CHAPTER 9

Internet Control Message Protocol (ICMP)

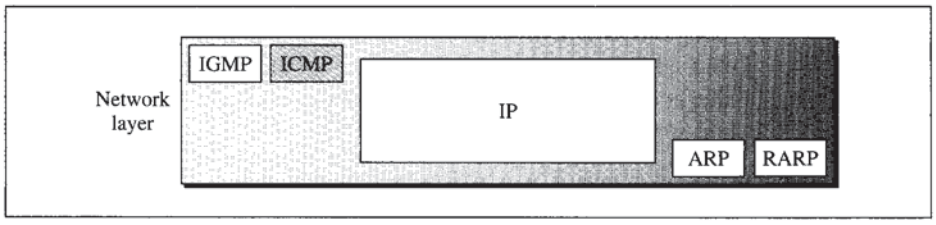
As discussed in Chapter 8, the IP provides unreliable and connectionless datagram delivery. It was designed this way to make efficient use of network resources. The IP protocol is a best-effort delivery service that delivers a datagram from its original source to its final destination. However, it has two deficiencies: lack of error control and lack of assistance mechanisms.

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.

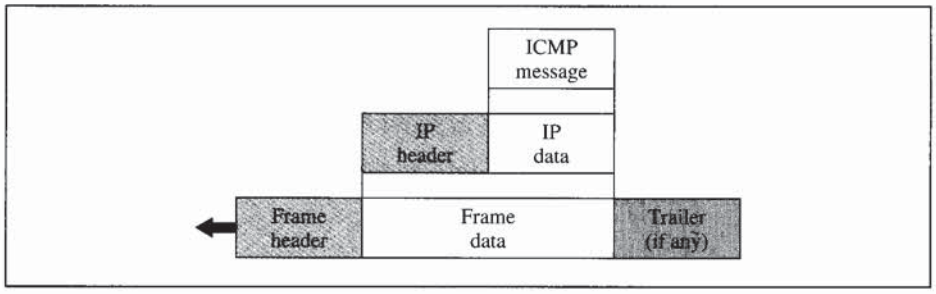
The Internet Control Message Protocol (ICMP) has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol. Figure 9.1 shows the position of ICMP in relation to IP and other protocols in the network layer.

Figure 9.1 *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 9.2).

Figure 9.2 *ICMP encapsulation*

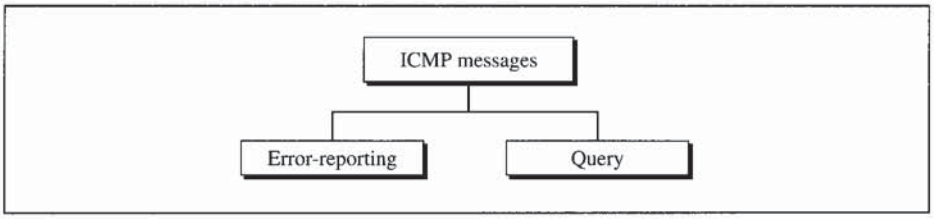


The value of the protocol field in the IP datagram is 1 to indicate that the IP data is an ICMP message.

9.1 TYPES OF MESSAGES

ICMP messages are divided into two broad categories: error-reporting messages and query messages as shown in Figure 9.3.

Figure 9.3 *ICMP 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 9.1 lists the ICMP messages in each category.

Table 9.1 *ICMP messages*

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

Table 9.1 *ICMP messages (continued)*

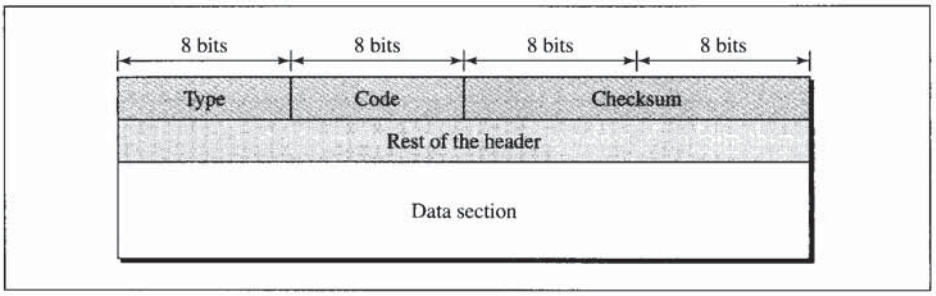
Category	*Type*	*Message*
Query messages	8 or 0	Echo request or reply
	13 or 14	Timestamp request or reply
	17 or 18	Address mask request or reply
	10 or 9	Router solicitation or advertisement

9.2 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 9.4 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 (to be discussed later in the chapter). 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 9.4 *General format of ICMP messages*



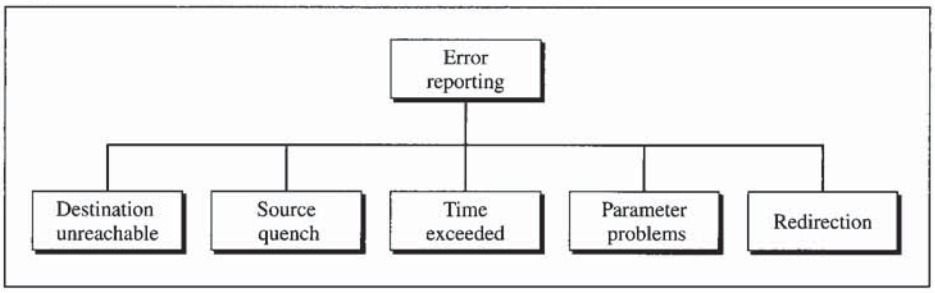
9.3 ERROR REPORTING

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 8, 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.

ICMP always reports error messages to the original source.

Five types of errors are handled: destination unreachable, source quench, time exceeded, parameter problems, and redirection (see Figure 9.5).

Figure 9.5 *Error-reporting messages*



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.
- 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 see in Chapters 11 and 12 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 9.6).

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 9.7 shows the format of the

Figure 9.6 *Contents of data field for the error messages*

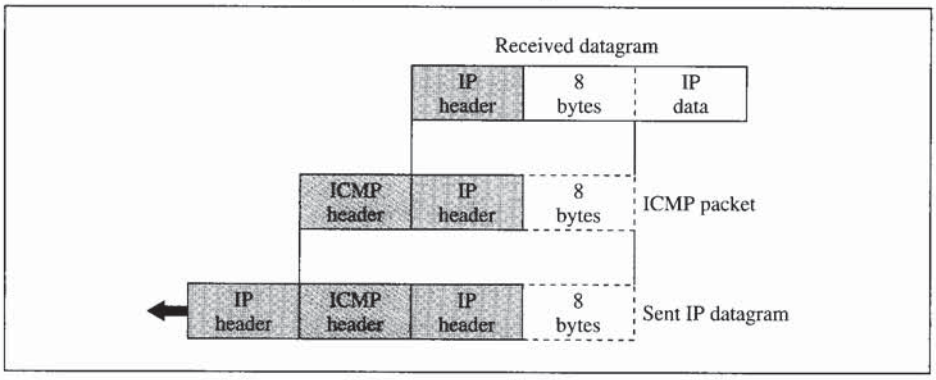
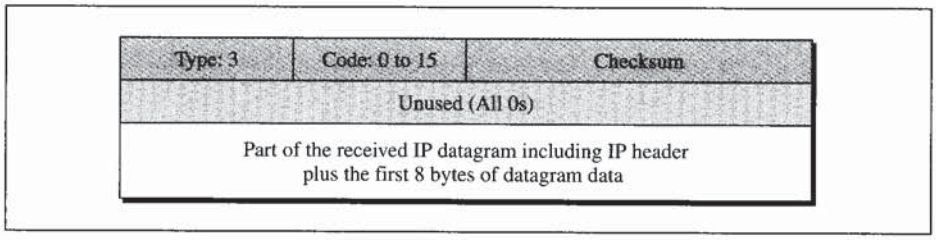


Figure 9.7 *Destination-unreachable format*



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. This type of message can only be generated by a router.
- **Code 1.** The host is unreachable. This can also be due to hardware failure. This type of message can only be generated by a router.
- **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. This type of message is generated only by the destination host.
- **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 precedence 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.

> Destination-unreachable messages with codes 2 or 3 can be created only by the destination host.
>
> Other destination-unreachable messages can be created only 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.

> A router cannot detect all problems that prevent the delivery of a packet.

Source Quench

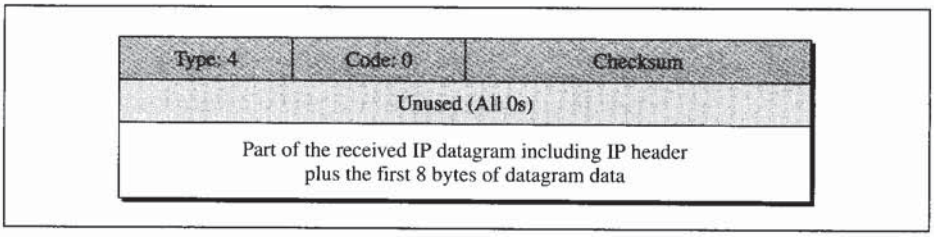
The IP protocol is a connectionless protocol. There is no communication between the source host, which produces the datagram, the routers, which forward it, and the destination host, which processes it. One of the ramifications of this absence of communication is the lack of *flow control*. IP does not have a flow-control mechanism embedded in the protocol. The lack of flow control can create a major problem in the operation of IP: congestion. The source host never knows if the routers or the destination host have been overwhelmed with datagrams. The source host never knows if it is producing datagrams faster than can be forwarded by routers or processed by the destination host.

There is no flow-control mechanism in the IP protocol.

The lack of flow control can create congestion in routers or the destination host. A router or a host has a limited-size queue (buffer) for incoming datagrams waiting to be forwarded (in the case of a router) or to be processed (in the case of a host). If the datagrams are received much faster than they can be forwarded or processed, the queue may overflow. In this case, the router or the host has no choice but to discard some of the datagrams.

The source-quench message in ICMP has been designed to add a kind of flow 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 9.8.

Figure 9.8 *Source-quench format*



A source-quench message informs the source that a datagram has been discarded due to congestion in a router or the destination host.

The source must slow down the sending of datagrams until the congestion is relieved.

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. In this case, 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.

> One source-quench message should be sent for each datagram that is discarded due to congestion.

Time Exceeded

The time-exceeded message is generated in two cases:

- As we saw in Chapter 6, 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 8, 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 is 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.

  > Whenever a router receives a datagram with a time-to-live value of zero, it discards the datagram and sends a time-exceeded message to the original source.

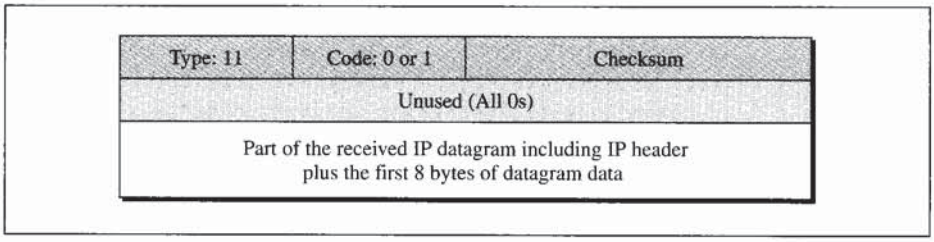
- Second, a time-exceeded message is also generated when all fragments that make up a message do not arrive at the destination host within a certain time limit. When the first fragment arrives, the destination host starts a timer. If all the fragments have not arrived when the time expires, the destination discards all the fragments and sends a time-exceeded message to the original sender.

  > When the final destination does not receive all of the fragments in a set time, it discards the received fragments and sends a time-exceeded message to the original source.

Figure 9.9 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.

In a time-exceeded message, code 0 is used only by routers to show that the value of the time-to-live field is zero. Code 1 is used only by the destination host to show that not all of the fragments have arrived within a set time.

Figure 9.9 *Time-exceeded message format*



Parameter Problem

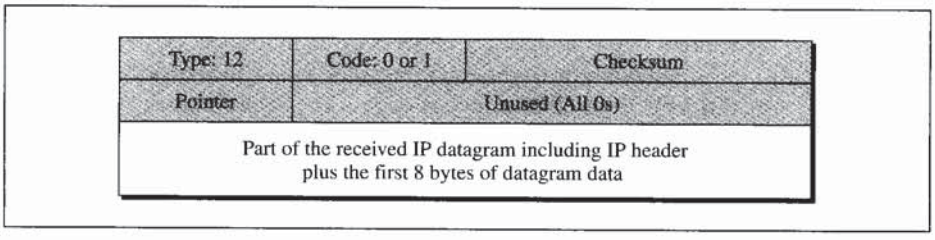
Any ambiguity in the header part of a datagram can create serious problems as the datagram travels through the Internet. If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter-problem message back to the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

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

Figure 9.10 shows the format of the parameter-problem message. The code field in this case specifies the reason for discarding the datagram and shows exactly what has failed:

- **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 9.10 *Parameter-problem message format*

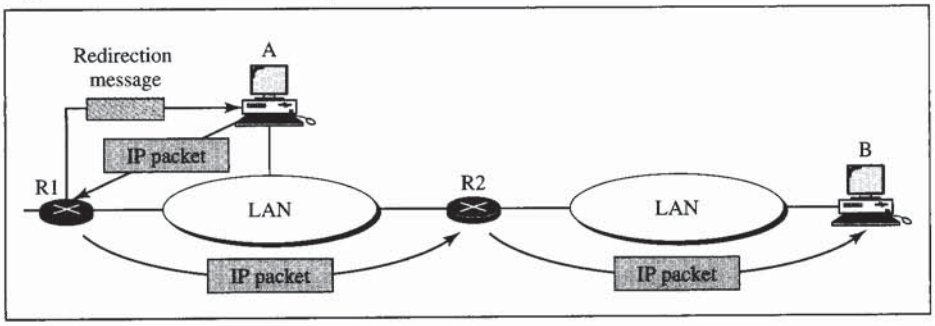


Redirection

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 as we will see in Chapter 13 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 9.11. 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 9.11 *Redirection concept*



A host usually starts with a small routing table that is gradually augmented and updated. One of the tools to accomplish this is the redirection message.

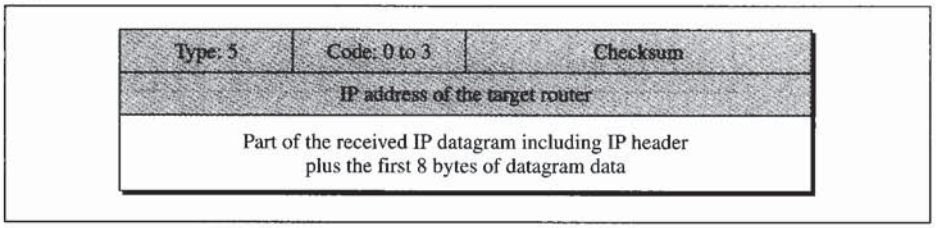
The format of the redirection message is shown in Figure 9.12. Note that the IP address of the appropriate target is given in the second row.

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 the host-specific route based on a specified type of service.

Figure 9.12 *Redirection message format*

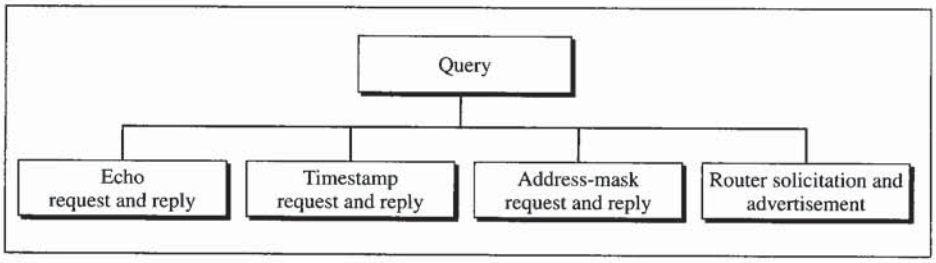


A redirection message is sent from a router to a host on the same local network.

9.4 QUERY

In addition to error reporting, ICMP can also diagnose some network problems. This is accomplished through the query messages, a group of four different pairs of messages, as shown in Figure 9.13. In this type of ICMP message, a node sends a message that is answered in a specific format by the destination node. Note that originally two other types of messages (information request and information reply) were defined, but they are now obsolete. They were designed to allow a host to get its Internet address at start-up; this function is now performed by RARP (see Chapter 7) and BOOTP (see Chapter 17).

Figure 9.13 *Query messages*



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.

> An echo-request message can be sent by a host or router. An echo-reply message is sent by the host or router which receives an echo-request message.

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.

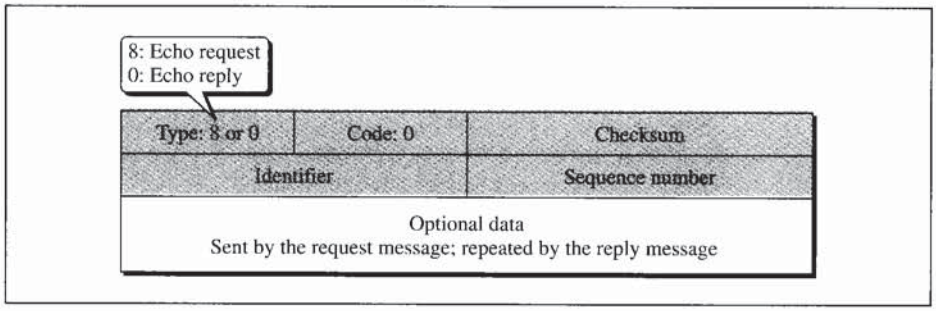
> Echo-request and echo-reply messages can be used by network managers to check the operation of the IP protocol.

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 and echo-reply messages can test the reachability of a host. This is usually done by invoking the ping command.

Echo request, together with echo reply, can validate 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 9.14 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. For example, the identifier field can define a group of problems and the sequence number can keep track of the particular echo-request messages sent. The identifier is often the same as the process ID (see Chapter 15 for the definition of process ID) of the process that originated the request.

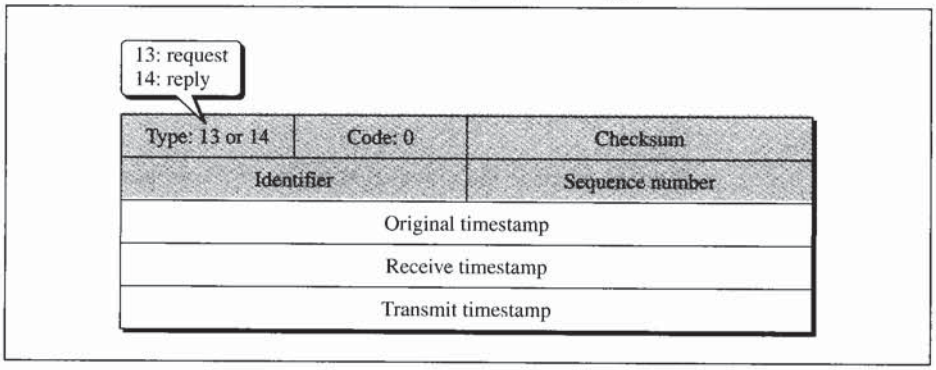
Figure 9.14 *Echo-request and echo-reply messages*



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 9.15.

Figure 9.15 *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 = value of receive timestamp – value of original timestamp
Receiving time = time the packet returned – value of 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.

Timestamp-request and timestamp-reply messages can be used to calculate the round-trip time between a source and a destination machine even if their clocks are not synchronized.

For example, given the following information:

Value of original timestamp: 46
Value of receive timestamp: 59
Value of transmit timestamp: 60
Time the packet arrived: 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

The timestamp-request and timestamp-reply messages can be used to synchronize two clocks in two machines if the exact one-way time duration is known.

Address-Mask Request and Reply

The IP address of a host contains a network address, subnet address, and host identifier. A host may know its full IP address, but it may not know which part of the address defines the network and subnetwork address and which part corresponds to the host identifier. For example, a host may know its 32-bit IP address as

```
10011111 00011111 11100010 10101011
```

But it may not know that the left 20 bits are network and subnetwork addresses and the remaining 12 bits are its host identifier. In this case, the host needs the following mask:

```
11111111 11111111 11110000 00000000
```

The 1s in the mask, as we saw in Chapter 5, identify the position of the bits used for the netid and subnetid. The 0s identify the position of the bits for the hostid.

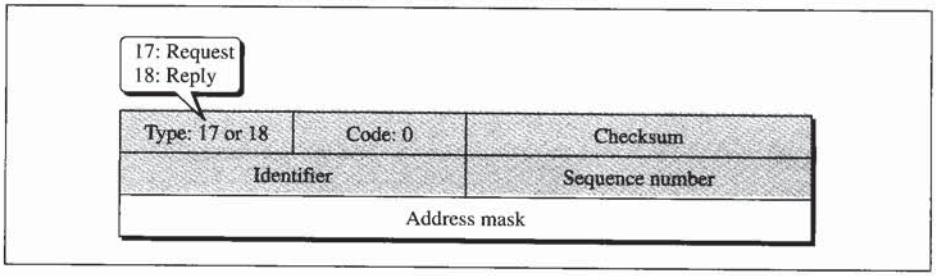
For example, applying the above mask to the above IP address, we get

```
Netid and subnetid ➡ 10011111 00011111 1110
Hostid             ➡ 0010 10101011
```

To obtain its mask, a host sends an address-mask-request message to a router on the LAN. If the host knows the address of the router, it sends the request directly to the router. If it does not know, it broadcasts the message. The router receiving the address-mask-request message responds with an address-mask-reply message, providing the necessary mask for the host. This can be applied to its full IP address to get its subnet address.

The format of the address-mask request and address-mask reply is shown in Figure 9.16. The address-mask field is filled with zeros in the request message. When the router sends the address-mask reply back to the host, this field contains the actual mask (1s for the netid and subnetid and 0s for the hostid).

Figure 9.16 *Mask-request and mask-reply message format*



Masking is needed for diskless stations at start-up time. When a diskless station comes up for the first time, it may ask for its full IP address using the RARP protocol (see Chapter 7); after receiving its IP address, it may use the address-mask request and reply to find out which part of the address defines the subnet.

Another way to get subnet mask information is through the use of the BOOTP protocol, as we will see in Chapter 17.

Router Solicitation and Advertisement

As we discussed in the redirection message section, a host that wants to send data to a host on another network needs to know the address of routers connected to its own network. Also, the host must know if the routers are alive and functioning. The router-solicitation and router-advertisement messages can help in this situation. A host can broadcast (or multicast) a router-solicitation message. The router or routers that receive the solicitation message broadcast their routing information using the router-advertisement message. A router can also periodically send router-advertisement messages even if no host has solicited. Note that when a router sends out an advertisement, it announces not only its own presence but also the presence of all routers on the network of which it is aware. Figure 9.17 shows the format of the router-solicitation message.

Figure 9.17 *Router-solicitation message format*

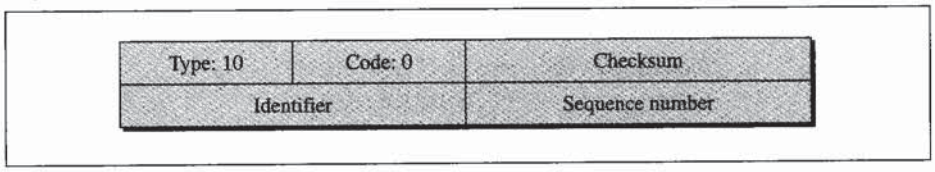
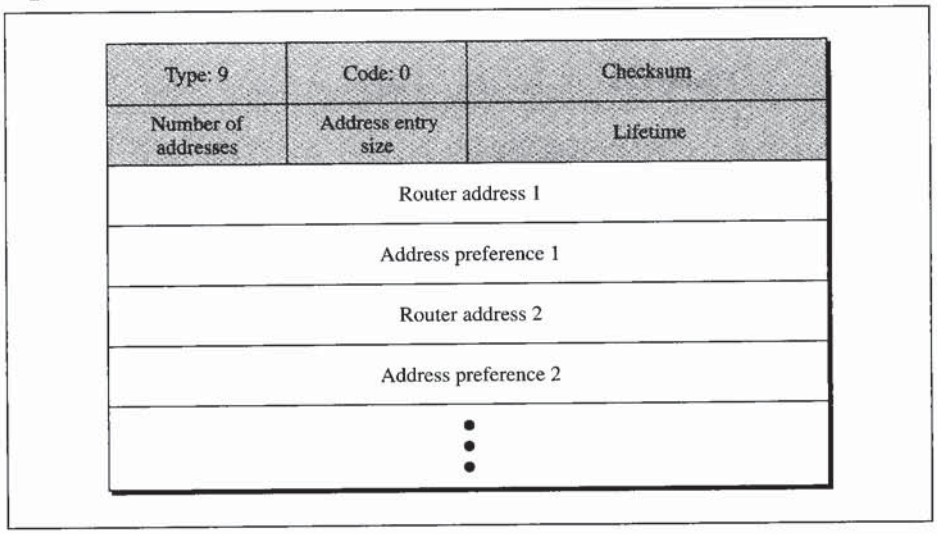


Figure 9.18 shows the format of the router-advertisement message. The lifetime field shows the number of seconds that the entries are considered to be valid. Each router entry in the advertisement contains at least two fields: the router address and the address preference level. The address preference level defines the ranking of the router.

Figure 9.18 *Router-advertisement message format*



The preference level is used to select a router as the default router. If the address preference level is zero, that router is considered the default router. If the address preference level is 80000000_{16}, the router should never be selected as the default router.

9.5 CHECKSUM

In Chapter 7, we learned the concept and idea of the checksum. In ICMP the checksum is calculated over the entire message (header and data).

Checksum Calculation

The sender follows these steps using one's complement arithmetic:

1. The checksum field is set to zero.
2. The sum of all the 16-bit words (header and data) is calculated.
3. The sum is complemented to get the checksum.
4. The checksum is stored in the checksum field.

Checksum Testing

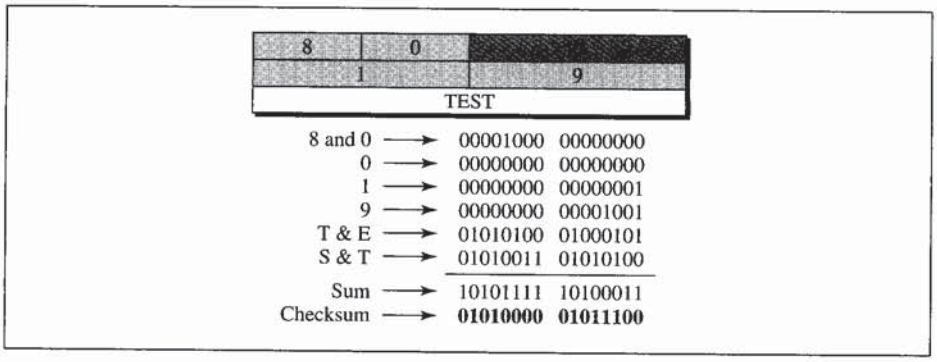
The receiver follows these steps using one's complement arithmetic:

1. The sum of all words (header and data) is calculated.
2. The sum is complemented.
3. If the result obtained in step 2 is 16 0s, the message is accepted; otherwise, it is rejected.

Example

Figure 9.19 shows an example of checksum calculation for a simple echo-request message (see the section on echo request and reply). The message is divided into 16-bit (2-byte) words. The words are added together and the sum is complemented. Now the sender can put this value in the checksum field.

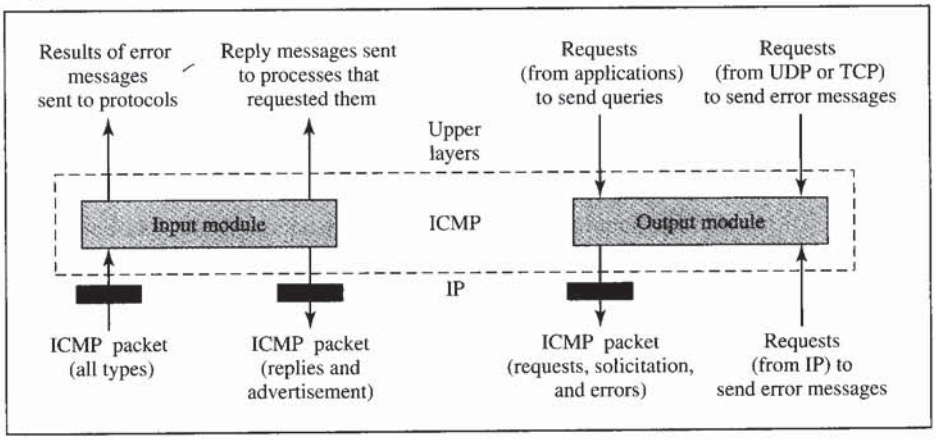
Figure 9.19 *Example of checksum calculation*



9.6 ICMP PACKAGE

To give an idea of how ICMP can handle the sending and receiving of ICMP messages, we present our version of an ICMP package made of two modules: an input module and an output module. Figure 9.20 shows these two modules.

Figure 9.20 *ICMP package*



Input Module

The input module handles all received ICMP messages. It is invoked when an ICMP packet is delivered to it from the IP layer. If the received packet is a request or solicitation, the module creates a reply or an advertisement and sends it out.

If the received packet is a redirection message, the module uses the information to update the routing table. If the received packet is an error message, the module informs the protocol about the situation that caused the error. The pseudocode is shown below:

```
Input Module
Receive: an ICMP packet from the IP layer
1. If (the type is any of the three request types)
    1. Create a reply.
    2. Send the reply.
2. If (the type is a router solicitation)
    1. If (station is a router)
        1. Create a router advertisement.
        2. Send the advertisement.
3. If (the type is one of the three reply messages or router advertisement)
    1. Extract information in the data section of the packet.
    2. Deliver extracted information to the process that requested it.
```

```
Input Module (continued)
4. If (the type defines a redirection)
     1. Modify the routing table.
5. If (the type defines an error message other than a redirection)
     1. Inform the appropriate source protocol about the situation.
6. Return.
```

Output Module

The output module is responsible for creating request, solicitation, or error messages requested by a higher level or the IP protocol. The module receives a demand from IP, UDP, or TCP to send one of the ICMP error messages. If the demand is from IP, the output module must first check that the request is allowed. Remember, an ICMP message cannot be created for four situations: an IP packet carrying an ICMP error message, a fragmented IP packet, a multicast IP packet, or an IP packet having IP address 0.0.0.0 or 127.X.Y. Z.

The output module may also receive a demand from an application program to send one of the ICMP request or solicitation messages. The pseudocode is shown below:

```
Output Module
Receive: a demand
1. If (the demand defines an error message)
     1. If (the demand is from IP )
          1. If (the demand is forbidden)
               1. Return.
     2. If (the type defines a redirection message)
          1. If (the station is not a router)
               1. Return.
     3. Create the error message using the type, the code, and the IP packet.
2. If (the demand defines a request or solicitation)
     1. Create a request or solicitation message.
3. Send the message.
4. Return.
```