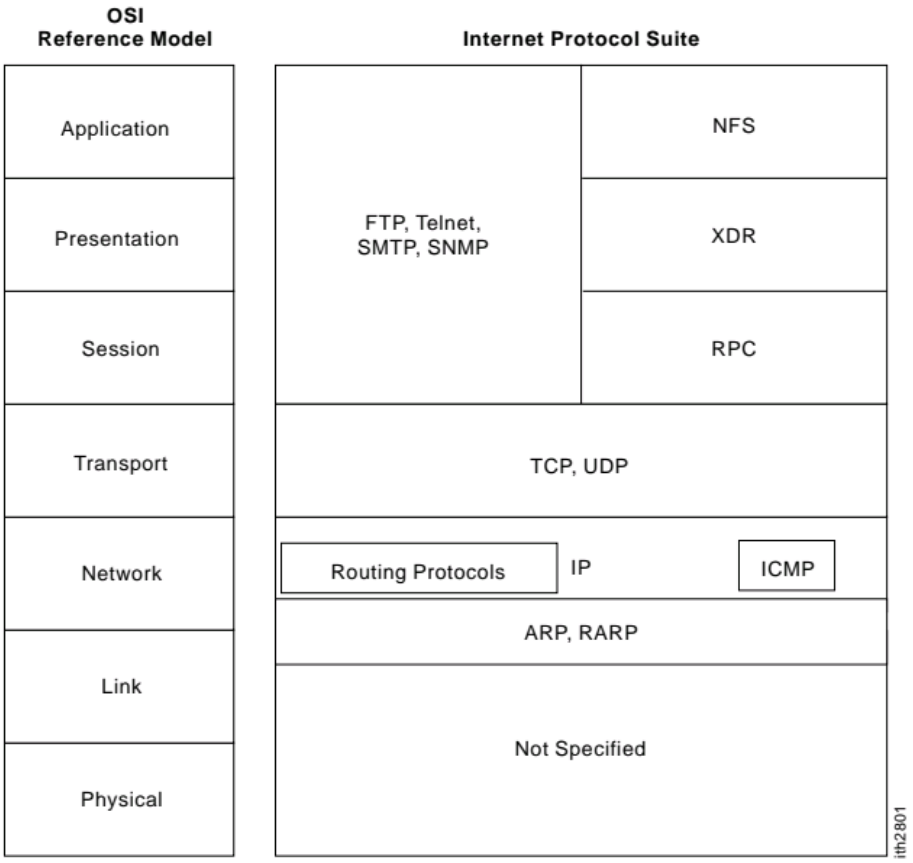
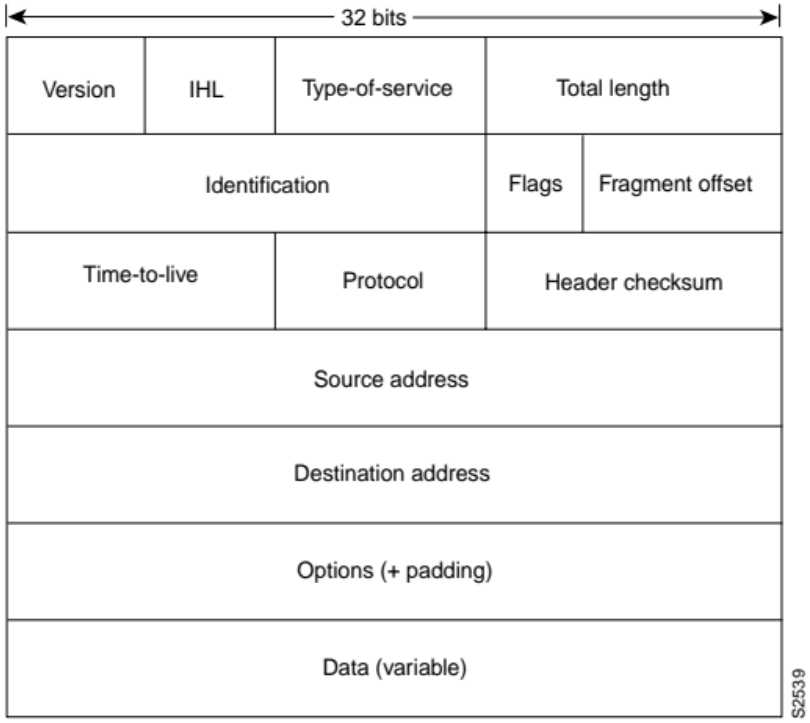


Internet Protocol (IP)

The Internet Protocol (IP) is a network-layer (Layer 3) protocol that contains addressing information and some control information that enables packets to be routed. IP is documented in RFC 791 and is the primary network-layer protocol in the Internet protocol suite. Along with the Transmission Control Protocol (TCP), IP represents the heart of the Internet protocols. IP has two primary responsibilities: providing connectionless, best-effort delivery of datagrams through an internetwork; and providing fragmentation and reassembly of datagrams to support data links with different maximum-transmission unit (MTU) sizes.

IP Packet Format

An IP packet contains several types of information, as illustrated in Figure 30-2.



The following discussion describes the IP packet fields illustrated in Figure 30-2:

- *Version*—Indicates the version of IP currently used.
- *IP Header Length (*IHL)—Indicates the datagram header length in 32-bit words.
- *Type-of-Service*—Specifies how an upper-layer protocol would like a current datagram to be handled, and assigns datagrams various levels of importance.
- *Total Length*—Specifies the length, in bytes, of the entire IP packet, including the data and header.
- *Identification*—Contains an integer that identifies the current datagram. This field is used to help piece together datagram fragments.
- *Flags*—Consists of a 3-bit field of which the two low-order (least-significant) bits control fragmentation. The low-order bit specifies whether the packet can be fragmented. The middle bit specifies whether the packet is the last fragment in a series of fragmented packets. The third or high-order bit is not used.
- *Fragment Offset*—Indicates the position of the fragment's data relative to the beginning of the data in the original datagram, which allows the destination IP process to properly reconstruct the original datagram.
- *Time-to-Live*—Maintains a counter that gradually decrements down to zero, at which point the datagram is discarded. This keeps packets from looping endlessly.
- *Protocol*—Indicates which upper-layer protocol receives incoming packets after IP processing is complete.
- *Header Checksum*—Helps ensure IP header integrity.
- S*ource Address*—Specifies the sending node.
- *Destination Address*—Specifies the receiving node.

- *Options*—Allows IP to support various options, such as security.
- *Data*—Contains upper-layer information.

IP Addressing

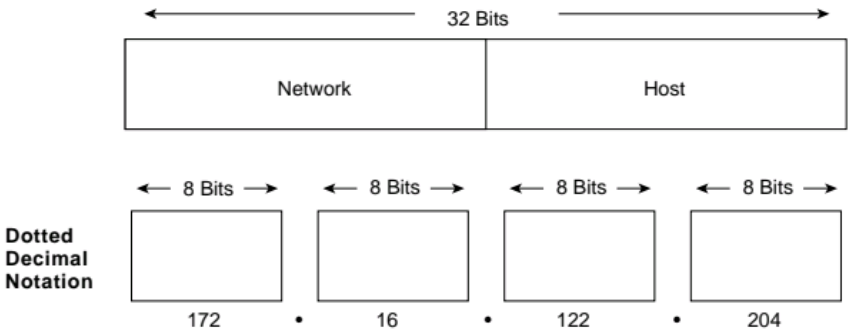
As with any other network-layer protocol, the IP addressing scheme is integral to the process of routing IP datagrams through an internetwork. Each IP address has specific components and follows a basic format. These IP addresses can be subdivided and used to create addresses for subnetworks, as discussed in more detail later in this chapter.

Each host on a TCP/IP network is assigned a unique 32-bit logical address that is divided into two main parts: the network number and the host number. The network number identifies a network and must be assigned by the Internet Network Information Center (InterNIC) if the network is to be part of the Internet. An Internet Service Provider (ISP) can obtain blocks of network addresses from the InterNIC and can itself assign address space as necessary. The host number identifies a host on a network and is assigned by the local network administrator.

IP Address Format

The 32-bit IP address is grouped eight bits at a time, separated by dots, and represented in decimal format (known as *dotted decimal notation*). Each bit in the octet has a binary weight (128, 64, 32, 16, 8, 4, 2, 1). The minimum value for an octet is 0, and the maximum value for an octet is 255. Figure 30-3 illustrates the basic format of an IP address.

Figure 30-3 An IP address consists of 32 bits, grouped into four octets.



IP Address Classes

IP addressing supports five different address classes: A, B,C, D, and E. Only classes A, B, and C are available for commercial use. The left-most (high-order) bits indicate the network class. Table 30-1 provides reference information about the five IP address classes.

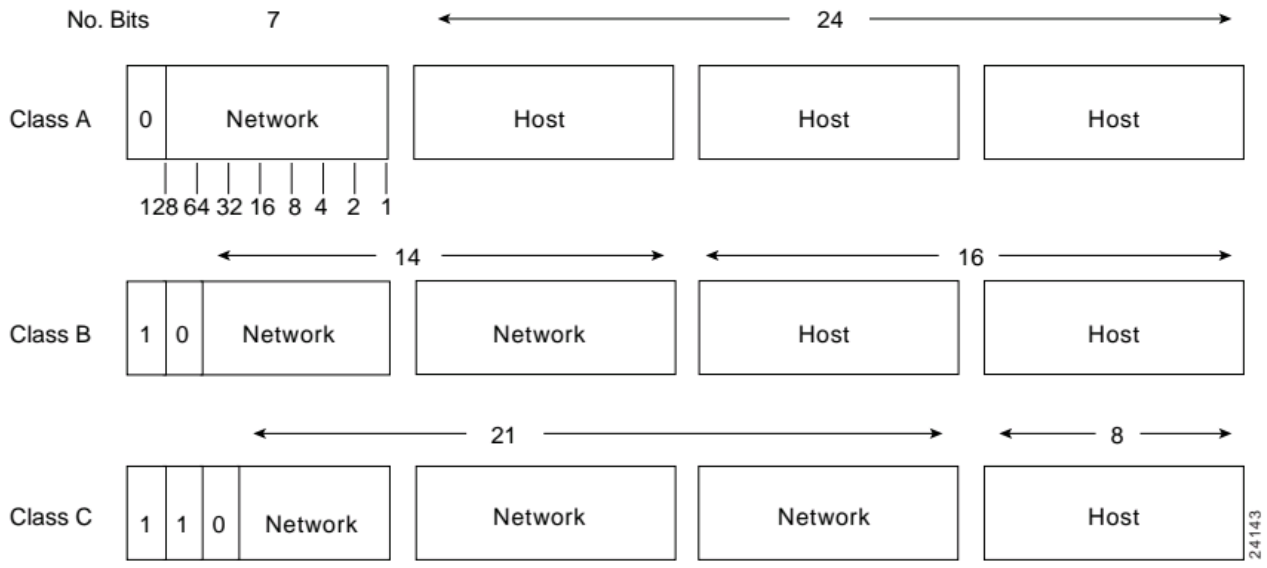
IP Address Class	Format	Purpose	High-Order Bit(s)	Address Range	No. Bits Network/Host	Max. Hosts
A	N.H.H.H[1]	Few large organizations	0	1.0.0.0 to 126.0.0.0	7/24	16,777, 214[2] ($2^{24} - 2$)
B	N.N.H.H	Medium-size organizations	1, 0	128.1.0.0 to 191.254.0.0	14/16	65, 543 ($2^{16} - 2$)
C	N.N.N.H	Relatively small organizations	1, 1, 0	192.0.1.0 to 223.255.254.0	22/8	245 ($2^{8} - 2$)
D	N/A	Multicast groups (RFC 1112)	1, 1, 1, 0	224.0.0.0 to 239.255.255.255	N/A (not for commercial use)	N/A
E	N/A	Experimental	1, 1, 1, 1	240.0.0.0 to 254.255.255.255	N/A	N/A

1 N = Network number, H = Host number.

2 One address is reserved for the broadcast address, and one address is reserved for the network.

Figure 30-4 illustrates the format of the commercial IP address classes. (Note the high-order bits in each class.)

Figure 30-4 IP address formats A, B, and C are available for commercial use.



The class of address can be determined easily by examining the first octet of the address and mapping that value to a class range in the following table. In an IP address of 172.31.1.2, for example, the first octet is 172. Because 172 falls between 128 and 191, 172.31.1.2 is a Class B address. Figure 30-5 summarizes the range of possible values for the first octet of each address class.

Address Class	First Octet in Decimal	High-Order Bits
Class A	1 Ð 126	0
Class B	128 Ð 191	10
Class C	192 Ð 223	110
Class D	224 Ð 239	1110
Class E	240 Ð 254	1111

24144

IP Subnet Addressing

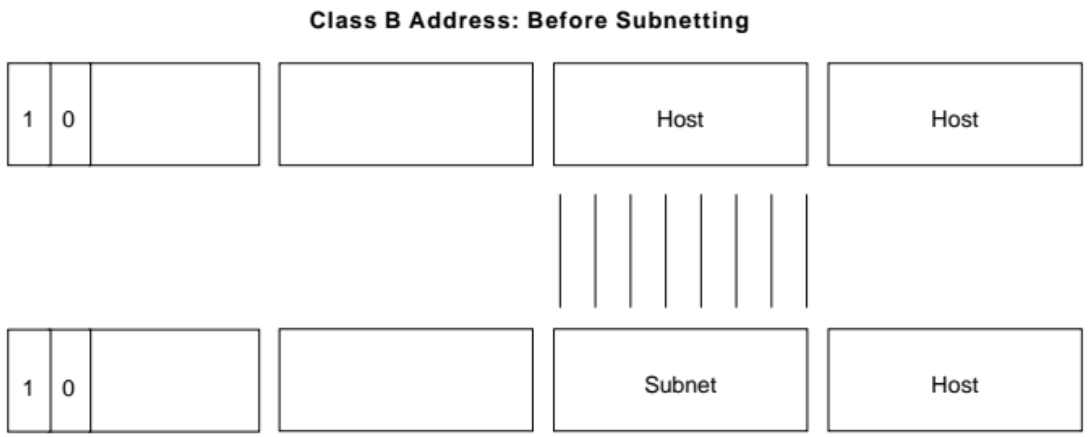
IP networks can be divided into smaller networks called subnetworks (or subnets). Subnetting provides the network administrator with several benefits, including extra flexibility, more efficient use of network addresses, and the capability to contain broadcast traffic (a broadcast will not cross a router).

Subnets are under local administration. As such, the outside world sees an organization as a single network and has no detailed knowledge of the organization's internal structure.

A given network address can be broken up into many subnetworks. For example, 172.16.1.0, 172.16.2.0, 172.16.3.0, and 172.16.4.0 are all subnets within network 171.16.0.0. (All 0s in the host portion of an address specifies the entire network.)

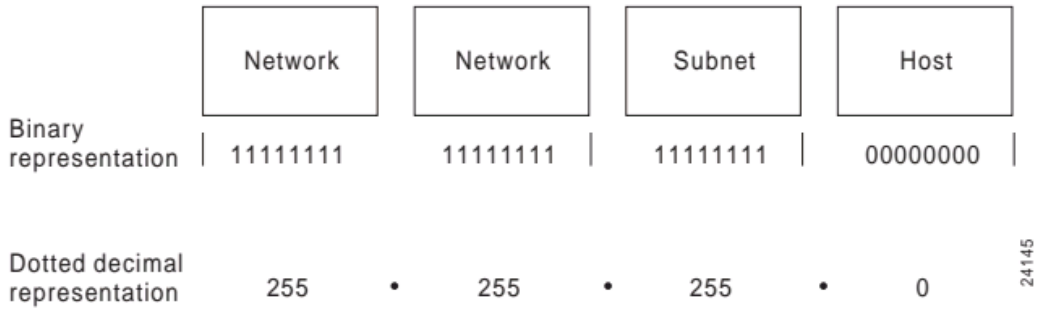
IP Subnet Mask

A subnet address is created by "borrowing" bits from the host field and designating them as the subnet field. The number of borrowed bits varies and is specified by the subnet mask. Figure 30-6 shows how bits are borrowed from the host address field to create the subnet address field.



Subnet masks use the same format and representation technique as IP addresses. The subnet mask, however, has binary 1s in all bits specifying the network and subnetwork fields, and binary 0s in all bits specifying the host field. Figure 30-7 illustrates a sample subnet mask.

Figure 30-7 A sample subnet mask consists of all binary 1s and 0s.



Subnet mask bits should come from the high-order (left-most) bits of the host field, as Figure 30-8 illustrates. Details of Class B and C subnet mask types follow. Class A addresses are not discussed in this chapter because they generally are subnetted on an 8-bit boundary.

128	64	32	16	8	4	2	1		
1	0	0	0	0	0	0	0	=	128
1	1	0	0	0	0	0	0	=	192
1	1	1	0	0	0	0	0	=	224
1	1	1	1	0	0	0	0	=	240
1	1	1	1	1	0	0	0	=	248
1	1	1	1	1	1	0	0	=	252
1	1	1	1	1	1	1	0	=	254
1	1	1	1	1	1	1	1	=	255

24146

Various types of subnet masks exist for Class B and C subnets.

The default subnet mask for a Class B address that has no subnetting is 255.255.0.0, while the subnet mask for a Class B address 171.16.0.0 that specifies eight bits of subnetting is 255.255.255.0. The reason for this is that eight bits of subnetting or $2^8 - 2$ (1 for the network address and 1 for the broadcast address) = 254 subnets possible, with $2^8 - 2 = 254$ hosts per subnet.

The subnet mask for a Class C address 192.168.2.0 that specifies five bits of subnetting is 255.255.255.248.With five bits available for subnetting, $2^5 - 2 = 30$ subnets possible, with $2^3 - 2 = 6$ hosts per subnet.

The reference charts shown in table 30–2 and table 30–3 can be used when planning Class B and C networks to determine the required number of subnets and hosts, and the appropriate subnet mask.

Table 30-2 Class B Subnetting Reference Chart

Number of Bits	Subnet Mask	Number of Subnets	Number of Hosts
2	255.255.192.0	2	16382
3	255.255.224.0	6	8190
4	255.255.240.0	14	4094
5	255.255.248.0	30	2046
6	255.255.252.0	62	1022
7	255.255.254.0	126	510
8	255.255.255.0	254	254
9	255.255.255.128	510	126
10	255.255.255.192	1022	62
11	255.255.255.224	2046	30
12	255.255.255.240	4094	14

Number of Bits	Subnet Mask	Number of Subnets	Number of Hosts
13	255.255.255.248	8190	6
14	255.255.255.252	16382	2

Table 30-3 Class C Subnetting Reference Chart

Number of Bits	Subnet Mask	Number of Subnets	Number of Hosts
2	255.255.255.192	2	62
3	255.255.255.224	6	30
4	255.255.255.240	14	14
5	255.255.255.248	30	6
6	255.255.255.252	62	2

How Subnet Masks are Used to Determine the Network Number

The router performs a set process to determine the network (or more specifically, the subnetwork) address. First, the router extracts the IP destination address from the incoming packet and retrieves the internal subnet mask. It then performs a *logical AND* operation to obtain the network number. This causes the host portion of the IP destination address to be removed, while the destination network number remains. The router then looks up the destination network number and matches it with an outgoing interface. Finally, it forwards the frame to the destination IP address. Specifics regarding the logical AND operation are discussed in the following section.

Logical AND Operation

Three basic rules govern logically "ANDing" two binary numbers. First, 1 "ANDed" with 1 yields 1. Second, 1 "ANDed" with 0 yields 0. Finally, 0 "ANDed" with 0 yields 0. The truth table provided in table 30–4 illustrates the rules for logical AND operations.

Table 30-4 Rules for Logical AND Operations

Input	Input	Output
1	1	1
1	0	0
0	1	0
0	0	0

Two simple guidelines exist for remembering logical AND operations: Logically "ANDing" a 1 with a 1 yields the original value, and logically "ANDing" a 0 with any number yields 0.

Figure 30-9 illustrates that when a logical AND of the destination IP address and the subnet mask is performed, the subnetwork number remains, which the router uses to forward the packet.

Mapping Internet Addresses To Physical Addresses (ARP)

5.1 Introduction

We described the TCP/IP address scheme in which each host is assigned a 32-bit address, and said that an internet behaves like a virtual network, using only the assigned addresses when sending and receiving packets. We also reviewed several network hardware technologies, and noted that two machines on a given physical network can communicate ***only if they know each other's physical network address.*** What we have not mentioned is how a host or a router maps an IP address to the correct physical address when it needs to send a packet across a physical net. This chapter considers that mapping, showing how it is implemented for the two most common physical network address schemes.

5.2 The Address Resolution Problem

Consider two machines A and B that connect to the same physical network. Each has an assigned IP address Z_A and Z_B and a physical address P_A and P_B. The goal is to devise low-level software that hides physical addresses and allows higher-level programs to work only with internet addresses. Ultimately, however, communication must be carried out by physical networks using whatever physical address scheme the underlying network hardware supplies. Suppose machine A wants to send a packet to

machine B across a physical network to which they both attach, but A has only B's internet address I_B. The question arises: how does A map that address to B's physical address, P_B?

Address mapping must be performed at each step along a path from the original source to the ultimate destination. In particular, two cases arise. First, at the last step of delivering a packet, the packet must be sent across one physical network to its final destination. The computer sending the packet must map the final destination's Internet address to the destination's physical address. Second, at any point along the path from the source to the destination other than the final step, the packet must be sent to an intermediate router. Thus, the sender must map the intermediate router's Internet address to a physical address.

The problem of mapping high-level addresses to physical addresses is known as the ***address resolution problem*** and has been solved in several ways. Some protocol suites keep tables in each machine that contain pairs of high-level and physical addresses. Others solve the problem by encoding hardware addresses in high-level addresses. Using either approach exclusively makes high-level addressing awkward at best. This chapter discusses two techniques for address resolution used by TCP/IP protocols and shows when each is appropriate.

5.3 Two Types Of Physical Addresses

There are two basic types of physical addresses, exemplified by the Ethernet, which has large, fixed physical addresses, and proNET, which has small, easily configured physical addresses. Address resolution is difficult for Ethernet-like networks, but easy for networks like proNET. We will consider the easy case first.

5.4 Resolution Through Direct Mapping

Consider a proNET token ring network. Recall from Chapter **2** that proNET uses small integers for physical addresses and allows the user to choose a hardware address when installing an interface board in a computer. The key to making address resolution easy with such network hardware lies in observing that as long as one has the freedom to choose both IP and physical addresses, they can be selected such that parts of them are the same. Typically, one assigns IP addresses with the **hostid** portion equal to **1, 2, 3,** and so on, and then, when installing network interface hardware, selects a physical address that corresponds to the IP address. For example, the system administrator would select physical address **3** for a computer with the IP address **192.5.48.3** because **192.5.48.3**is a class C address with the host portion equal to **3.**

For networks like proNET, computing a physical address from an IP address is trivial. The computation consists of extracting the host portion of the IP address. Extraction is computationally efficient on most architectures because it requires only a few machine instructions. The mapping is easy to maintain because it can be performed

without reference to external data. Finally, new computers can be added to the network without changing existing assignments or recompiling code.

Conceptually, choosing a numbering scheme that makes address resolution efficient means selecting a function f that maps IP addresses to physical addresses. The designer may be able to select a physical address numbering scheme as well, depending on the hardware. Resolving IP address I_A means computing

$$P_A = f(I_A)$$

We want the computation of f to be efficient. If the set of physical addresses is constrained, it may be possible to arrange efficient mappings other than the one given in the example above. For instance, when using IP over a connection-oriented network such as ATM, one cannot choose physical addresses. On such networks, one or more computers (servers) store pairs of addresses, where each pair contains an Internet address and the corresponding physical address. Typically, such servers store the pairs in a table in memory to speed searching. To guarantee efficient address resolution in such cases, software can use a conventional hash function to search the table. Exercise 5.1 suggests a related alternative.

5.5 Resolution Through Dynamic Binding

To understand why address resolution is difficult for some networks, consider Ethernet technology. Recall from Chapter 2 that each Ethernet interface is assigned a 48-bit physical address when the device is manufactured. As a consequence, when hardware fails and requires that an Ethernet interface be replaced, the machine's physical address changes. Furthermore, because the Ethernet address is 48 bits long, there is no hope it can be encoded in a 32-bit IP address†.

Designers of TCP/IP protocols found a creative solution to the address resolution problem for networks like the Ethernet that have broadcast capability. The solution allows new hosts or routers to be added to the network without recompiling code, and does not require maintenance of a centralized database. To avoid maintaining a table of mappings, the designers chose to use a low-level protocol to bind addresses dynamically. Termed the ***Address Resolution Protocol (ARP)***, the protocol provides a mechanism that is both reasonably efficient and easy to maintain.

As Figure 5.1 shows, the idea behind dynamic resolution with ARP is simple: when host A wants to resolve IP address I_B, it broadcasts a special packet that asks the host with IP address I_B to respond with its physical address, P_B. All hosts, including B, receive the request, but only host B recognizes its IP address and sends a reply that contains its physical address. When A receives the reply, it uses the physical address to send the internet packet directly to B. We can summarize:

*The Address Resolution Protocol, **ARP**, allows a host to find the physical address of a target host on the same physical network, given only the target's **IP** address.*

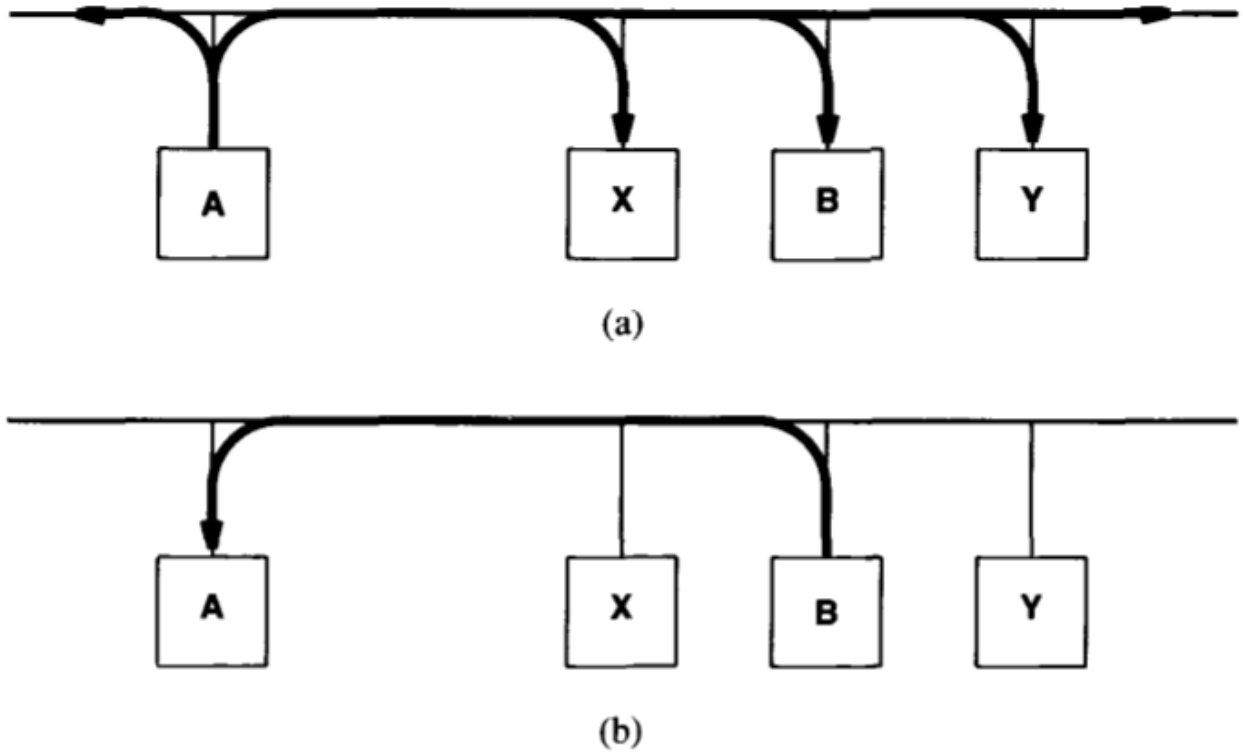


Figure 5.1 The ARP protocol. To determine P_B, B's physical address, from I_B, its IP address, (a) host A broadcasts an ARP request containing I_B to all machines on the net, and (b) host B responds with an ARP reply that contains the pair (I_B, P_B).

5.6 The Address Resolution Cache

It may seem silly that for *A* to send a packet to B it first sends a broadcast that reaches B. Or it may seem even sillier that *A* broadcasts the question, "how can I reach you?" instead of just broadcasting the packet it wants to deliver. But there is an important reason for the exchange. Broadcasting is far too expensive to be used every time one machine needs to transmit a packet to another because every machine on the network must receive and process the broadcast packet.

5.7 ARP Cache Timeout

To reduce communication costs, computers that use ARP maintain a cache of recently acquired IP-to-physical address bindings. That is, whenever a computer sends an ARP request and receives an *ARP* reply, it saves the IP address and corresponding hardware address information in its cache for successive lookups. When transmitting a packet, a computer always looks in its cache for a binding before sending an **ARP** request. **If** it finds the desired binding in its ARP cache, the computer need not broadcast on the network. Thus, when two computers on a network communicate, they begin with an ARP request and response, and then repeatedly transfer packets without using ARP for each one. Experience shows that because most network communication involves more than one packet transfer, even a small cache is worthwhile.

The AFW cache provides an example of *soft* state, a technique commonly used in network protocols. The name describes a situation in which information can become "stale" without warning. In the case of *ARP*, consider two computers, A and B, both connected to an Ethernet. Assume A has sent **an** ARP request, and B has replied. Further assume that after the exchange B crashes. Computer A will not receive any notification of the crash. Moreover, because it already has address binding information for B in its ARP cache, computer A will continue to send packets to B. The Ethernet hardware provides no indication that *B* is not on-line because Ethernet does not have guaranteed delivery. Thus, A has no way of knowing when information in its **ARP** cache has become incorrect.

To accommodate soft state, responsibility for correctness lies with the owner of the information. Typically, protocols that implement soft state use timers, with the state information being deleted when the timer expires. For example, whenever address binding information is placed in an **ARP** cache, the protocol requires a timer to be set, with a typical **timeout** being 20 minutes. When the timer expires, the information must be removed. After removal there are two possibilities. **If** no further packets are sent to the destination, nothing occurs. If a packet must be sent to the destination and there is no binding present in the cache, the computer follows the normal procedure of broadcasting an ARP request and obtaining the binding. If the destination is still reachable, the binding will again be placed in the ARP cache. **If** not, the sender will discover that the destination is off-line.

The use of soft state in *ARP* has advantages and disadvantages. The chief advantage arises from autonomy. First, a computer can determine when information in its ARP cache should be revalidated independent of other computers. Second, a sender does not need successful communication with the receiver or a third party to determine that a binding has become invalid; if a target does not respond to an ARP request, the sender will declare the target to be down. Third, the scheme does not rely on network hardware to provide reliable transfer. The chief disadvantage of soft state arises from delay — if the timer interval is N seconds, a sender may not detect that a receiver has crashed until N seconds elapse.

5.8 ARP Refinements

Several refinements of ARP have been included in the protocol. First, observe that if host A is about to use ARP because it needs to send to B, there is a high probability that host B will need to send to A in the near future. To anticipate B's need and avoid extra network traffic, A includes its IP-to-physical address binding when sending B a request. B extracts A's binding from the request, saves the binding in its ARP cache, and then sends a reply to A. Second, notice that because A broadcasts its initial request, all machines on the network receive it and can extract and update A's IP-to-physical address binding in their cache. Third, when a computer has its host interface replaced, (e.g., because the hardware has failed) its physical address changes. Other computers on the net that have stored a binding in their ARP cache need to be informed so they can change the entry. The computer can notify others of a new address by sending an ARP broadcast when it boots.

The following rule summarizes refinements:

> ***The sender's IP-to-physical address binding is included in every ARP broadcast; receivers update the IP-to-physical address binding information in their cache before processing an ARP packet.***

5.9 Relationship Of ARP To Other Protocols

ARP provides one possible mechanism to map from IP addresses to physical addresses; we have already seen that some network technologies do not need it. The point is that ARP would be completely unnecessary if we could make all network hardware recognize IP addresses. Thus, ARP merely imposes a new address scheme on top of whatever low-level address mechanism the hardware uses. The idea can be summarized:

> ***ARP is a low-level protocol that hides the underlying network physical addressing, permitting one to assign an arbitrary IP address to every machine. We think of ARP as part of the physical network system, and not as part of the internet protocols.***

5.10 ARP Implementation

Functionally, ARP is divided into two parts. The first part maps an IP address to a physical address when sending a packet, and the second part answers requests from other machines. Address resolution for outgoing packets seems straightforward, but small details complicate an implementation. Given a destination IP address the software consults its ARP cache to see if it knows the mapping from IP address to physical address.

If it does, the software extracts the physical address, places the data in a frame using that address, and sends the frame. If it does not know the mapping, the software must broadcast an ARP request and wait for a reply.

Broadcasting an ARP request to find an address mapping can become complex. The target machine can be down or just too busy to accept the request. If so, the sender may not receive a reply or the reply may be delayed. Because the Ethernet is a **best-effort** delivery system, the initial ARP broadcast request can also be lost (in which case the sender should retransmit, at least once). Meanwhile, the host must store the original outgoing packet so it can be sent once the address has been **resolved†**. In fact, the host must decide whether to allow other application programs to proceed while it processes an ARP request (most do). If so, the software must handle the case where an application generates additional ARP requests for the same address without broadcasting multiple requests for a given target.

Finally, consider the case where machine A has obtained a binding for machine B, but then B's hardware fails and is replaced. Although B's address has changed, A's cached binding has not, so A uses a nonexistent hardware address, making successful reception impossible. This case shows why it is important to have ARP software treat its table of bindings as a cache and remove entries after a fixed period. Of course, the timer for an entry in the cache must be reset whenever an *ARP* broadcast arrives containing the binding (but it is not reset when the entry is used to send a packet).

The second part of the ARP code handles ARP packets that arrive from the network. When an ARP packet arrives, the software first extracts the sender's IP address and hardware address pair, and examines the local cache to see if it already has an entry for the sender. If a cache entry exists for the given IP address, the handler updates that entry by overwriting the physical address with the physical address obtained from the packet. The receiver then processes the rest of the ARP packet.

A receiver must handle two types of incoming ARP packets. If an ARP request arrives, the receiving machine must see if it is the target of the request **(i.e.,** some other machine has broadcast a request for the receiver's physical address). If so, the ARP software **forms** a reply by supplying its physical hardware address, and sends the reply directly back to the requester. The receiver also adds the sender's address pair to its cache if the pair is not already present. If the IP address mentioned in the ARP request does not match the local IP address, the packet is requesting a mapping for some other machine on the network and can be ignored.

The other interesting case occurs when an ARP reply arrives. Depending on the implementation, the handler may need to create a cache entry, or the entry may have been created when the request was generated. In any case, once the cache has been updated, the receiver tries to match the reply with a previously issued request. Usually, replies arrive in response to a request, which was generated because the machine has a packet to deliver. Between the time a machine broadcasts its *ARP* request and receive the reply, application programs or higher-level protocols may generate additiona quests for the same address; the software must remember that it has already sen quest and not send more. Usually, ARP software places the additional packe queue. Once the reply arrives and the address binding is known, the ARP softw

Figure 5.3 shows an ARP message with 4 octets per line, a format that is standard throughout this text. Unfortunately, unlike most of the remaining protocols, the variable-length fields in ARP packets do not align neatly on 32-bit boundaries, making the diagram difficult to read. For example, the sender's hardware address, labeled *SENDER* HA, occupies 6 contiguous octets, so it spans two lines in the diagram.

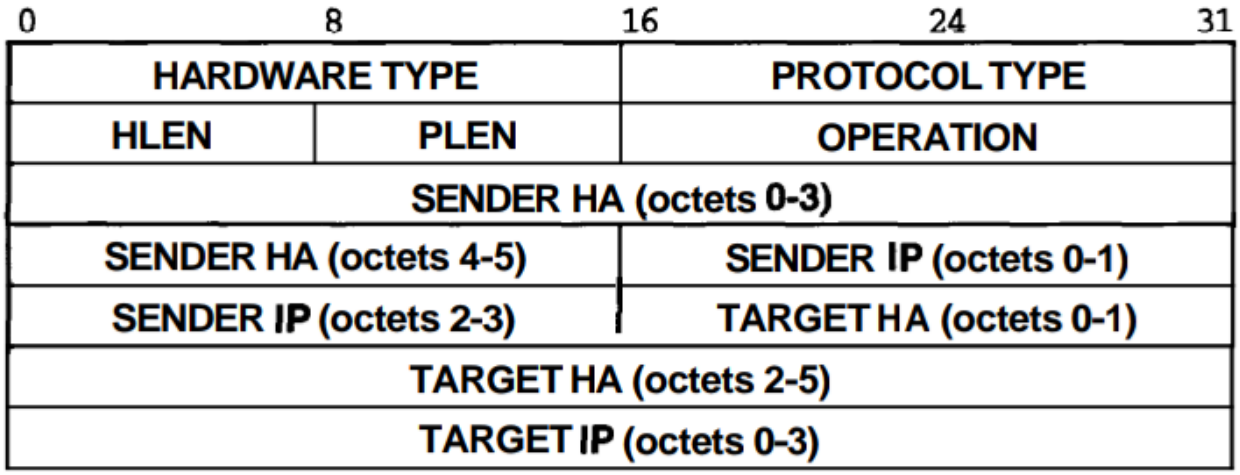


Figure 5.3 An example of the ARP/RARP message format when used for IP-to-Ethernet address resolution. The length of fields depends on the hardware and protocol address lengths, which are *6* octets for an Ethernet address and 4 octets for an IP address.

Field ***HARDWARE TYPE*** specifies a hardware interface type for which the sender seeks an answer; it contains the value *1* for Ethernet. Similarly, field ***PROTOCOL*** TYPE specifies the type of high-level protocol address the sender has supplied; it contains 0800_{16} for IP addresses. Field ***OPERATION*** specifies an ARP request (1), ARP response (2), RARP† request (3), or RARP response (4). Fields ***HLEN*** and ***PLEN*** allow ARP to be used with arbitrary networks because they specify the length of the hardware address and the length of the high-level protocol address. The sender supplies its hardware address and IP address, if known, in fields ***SENDER*** HA and ***SENDER*** IP.

When making a request, the sender also supplies the target hardware address (RARP) or target IP address (ARP), using fields ***TARGET*** HA or ***TARGET IP.*** Before the target machine responds, it fills in the missing addresses, swaps the target and sender pairs, and changes the operation to a reply. Thus, a reply carries the IP and hardware addresses of the original requester, as well as the IP and hardware addresses of the machine for which a binding was sought.

9.2 The Internet Control Message Protocol

In the connectionless system we have described so far, each router operates autonomously, routing or delivering datagrams that arrive without coordinating with the original sender. The system works well if all machines operate correctly and agree on routes. Unfortunately, no large communication system works correctly all the time. Besides failures of communication lines and processors, IP fails to deliver datagrams when the destination machine is temporarily or permanently disconnected from the network, when the time-to-live counter expires, or when intermediate routers become so

congested that they cannot process the incoming traffic. The important difference between having a single network implemented with dedicated hardware and an internet implemented with software is that in the former, the designer can add special hardware to inform attached hosts when problems arise. In an internet, which has no such hardware mechanism, a sender cannot tell whether a delivery failure resulted from a local malfunction or a remote one. Debugging becomes extremely d cult. The IP protocol itself contains nothing to help the sender test connectivity or learn about such failures.

To allow routers in an internet to report errors or provide information about unexpected circumstances, the designers added a special-purpose message mechanism to the TCP/IP protocols. The mechanism, known as the ***Internet Control Message Protocol (ICMP),*** is considered a required part of IP and must be included in every IP implementation.

Like all other traffic, ICMP messages travel across the internet in the data portion of IP datagrams. The ultimate destination of an ICMP message is not an application program or user on the destination machine, however, but the Internet Protocol software on that machine. That is, when an ICMP error message arrives, the ICMP software module handles it. Of course, if ICMP determines that a particular higher-level protocol or application program has caused a problem, it will inform the appropriate module. We can summarize:

> ***The Internet Control Message Protocol allows routers to send error or control messages to other routers or hosts; ICMP provides communication between the Internet Protocol software on one machine and the Internet Protocol sofrware on another.***

Initially designed to allow routers to report the cause of delivery errors to hosts, ICMP is not restricted to routers. Although guidelines restrict the use of some ICMP messages, an arbitrary machine can send an ICMP message to any other machine. Thus, a host can use ICMP to correspond with a router or another host. The chief advantage of allowing hosts to use ICMP is that it provides a single mechanism used for all control and information messages.

9.4 ICMP Message Delivery

ICMP messages require two levels of encapsulation as Figure 9.1 shows. Each ICMP message travels across the internet in the data portion of an IP datagram, which itself travels across each physical network in the data portion of a frame. Datagrams carrying ICMP messages are routed exactly like datagrams carrying information for users; there is no additional reliability or priority. Thus, error messages themselves may be lost or discarded. Furthermore, in an already congested network, the error message may cause additional congestion. An exception is made to the error handling procedures if an IP datagram carrying an ICMP message causes an error. The exception, established to avoid the problem of having error messages about error messages, specifies that ICMP messages are not generated for errors that result from datagrams carrying ICMP error messages.

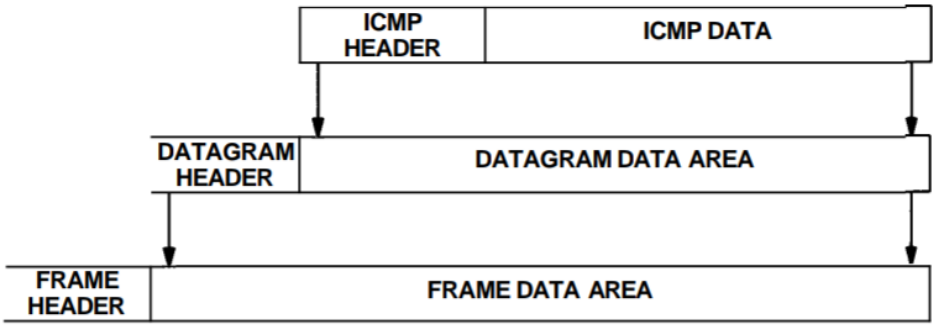


Figure 9.1 Two levels of ICMP encapsulation. The ICMP message is encapsulated in an IP datagram, which is further encapsulated in a frame for transmission. To identify ICMP, the datagram protocol field contains the value 1.

It is important to keep in mind that even though ICMP messages are encapsulated and sent using IP, ICMP is not considered a higher level protocol — it is a required part of IP. The reason for using IP to deliver ICMP messages is that they may need to travel across several physical networks to reach their final destination. Thus, they cannot be delivered by the physical transport alone.

9.5 ICMP Message Format

Although each ICMP message has its own format, they all begin with the same three fields: an 8-bit integer message *TYPE* field that identifies the message, an 8-bit ***CODE*** field that provides further information about the message type, and a 16-bit ***CHECKSUM*** field (ICMP uses the same additive checksum algorithm as IP, but the ICMP checksum only covers the ICMP message). In addition, ICMP messages that report errors always include the header and first 64 data bits of the datagram causing the problem.

The reason for returning more than the datagram header alone is to allow the receiver to determine more precisely which protocol(s) and which application program were responsible for the datagram. As we will see later, higher-level protocols in the TCP/IP suite are designed so that crucial information is encoded in the first 64 bits.

The ICMP ***TYPE*** field defines the meaning of the message as well as its format. The types include:

Type Field	ICMP Message Type
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect (change a route)
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded for a Datagram
12	Parameter Problem on a Datagram
13	Timestamp Request
14	Timestamp Reply
15	Information Request (obsolete)
16	Information Reply (obsolete)
17	Address Mask Request
18	Address Mask Reply

The next sections describe each of these messages, giving details of the message format and its meaning.

9.6 Testing Destination Reachability And Status (Ping)

TCP/IP protocols provide facilities to help network managers or users identify network problems. One of the most frequently used debugging tools invokes the ICMP ***echo request*** and ***echo reply*** messages. A host or router sends an ICMP echo request message to a specified destination. Any machine that receives an echo request formulates an echo reply and returns it to the original sender. The request contains an optional data area; the reply contains a copy of the data sent in the request. The echo request and associated reply can be used to test whether a destination is reachable and responding. Because both the request and reply travel in IP datagrams, successful receipt of a reply verifies that major pieces of the transport system work. First, IP software on the source computer must route the datagram. Second, intermediate routers between the source and destination must be operating and must route the datagram correctly. Third, the destination machine must be running (at least it must respond to interrupts), and both ICMP and IP software must be working. Finally, all routers along the return path must have correct routes.

On many systems, the command users invoke to send ICMP echo requests is named *ping*†. Sophisticated versions of ping send a series of ICMP echo requests, capture responses, and provide statistics about datagram loss. They allow the user to specify the length of the data being sent and the interval between requests. Less sophisticated versions merely send one ICMP echo request and await a reply.

9.7 Echo Request And Reply Message Format

Figure 9.2 shows the format of echo request and reply messages.

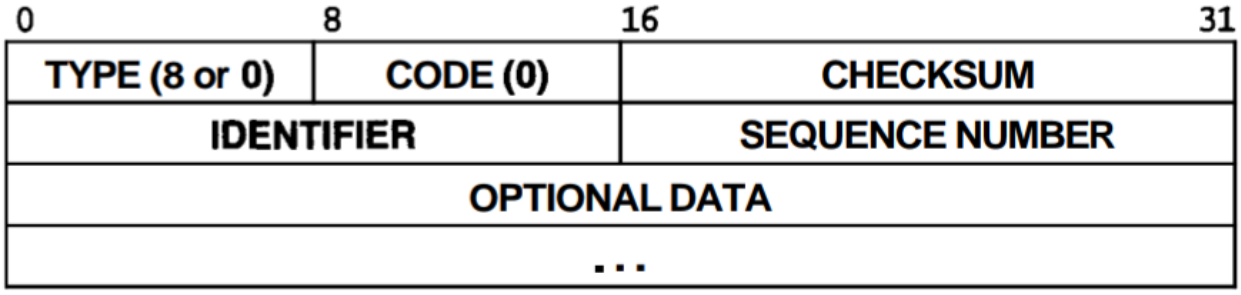


Figure 9.2 ICMP echo request or reply message format.

The field listed as ***OPTIONAL DATA*** is a variable length field that contains data to be returned to the sender. An echo reply always returns exactly the same data as was received in the request. Fields ***IDENTIFIER*** and ***SEQUENCE NUMBER*** are used by the sender to match replies to requests. The value of the ***TYPE*** field specifies whether the message is a request (8) or a reply (*0*).

9.8 Reports Of Unreachable Destinations

When a router cannot forward or deliver an IP datagram, it sends a *destination* unreachable message back to the original source, using the format shown in Figure 9.3.

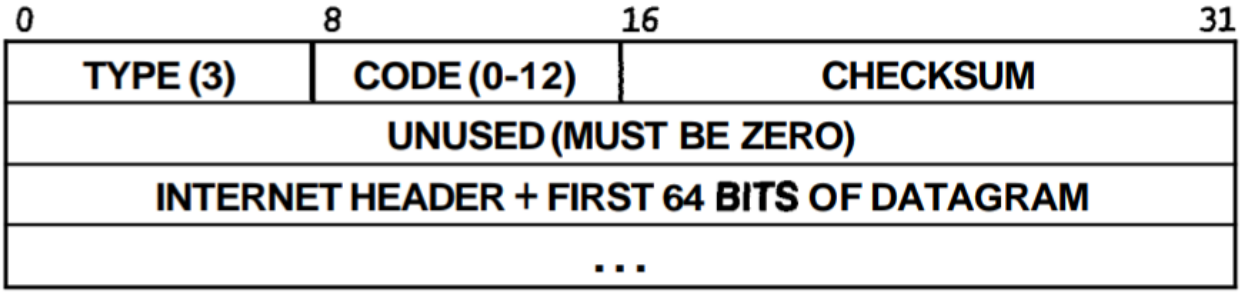


Figure 9.3 ICMP destination unreachable message format.

The CODE field in a destination unreachable message contains an integer that further describes the problem. Possible values are:

Code Value	Meaning
0	Network unreachable
1	Host unreachable
2	Protocol unreachable
3	Port unreachable
4	Fragmentation needed and DF set
5	Source route failed
6	Destination network unknown
7	Destination host unknown
8	Source host isolated
9	Communication with destination network administratively prohibited
10	Communication with destination host administratively prohibited
11	Network unreachable for type of service
12	Host unreachable for type of service

Although IP is a best-effort delivery mechanism, discarding datagrams should not be taken lightly. Whenever an error prevents a router from routing or delivering a datagram, the router sends a destination unreachable message back to the source and then ***drops*** (i.e., discards) the datagram. Network unreachable errors usually imply routing failures; host unreachable errors imply delivery failures†. Because the ICMP error message contains a short prefix of the datagram that caused the problem, the source will know exactly which address is unreachable.

Destinations may be unreachable because hardware is temporarily out of service, because the sender specified a nonexistent destination address, or (in rare circumstances) because the router does not have a route to the destination network. Note that although routers report failures they encounter, they may not know of all delivery failures. For example, if the destination machine connects to an Ethernet network, the network hardware does not provide acknowledgements. Therefore, a router can continue to send packets to a destination after the destination is powered down without receiving any indication that the packets are not being delivered. To summarize:

> ***Although a router sends a destination unreachable message when it encounters a datagram that cannot be forwarded or delivered, a router cannot detect all such errors.***

The meaning of protocol and port unreachable messages will become clear when we study how higher level protocols use abstract destination points called ***ports.*** Most of the remaining messages are self explanatory. If the datagram contains the source route option with an incorrect route, it may trigger a ***source route*** failure message. *If* a router needs to fragment a datagram but the "don't fragment" bit is set, the router sends a ***fragmentation needed*** message back to the source.

Distance Vector Routing (DVR) Protocol

A **distance-vector routing (DVR)** protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

Bellman Ford Basics – Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances,based on a chosen metric, are computed using information from the neighbors' distance vectors.

- Each router has an ID
- Associated with each link connected to a router, there is a link cost (static or dynamic).
- Intermediate hops

Distance Vector Table Initialization -

- Distance to itself = 0
- Distance to ALL other routers = infinity number.

Distance Vector Algorithm –

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:

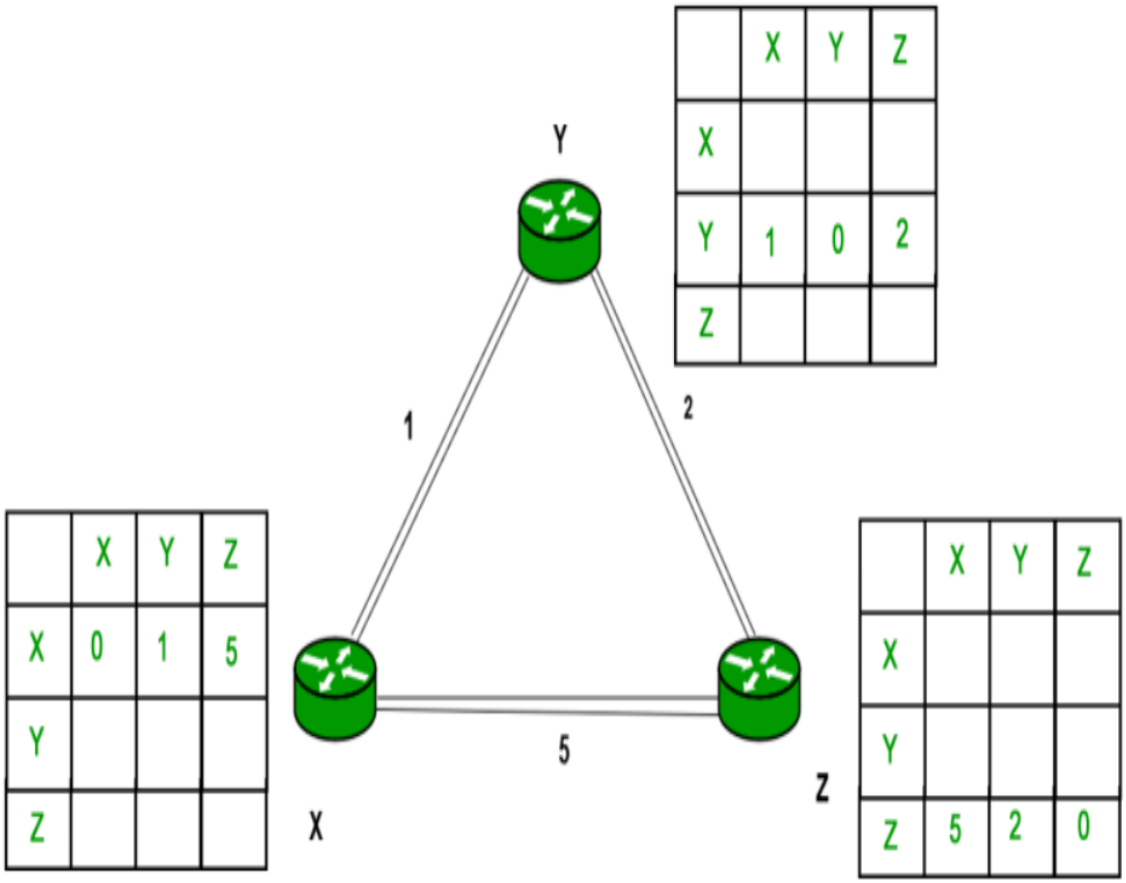
- It receives a distance vector from a neighbor containing different information than before.
- It discovers that a link to a neighbor has gone down.

The DV calculation is based on minimizing the cost to each destination

- From time-to-time, each node sends its own distance vector estimate to neighbors.
- When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation:

```
Dx(y) = min { C(x,v) + Dv(y), Dx(y) } f
```

Example – Consider 3-routers X, Y and Z as shown in figure. Each router have their routing table. Every routing table will contain distance to the destination nodes.



Consider router X , X will share it routing table to neighbors and neighbors will share it routing table to it to X and distance from node X to destination will be calculated using bellmen- ford equation.

Link State Routing

Link state routing is a technique in which each router shares the knowledge of its neighborhood with every other router in the internetwork.

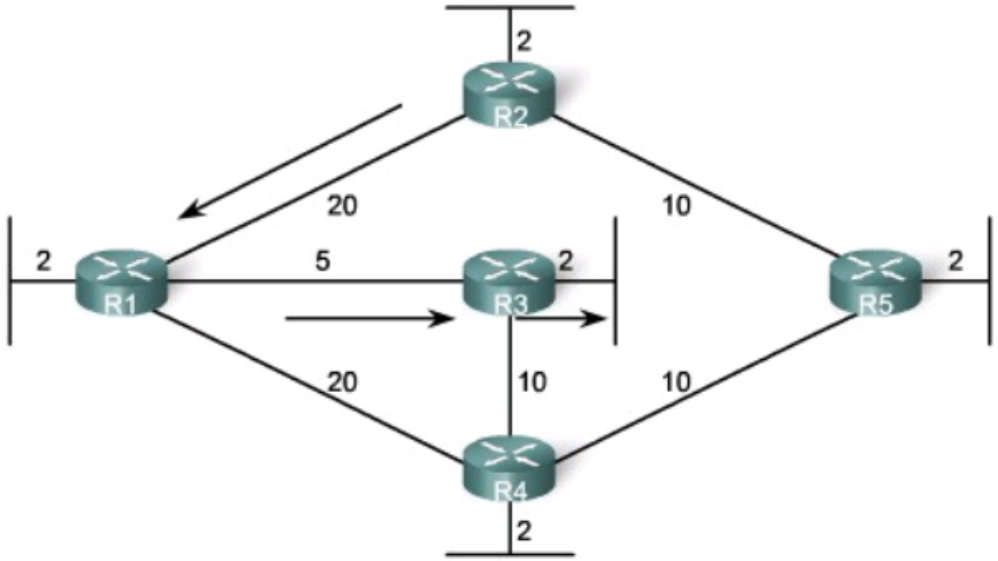
The three keys to understand the Link State Routing algorithm:

- **Knowledge about the neighborhood:** Instead of sending its routing table, a router sends the information about its neighborhood only. A router broadcast its identities and cost of the directly attached links to other routers.
- **Flooding:** Each router sends the information to every other router on the internetwork except its neighbors. This process is known as Flooding. Every router that receives the packet sends the copies to all its neighbors. Finally, each and every router receives a copy of the same information.
- **Information sharing:** A router sends the information to every other router only when the change occurs in the information.

Link states:

Information about the state of (Router interfaces) links is known as link-states. As you can see in the figure, this information includes:

- The interface's IP address and subnet mask.
- The type of network, such as Ethernet (broadcast) or Serial point-to-point link.
- The cost of that link.
- Any neighbor routers on that link.



Shortest Path for host on R2 LAN to reach host on R3 LAN:
R2 to R1 (20) + R1 to R3 (5) + R3 to LAN (2) = 27

So exactly ho g generic link-

1. **Each router learns about its own links, its own directly connected networks.** This is done by detecting that an interface is in the up state.

2. **Each router is responsible for meeting its neighbors on directly connected networks.** link state routers do this by exchanging Hello packets with other link-state routers on directly connected networks.

3. **Each router builds a Link-State Packet (LSP) containing the state of each directly connected link.** This is done by recording all the pertinent information about each neighbor, including neighbor ID, link type, and bandwidth.

4. **Each router floods the LSP to all neighbors, who then store all LSPs received in a database.** Neighbors then flood the LSPs to their neighbors until all routers in the area have received the LSPs. Each router stores a copy of each LSP received from its neighbors in a local database.

5. **Each router uses the database to construct a complete map of the topology and computes the best path to each destination network**. Like having a road map, the router now has a complete map of all destinations in the topology and the routes to reach them. The SPF algorithm is used to construct the map of the topology and to determine the best path to each network.

Advantages of Link state Routing protocol:

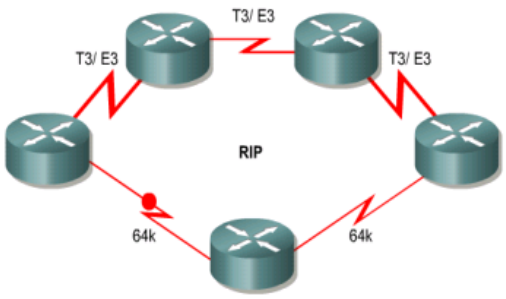
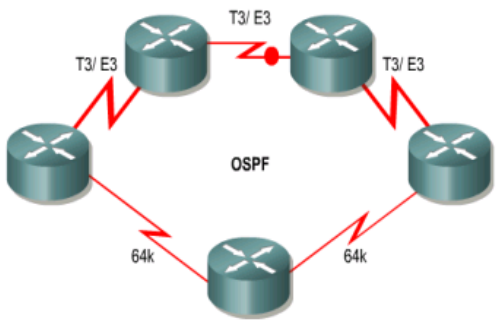
Build the topological map:
Link-state routing protocols create a topological map, or SPF tree of the network topology. Distance vector routing protocols do not have a topological map of the network.

Faster Convergence:
When receiving a Link-state Packet (LSP), link-state routing protocols immediately flood the LSP out all interfaces except for the interface from which the LSP was received. This way, it achieve the faster convergence. With distance vector routing algorithm, router needs to process each routing update and update its routing table before flooding them out other interfaces.

Event Driven Updates:
After the initial flooding of LSPs, link-state routing protocols only send out an LSP when there is a change in the topology. The LSP contains only the information regarding the affected link. Unlike some distance vector routing protocols, link-state routing protocols do not send periodic updates.

Distance vector vs. Link state:

Sno.	Distance Vector	Link State
1	Uses hop count as Metric.	Uses shortest path.
2	View the network from the perspective of neighbor.	Gets common view of entire network topology.
3	Has frequent and periodic updates	Has event triggered updates.
4	Slow convergence	Faster convergence

5	Susceptible to routing loops.	Not as susceptible to routing loops.
6	Easy to configure and administer.	Difficult to configure and administer.
7	Requires less memory and processing power of routers.	Requires more precessing power and memory than distance vector.
8	Consumes a lot of Bandwidth.	Consumes less BW than distance vector.
9	Passes copies of routing table to neighbor routers.	Passes link-state routing updates to other routers.
10	Eg. RIP 	Eg. OSPF 

Flow based routing:
A flooding algorithm is an algorithm for distributing material to every part of a connected network. The name derives from the concept of inundation by a flood. Its implemented by the ospf:

Advantages of Flooding
The main advantage of flooding the increased reliability provided by this routing method. Since the message will be sent at least once to every host it is almost guaranteed to reach its destination. In addition, the message will reach the host through the shortest possible path.

Disadvantages of Flooding
There are several disadvantages with this approach to routing. It is very wasteful in terms of the networks total bandwidth. While a message may only have one destination it has to be sent to every host. This increases the maximum load placed upon the network.

Messages can also become duplicated in the network further increasing the load on the networks bandwidth as well as requiring an increase in processing complexity to disregard duplicate messages.

A variant of flooding called *selective flooding* partially addresses these issues by only sending packets to routers in the same direction.

RIP (Routing Information Protocol)

RIP is a standardized Distance Vector protocol, designed for use on smaller networks. RIP was one of the first true Distance Vector routing protocols, and is supported on a wide variety of systems.

RIP adheres to the following Distance Vector characteristics:

- RIP sends out periodic routing updates (every **30 seconds**)
- RIP sends out the full routing table every periodic update
- RIP uses a form of distance as its metric (in this case, **hopcount**)
- RIP uses the Bellman-Ford Distance Vector algorithm to determine the best "path" to a particular destination

Other characteristics of RIP include:

- RIP supports IP and IPX routing.
- RIP utilizes UDP port 520
- RIP routes have an administrative distance of **120**.
- RIP has a maximum hopcount of **15 hops.**

Any network that is 16 hops away or more is considered unreachable to RIP, thus the maximum diameter of the network is 15 hops. A metric of 16 hops in RIP is considered a **poison route** or **infinity metric.**

If multiple paths exist to a particular destination, RIP will load balance between those paths (by default, up to **4**) only if the metric (hopcount) is **equal**. RIP uses a round-robin system of load-balancing between equal metric routes, which can lead to **pinhole congestion**.

For example, two paths might exist to a particular destination, one going through a 9600 baud link, the other via a T1. If the metric (hopcount) is equal, RIP will load-balance, sending an equal amount of traffic down the 9600 baud link and the T1. This will (obviously) cause the slower link to become congested.

RIP Versions

RIP has two versions, **Version 1** (**RIPv1**) and **Version 2** (**RIPv2**).

RIPv1 (RFC 1058) is **classful**, and thus does not include the subnet mask with its routing table updates. Because of this, RIPv1 does not support **Variable Length Subnet Masks (VLSMs**). When using RIPv1, networks must be contiguous, and subnets of a major network must be configured with identical subnet masks. Otherwise, route table inconsistencies (or worse) will occur.

RIPv1 sends updates as **broadcasts** to address 255.255.255.255.

RIPv2 (RFC 2543) is **classless**, and thus *does* include the subnet mask with its routing table updates. RIPv2 fully supports VLSMs, allowing discontiguous networks and varying subnet masks to exist.

Other enhancements offered by RIPv2 include:

- Routing updates are sent via **multicast**, using address **224.0.0.9**
- Encrypted authentication can be configured between RIPv2 routers
- Route tagging is supported (explained in a later section)

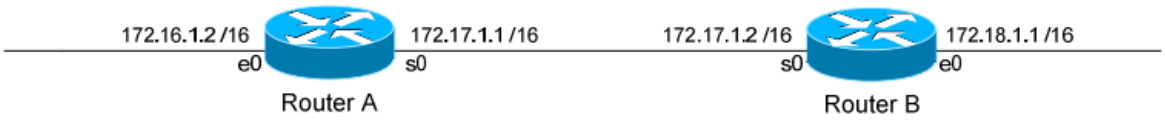
RIPv2 can interoperate with RIPv1. By default:

- RIPv1 routers will sent only Version 1 packets
- RIPv1 routers will receive both Version 1 and 2 updates
- RIPv2 routers will both send and receive only Version 2 updates

We can control the version of RIP a particular interface will "send" or "receive."

Unless RIPv2 is manually specified, a Cisco will default to RIPv1 when configuring RIP.

RIPv1 Basic Configuration



Routing protocol configuration occurs in Global Configuration mode. On Router A, to configure RIP, we would type:

Router(config)# *router rip*
Router(config-router)# *network 172.16.0.0*
Router(config-router)# *network 172.17.0.0*

The first command, *router rip*, enables the RIP process.

The *network* statements tell RIP which networks you wish to advertise to other RIP routers. We simply list the networks that are directly connected to our router. Notice that we specify the networks at their classful boundaries, and we do not specify a subnet mask.

To configure Router B:

Router(config)# *router rip*
Router(config-router)# *network 172.17.0.0*
Router(config-router)# *network 172.18.0.0*

The routing table on Router A will look like:

RouterA# *show ip route*

```
<eliminated irrelevant header>

Gateway of last resort is not set

C     172.16.0.0 is directly connected, Ethernet0
C     172.17.0.0 is directly connected, Serial0
R     172.18.0.0 [120/1] via 172.17.1.2, 00:00:00, Serial0
```

The routing table on Router B will look like:

RouterB# *show ip route*

```
<eliminated irrelevant header>

Gateway of last resort is not set

C     172.17.0.0 is directly connected, Serial0
C     172.18.0.0 is directly connected, Ethernet0
R     172.16.0.0 [120/1] via 172.17.1.1, 00:00:00, Serial0
```

* * *

RIP Timers

RIP has four basic timers:

Update Timer (default **30 seconds**) – indicates how often the router will send out a routing table update.

Invalid Timer (default **180 seconds**) – indicates how long a route will remain in a routing table before being marked as invalid, if no new updates are heard about this route. The invalid timer will be reset if an update is received for that particular route *before* the timer expires.

A route marked as invalid is *not* immediately removed from the routing table. Instead, the route is marked (and advertised) with a metric of 16, indicating it is unreachable, and placed in a **hold-down** state.

Hold-down Timer (default **180 seconds**) – indicates how long RIP will "suppress" a route that it has placed in a **hold-down** state. RIP will not accept any new updates for routes in a hold-down state, until the hold-down timer expires.

A route will enter a hold-down state for one of three reasons:

- The invalid timer has expired.
- An update has been received from another router, marking that route with a metric of 16 (or *unreachable*).
- An update has been received from another router, marking that route with a *higher* metric than what is currently in the routing table. This is to prevent loops.

Flush Timer (default **240 seconds**) **–** indicates how long a route can remain in a routing table before being flushed, if no new updates are heard about this route. The flush timer runs **concurrently with the invalid timer**, and thus will flush out a route 60 seconds after it has been marked invalid.

RIP timers must be identical on **all** routers on the RIP network, otherwise massive instability will occur.

- Open Shortest Path First -

OSPF (Open Shortest Path First)

OSPF is a standardized Link-State routing protocol, designed to scale efficiently to support larger networks.

OSPF adheres to the following Link State characteristics:

- OSPF employs a hierarchical network design using **Areas.**
- OSPF will form **neighbor** relationships with adjacent routers in the same **Area.**
- Instead of advertising the *distance* to connected networks, OSPF advertises the *status* of directly connected **links** using **Link-State Advertisements (LSAs)**.
- OSPF sends updates (LSAs) when there is a change to one of its links, and will *only* send the change in the update. LSAs are additionally refreshed every **30 minutes**.
- OSPF traffic is multicast either to address **224.0.0.5 (**all OSPF routers) or **224.0.0.6** (all Designated Routers).
- OSPF uses the **Dijkstra Shortest Path First** algorithm to determine the shortest path.
- OSPF is a classless protocol, and thus supports VLSMs.

Other characteristics of OSPF include:

- OSPF supports only IP routing.
- OSPF routes have an administrative distance is **110**.
- OSPF uses **cost** as its metric, which is computed based on the bandwidth of the link. OSPF has no hop-count limit.

The OSPF process builds and maintains three separate tables:

- A **neighbor table** – contains a list of all neighboring routers.
- A **topology table** – contains a list of *all* possible routes to all known networks within an area.
- A **routing table** – contains the *best* route for each known network.

OSPF Neighbors

OSPF forms neighbor relationships, called **adjacencies,** with other routers in the same **Area** by exchanging **Hello** packets to multicast address **224.0.0.5**. Only after an adjacency is formed can routers share routing information.

Each OSPF router is identified by a unique **Router ID.** The Router ID can be determined in one of three ways:

- The Router ID can be **manually** specified.
- If not manually specified, the highest IP address configured on any **Loopback interface** on the router will become the Router ID.
- If no loopback interface exists, the highest IP address configured on any **Physical interface** will become the Router ID.

By default, Hello packets are sent out OSPF-enabled interfaces every **10 seconds** for broadcast and point-to-point interfaces, and **30 seconds** for non-broadcast and point-to-multipoint interfaces.

OSPF also has a **Dead Interval**, which indicates how long a router will wait without hearing any hellos before announcing a neighbor as "down." Default for the Dead Interval is **40 seconds** for broadcast and point-to-point interfaces, and **120** seconds for non-broadcast and point-to-multipoint interfaces. Notice that, by default, the dead interval timer is four times the Hello interval.

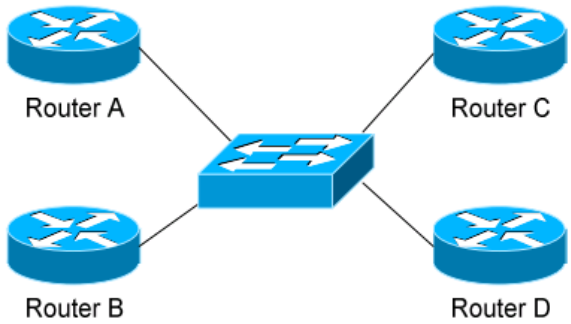
These timers can be adjusted on a *per interface* basis:

Router(config-if)# *ip ospf hello-interval 15*
Router(config-if)# *ip ospf dead-interval 60*

OSPF Designated Routers

In multi-access networks such as Ethernet, there is the possibility of *many* neighbor relationships on the same physical segment. In the above example, four routers are connected into the same multi-access segment. Using the following formula (where "n" is the number of routers):

n(n-1)/2



…..it is apparent that **6** separate adjacencies are needed for a fully meshed network. Increase the number of routers to five, and **10** separate adjacencies would be required. This leads to a considerable amount of unnecessary Link State Advertisement (LSA) traffic.

If a link off of Router A were to fail, it would flood this information to all neighbors. Each neighbor, in turn, would then flood that same information to all other neighbors. This is a waste of bandwidth and processor load.

To prevent this, OSPF will elect a **Designated Router (DR)** for each multi-access networks, accessed via multicast address **224.0.0.6**. For redundancy purposes, a **Backup Designated Router (BDR)** is also elected.

OSPF routers will form adjacencies with the DR and BDR. If a change occurs to a link, the update is forwarded only to the DR, which then forwards it to all other routers. This greatly reduces the flooding of LSAs.

DR and BDR elections are determined by a router's **OSPF priority**, which is configured on a per-interface basis (a router can have interfaces in multiple multi-access networks). The router with the **highest priority** becomes the DR; second highest becomes the BDR. If there is a tie in priority, whichever router has the **highest Router ID** will become the DR. To change the priority on an interface:

Router(config-if)# *ip ospf priority 125*

Default priority on Cisco routers is **1**. A priority of **0** will prevent the router from being elected DR or BDR. **Note:** The DR election process is ***not preemptive***. Thus, if a router with a higher priority is added to the network, it will *not* automatically supplant an existing DR. Thus, a router that should never become the DR should always have its priority set to 0.

* * *

OSPF Network Types

OSPF's functionality is different across several different network topology types. OSPF's interaction with Frame Relay will be explained in another section

Broadcast Multi-Access – indicates a topology where broadcast occurs.

- Examples include Ethernet, Token Ring, and ATM.
- OSPF *will* elect DRs and BDRs.
- Traffic to DRs and BDRs is multicast to 224.0.0.6. Traffic from DRs and BDRs to other routers is multicast to 224.0.0.5.
- Neighbors *do not* need to be manually specified.

Point-to-Point – indicates a topology where two routers are directly connected.

- An example would be a point-to-point T1.
- OSPF *will not* elect DRs and BDRs.
- All OSPF traffic is multicast to 224.0.0.5.
- Neighbors *do not* need to be manually specified.

Point-to-Multipoint – indicates a topology where one interface can connect to multiple destinations. Each connection between a source and destination is treated as a point-to-point link.

- An example would be Point-to-Multipoint Frame Relay.
- OSPF *will not* elect DRs and BDRs.
- All OSPF traffic is multicast to 224.0.0.5.
- Neighbors *do not* need to be manually specified.

Non-broadcast Multi-access Network (NBMA) – indicates a topology where one interface can connect to multiple destinations; however, broadcasts cannot be sent across a NBMA network.

- An example would be Frame Relay.
- OSPF *will* elect DRs and BDRs.
- OSPF neighbors must be *manually* defined, thus All OSPF traffic is unicast instead of multicast.

Remember: on *non-broadcast* networks, neighbors must be **manually specified**, as multicast Hello's are not allowed.