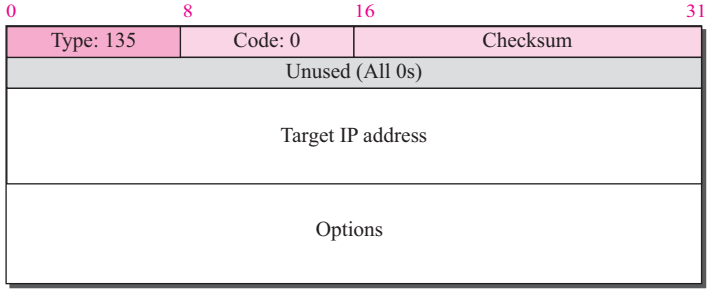
- **Hop Limit.** This 8-bit field limits the number of hops that the requestor should use as the hop limit in its IPv6 datagram.
- **M.** This 1-bit field is the “manage address configuration” field. When this bit is set to 1, the host needs to use the administration configuration.

- **O.** This 1-bit field is the "other address configuration" field. When this bit is set to 1, the host needs to use the appropriate protocol for configuration.
- **Router Lifetime.** This 16-bit field defines the lifetime (in units of seconds) of the router as the default router. When the value of this field is 0, it means that the router is not a default router.
- **Reachable Time.** This 32-bit field defines the time (in units of seconds) that the router is reachable.
- **Retransmission Interval.** This 32-bit field defines the retransmission interval (in units of seconds).
- **Option.** Some possible options are the link layer address of the link from which the message is sent, the MTU of the link, and address prefix information.

Neighbor-Solicitation Message

Figure 4.48 shows the format of **neighbor-solicitation message.** As previously mentioned, the network layer in version 4 contains an independent protocol called Address Resolution Protocol (ARP). In version 6, this protocol is eliminated, and its duties are included in ICMPv6. The neighbor solicitation message has the same duty as the ARP request message. This message is sent when a host or router has a message to send to a neighbor. The sender knows the IP address of the receiver, but needs the data link address of the receiver. The data link address is needed for the IP datagram to be encapsulated in a frame. The only option announces the sender data link address for the convenience of the receiver. The receiver can use the sender data link address to use a unicast response.

Figure 4.48 *Neighbor-solicitation message*

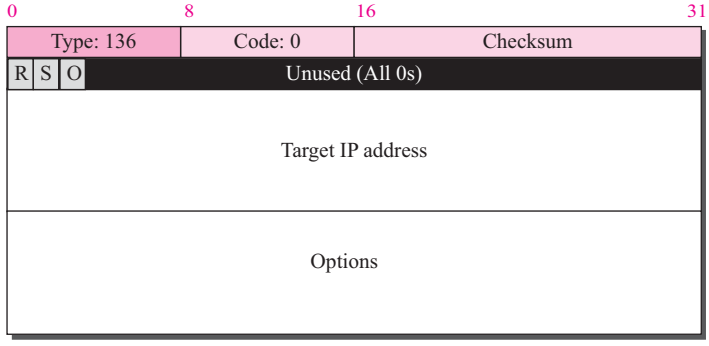


Neighbor-Advertisement Message

The **neighbor-advertisement message** is sent in response to the neighbor-solicitation message. This is equivalent to the ARP reply message in IPv4. Figure 4.49 shows the format of this message.

The fields are explained below:

Figure 4.49 *Neighbor-advertisement message*

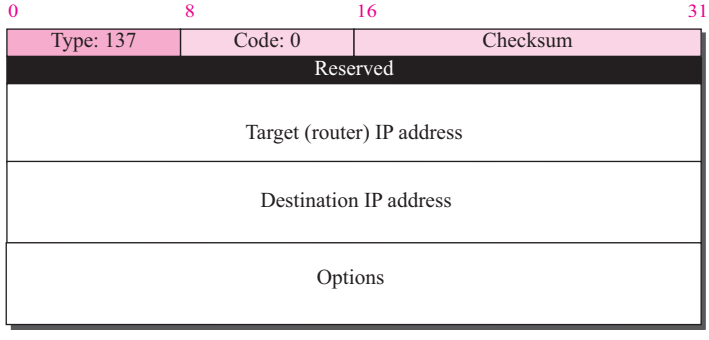


- **R.** This 1-bit field is the "router" flag. When it is set to 1, it means the sender of this message is a router.
- **S.** This 1-bit field is the "solicitation" flag. When it is set to 1, it means that the sender is sending this advertisement in response to a neighbor solicitation. An advertisement can be sent by a host or router without solicitation.
- **O.** This 1-bit field is the "override" flag. When it is set, it means that the advertisement should override existing information in the cache.
- **Option.** The only possible option is the link layer address of the advertiser.

Redirection Message

The purpose of the redirection message is the same as described for version 4. However, the format of the packet now accommodates the size of the IP address in version 6. Also, an option is added to let the host know the physical address of the target router (see Figure 4.50).

Figure 4.50 *Redirection message*

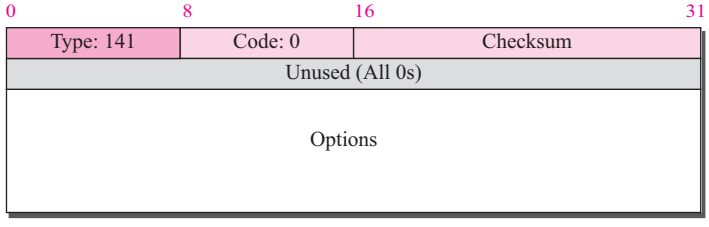


The possible option is the inclusion of the sender data link address and the part of the redirected IP header as long as the total size of the message does not exceed the MTU.

Inverse-Neighbor-Solicitation Message

The **inverse-neighbor-solicitation message** is sent by a node that knows the link layer address of a neighbor, but not the neighbor's IP address. The message is encapsulated in an IPv6 datagram using an all-node multicast address. The sender must send the following two pieces of information in the option field: its link-layer address and the link-layer address of the target node. The sender can also include its IP address and the MTU value for the link. Figure 4.51 shows the format of this message.

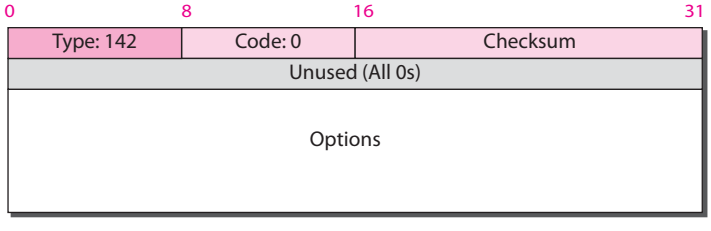
Figure 4.51 *Inverse-neighbor-solicitation message*



Inverse-Neighbor-Advertisement Message

The **inverse-neighbor-advertisement message** is sent in response to the inverse-neighbor-discovery message. The sender of this message must include the link layer address of the sender and the link layer address of the target node in the option section. Figure 4.52 shows the format of this message.

Figure 4.52 *Inverse-neighbor-advertisement message*



Group Membership Messages

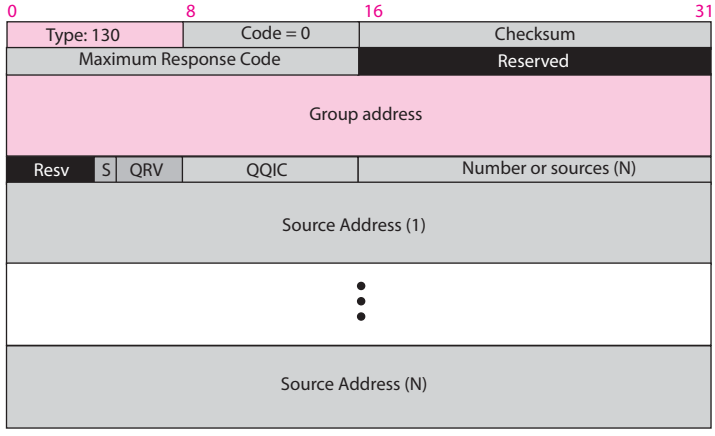
The management of multicast delivery handling in IPv4 is given to the IGMPv3 protocol. In IPv6, this responsibility is given to the **Multicast Listener Delivery** protocol. MLDv1 is the counterpart to IGMPv2; MLDv2 is the counterpart to IGMPv3. The material discussed in this section is taken from RFC 3810. The idea is the same as we discussed in IGMPv3, but the sizes and formats of the messages have been changed to

fit the larger multicast address size in IPv6. Like IGMPv3, MLDv2 has two types of messages: *membership-query message* and *membership- report message*. The first type can be divided into three subtypes: *general*, *group-specific*, and *group-and-source specific*.

Membership-Query Message

A membership-query message is sent by a router to find active group members in the network. Figure 4.53 shows the format of this message.

Figure 4.53 *Membership-query message format*



The fields are almost the same as the ones in IGMPv3 except that the size of the multicast address and the source address has been changed from 32 bits to 128 bits. Another noticeable change in the field size is in the *maximum response code* field, in which the size has been changed from 8 bits to 16 bits. We will discuss this field shortly. Also note that the format of the first 8 bytes matches the format for other ICMPv6 packets because MLDv2 is considered to be part of ICMPv6.

Membership-Report Message

Figure 4.54 shows the format of a membership report message format. Note that the format of the membership report in MLDv2 is exactly the same as the one in IGMPv3 except that the sizes of the fields are changed because of the address size. In particular, the record type is the same as the one defined for IGMPv3 (types 1 to 6).

Functionality

MDLv2 protocol behaves in the same way as IGMPv3. However, there are a few differences that we discuss here.

Calculation of Maximum Response Time As we mentioned, the size of the Max Resp Code in MLV2 is twice the size of the same field in IGMPv3. For this reason, the calculation of maximum response time is slightly different in this protocol as shown in Figure 4.55.

Figure 4.54 *Membership-report message format*

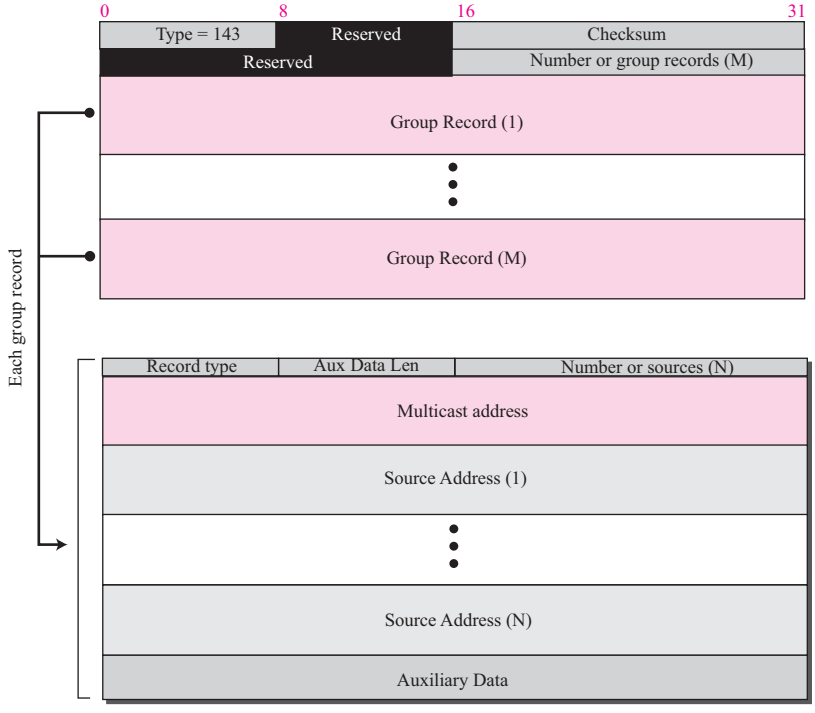
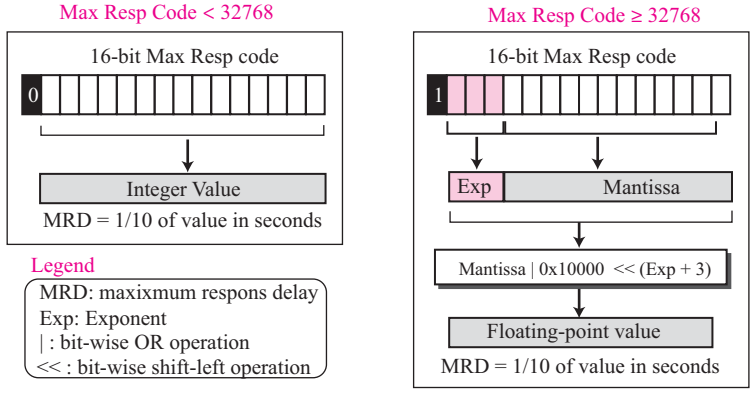


Figure 4.55 *Calculation of maximum response time*



Calculation of Query Interval The calculation of query interval follows the same process as the calculation of maximum response delay; it is calculated from the value of the QQIC field as shown in Figure 4.56.

Figure 4.56 *Calculation of maximum response time*