A physical address is a local address. Its jurisdiction is a local network. It must be unique locally, but is not necessarily unique universally. It is called a *physical* address because it is usually (but not always) implemented in hardware. An example of a physical address is the 48-bit MAC address in the Ethernet protocol, which is imprinted on the NIC installed in the host or router.

The physical address and the logical address are two different identifiers. We need both because a physical network such as Ethernet can have two different protocols at the network layer such as IP and IPX (Novell) at the same time. Likewise, a packet at a network layer such as IP may pass through different physical networks such as Ethernet and LocalTalk (Apple).

This means that delivery of a packet to a host or a router requires two levels of addressing: logical and physical. We need to be able to map a logical address to its corresponding physical address and vice versa. These can be done by using either static or dynamic mapping.

Static mapping involves in the creation of a table that associates a logical address with a physical address. This table is stored in each machine on the network. Each machine that knows, for example, the IP address of another machine but not its physical address can look it up in the table. This has some limitations because physical addresses may change in the following ways:

1. A machine could change its NIC, resulting in a new physical address.
2. In some LANs, such as LocalTalk, the physical address changes every time the computer is turned on.
3. A mobile computer can move from one physical network to another, resulting in a change in its physical address.

To implement these changes, a static mapping table must be updated periodically. This overhead could affect network performance.

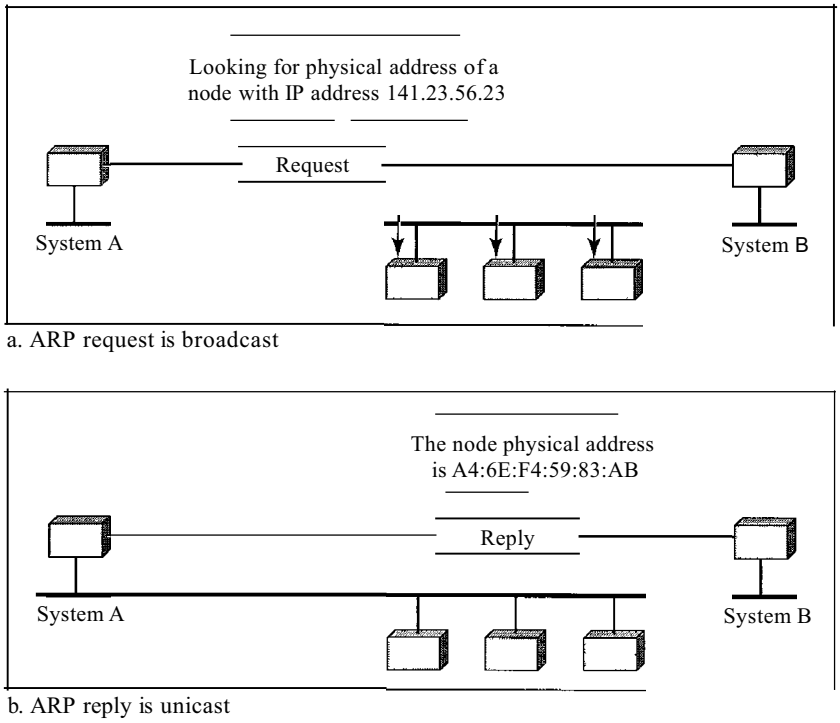
In dynamic mapping each time a machine knows one of the two addresses (logical or physical), it can use a protocol to find the other one.

Mapping Logical to Physical Address: ARP

Anytime a host or a router has an IP datagram to send to another host or router, it has the logical (IP) address of the receiver. The logical (IP) address is obtained from the DNS (see Chapter 25) if the sender is the host or it is found in a routing table (see Chapter 22) if the sender is a router. But the IP datagram must be encapsulated in a frame to be able to pass through the physical network. This means that the sender needs the physical address of the receiver. The host or the router sends an ARP query packet. The packet includes the physical and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the physical address of the receiver, the query is broadcast over the network (see Figure 21.1).

Every host or router on the network receives and processes the ARP query packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and physical addresses. The packet is unicast directly to the inquirer by using the physical address received in the query packet.

Figure 21.1 *ARP operation*



In Figure 21.1a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address 141.23.56.23. System A needs to pass the packet to its data link layer for the actual delivery, but it does not know the physical address of the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IF address of 141.23.56.23.

This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 21.1 b. System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination by using the physical address it received.

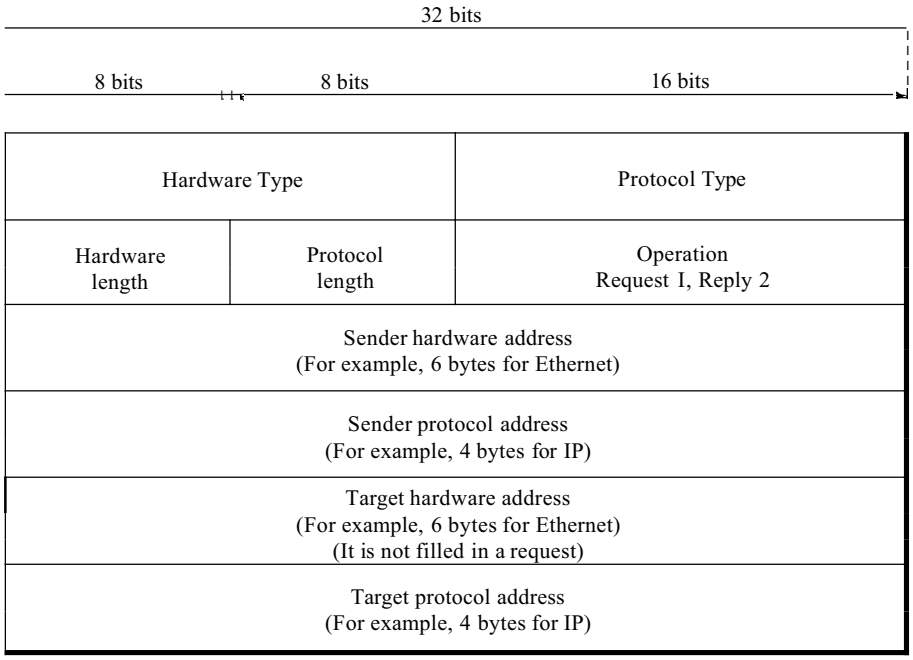
Cache Memory

Using ARP is inefficient if system A needs to broadcast an ARP request for each IP packet it needs to send to system B. It could have broadcast the IP packet itself. ARP can be useful if the ARP reply is cached (kept in cache memory for a while) because a system normally sends several packets to the same destination. A system that receives an ARP reply stores the mapping in the cache memory and keeps it for 20 to 30 minutes unless the space in the cache is exhausted. Before sending an ARP request, the system first checks its cache to see if it can find the mapping.

Packet Format

Figure 21.2 shows the format of an ARP packet.

Figure 21.2 *ARP packet*



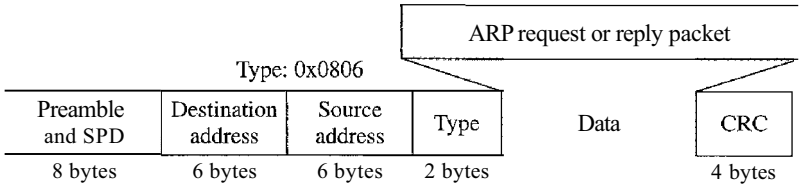
Thc fields arc as follows:

- Hardware type. This is a 16-bit field defining the type of the network on which ARP is running. Each LAN has been assigned an integer based on its type. For example, Ethernet is given type 1. ARP can be used on any physical network.
- Protocol type. This is a 16-bit field defining the protocol. For example, the value of this field for the IPv4 protocol is 0800_{16}, ARP can be used with any higher-level protocol.
- Hardware length. This is an 8-bit field defining the length of the physical address in bytes. For example, for Ethernet the value is 6.
- Protocol length. This is an 8-bit field defining the length of the logical address in bytes. For example, for the IPv4 protocol the value is 4.
- Operation. This is a 16-bit field defining the type of packet. Two packet types are defined: ARP request (1) andARP reply (2).
- Sender hardware address. This is a variable-length field defining the physical address of the sender. For example, for Ethernet this field is 6 bytes long.
- Sender protocol address. This is a variable-length field defining the logical (for example, IP) address of the sender. For the IP protocol, this field is 4 bytes long.
- Target hardware address. This is a variable-length field defining the physical address of the target. For example, for Ethernet this field is 6 bytes long. For an ARP request message, this field is alIOs because the sender does not know the physical address of the target.
- Target protocol address. This is a variable-length field defining the logical (for example, IP) address of the target. For the IPv4 protocol, this field is 4 bytes long.

Encapsulation

An ARP packet is encapsulated directly into a data link frame. For example, in Figure 21.3 an ARP packet is encapsulated in an Ethernet frame. Note that the type field indicates that the data carried by the frame are an ARP packet.

Figure 21.3 *Encapsulation of ARP packet*



Operation

Let us see how ARP functions on a typical internet. First we describe the steps involved. Then we discuss the four cases in which a host or router needs to use ARP. These are the steps involved in an ARP process:

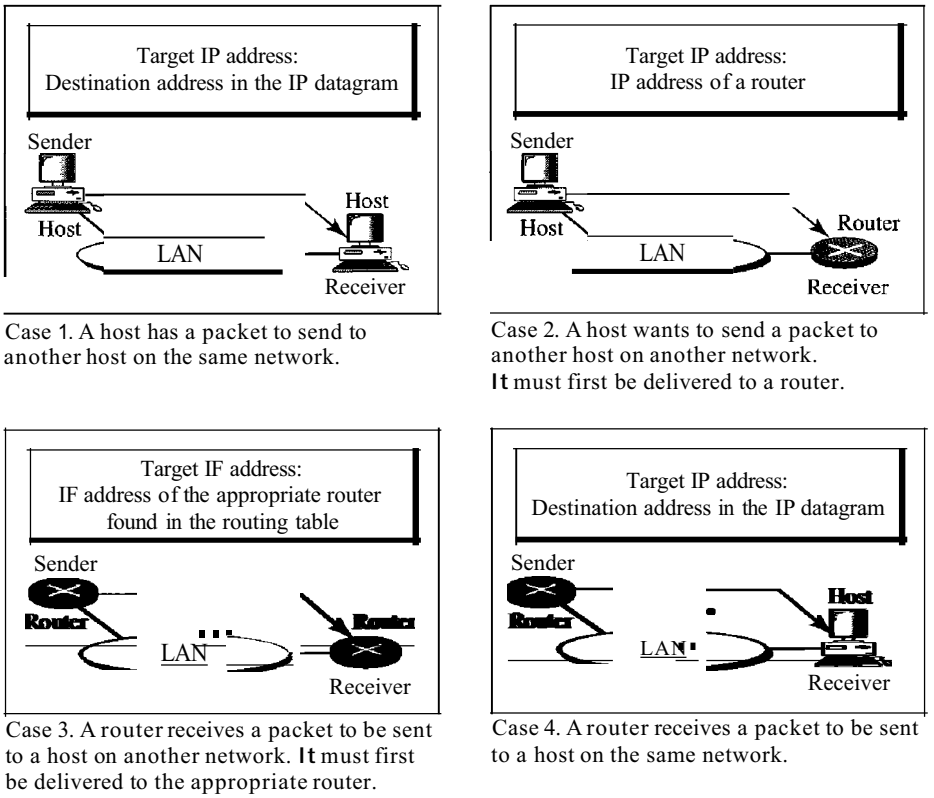
1. The sender knows the IP address of the target. We will see how the sender obtains this shortly.
2. IP asks ARP to create an ARP request message, filling in the sender physical address, the sender IP address, and the target IP address. The target physical address field is filled with 0s.
3. The message is passed to the data link layer where it is encapsulated in a frame by using the physical address of the sender as the source address and the physical broadcast address as the destination address.
4. Every host or router receives the frame. Because the frame contains a broadcast destination address, all stations remove the message and pass it to ARP. All machines except the one targeted drop the packet. The target machine recognizes its IP address.
5. The target machine replies with an ARP reply message that contains its physical address. The message is unicast.
6. The sender receives the reply message. It now knows the physical address of the target machine.
7. The IP datagram, which carries data for the target machine, is now encapsulated in a frame and is unicast to the destination.

Four Different Cases

The following are four different cases in which the services of ARP can be used (see Figure 21.4).

1. The sender is a host and wants to send a packet to another host on the same network. In this case, the logical address that must be mapped to a physical address is the destination IP address in the datagram header.

Figure 21.4 *Four cases using ARP*



Case 1. A host has a packet to send to another host on the same network.

Case 2. A host wants to send a packet to another host on another network. It must first be delivered to a router.

Case 3. A router receives a packet to be sent to a host on another network. It must first be delivered to the appropriate router.

Case 4. A router receives a packet to be sent to a host on the same network.

2. The sender is a host and wants to send a packet to another host on another network. In this case, the host looks at its routing table and finds the IP address of the next hop (router) for this destination. If it does not have a routing table, it looks for the IP address of the default router. The IP address of the router becomes the logical address that must be mapped to a physical address.
3. The sender is a router that has received a datagram destined for a host on another network. It checks its routing table and finds the IP address of the next router. The IP address of the next router becomes the logical address that must be mapped to a physical address.
4. The sender is a router that has received a datagram destined for a host on the same network. The destination IP address of the datagram becomes the logical address that must be mapped to a physical address.

An ARP request is broadcast; an ARP reply is unicast.

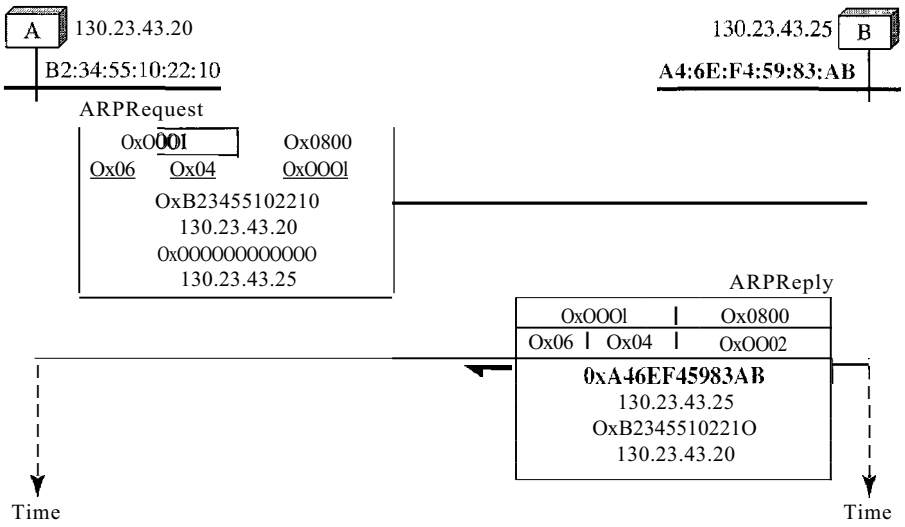
Example 21.1

A host with IP address 130.23.43.20 and physical address B2:34:55: 10:22: 10 has a packet to send to another host with IP address 130.23.43.25 and physical address A4:6E:F4:59:83:AB (which is unknown to the first host). The two hosts are on the same Ethernet network. Show the ARP request and reply packets encapsulated in Ethernet frames.

Solution

Figure 21.5 shows the ARP request and reply packets. Note that the ARP data field in this case is 28 bytes, and that the individual addresses do not fit in the 4-byte boundary. That is why we do not show the regular 4-byte boundaries for these addresses.

Figure 21.5 *Example* 21.1, *anARP request and reply*

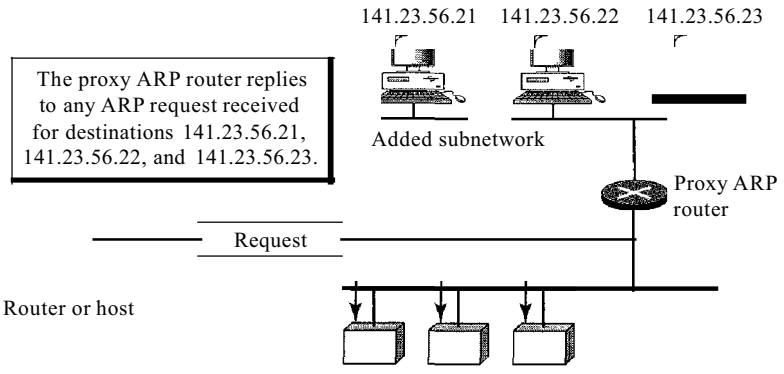


ProxyARP

A technique called *proxy* ARP is used to create a subnetting effect. A proxy ARP is an ARP that acts on behalf of a set of hosts. Whenever a router running a proxy ARP receives an ARP request looking for the IP address of one of these hosts, the router sends an ARP reply announcing its own hardware (physical) address. After the router receives the actual IP packet, it sends the packet to the appropriate host or router.

Let us give an example. In Figure 21.6 the ARP installed on the right-hand host will answer only to anARP request with a target IP address of 141.23.56.23.

Figure 21.6 *Proxy ARP*



However, the administrator may need to create a subnet without changing the whole system to recognize subnetted addresses. One solution is to add a router running a proxy ARP. In this case, the router acts on behalf of all the hosts installed on the subnet. When it receives an ARP request with a target IP address that matches the address of one of its protégés (141.23.56.21, 141.23.56.22, or 141.23.56.23), it sends an ARP reply and announces its hardware address as the target hardware address. When the router receives the IP packet, it sends the packet to the appropriate host.

Mapping Physical to Logical Address: RARP, BOOTP, and DHCP

There are occasions in which a host knows its physical address, but needs to know its logical address. This may happen in two cases:

1. A diskless station is just booted. The station can find its physical address by checking its interface, but it does not know its IP address.
2. An organization does not have enough IP addresses to assign to each station; it needs to assign IP addresses on demand. The station can send its physical address and ask for a short time lease.

RARP

Reverse Address Resolution Protocol (RARP) finds the logical address for a machine that knows only its physical address. Each host or router is assigned one or more logical (IP) addresses, which are unique and independent of the physical (hardware) address of the machine. To create an IP datagram, a host or a router needs to know its own IP address or addresses. The IP address of a machine is usually read from its configuration file stored on a disk file.

However, a diskless machine is usually booted from ROM, which has minimum booting information. The ROM is installed by the manufacturer. It cannot include the IP address because the IP addresses on a network are assigned by the network administrator.

The machine can get its physical address (by reading its NIC, for example), which is unique locally. It can then use the physical address to get the logical address by using the RARP protocol. A RARP request is created and broadcast on the local network. Another machine on the local network that knows all the IP addresses will respond with a RARP reply. The requesting machine must be running a RARP client program; the responding machine must be running a RARP server program.

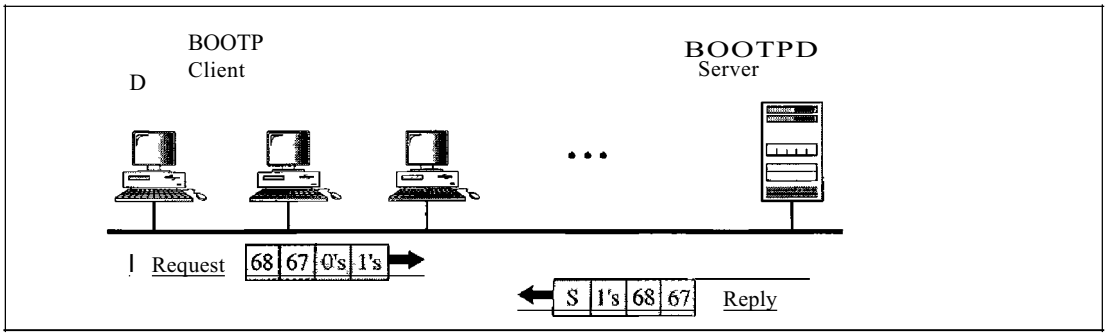
There is a serious problem with RARP: Broadcasting is done at the data link layer. The physical broadcast address, all is in the case of Ethernet, does not pass the boundaries of a network. This means that if an administrator has several networks or several subnets, it needs to assign a RARP server for each network or subnet. This is the reason that RARP is almost obsolete. Two protocols, BOOTP and DHCp, are replacing RARP.

BOOTP

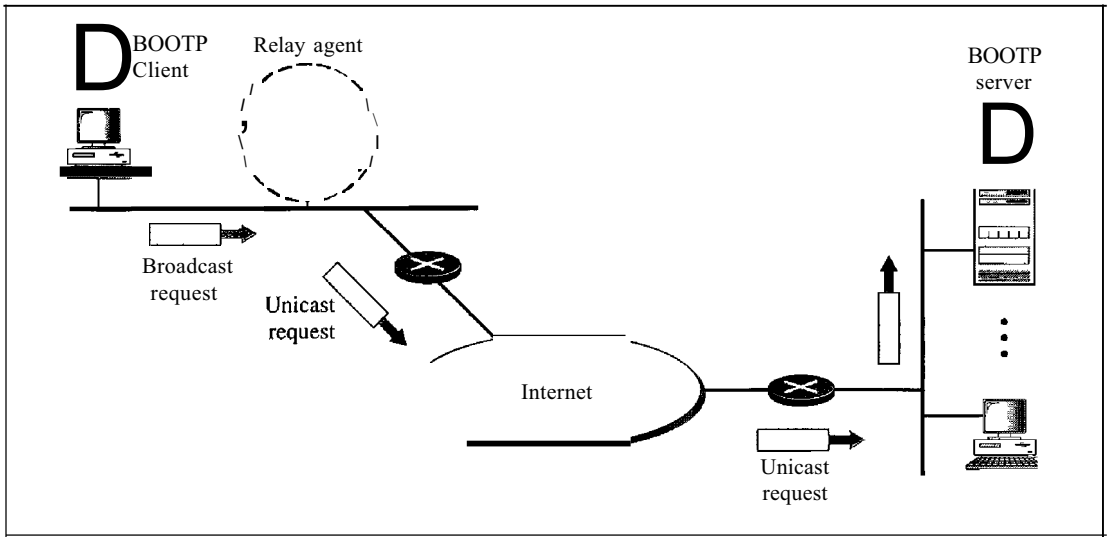
The Bootstrap Protocol (BOOTP) is a client/server protocol designed to provide physical address to logical address mapping. BOOTP is an application layer protocol. The administrator may put the client and the server on the same network or on different

networks, as shown in Figure 21.7. BOOTP messages are encapsulated in a UDP packet, and the UDP packet itself is encapsulated in an IP packet.

Figure 21.7 *BOOTP client and server on the same and different network*



a. Client and server on the same network



b. Client and server on different networks

The reader may ask how a client can send an IP datagram when it knows neither its own IP address (the source address) nor the server's IP address (the destination address). The client simply uses all as as the source address and a11Is as the destination address.

One of the advantages of BOOTP over RARP is that the client and server are application-layer processes. As in other application-layer processes, a client can be in one network and the server in another, separated by several other networks. However, there is one problem that must be solved. The BOOTP request is broadcast because the client does not know the IP address of the server. A broadcast IP datagram cannot pass through any router. To solve the problem, there is a need for an intermediary. One of the hosts (or a router that can be configured to operate at the application layer) can be used as a relay. The host in this case is called a relay agent. The relay agent knows the unicast address of a BOOTP server. When it receives this type of packet, it encapsulates the message in a unicast datagram and sends the request to the BOOTP server. The packet,

carrying a unicast destination address, is routed by any router and reaches the BOOTP server. The BOOTP server knows the message comes from a relay agent because one of the fields in the request message defines the IP address of the relay agent. The relay agent, after receiving the reply, sends it to the BOOTP client.

DHCP

BOOTP is not a dynamic configuration protocol. When a client requests its IP address, the BOOTP server consults a table that matches the physical address of the client with its IP address. This implies that the binding between the physical address and the IP address of the client already exists. The binding is predetermined.

However, what if a host moves from one physical network to another? What if a host wants a temporary IP address? BOOTP cannot handle these situations because the binding between the physical and IP addresses is static and fixed in a table until changed by the administrator. BOOTP is a static configuration protocol.

The Dynamic Host Configuration Protocol (DHCP) has been devised to provide static and dynamic address allocation that can be manual or automatic.

DHCP provides static and dynamic address allocation that can be manual or automatic.

Static Address Allocation In this capacity DHCP acts as BOOTP does. It is backward-compatible with BOOTP, which means a host running the BOOTP client can request a static address from a DHCP server. A DHCP server has a database that statically binds physical addresses to IP addresses.

Dynamic Address Allocation DHCP has a second database with a pool of available IP addresses. This second database makes DHCP dynamic. When a DHCP client requests a temporary IP address, the DHCP server goes to the pool of available (unused) IP addresses and assigns an IP address for a negotiable period of time.

When a DHCP client sends a request to a DHCP server, the server first checks its static database. If an entry with the requested physical address exists in the static database, the permanent IP address of the client is returned. On the other hand, if the entry does not exist in the static database, the server selects an IP address from the available pool, assigns the address to the client, and adds the entry to the dynamic database.

The dynamic aspect of DHCP is needed when a host moves from network to network or is connected and disconnected from a network (as is a subscriber to a service provider). DHCP provides temporary IP addresses for a limited time.

The addresses assigned from the pool are temporary addresses. The DHCP server issues a lease for a specific time. When the lease expires, the client must either stop using the IP address or renew the lease. The server has the option to agree or disagree with the renewal. If the server disagrees, the client stops using the address.

Manual and Automatic Configuration One major problem with the BOOTP protocol is that the table mapping the IP addresses to physical addresses needs to be manually configured. This means that every time there is a change in a physical or IP address, the administrator needs to manually enter the changes. DHCP, on the other hand, allows both manual and automatic configurations. Static addresses are created manually; dynamic addresses are created automatically.

21.2 ICMP

As discussed in Chapter 20, 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 administrator 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.

Types of Messages

ICMP messages are divided into two broad categories: error-reporting messages and query messages.

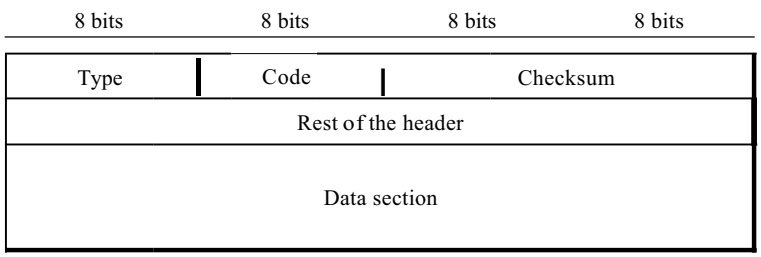
The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet.

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.

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 21.8 shows, the first field, ICMP type, defines the type of the

Figure 21.8 *General format of ICMP messages*



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.

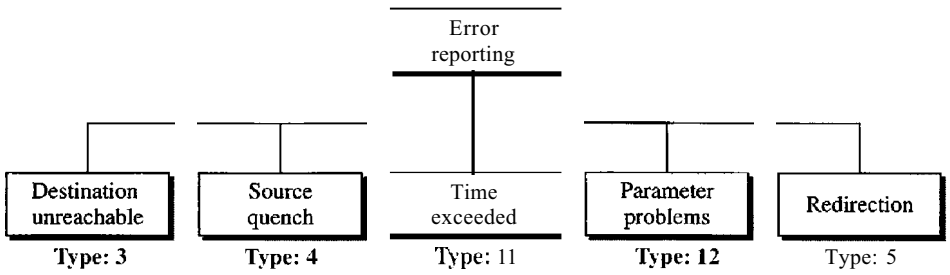
Error Reporting

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 20, 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 21.9).

Figure 21.9 *Error-reporting messages*

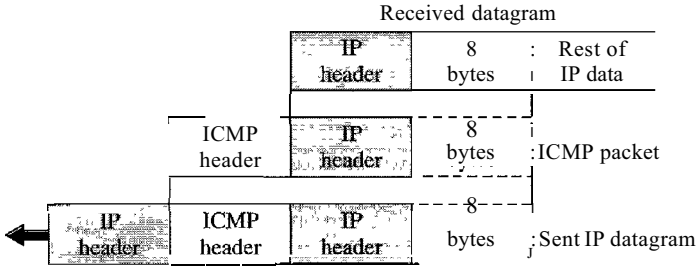


The following are important points about ICMP error messages:

- 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 IeMP 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 Chapter 23 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 21.10).

Figure 21.10 *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. Note that destination-unreachable messages can be created by either a router or the destination host.

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 has 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.

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 was 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.

Time Exceeded

The time-exceeded message is generated in two cases: As we see in Chapter 22, 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 20, 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 not all fragments that make up a message arrive at the destination host within a certain time limit.

Parameter Problem

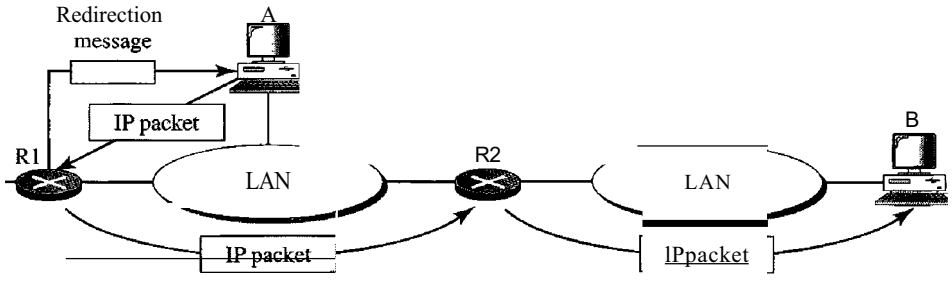
Any ambiguity in the header part of a datagram can Create serious problems as the datagram travels through the Internet. If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter-problem message back to the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

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 22, 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 the IP address of only 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 21.11. Host A wants to send a datagram to host B.

Figure **21.11** *Redirection concept*



Router R2 is obviously the most efficient routing choice, but host A did not choose router R2. The datagram goes to R1 instead. Router 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.

Query

In addition to error reporting, ICMP can diagnose some network problems. This is accomplished through the query messages, a group of four different pairs of messages, as shown in Figure 21.12. In this type of ICMP message, a node sends a message that is answered in a specific format by the destination node. A query message is encapsulated in an IP packet, which in tum is encapsulated in a data link layer frame. However, in this case, no bytes of the original IP are included in the message, as shown in Figure 21.13.

Figure 21.12 *Query messages*

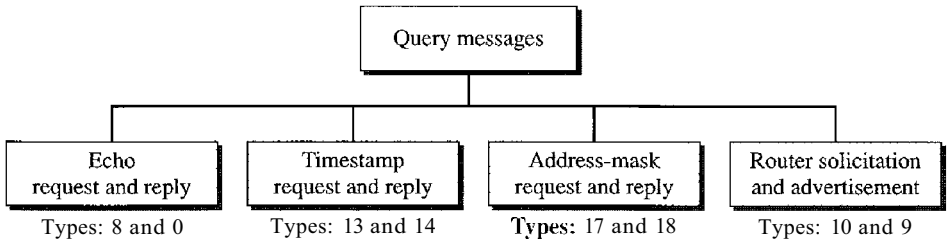
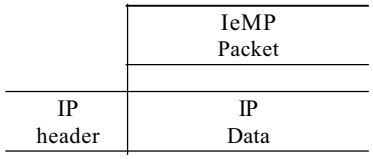


Figure 21.13 *Encapsulation of ICMP query messages*



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 detennines whether two systems (hosts or routers) can communicate with each other. 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. 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. We will see the use of this program at the end of the chapter.

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.

Address-Mask Request and Reply

A host may know its IP address, but it may not know the corresponding mask. For example, a host may know its IP address as 159.31.17.24, but it may not know that the corresponding mask is /24. 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.

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.

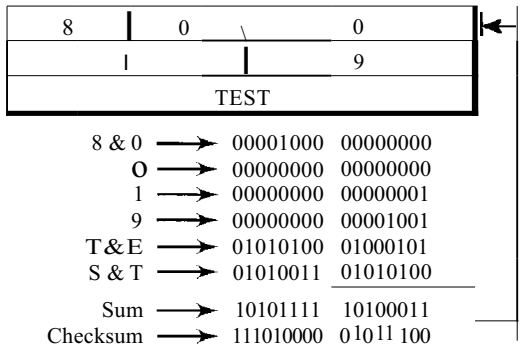
Checksum

In Chapter 10, we learned the concept and idea of the checksum. In ICMP the checksum is calculated over the entire message (header and data).

Example 21.2

Figure 21.14 shows an example of checksum calculation for a simple echo-request message. We randomly chose the identifier to be 1 and the sequence number to be 9. The message is divided

Figure 21.14 *Example of checksum calculation*



into 16-bit (2-byte) words. The words are added and the sum is complemented. Now the sender can put this value in the checksum field.

Debugging Tools

There are several tools that can be used in the Internet for debugging. We can determine the viability of a host or router. We can trace the route of a packet. We introduce two tools that use ICMP for debugging: *ping* and *traceroute.* We will introduce more tools in future chapters after we have discussed the corresponding protocols.

Ping

We can use the *ping* program to find if a host is alive and responding. We use *ping* here to see how it uses ICMP packets.

The source host sends ICMP echo-request messages (type: 8, code: 0); the destination, if alive, responds with ICMP echo-reply messages. The *ping* program sets the identifier field in the echo-request and echo-reply message and stans the sequence number from 0; this number is incremented by 1 each time a new message is sent. Note that *ping* can calculate the round-trip time. It inserts the sending time in the data section of the message. When the packet arrives, it subtracts the arrival time from the departure time to get the round-trip time (RTT).

Example 21.3

We use the *ping* program to test the server fhda.edu. The result is shown below:

```
$ ping thda.edu
PING thda.edu (153.18.8.1)    56 (84)  bytes of data.
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=O     ttl=62     time=1.91 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=l     ttl=62     time=2.04 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=2     ttl=62     time=1.90 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=3     ttl=62     time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=4     ttl=62     time=1.93 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=5     ttl=62     time=2.00 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=6     tt1=62     time=1.94 ms
64 bytes from tiptoe.:fhda.edu (153.18.8.1): icmp_seq=7    ttl=62     time=1.94 ms
64 bytes from tiptoe.:fhda.edu (153.18.8.1): icmp_seq=8    ttl=62     time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=9     ttl=62     time=1.89 ms
64 bytes from tiptoe.:fhda.edu (153.18.8.1): icmp_seq=lO    ttl=62     time=1.98 ms

--- thda.edu ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10103ms
    rtt minJavg/max = 1.899/1.955/2.041 ms
```

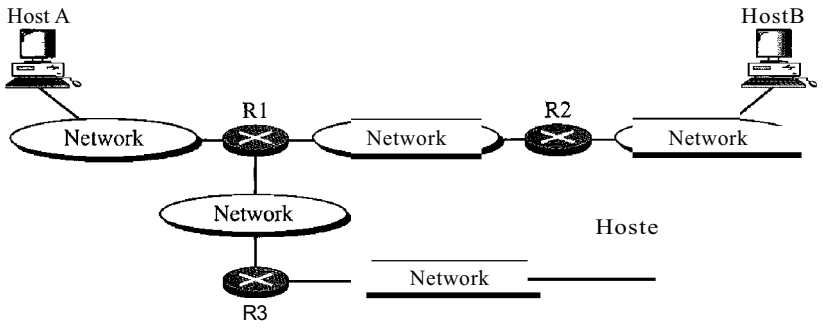
The *ping* program sends messages with sequence numbers starting from 0. For each probe it gives us the RTT time. The TTL (time to live) field in the IP datagram that encapsulates an ICMP message has been set to 62, which means the packet cannot travel mare than 62 hops. At the beginning, *ping* defines the number of data bytes as 56 and the total number of bytes as 84. It is obvious that if we add 8 bytes of ICMP header and 20 bytes of IP header to 56, the result is 84. However,

note that in each probe *ping* defines the number of bytes as 64. This is the total number of bytes in the ICMP packet (56 + 8). The *ping* program continues to send messages, if we do not stop it by using the interrupt key (ctrl + c, for example). After it is interrupted, it prints the statistics of the probes. It tells us the number of packets sent, the number of packets received, the total time, and the RTT minimum, maximum, and average. Some systems may print more infonnation.

Tracerollte

The *traceroute* program in UNIX or *tracert* in Windows can be used to trace the route of a packet from the source to the destination. We have seen an application of the *traceroute* program to simulate the loose source route and strict source route options of an IP datagram in Chapter 20. We use this program in conjunction with ICMP packets in this chapter. The program elegantly uses two ICMP messages, time exceeded and destination unreachable, to find the route of a packet. This is a program at the application level that uses the services of UDP (see Chapter 23). Let us show the idea of the *traceroute* program by using Figure 21.15.

Figure 21.15 *The traceroute program operation*



Given the topology, we know that a packet from host A to host B travels through routers R1 and R2. However, most of the time, we are not aware of this topology. There could be several routes from A to B. The *traceroute* program uses the ICMP messages and the TTL (time to live) field in the IP packet to find the route.

1. The *traceroute* program uses the following steps to find the address of the router R1 and the round-trip time between host A and router R1.
   a. The *traceroute* application at host A sends a packet to destination Busing UDP; the message is encapsulated in an IP packet with a TTL value of 1. The program notes the time the packet is sent.
   b. Router R1 receives the packet and decrements the value of TTL to 0. It then discards the packet (because TTL is 0). The router, however, sends a time-exceeded ICMP message (type: 11, code: 0) to show that the TTL value is 0 and the packet was discarded.
   c. The *traceroute* program receives the ICMP messages and uses the destination address of the IP packet encapsulating ICMP to find the IP address of router R1. The program also makes note of the time the packet has arrived. The difference between this time and the time at step a is the round-trip time.

The *traceroute* program repeats steps a to c three times to get a better average round-trip time. The first trip time may be much longer than the second or third because it takes time for the ARP program to find the physical address of router RI. For the second and third trips, ARP has the address in its cache.

2. The *traceroute* program repeats the previous steps to find the address of router R2 and the round-trip time between host A and router R2. However, in this step, the value of TTL is set to 2. So router RI forwards the message, while router R2 discards it and sends a time-exceeded ICMP message.
3. The *traceroute* program repeats step 2 to find the address of host B and the round-trip time between host A and host B. When host B receives the packet, it decrements the value of TIL, but it does not discard the message since it has reached its final destination. How can an ICMP message be sent back to host A? The *traceroute* program uses a different strategy here. The destination port of the UDP packet is set to one that is not supported by the UDP protocol. When host B receives the packet, it cannot find an application program to accept the delivery. It discards the packet and sends an ICMP destination-unreachable message (type: 3, code: 3) to host A. Note that this situation does not happen at router RI or R2 because a router does not check the UDP header. The *traceroute* program records the destination address of the arrived IP datagram and makes note of the round-trip time. Receiving the destination-unreachable message with a code value 3 is an indication that the whole route has been found and there is no need to send more packets.

Example 21.4

We use the *traceroute* program to find the route from the computer voyager.deanza.edu to the server fhda.edu. The following shows the result:

```
$ traceroute fbda.edu
traceroute to fbda.edu   (153.18.8.1),30 hops max, 38 byte packets
 1 Dcore.fhda.edu        (153.18.31.254)   0.995 ms   0.899 ms   0.878 ms
 2 Dbackup.fhda.edu      (153.18.251.4)    1.039 fiS  1.064 fiS  1.083 fiS
 3 tiptoe.fhda.edu       (153.18.8.1)      1.797 illS 1.642 ms   1.757 fiS
```

The unnumbered line after the command shows that the destination is 153.18.8.1. The TIL value is 30 hops. The packet contains 38 bytes: 20 bytes of IP header, 8 bytes of UDP header, and 10 bytes of application data. The application data are used by *traceroute* to keep track of the packets.

The first line shows the first router visited. The router is named Dcore.fhda.edu with IP address 153.18.31.254. The first round-trip time was 0.995 ms, the second was 0.899 ms, and the third was 0.878 ms.

The second line shows the second router visited. The router is named Dbackup.fhda.edu with IP address 153.18.251.4. The three round-trip times are also shown.

The third line shows the destination host. We know that this is the destination host because there are no more lines. The destination host is the server thda.edu, but it is named tiptoe.fhda.edu with the IP address 153.18.8.1. The three round-trip times are also shown.

Example 21.5

In this example, we trace a longer route, the route to xerox.com.

```
$ traceroute xerox.com
traceroute to xerox.com (13.1.64.93), 30 hops max, 38 byte packets
 1  Dcore.fbda.edu      (153.18.31.254)     0.622 ms   0.891 ms   0.875 ms
 2  Ddmz.fbda.edu       (153.18.251.40)     2.132 ms   2.266 ms   2.094ms
 3  Cinic.fhda.edu      (153.18.253.126)    2.110 ms   2.145 ms   1.763 ms
 4  cenic.net           (137.164.32.140)    3.069 ms   2.875 ms   2.930ms
 5  cenic.net           (137.164.22.31)     4.205 ms   4.870 ms   4.197 ms

14  snfc21.pbi.net      (151.164.191.49)    7.656 ms   7.129 ms   6.866ms
15  sbcglobaLnet        (151.164.243.58)    7.844 ms   7.545 ms   7.353 ms
16  pacbell.net         (209.232.138.114)   9.857 ms   9.535 ms   9.603 ms
17  209.233.48.223      (209.233.48.223)   10.634ms   10.771 ms  10.592 ms
18  alpha.Xerox.COM     (13.1.64.93)       11.172 ms   11.048 ms  10.922ms
```

Here there are 17 hops between source and destination. Note that some round-trip times look unusual. It could be that a router was too busy to process the packet immediately.

21.3 IGMP

The IP protocol can be involved in two types of communication: unicasting and multicasting. Unicasting is the communication between one sender and one receiver. It is a one-to-one communication. However, some processes sometimes need to send the same message to a large number of receivers simultaneously. This is called multicasting, which is a one-to-many communication. Multicasting has many applications. For example, multiple stockbrokers can simultaneously be informed of changes in a stock price, or travel agents can be informed of a plane cancellation. Some other applications include distance learning and video-on-demand.

The Internet Group Management Protocol (IGMP) is one of the necessary, but not sufficient (as we will see), protocols that is involved in multicasting. IGMP is a companion to the IP protocol.

Group Management

For multicasting in the Internet we need routers that are able to route multicast packets. The routing tables of these routers must be updated by using one of the multicasting routing protocols that we discuss in Chapter 22.

IGMP is not a multicasting routing protocol; it is a protocol that manages group membership. In any network, there are one or more multicast routers that distribute multicast packets to hosts or other routers. The IGMP protocol gives the multicast routers information about the membership status of hosts (routers) connected to the network.

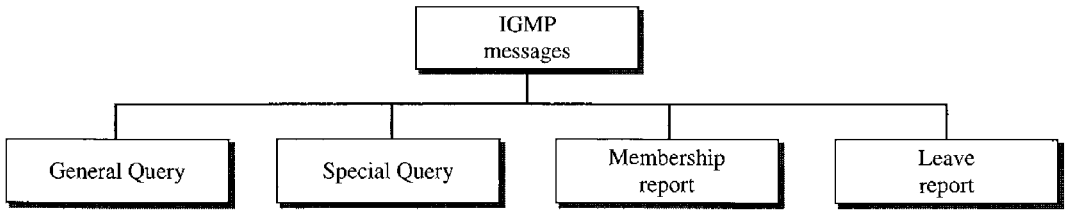
A multicast router may receive thousands of multicast packets every day for different groups. If a router has no knowledge about the membership status of the hosts, it must broadcast all these packets. This creates a lot of traffic and consumes bandwidth. A better solution is to keep a list of groups in the network for which there is at least one loyal member. IGMP helps the multicast router create and update this list.

IGMP is a group management protocol. It helps a multicast router create and update a list of loyal members related to each router interface.

IGMP Messages

IOMP has gone through two versions. We discuss IOMPv2, the current version. IOMPv2 has three types of messages: the query, the membership report, and the leave report. There are two types of query messages: general and special (see Figure 21.16).

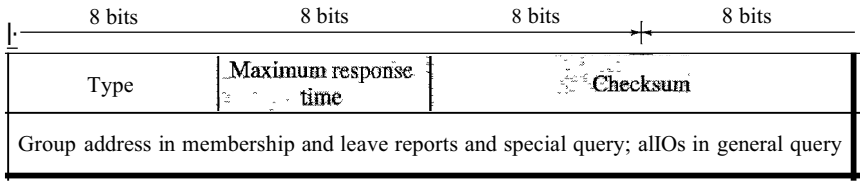
Figure 21.16 *IGMP message types*



Message Format

Figure 21.17 shows the format of an IOMP (version 2) message.

Figure 21.17 *IGMP message fonnat*



- Type. This 8-bit field defines the type of message, as shown in Table 21.1. The value of the type is shown in both hexadecimal and binary notation.

Table 21.1 *IGMP type field*

Type	*Value*
General or special query	0x11 or 00010001
Membership report	0x16 or 00010110
Leave report	0x17 or 00010111

- Maximum Response Time. This 8-bit field defines the amount of time in which a query must be answered. The value is in tenths of a second; for example, if the

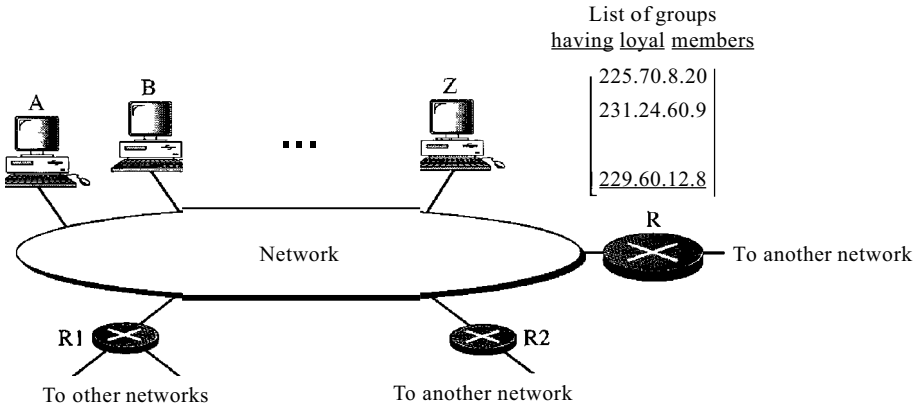
value is 100, it means 10 s. The value is nonzero in the query message; it is set to zero in the other two message types. We will see its use shortly.

- Checksum. This is a 16-bit field carrying the checksum. The checksum is calculated over the 8-byte message.
- Group address. The value of this field is 0 for a general query message. The value defines the groupid (multicast address of the group) in the special query, the membership report, and the leave report messages.

IGMP Operation

IGMP operates locally. A multicast router connected to a network has a list of multicast addresses of the groups with at least one loyal member in that network (see Figure 21.18).

Figure 21.18 *IGMP operation*



For each group, there is one router that has the duty of distributing the multicast packets destined for that group. This means that if there are three multicast routers connected to a network, their lists of groupids are mutually exclusive. For example, in Figure 21.18 only router R distributes packets with the multicast address of225.70.8.20.

A host or multicast router can have membership in a group. When a host has membership, it means that one of its processes (an application program) receives multicast packets from some group. When a router has membership, it means that a network connected to one of its other interfaces receives these multicast packets. We say that the host or the router has an *interest* in the group. In both cases, the host and the router keep a list of groupids and relay their interest to the distributing router.

For example, in Figure 21.18, router R is the distributing router. There are two other multicast routers (R1 and R2) that, depending on the group list maintained by router R, could be the recipients of router R in this network. Routers R1 and R2 may be distributors for some of these groups in other networks, but not on this network.

Joining a Group

A host or a router can join a group. A host maintains a list of processes that have membership in a group. When a process wants to join a new group, it sends its request to the host.

The host adds the name of the process and the name of the requested group to its list. If this is the first entry for this particular group, the host sends a membership report message. If this is not the first entry, there is no need to send the membership report since the host is already a member of the group; it already receives multicast packets for this group.

The protocol requires that the membership report be sent twice, one after the other within a few moments. In this way, if the first one is lost or damaged, the second one replaces it.

In IGMP, a membership report is sent twice, one after the other.

Leaving a Group

When a host sees that no process is interested in a specific group, it sends a leave report. Similarly, when a router sees that none of the networks connected to its interfaces is interested in a specific group, it sends a leave report about that group.

However, when a multicast router receives a leave report, it cannot immediately purge that group from its list because the report comes from just one host or router; there may be other hosts or routers that are still interested in that group. To make sure, the router sends a special query message and inserts the groupid, or multicast address, related to the group. The router allows a specified time for any host or router to respond. If, during this time, no interest (membership report) is received, the router assumes that there are no loyal members in the network and purges the group from its list.

Monitoring Membership

A host or router can join a group by sending a membership report message. It can leave a group by sending a leave report message. However, sending these two types of reports is not enough. Consider the situation in which there is only one host interested in a group, but the host is shut down or removed from the system. The multicast router will never receive a leave report. How is this handled? The multicast router is responsible for monitoring all the hosts or routers in a LAN to see if they want to continue their membership in a group.

The router periodically (by default, every 125 s) sends a general query message. In this message, the group address field is set to 0.0.0.0. This means the query for membership continuation is for all groups in which a host is involved, not just one.

The general query message does not define a particular group.

The router expects an answer for each group in its group list; even new groups may respond. The query message has a maximum response time of 10 s (the value of the field is actually 100, but this is in tenths of a second). When a host or router receives the general query message, it responds with a membership report if it is interested in a group. However, if there is a common interest (two hosts, for example, are interested in the same group), only one response is sent for that group to prevent unnecessary traffic. This is called a delayed response. Note that the query message must be sent by only one

router (normally called the query router), also to prevent unnecessary traffic. We discuss this issue shortly.

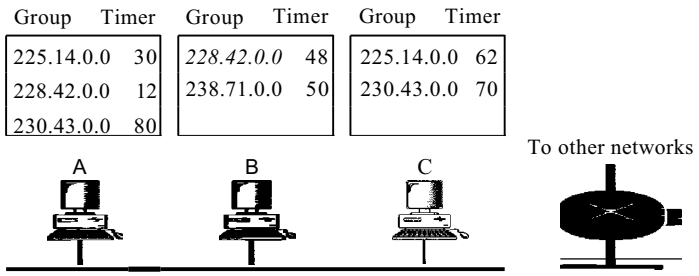
Delayed Response

To prevent unnecessary traffic, IGMP uses a delayed response strategy. When a host or router receives a query message, it does not respond immediately; it delays the response. Each host or router uses a random number to create a timer, which expires between 1 and 10s. The expiration time can be in steps of 1 s or less. A timer is set for each group in the list. For example, the timer for the first group may expire in 2 s, but the timer for the third group may expire in 5 s. Each host or router waits until its timer has expired before sending a membership report message. During this waiting time, if the timer of another host or router, for the same group, expires earlier, that host or router sends a membership report. Because, as we will see shortly, the report is broadcast, the waiting host or router receives the report and knows that there is no need to send a duplicate report for this group; thus, the waiting station cancels its corresponding timer.

Example 21.6

Imagine there are three hosts in a network, as shown in Figure 21.19.

Figure 21.19 *Example 21.6*



A query message was received at time 0; the random delay time (in tenths of seconds) for each group is shown next to the group address. Show the sequence of report messages.

Solution

The events occur in this sequence:

a. Time 12: The timer for 228.42.0.0 in host A expires, and a membership report is sent, which is received by the router and every host including host B which cancels its timer for 228.42.0.0.

b. Time 30: The timer for 225.14.0.0 in host A expires, and a membership report is sent, which is received by the router and every host including host C which cancels its timer for 225.14.0.0.

c. Time 50: The timer for 238.71.0.0 in host B expires, and a membership report is sent, which is received by the router and every host.

d. Time 70: The timer for 230.43.0.0 in host C expires, and a membership report is sent, which is received by the router and every host including host A which cancels its timer for 230.43.0.0.

Note that if each host had sent a report for every group in its list, there would have been seven reports; with this strategy only four reports are sent.

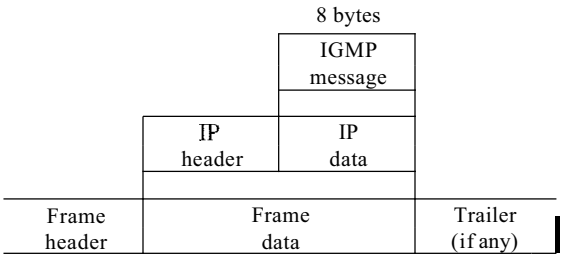
Query Router

Query messages may create a lot of responses. To prevent unnecessary traffic, IGMP designates one router as the query router for each network. Only this designated router sends the query message, and the other routers are passive (they receive responses and update their lists).

Encapsulation

The IGMP message is encapsulated in an IP datagram, which is itself encapsulated in a frame. See Figure 21.20.

Figure 21.20 *Encapsulation of IGMP packet*



Encapsulation at Network Layer

The value of the protocol field is 2 for the IGMP protocol. Every IP packet carrying this value in its protocol field has data delivered to the IGMP protocol. When the message is encapsulated in the IP datagram, the value of TTL must be 1. This is required because the domain of IGMP is the LAN. No IGMP message must travel beyond the LAN. A TTL value of 1 guarantees that the message does not leave the LAN since this value is decremented to 0 by the next router and, consequently, the packet is discarded. Table 21.2 shows the destination IP address for each type of message.

The IP packet that carries an IGMP packet has a value of 1 in its TTL field.

Table 21.2 *Destination IP addresses*

Type	*IP Destination Address*
Query	224.0.0.1 All systems on this subnet
Membership report	The multicast address of the group
Leave report	224.0.0.2 All routers on this subnet

A query message is multicast by using the multicast address 224.0.0.1 All hosts and all routers will receive the message. A membership report is multicast using a

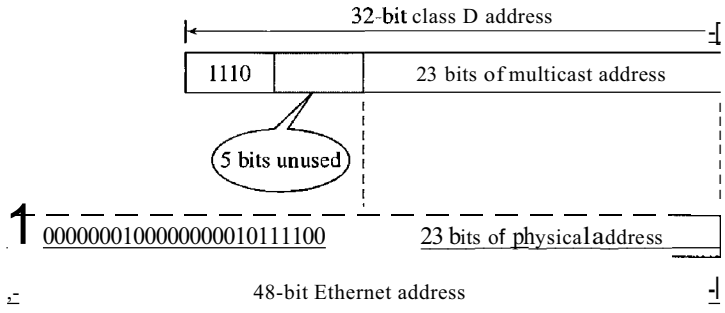
destination address equal to the multicast address being reported (groupid). Every station (host or router) that receives the packet can immediately determine (from the header) the group for which a report has been sent. As discussed previously, the timers for the corresponding unsent reports can then be canceled. Stations do not need to open the packet to find the groupid. This address is duplicated in a packet; it's part of the message itself and also a field in the IP header. The duplication prevents errors. A leave report message is multicast using the multicast address 224.0.0.2 (all routers on this subnet) so that routers receive this type of message. Hosts receive this message too, but disregard it.

Encapsulation at Data Link Layer

At the network layer, the IGMP message is encapsulated in an IP packet and is treated as an IP packet. However, because the IP packet has a multicast IP address, the ARP protocol cannot find the corresponding MAC (physical) address to forward the packet at the data link layer. What happens next depends on whether the underlying data link layer supports physical multicast addresses.

Physical Multicast Support Most LANs support physical multicast addressing. Ethernet is one of them. An Ethernet physical address (MAC address) is six octets (48 bits) long. If the first 25 bits in an Ethernet address are 00000001000000000010111100, this identifies a physical multicast address for the TCP/IP protocol. The remaining 23 bits can be used to define a group. To convert an IP multicast address into an Ethernet address, the multicast router extracts the least significant 23 bits of a class D IP address and inserts them into a multicast Ethernet physical address (see Figure 21.21).

Figure 21.21 *Mapping class D to Ethernet physical address*



However, the group identifier of a class D IP address is 28 bits long, which implies that 5 bits is not used. This means that 32 (25) multicast addresses at the IP level are mapped to a single multicast address. In other words, the mapping is many-to-one instead of one-to-one. If the 5 leftmost bits of the group identifier of a class D address are not all zeros, a host may receive packets that do not really belong to the group in which it is involved. For this reason, the host must check the IP address and discard any packets that do not belong to it.

Other LANs support the same concept but have different methods of mapping.

An Ethernet multicast physical address is in the range
01:00:5E:00:OO:OO to 01:00:5E:7F:FF:FF.

Example 21.7

Change the multicast IP address 230.43.14.7 to an Ethernet multicast physical address.

Solution

We can do this in two steps:

a. We write the rightmost 23 bits of the IP address in hexadecimal. This can be done by changing the rightmost 3 bytes to hexadecimal and then subtracting 8 from the leftmost digit if it is greater than or equal to 8. In our example, the result is 2B:OE:07.

b. We add the result of part a to the starting Ethernet multicast address, which is 01:00:5E:00:00:00. The result is

01:00:5E:2B:OE:07

Example 21.8

Change the multicast IP address 238.212.24.9 to an Ethernet multicast address.

Solution

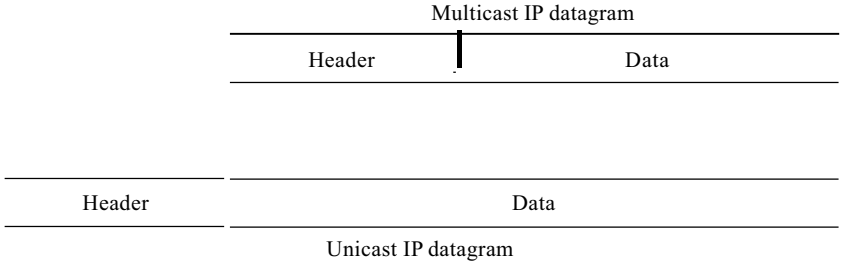
a. The rightmost 3 bytes in hexadecimal is D4: 18:09. We need to subtract 8 from the leftmost digit, resulting in 54:18:09.

b. We add the result of part a to the Ethernet multicast starting address. The result is

01:00:5E:54: 18:09

No Physical Multicast Support Most WANs do not support physical multicast addressing. To send a multicast packet through these networks, a process called *tunneling* is used. In tunneling, the multicast packet is encapsulated in a unicast packet and sent through the network, where it emerges from the other side as a multicast packet (see Figure 21.22).

Figure 21.22 *Tunneling*



Netstat Utility

The *netstat* utility can be used to find the multicast addresses supported by an interface.