CHAPTER 18

INTERNET PROTOCOLS

The map of the London Underground, which can be seen inside every train, has been called a model of its kind, a work of art. It presents the underground network as a geometric grid. The tube lines do not, of course, lie at right angles to one another like the streets of Manhattan. Nor do they branch off at acute angles or form perfect oblongs.

—*King Solomon's Carpet*. Barbara Vine (Ruth Rendell)

KEY POINTS

- Key functions typically performed by a protocol include encapsulation, fragmentation and reassembly, connection control, ordered delivery, flow control, error control, addressing, and multiplexing.
- An internet consists of multiple separate networks that are interconnected by routers. Data are transmitted in packets from a source system to a destination across a path involving multiple networks and routers. Typically, a connectionless or datagram operation is used. A router accepts datagrams and relays them on toward their destination and is responsible for determining the route, much the same way as packet-switching nodes operate.
- The most widely used protocol for internetworking is the Internet Protocol (IP). IP attaches a header to upper-layer (e.g., TCP) data to form an IP datagram. The header includes source and destination addresses, information used for fragmentation and reassembly, a time-to-live field, a type-of-service field, and a checksum.
- A next-generation IP, known as IPv6, has been defined. IPv6 provides longer address fields and more functionality than the current IP.

The purpose of this chapter is to examine the Internet Protocol, which is the foundation on which all of the internet-based protocols and on which internetworking is based. First, it will be useful to review the basic functions of networking protocols. This review serves to summarize some of the material introduced previously and to set the stage for the study of internet-based protocols in Parts Five and Six. We then move to a discussion of internetworking. Next, the chapter focuses on the two standard internet protocols: IPv4 and IPv6. Finally, the topic of IP security is introduced.

Refer to Figure 2.5 to see the position within the TCP/IP suite of the protocols discussed in this chapter.

18.1 BASIC PROTOCOL FUNCTIONS

Before turning to a discussion of internet protocols, let us consider a rather small set of functions that form the basis of all protocols. Not all protocols have all functions; this would involve a significant duplication of effort. There are, nevertheless, many instances of the same type of function being present in protocols at different levels.

We can group protocol functions into the following categories:

- Encapsulation
- Fragmentation and reassembly
- Connection control
- Ordered delivery
- Flow control
- Error control
- Addressing
- Multiplexing
- Transmission services

Encapsulation

For virtually all protocols, data are transferred in blocks, called protocol data units (PDUs). Each PDU contains not only data but also control information. Indeed, some PDUs consist solely of control information and no data. The control information falls into three general categories:

- **Address:** The address of the sender and/or receiver may be indicated.
- **Error-detecting code:** Some sort of frame check sequence is often included for error detection.
- **Protocol control:** Additional information is included to implement the protocol functions listed in the remainder of this section.

The addition of control information to data is referred to as **encapsulation**. Data are accepted or generated by an entity and encapsulated into a PDU containing that data plus control information. Typically, the control information is contained in a PDU header; some data link layer PDUs include a trailer as well. Numerous examples of PDUs appear in the preceding chapters [e.g., TFTP (Figure 2.13), HDLC (Figure 7.7), frame relay (Figure 10.16), ATM (Figure 11.4), LLC (Figure 15.7), IEEE 802.3 (Figure 16.3), IEEE 802.11 (Figure 17.8)].

Fragmentation and Reassembly[1]

A protocol is concerned with exchanging data between two entities. Usually, the transfer can be characterized as consisting of a sequence of PDUs of some bounded size. Whether the application entity sends data in messages or in a continuous

[1]The term *segmentation* is used in OSI-related documents, but in protocol specifications related to the TCP/IP protocol suite, the term *fragmentation* is used. The meaning is the same.

stream, lower-level protocols typically organize the data into blocks. Further, a protocol may need to divide a block received from a higher layer into multiple blocks of some smaller bounded size. This process is called fragmentation.

There are a number of motivations for fragmentation, depending on the context. Among the typical reasons for fragmentation are the following:

- The communications network may only accept blocks of data up to a certain size. For example, an ATM network is limited to blocks of 53 octets; Ethernet imposes a maximum size of 1526 octets.
- Error control may be more efficient with a smaller PDU size. With smaller PDUs, fewer bits need to be retransmitted when a PDU suffers an error.
- More equitable access to shared transmission facilities, with shorter delay, can be provided. For example, without a maximum block size, one station could monopolize a multipoint medium.
- A smaller PDU size may mean that receiving entities can allocate smaller buffers.
- An entity may require that data transfer comes to some sort of "closure" from time to time, for checkpoint and restart/recovery operations.

There are several disadvantages to fragmentation that argue for making PDUs as large as possible:

- Because each PDU contains a certain amount of control information, smaller blocks have a greater percentage of overhead.
- PDU arrival may generate an interrupt that must be serviced. Smaller blocks result in more interrupts.
- More time is spent processing smaller, more numerous PDUs.

All of these factors must be taken into account by the protocol designer in determining minimum and maximum PDU size.

The counterpart of fragmentation is reassembly. Eventually, the segmented data must be reassembled into messages appropriate to the application level. If PDUs arrive out of order, the task is complicated.

Connection Control

An entity may transmit data to another entity in such a way that each PDU is treated independently of all prior PDUs. This is known as connectionless data transfer; an example is the use of the datagram, described in Chapter 10. While this mode is useful, an equally important technique is connection-oriented data transfer, of which the virtual circuit, also described in Chapter 10, is an example.

Connection-oriented data transfer is preferred (even required) if stations anticipate a lengthy exchange of data and/or certain details of their protocol must be worked out dynamically. A logical association, or connection, is established between the entities. Three phases occur (Figure 18.1):

- Connection establishment
- Data transfer
- Connection termination

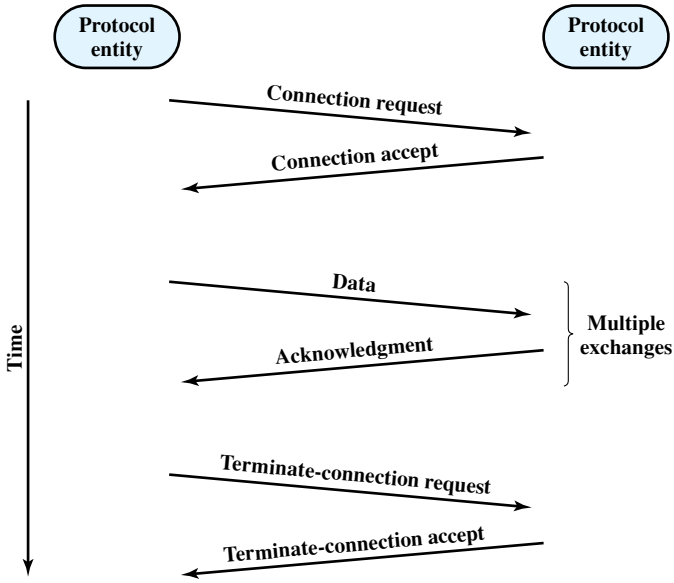


Figure 18.1 The Phases of a Connection-Oriented Data Transfer

With more sophisticated protocols, there may also be connection interrupt and recovery phases to cope with errors and other sorts of interruptions.

During the connection establishment phase, two entities agree to exchange data. Typically, one station will issue a connection request (in connectionless fashion) to the other. A central authority may or may not be involved. In simpler protocols, the receiving entity either accepts or rejects the request and, in the former case, the connection is considered to be established. In more complex proposals, this phase includes a negotiation concerning the syntax, semantics, and timing of the protocol. Both entities must, of course, be using the same protocol. But the protocol may allow certain optional features and these must be agreed upon by means of negotiation. For example, the protocol may specify a PDU size of up to 8000 octets; one station may wish to restrict this to 1000 octets.

Following connection establishment, the data transfer phase is entered. During this phase both data and control information (e.g., flow control, error control) are exchanged. Figure 18.1 shows a situation in which all of the data flow in one direction, with acknowledgments returned in the other direction. More typically, data and acknowledgments flow in both directions. Finally, one side or the other wishes to terminate the connection and does so by sending a termination request. Alternatively, a central authority might forcibly terminate a connection.

A key characteristic of many connection-oriented data transfer protocols is that sequencing is used (e.g., HDLC, IEEE 802.11). Each side sequentially numbers the PDUs that it sends to the other side. Because each side remembers that it is engaged in a logical connection, it can keep track of both outgoing numbers, which it generates, and incoming numbers, which are generated by the other side. Indeed, one can essentially define a connection-oriented data transfer as a transfer in which both sides

number PDUs and keep track of both incoming and outgoing numbers. Sequencing supports three main functions: ordered deliver, flow control, and error control.

Sequencing is not found in all connection-oriented protocols. Examples include frame relay and ATM. However, all connection-oriented protocols include in the PDU format some way of identifying the connection, which may be a unique connection identifier or a combination of source and destination addresses.

Ordered Delivery

If two communicating entities are in different hosts[2] connected by a network, there is a risk that PDUs will not arrive in the order in which they were sent, because they may traverse different paths through the network. In connection-oriented protocols, it is generally required that PDU order be maintained. For example, if a file is transferred between two systems, we would like to be assured that the records of the received file are in the same order as those of the transmitted file, and not shuffled. If each PDU is given a unique number, and numbers are assigned sequentially, then it is a logically simple task for the receiving entity to reorder received PDUs on the basis of sequence number. A problem with this scheme is that, with a finite sequence number field, sequence numbers repeat (modulo some maximum number). Evidently, the maximum sequence number must be greater than the maximum number of PDUs that could be outstanding at any time. In fact, the maximum number may need to be twice the maximum number of PDUs that could be outstanding (e.g., selective-repeat ARQ; see Chapter 7).

Flow Control

Flow control is a function performed by a receiving entity to limit the amount or rate of data that is sent by a transmitting entity.

The simplest form of flow control is a stop-and-wait procedure, in which each PDU must be acknowledged before the next can be sent. More efficient protocols involve some form of credit provided to the transmitter, which is the amount of data that can be sent without an acknowledgment. The HDLC sliding-window technique is an example of this mechanism (Chapter 7).

Flow control is a good example of a function that must be implemented in several protocols. Consider Figure 18.2, which repeats Figure 2.1. The network will need to exercise flow control over host A via the network access protocol, to enforce network traffic control. At the same time, B's network access module has finite buffer space and needs to exercise flow control over A's transmission; it can do this via the transport protocol. Finally, even though B's network access module can control its data flow, B's application may be vulnerable to overflow. For example, the application could be hung up waiting for disk access. Thus, flow control is also needed over the application-oriented protocol.

Error Control

Error control techniques are needed to guard against loss or damage of data and control information. Typically, error control is implemented as two separate

[2]The term *host* refers to any end system attached to a network, such as a PC, workstation, or server.

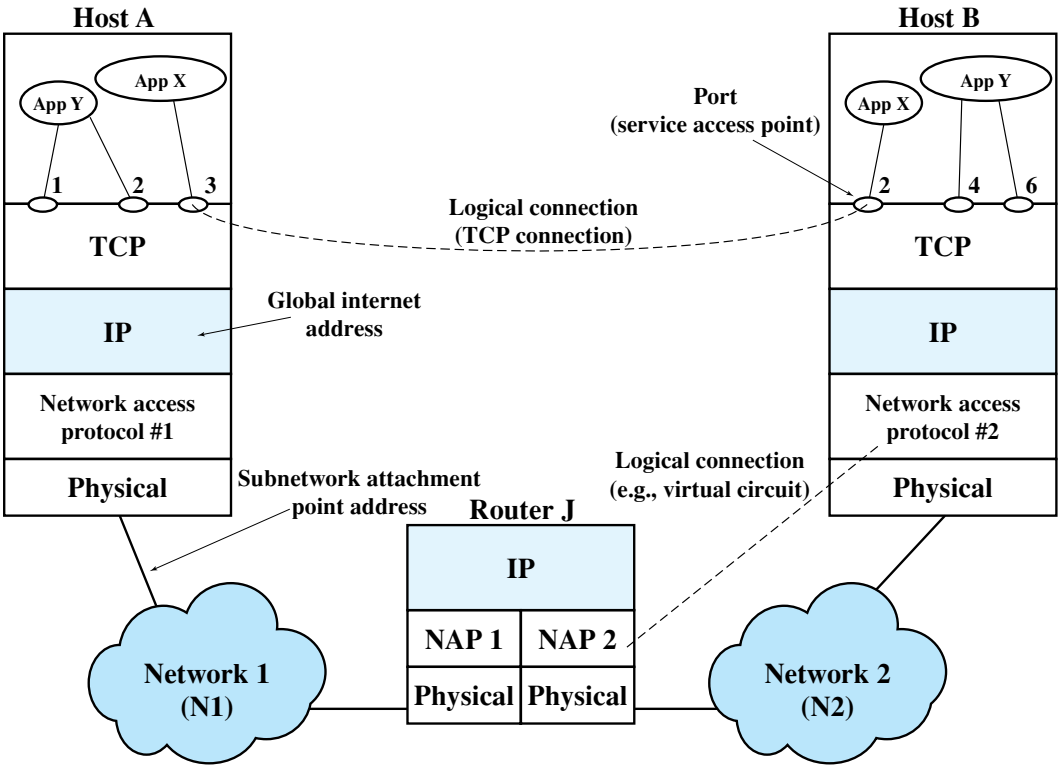


Figure 18.2 TCP/IP Concepts

functions: error detection and retransmission. To achieve error detection, the sender inserts an error-detecting code in the transmitted PDU, which is a function of the other bits in the PDU. The receiver checks the value of the code on the incoming PDU. If an error is detected, the receiver discards the PDU. Upon failing to receive an acknowledgment to the PDU in a reasonable time, the sender retransmits the PDU. Some protocols also employ an error correction code, which enables the receiver not only to detect errors but, in some cases, to correct them.

As with flow control, error control is a function that must be performed at various layers of protocol. Consider again Figure 18.2. The network access protocol should include error control to assure that data are successfully exchanged between station and network. However, a packet of data may be lost inside the network, and the transport protocol should be able to recover from this loss.

Addressing

The concept of addressing in a communications architecture is a complex one and covers a number of issues, including

- Addressing level
- Addressing scope

- Connection identifiers
- Addressing mode

During this discussion, we illustrate the concepts using Figure 18.2, which shows a configuration using the TCP/IP architecture. The concepts are essentially the same for the OSI architecture or any other communications architecture.

Addressing level refers to the level in the communications architecture at which an entity is named. Typically, a unique address is associated with each end system (e.g., workstation or server) and each intermediate system (e.g., router) in a configuration. Such an address is, in general, a network-level address. In the case of the TCP/IP architecture, this is referred to as an IP address, or simply an internet address. In the case of the OSI architecture, this is referred to as a network service access point (NSAP). The network-level address is used to route a PDU through a network or networks to a system indicated by a network-level address in the PDU.

Once data arrive at a destination system, they must be routed to some process or application in the system. Typically, a system will support multiple applications and an application may support multiple users. Each application and, perhaps, each concurrent user of an application is assigned a unique identifier, referred to as a port in the TCP/IP architecture and as a service access point (SAP) in the OSI architecture. For example, a host system might support both an electronic mail application and a file transfer application. At minimum each application would have a port number or SAP that is unique within that system. Further, the file transfer application might support multiple simultaneous transfers, in which case, each transfer is dynamically assigned a unique port number or SAP.

Figure 18.2 illustrates two levels of addressing within a system. This is typically the case for the TCP/IP architecture. However, there can be addressing at each level of an architecture. For example, a unique SAP can be assigned to each level of the OSI architecture.

Another issue that relates to the address of an end system or intermediate system is **addressing scope**. The internet address or NSAP address referred to previously is a global address. The key characteristics of a global address are as follows:

- **Global nonambiguity:** A global address identifies a unique system. Synonyms are permitted. That is, a system may have more than one global address.
- **Global applicability:** It is possible at any global address to identify any other global address, in any system, by means of the global address of the other system.

Because a global address is unique and globally applicable, it enables an internet to route data from any system attached to any network to any other system attached to any other network.

Figure 18.2 illustrates that another level of addressing may be required. Each network must maintain a unique address for each device interface on the network. Examples are a MAC address on an IEEE 802 network and an ATM host address. This address enables the network to route data units (e.g., MAC frames, ATM cells) through the network and deliver them to the intended attached system. We can refer to such an address as a *network attachment point address*.

The issue of addressing scope is generally only relevant for network-level addresses. A port or SAP above the network level is unique within a given system

but need not be globally unique. For example, in Figure 18.2, there can be a port 1 in system A and a port 1 in system B. The full designation of these two ports could be expressed as A.1 and B.1, which are unique designations.

The concept of **connection identifiers** comes into play when we consider connection-oriented data transfer (e.g., virtual circuit) rather than connectionless data transfer (e.g., datagram). For connectionless data transfer, a global identifier is used with each data transmission. For connection-oriented transfer, it is sometimes desirable to use only a connection identifier during the data transfer phase. The scenario is this: Entity 1 on system A requests a connection to entity 2 on system B, perhaps using the global address B.2. When B.2 accepts the connection, a connection identifier (usually a number) is provided and is used by both entities for future transmissions. The use of a connection identifier has several advantages:

- **Reduced overhead:** Connection identifiers are generally shorter than global identifiers. For example, in the frame relay protocol (discussed in Chapter 10), connection request packets contain both source and destination address fields. After a logical connection, called a data link connection, is established, data frames contain a data link connection identifier (DLCI) of 10, 16, or 23 bits.
- **Routing:** In setting up a connection, a fixed route may be defined. The connection identifier serves to identify the route to intermediate systems, such as packet-switching nodes, for handling future PDUs.
- **Multiplexing:** We address this function in more general terms later. Here we note that an entity may wish to enjoy more than one connection simultaneously. Thus, incoming PDUs must be identified by connection identifier.
- **Use of state information:** Once a connection is established, the end systems can maintain state information relating to the connection. This enables such functions as flow control and error control using sequence numbers. We see examples of this with HDLC (Chapter 7) and IEEE 802.11 (Chapter 17).

Figure 18.2 shows several examples of connections. The logical connection between router J and host B is at the network level. For example, if network 2 is a frame relay network, then this logical connection would be a data link connection. At a higher level, many transport-level protocols, such as TCP, support logical connections between users of the transport service. Thus, TCP can maintain a connection between two ports on different systems.

Another addressing concept is that of **addressing mode**. Most commonly, an address refers to a single system or port; in this case it is referred to as an individual or **unicast** address. It is also possible for an address to refer to more than one entity or port. Such an address identifies multiple simultaneous recipients for data. For example, a user might wish to send a memo to a number of individuals. The network control center may wish to notify all users that the network is going down. An address for multiple recipients may be **broadcast**, intended for all entities within a domain, or **multicast**, intended for a specific subset of entities. Table 18.1 illustrates the possibilities.

Multiplexing

Related to the concept of addressing is that of multiplexing. One form of multiplexing is supported by means of multiple connections into a single system. For

Table 18.1 Addressing Modes

Destination	Network Address	System Address	Port/SAP Address
Unicast	Individual	Individual	Individual
Multicast	Individual	Individual	Group
	Individual	All	Group
	All	All	Group
Broadcast	Individual	Individual	All
	Individual	All	All
	All	All	All

example, with frame relay, there can be multiple data link connections terminating in a single end system; we can say that these data link connections are multiplexed over the single physical interface between the end system and the network. Multiplexing can also be accomplished via port names, which also permit multiple simultaneous connections. For example, there can be multiple TCP connections terminating in a given system, each connection supporting a different pair of ports.

Multiplexing is used in another context as well, namely the mapping of connections from one level to another. Consider again Figure 18.2. Network 1 might provide a connection-oriented service. For each process-to-process connection established at the next higher level, a data link connection could be created at the network access level. This is a one-to-one relationship, but it need not be so. Multiplexing can be used in one of two directions. Upward multiplexing, or inward multiplexing, occurs when multiple higher-level connections are multiplexed on, or share, a single lower-level connection. This may be needed to make more efficient use of the lower-level service or to provide several higher-level connections in an environment where only a single lower-level connection exists. Downward multiplexing, or splitting, means that a single higher-level connection is built on top of multiple lower-level connections, and the traffic on the higher connection is divided among the various lower connections. This technique may be used to provide reliability, performance, or efficiency.

Transmission Services

A protocol may provide a variety of additional services to the entities that use it. We mention here three common examples:

- **Priority:** Certain messages, such as control messages, may need to get through to the destination entity with minimum delay. An example would be a terminate-connection request. Thus, priority could be assigned on a message basis. Additionally, priority could be assigned on a connection basis.
- **Quality of service:** Certain classes of data may require a minimum throughput or a maximum delay threshold.
- **Security:** Security mechanisms, restricting access, may be invoked.

All of these services depend on the underlying transmission system and any intervening lower-level entities. If it is possible for these services to be provided from below, the protocol can be used by the two entities to exercise those services.

18.2 PRINCIPLES OF INTERNETWORKING

Packet-switching and packet-broadcasting networks grew out of a need to allow the computer user to have access to resources beyond that available in a single system. In a similar fashion, the resources of a single network are often inadequate to meet users' needs. Because the networks that might be of interest exhibit so many differences, it is impractical to consider merging them into a single network. Rather, what is needed is the ability to interconnect various networks so that any two stations on any of the constituent networks can communicate.

Table 18.2 lists some commonly used terms relating to the interconnection of networks, or internetworking. An interconnected set of networks, from a user's point of view, may appear simply as a larger network. However, if each of the constituent networks retains its identity and special mechanisms are needed for communicating across multiple networks, then the entire configuration is often referred to as an **internet**.

Each constituent network in an internet supports communication among the devices attached to that network; these devices are referred to as **end systems** (ESs). In addition, networks are connected by devices referred to in the ISO documents as **intermediate systems** (ISs). Intermediate systems provide a communications path

Table 18.2 Internetworking Terms

Communication Network
A facility that provides a data transfer service among devices attached to the network.

Internet
A collection of communication networks interconnected by bridges and/or routers.

Intranet
An internet used by a single organization that provides the key Internet applications, especially the World Wide Web. An intranet operates within the organization for internal purposes and can exist as an isolated, self-contained internet, or may have links to the Internet.

Subnetwork
Refers to a constituent network of an internet. This avoids ambiguity because the entire internet, from a user's point of view, is a single network.

End System (ES)
A device attached to one of the networks of an internet that is used to support end-user applications or services.

Intermediate System (IS)
A device used to connect two networks and permit communication between end systems attached to different networks.

Bridge
An IS used to connect two LANs that use similar LAN protocols. The bridge acts as an address filter, picking up packets from one LAN that are intended for a destination on another LAN and passing those packets on. The bridge does not modify the contents of the packets and does not add anything to the packet. The bridge operates at layer 2 of the OSI model.

Router
An IS used to connect two networks that may or may not be similar. The router employs an internet protocol present in each router and each end system of the network. The router operates at layer 3 of the OSI model.

and perform the necessary relaying and routing functions so that data can be exchanged between devices attached to different networks in the internet.

Two types of ISs of particular interest are bridges and routers. The differences between them have to do with the types of protocols used for the internetworking logic. In essence, a **bridge** operates at layer 2 of the open systems interconnection (OSI) seven-layer architecture and acts as a relay of frames between similar networks; bridges are discussed in Chapter 15. A **router** operates at layer 3 of the OSI architecture and routes packets between potentially different networks. Both the bridge and the router assume that the same upper-layer protocols are in use.

We begin our examination of internetworking with a discussion of the basic principles of internetworking. We then examine the most important architectural approach to internetworking: the connectionless router.

Requirements

The overall requirements for an internetworking facility are as follows (we refer to Figure 18.2 as an example throughout):

1. Provide a link between networks. At minimum, a physical and link control connection is needed. (Router J has physical links to N1 and N2, and on each link there is a data link protocol.)
2. Provide for the routing and delivery of data between processes on different networks. (Application X on host A exchanges data with application X on host B.)
3. Provide an accounting service that keeps track of the use of the various networks and routers and maintains status information.
4. Provide the services just listed in such a way as not to require modifications to the networking architecture of any of the constituent networks. This means that the internetworking facility must accommodate a number of differences among networks. These include

    - **Different addressing schemes:** The networks may use different endpoint names and addresses and directory maintenance schemes. Some form of global network addressing must be provided, as well as a directory service. (Hosts A and B and router J have globally unique IP addresses.)
    - **Different maximum packet size:** Packets from one network may have to be broken up into smaller pieces for another. This process is referred to as fragmentation. (N1 and N2 may set different upper limits on packet sizes.)
    - **Different network access mechanisms:** The network access mechanism between station and network may be different for stations on different networks. (For example, N1 may be a frame relay network and N2 an Ethernet network.)
    - **Different timeouts:** Typically, a connection-oriented transport service will await an acknowledgment until a timeout expires, at which time it will retransmit its block of data. In general, longer times are required for successful delivery across multiple networks. Internetwork timing procedures must allow successful transmission that avoids unnecessary retransmissions.

- **Error recovery:** Network procedures may provide anything from no error recovery up to reliable end-to-end (within the network) service. The internetwork service should not depend on nor be interfered with by the nature of the individual network's error recovery capability.
- **Status reporting:** Different networks report status and performance differently. Yet it must be possible for the internetworking facility to provide such information on internetworking activity to interested and authorized processes.
- **Routing techniques:** Intranetwork routing may depend on fault detection and congestion control techniques peculiar to each network. The internetworking facility must be able to coordinate these to route data adaptively between stations on different networks.
- **User access control:** Each network will have its own user access control technique (authorization for use of the network). These must be invoked by the internetwork facility as needed. Further, a separate internetwork access control technique may be required.
- **Connection, connectionless:** Individual networks may provide connection-oriented (e.g., virtual circuit) or connectionless (datagram) service. It may be desirable for the internetwork service not to depend on the nature of the connection service of the individual networks.

The Internet Protocol (IP) meets some of these requirements. Others require additional control and application software, as we shall see in this chapter and the next.

Connectionless Operation

In virtually all implementation, internetworking involves connectionless operation at the level of the Internet Protocol. Whereas connection-oriented operation corresponds to the virtual circuit mechanism of a packet-switching network (Figure 10.10), connectionless-mode operation corresponds to the datagram mechanism of a packet-switching network (Figure 10.9). Each network protocol data unit is treated independently and routed from source ES to destination ES through a series of routers and networks. For each data unit transmitted by A, A makes a decision as to which router should receive the data unit. The data unit hops across the internet from one router to the next until it reaches the destination network. At each router, a routing decision is made (independently for each data unit) concerning the next hop. Thus, different data units may travel different routes between source and destination ES.

All ESs and all routers share a common network-layer protocol known generically as the Internet Protocol. An Internet Protocol (IP) was initially developed for the DARPA internet project and published as RFC 791 and has become an Internet Standard. Below this Internet Protocol, a protocol is needed to access a particular network. Thus, there are typically two protocols operating in each ES and router at the network layer: an upper sublayer that provides the internetworking function, and a lower sublayer that provides network access. Figure 18.3 shows an example.

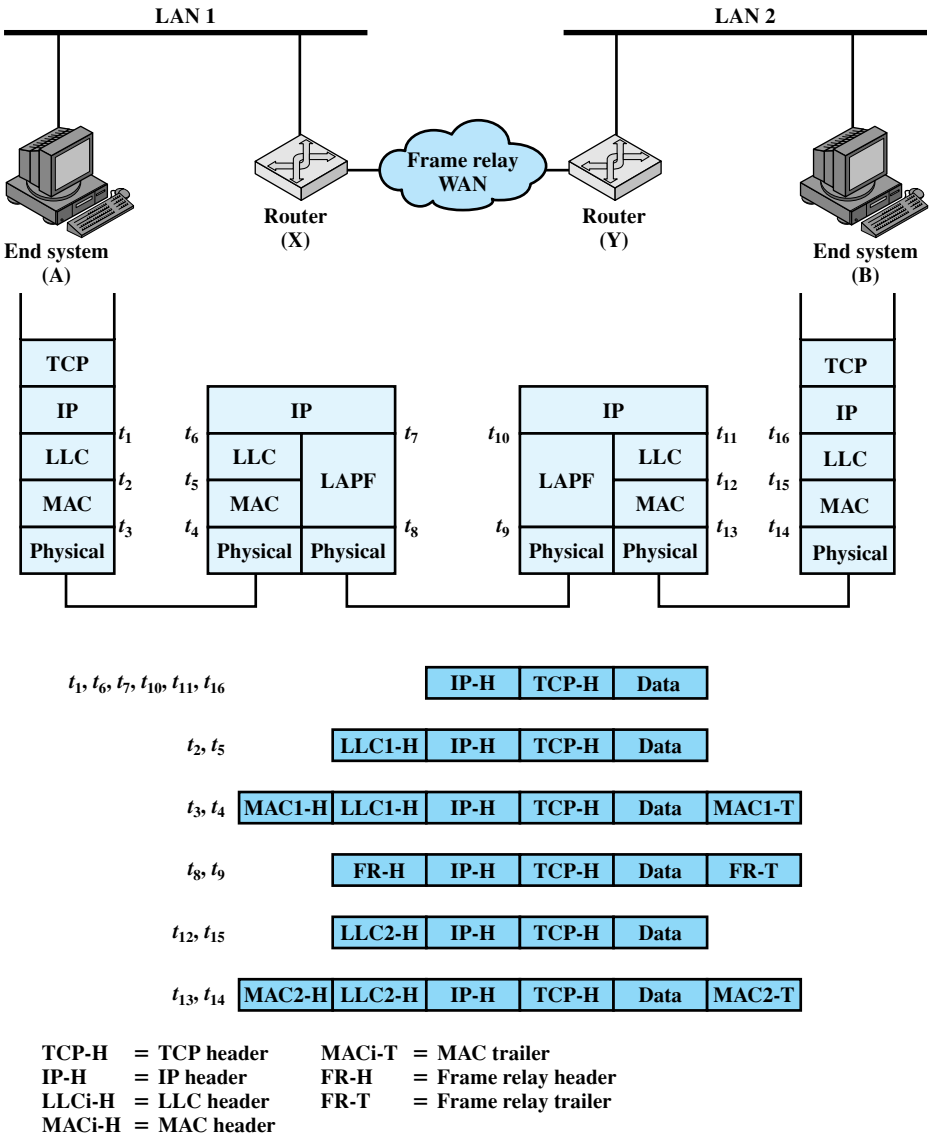


Figure 18.3 Example of Internet Protocol Operation

18.3 INTERNET PROTOCOL OPERATION

In this section, we examine the essential functions of an internetwork protocol. For convenience, we refer specifically to the Internet Standard IPv4, but the narrative in this section applies to any connectionless Internet Protocol, such as IPv6.

Operation of a Connectionless Internetworking Scheme

IP provides a connectionless, or datagram, service between end systems. There are a number of advantages to this approach:

- A connectionless internet facility is flexible. It can deal with a variety of networks, some of which are themselves connectionless. In essence, IP requires very little from the constituent networks.
- A connectionless internet service can be made highly robust. This is basically the same argument made for a datagram network service versus a virtual circuit service. For a further discussion, see Section 10.5.
- A connectionless internet service is best for connectionless transport protocols, because it does not impose unnecessary overhead.

Figure 18.3 depicts a typical example using IP, in which two LANs are interconnected by a frame relay WAN. The figure depicts the operation of the Internet Protocol for data exchange between host A on one LAN (network 1) and host B on another LAN (network 2) through the WAN. The figure shows the protocol architecture and format of the data unit at each stage. The end systems and routers must all share a common Internet Protocol. In addition, the end systems must share the same protocols above IP. The intermediate routers need only implement up through IP.

The IP at A receives blocks of data to be sent to B from a higher layers of software in A (e.g., TCP or UDP). IP attaches a header (at time t_1) specifying, among other things, the global internet address of B. That address is logically in two parts: network identifier and end system identifier. The combination of IP header and upper-level data is called an Internet Protocol data unit (PDU), or simply a datagram. The datagram is then encapsulated with the LAN protocol (LLC header at t_2; MAC header and trailer at t_3) and sent to the router, which strips off the LAN fields to read the IP header (t_6). The router then encapsulates the datagram with the frame relay protocol fields (t_8) and transmits it across the WAN to another router. This router strips off the frame relay fields and recovers the datagram, which it then wraps in LAN fields appropriate to LAN 2 and sends it to B.

Let us now look at this example in more detail. End system A has a datagram to transmit to end system B; the datagram includes the internet address of B. The IP module in A recognizes that the destination (B) is on another network. So the first step is to send the data to a router, in this case router X. To do this, IP passes the datagram down to the next lower layer (in this case LLC) with instructions to send it to router X. LLC in turn passes this information down to the MAC layer, which inserts the MAC-level address of router X into the MAC header. Thus, the block of data transmitted onto LAN 1 includes data from a layer or layers above TCP, plus a TCP header, an IP header, an LLC header, and a MAC header and trailer (time t_3 in Figure 18.3).

Next, the packet travels through network 1 to router X. The router removes MAC and LLC fields and analyzes the IP header to determine the ultimate destination of the data, in this case B. The router must now make a routing decision. There are three possibilities:

1. The destination station B is connected directly to one of the networks to which the router is attached. If so, the router sends the datagram directly to the destination.

2. To reach the destination, one or more additional routers must be traversed. If so, a routing decision must be made: To which router should the datagram be sent? In both cases 1 and 2, the IP module in the router sends the datagram down to the next lower layer with the destination network address. Please note that we are speaking here of a lower-layer address that refers to this network.
3. The router does not know the destination address. In this case, the router returns an error message to the source of the datagram.

In this example, the data must pass through router Y before reaching the destination. So router X constructs a new frame by appending a frame relay (LAPF) header and trailer to the IP datagram. The frame relay header indicates a logical connection to router Y. When this frame arrives at router Y, the frame header and trailer are stripped off. The router determines that this IP data unit is destined for B, which is connected directly to a network to which this router is attached. The router therefore creates a frame with a layer-2 destination address of B and sends it out onto LAN 2. The data finally arrive at B, where the LAN and IP headers can be stripped off.

At each router, before the data can be forwarded, the router may need to fragment the datagram to accommodate a smaller maximum packet size limitation on the outgoing network. If so, the data unit is split into two or more fragments, each of which becomes an independent IP datagram. Each new data unit is wrapped in a lower-layer packet and queued for transmission. The router may also limit the length of its queue for each network to which it attaches so as to avoid having a slow network penalize a faster one. Once the queue limit is reached, additional data units are simply dropped.

The process just described continues through as many routers as it takes for the data unit to reach its destination. As with a router, the destination end system recovers the IP datagram from its network wrapping. If fragmentation has occurred, the IP module in the destination end system buffers the incoming data until the entire original data field can be reassembled. This block of data is then passed to a higher layer in the end system.[3]

This service offered by IP is an unreliable one. That is, IP does not guarantee that all data will be delivered or that the data that are delivered will arrive in the proper order. It is the responsibility of the next higher layer (e.g., TCP) to recover from any errors that occur. This approach provides for a great deal of flexibility.

With the Internet Protocol approach, each unit of data is passed from router to router in an attempt to get from source to destination. Because delivery is not guaranteed, there is no particular reliability requirement on any of the networks. Thus, the protocol will work with any combination of network types. Because the sequence of delivery is not guaranteed, successive data units can follow different paths through the internet. This allows the protocol to react to both congestion and failure in the internet by changing routes.

[3]Appendix L provides a more detailed example, showing the involvement of all protocol layers.

Design Issues

With that brief sketch of the operation of an IP-controlled internet, we now examine some design issues in greater detail:

- Routing
- Datagram lifetime
- Fragmentation and reassembly
- Error control
- Flow control

As we proceed with this discussion, note the many similarities with design issues and techniques relevant to packet-switching networks. To see the reason for this, consider Figure 18.4, which compares an internet architecture with a packet-switching network architecture. The routers (R1, R2, R3) in the internet correspond

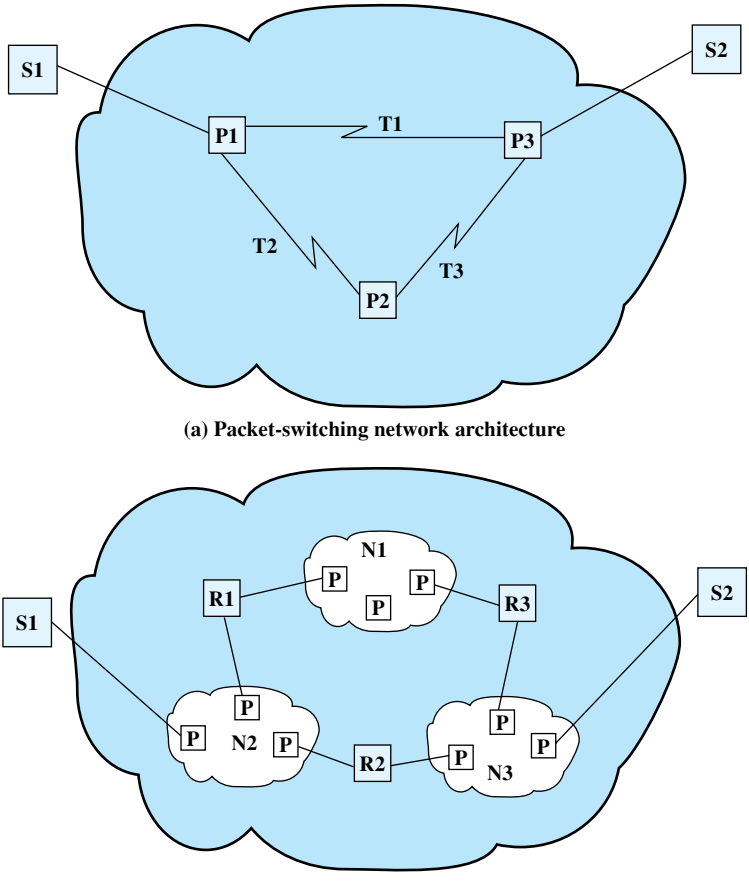


Figure 18.4 The Internet as a Network (based on [HIND83])

to the packet-switching nodes (P1, P2, P3) in the network, and the networks (N1, N2, N3) in the internet correspond to the transmission links (T1, T2, T3) in the networks. The routers perform essentially the same functions as packet-switching nodes and use the intervening networks in a manner analogous to transmission links.

Routing For the purpose of routing, each end system and router maintains a routing table that lists, for each possible destination network, the next router to which the internet datagram should be sent.

The routing table may be static or dynamic. A static table, however, could contain alternate routes if a particular router is unavailable. A dynamic table is more flexible in responding to both error and congestion conditions. In the Internet, for example, when a router goes down, all of its neighbors will send out a status report, allowing other routers and stations to update their routing tables. A similar scheme can be used to control congestion. Congestion control is particularly important because of the mismatch in capacity between local and wide area networks. Chapter 19 discusses routing protocols.

Routing tables may also be used to support other internetworking services, such as security and priority. For example, individual networks might be classified to handle data up to a given security classification. The routing mechanism must assure that data of a given security level are not allowed to pass through networks not cleared to handle such data.

Another routing technique is source routing. The source station specifies the route by including a sequential list of routers in the datagram. This, again, could be useful for security or priority requirements.

Finally, we mention a service related to routing: route recording. To record a route, each router appends its internet address to a list of addresses in the datagram. This feature is useful for testing and debugging purposes.

Datagram Lifetime If dynamic or alternate routing is used, the potential exists for a datagram to loop indefinitely through the internet. This is undesirable for two reasons. First, an endlessly circulating datagram consumes resources. Second, we will see in Chapter 20 that a transport protocol may depend on the existence of an upper bound on datagram lifetime. To avoid these problems, each datagram can be marked with a lifetime. Once the lifetime expires, the datagram is discarded.

A simple way to implement lifetime is to use a hop count. Each time that a datagram passes through a router, the count is decremented. Alternatively, the lifetime could be a true measure of time. This requires that the routers must somehow know how long it has been since the datagram or fragment last crossed a router, to know by how much to decrement the lifetime field. This would seem to require some global clocking mechanism. The advantage of using a true time measure is that it can be used in the reassembly algorithm, described next.

Fragmentation and Reassembly Individual networks within an internet may specify different maximum packet sizes. It would be inefficient and unwieldy to try to dictate uniform packet size across networks. Thus, routers may need to fragment incoming datagrams into smaller pieces, called segments or fragments, before transmitting on to the next network.

If datagrams can be fragmented (perhaps more than once) in the course of their travels, the question arises as to where they should be reassembled. The easiest solution is to have reassembly performed at the destination only. The principal disadvantage of this approach is that fragments can only get smaller as data move through the internet. This may impair the efficiency of some networks. However, if intermediate router reassembly is allowed, the following disadvantages result:

1. Large buffers are required at routers, and there is the risk that all of the buffer space will be used up storing partial datagrams.
2. All fragments of a datagram must pass through the same router. This inhibits the use of dynamic routing.

In IP, datagram fragments are reassembled at the destination end system. The IP fragmentation technique uses the following information in the IP header:

- Data Unit Identifier (ID)
- Data Length[4]
- Offset
- More Flag

The *ID* is a means of uniquely identifying an end-system-originated datagram. In IP, it consists of the source and destination addresses, a number that corresponds to the protocol layer that generated the data (e.g., TCP), and an identification supplied by that protocol layer. The *Data Length* is the length of the user data field in octets, and the *Offset* is the position of a fragment of user data in the data field of the original datagram, in multiples of 64 bits.

The source end system creates a datagram with a *Data Length* equal to the entire length of the data field, with *Offset* = 0, and a *More Flag* set to 0 (false). To fragment a long datagram into two pieces, an IP module in a router performs the following tasks:

1. Create two new datagrams and copy the header fields of the incoming datagram into both.
2. Divide the incoming user data field into two portions along a 64-bit boundary (counting from the beginning), placing one portion in each new datagram. The first portion must be a multiple of 64 bits (8 octets).
3. Set the *Data Length* of the first new datagram to the length of the inserted data, and set *More Flag* to 1 (true). The *Offset* field is unchanged.
4. Set the *Data Length* of the second new datagram to the length of the inserted data, and add the length of the first data portion divided by 8 to the *Offset* field. The *More Flag* remains the same.

Figure 18.5 gives an example in which two fragments are created from an original IP datagram. The procedure is easily generalized to an *n*-way split. In this example, the payload of the original IP datagram is a TCP segment, consisting of a

[4]In the IPv6 header, there is a Payload Length field that corresponds to Data Length in this discussion. In the IPv4 header, there is Total Length field whose value is the length of the header plus data; the data length must be calculated by subtracting the header length.

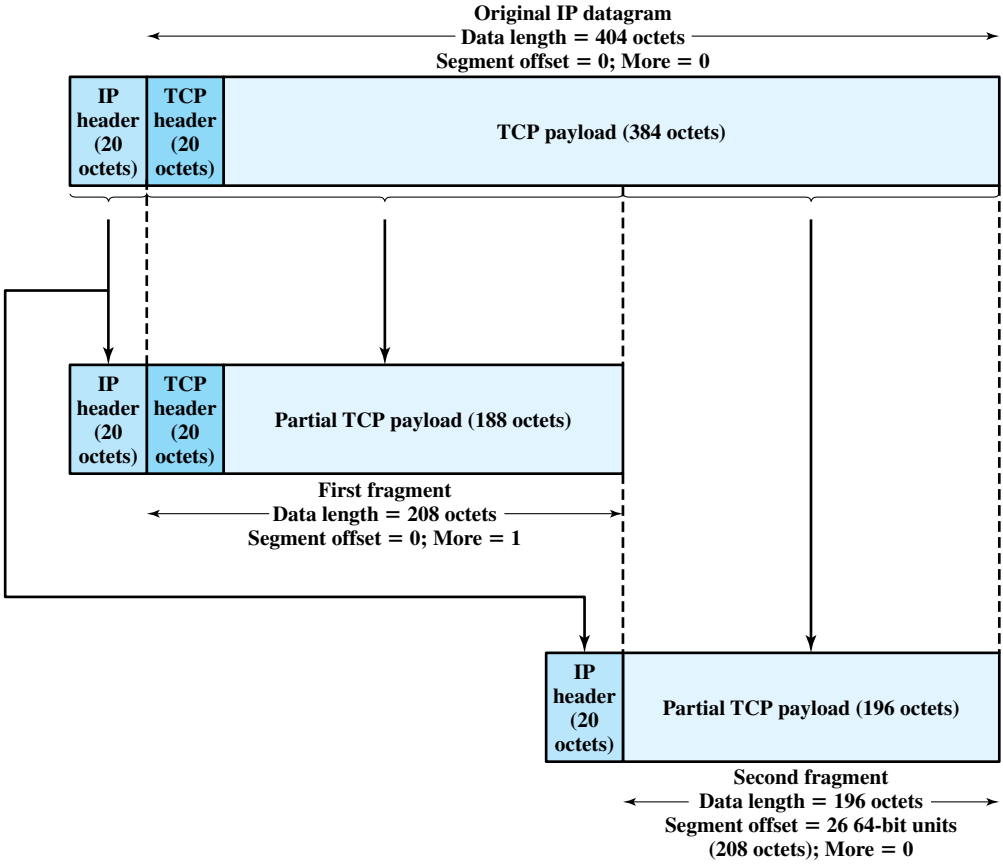


Figure 18.5 Fragmentation Example

TCP header and application data. The IP header from the original datagram is used in both fragments, with the appropriate changes to the fragmentation-related fields. Note that the first fragment contains the TCP header; this header is not replicated in the second fragment, because all of the IP payload, including the TCP header is transparent to IP. That is, IP is not concerned with the contents of the payload of the datagram.

To reassemble a datagram, there must be sufficient buffer space at the reassembly point. As fragments with the same ID arrive, their data fields are inserted in the proper position in the buffer until the entire data field is reassembled, which is achieved when a contiguous set of data exists starting with an *Offset* of zero and ending with data from a fragment with a false *More Flag*.

One eventuality that must be dealt with is that one or more of the fragments may not get through: The IP service does not guarantee delivery. Some method is needed to decide when to abandon a reassembly effort to free up buffer space. Two approaches are commonly used. First, assign a reassembly lifetime to the first fragment to arrive. This is a local, real-time clock assigned by the reassembly function and decremented while the fragments of the original datagram are being buffered.

If the time expires prior to complete reassembly, the received fragments are discarded. A second approach is to make use of the datagram lifetime, which is part of the header of each incoming fragment. The lifetime field continues to be decremented by the reassembly function; as with the first approach, if the lifetime expires prior to complete reassembly, the received fragments are discarded.

Error Control The internetwork facility does not guarantee successful delivery of every datagram. When a datagram is discarded by a router, the router should attempt to return some information to the source, if possible. The source Internet Protocol entity may use this information to modify its transmission strategy and may notify higher layers. To report that a specific datagram has been discarded, some means of datagram identification is needed. Such identification is discussed in the next section.

Datagrams may be discarded for a number of reasons, including lifetime expiration, congestion, and FCS error. In the latter case, notification is not possible because the source address field may have been damaged.

Flow Control Internet flow control allows routers and/or receiving stations to limit the rate at which they receive data. For the connectionless type of service we are describing, flow control mechanisms are limited. The best approach would seem to be to send flow control packets, requesting reduced data flow, to other routers and source stations. We will see one example of this with Internet Control Message Protocol (ICMP), discussed in the next section.

18.4 INTERNET PROTOCOL

In this section, we look at version 4 of IP, officially defined in RFC 791. Although it is intended that IPv4 will ultimately be replaced by IPv6, it is currently the standard IP used in TCP/IP networks.

The Internet Protocol (IP) is part of the TCP/IP suite and is the most widely used internetworking protocol. As with any protocol standard, IP is specified in two parts (see Figure 2.9):

- The interface with a higher layer (e.g., TCP), specifying the services that IP provides
- The actual protocol format and mechanisms

In this section, we examine first IP services and then the protocol. This is followed by a discussion of IP address formats. Finally, the Internet Control Message Protocol (ICMP), which is an integral part of IP, is described.

IP Services

The services to be provided across adjacent protocol layers (e.g., between IP and TCP) are expressed in terms of primitives and parameters. A primitive specifies the function to be performed, and the parameters are used to pass data and control information. The actual form of a primitive is implementation dependent. An example is a procedure call.

IP provides two service primitives at the interface to the next higher layer. The Send primitive is used to request transmission of a data unit. The Deliver primitive is used by IP to notify a user of the arrival of a data unit. The parameters associated with the two primitives are as follows:

- **Source address:** Internetwork address of sending IP entity.
- **Destination address:** Internetwork address of destination IP entity.
- **Protocol:** Recipient protocol entity (an IP user, such as TCP).
- **Type-of-service indicators:** Used to specify the treatment of the data unit in its transmission through component networks.
- **Identification:** Used in combination with the source and destination addresses and user protocol to identify the data unit uniquely. This parameter is needed for reassembly and error reporting.
- **Don't fragment identifier:** Indicates whether IP can fragment data to accomplish delivery.
- **Time to live:** Measured in seconds.
- **Data length:** Length of data being transmitted.
- **Option data:** Options requested by the IP user.
- **Data:** User data to be transmitted.

The *identification, don't fragment identifier*, and *time to live* parameters are present in the Send primitive but not in the Deliver primitive. These three parameters provide instructions to IP that are not of concern to the recipient IP user.

The options parameter allows for future extensibility and for inclusion of parameters that are usually not invoked. The currently defined options are as follows:

- **Security:** Allows a security label to be attached to a datagram.
- **Source routing:** A sequenced list of router addresses that specifies the route to be followed. Routing may be strict (only identified routers may be visited) or loose (other intermediate routers may be visited).
- **Route recording:** A field is allocated to record the sequence of routers visited by the datagram.
- **Stream identification:** Names reserved resources used for stream service. This service provides special handling for volatile periodic traffic (e.g., voice).
- **Timestamping:** The source IP entity and some or all intermediate routers add a timestamp (precision to milliseconds) to the data unit as it goes by.

Internet Protocol

The protocol between IP entities is best described with reference to the IP datagram format, shown in Figure 18.6. The fields are as follows:

- **Version (4 bits):** Indicates version number, to allow evolution of the protocol; the value is 4.

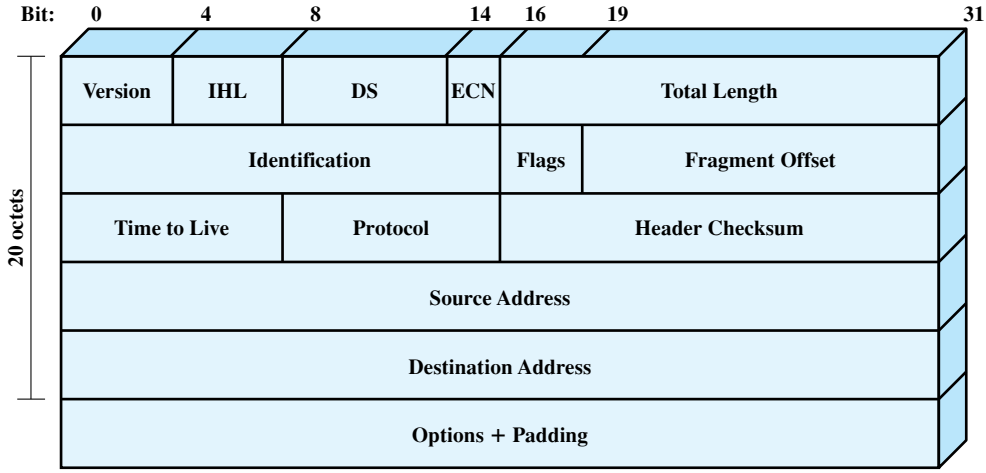


Figure 18.6 IPv4 Header

- **Internet Header Length (IHL) (4 bits):** Length of header in 32-bit words. The minimum value is five, for a minimum header length of 20 octets.
- **DS/ECN (8 bits):** Prior to the introduction of differentiated services, this field was referred to as the Type of Service field and specified reliability, precedence, delay, and throughput parameters. This interpretation has now been superseded. The first six bits of this field are now referred to as the DS (Differentiated Services) field, discussed in Chapter 19. The remaining 2 bits are reserved for an ECN (Explicit Congestion Notification) field, currently in the process of standardization. The ECN field provides for explicit signaling of congestion in a manner similar to that discussed for frame relay (Section 13.5).
- **Total Length (16 bits):** Total datagram length, including header plus data, in octets.
- **Identification (16 bits):** A sequence number that, together with the source address, destination address, and user protocol, is intended to identify a datagram uniquely. Thus, this number should be unique for the datagram's source address, destination address, and user protocol for the time during which the datagram will remain in the internet.
- **Flags (3 bits):** Only two of the bits are currently defined. The More bit is used for fragmentation and reassembly, as previously explained. The Don't Fragment bit prohibits fragmentation when set. This bit may be useful if it is known that the destination does not have the capability to reassemble fragments. However, if this bit is set, the datagram will be discarded if it exceeds the maximum size of an en route network. Therefore, if the bit is set, it may be advisable to use source routing to avoid networks with small maximum packet size.
- **Fragment Offset (13 bits):** Indicates where in the original datagram this fragment belongs, measured in 64-bit units. This implies that fragments other

than the last fragment must contain a data field that is a multiple of 64 bits in length.

- **Time to Live (8 bits):** Specifies how long, in seconds, a datagram is allowed to remain in the internet. Every router that processes a datagram must decrease the TTL by at least one, so the TTL is similar to a hop count.
- **Protocol (8 bits):** Indicates the next higher level protocol that is to receive the data field at the destination; thus, this field identifies the type of the next header in the packet after the IP header. Example values are TCP = 6; UDP = 17; ICMP = 1. A complete list is maintained at **http://www.iana.org/assignments/protocol-numbers**.
- **Header Checksum (16 bits):** An error-detecting code applied to the header only. Because some header fields may change during transit (e.g., Time to Live, fragmentation-related fields), this is reverified and recomputed at each router. The checksum is formed by taking the ones complement of the 16-bit ones complement addition of all 16-bit words in the header. For purposes of computation, the checksum field is itself initialized to a value of zero.[5]
- **Source Address (32 bits):** Coded to allow a variable allocation of bits to specify the network and the end system attached to the specified network, as discussed subsequently.
- **Destination Address (32 bits):** Same characteristics as source address.
- **Options (variable):** Encodes the options requested by the sending user.
- **Padding (variable):** Used to ensure that the datagram header is a multiple of 32 bits in length.
- **Data (variable):** The data field must be an integer multiple of 8 bits in length. The maximum length of the datagram (data field plus header) is 65,535 octets.

It should be clear how the IP services specified in the Send and Deliver primitives map into the fields of the IP datagram.

IP Addresses

The source and destination address fields in the IP header each contain a 32-bit global internet address, generally consisting of a network identifier and a host identifier.

Network Classes The address is coded to allow a variable allocation of bits to specify network and host, as depicted in Figure 18.7. This encoding provides flexibility in assigning addresses to hosts and allows a mix of network sizes on an internet. The three principal network classes are best suited to the following conditions:

- **Class A:** Few networks, each with many hosts
- **Class B:** Medium number of networks, each with a medium number of hosts
- **Class C:** Many networks, each with a few hosts

[5]A discussion of this checksum is contained in Appendix K.

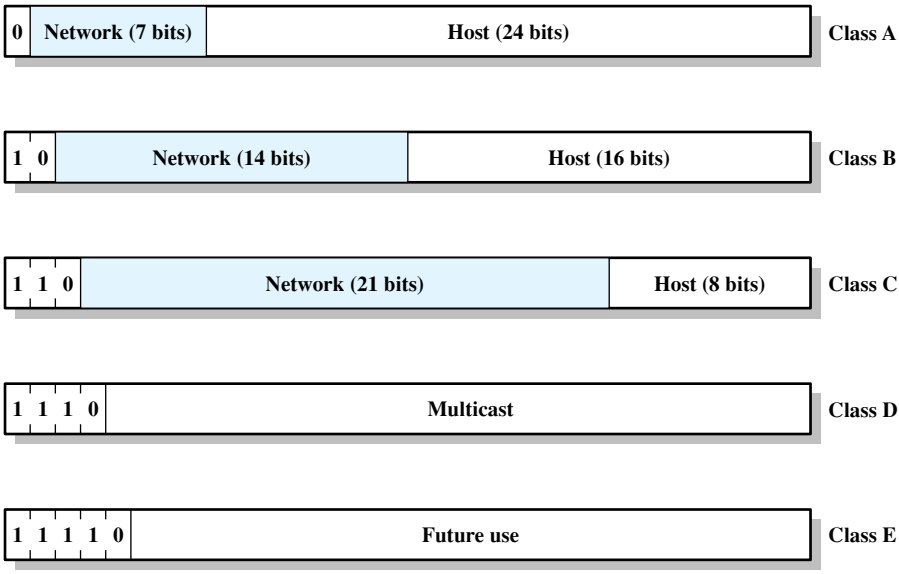


Figure 18.7 IPv4 Address Formats

In a particular environment, it may be best to use addresses all from one class. For example, a corporate internetwork that consist of a large number of departmental local area networks may need to use Class C addresses exclusively. However, the format of the addresses is such that it is possible to mix all three classes of addresses on the same internetwork; this is what is done in the case of the Internet itself. A mixture of classes is appropriate for an internetwork consisting of a few large networks, many small networks, plus some medium-sized networks.

IP addresses are usually written in **dotted decimal notation**, with a decimal number representing each of the octets of the 32-bit address. For example, the IP address 11000000 11100100 00010001 00111001 is written as 192.228.17.57.

Note that all Class A network addresses begin with a binary 0. Network addresses with a first octet of 0 (binary 00000000) and 127 (binary 01111111) are reserved, so there are 126 potential Class A network numbers, which have a first dotted decimal number in the range 1 to 126. Class B network addresses begin with a binary 10, so that the range of the first decimal number in a Class B address is 128 to 191(binary 10000000 to 10111111). The second octet is also part of the Class B address, so that there are $2^{14} = 16{,}384$ Class B addresses. For Class C addresses, the first decimal number ranges from 192 to 223 (11000000 to 11011111). The total number of Class C addresses is $2^{21} = 2{,}097{,}152$.

Subnets and Subnet Masks The concept of subnet was introduced to address the following requirement. Consider an internet that includes one or more WANs and a number of sites, each of which has a number of LANs. We would like to allow arbitrary complexity of interconnected LAN structures within an organization while insulating the overall internet against explosive growth in network numbers

and routing complexity. One approach to this problem is to assign a single network number to all of the LANs at a site. From the point of view of the rest of the internet, there is a single network at that site, which simplifies addressing and routing. To allow the routers within the site to function properly, each LAN is assigned a subnet number. The *host* portion of the internet address is partitioned into a subnet number and a host number to accommodate this new level of addressing.

Within the subnetted network, the local routers must route on the basis of an extended network number consisting of the *network* portion of the IP address and the subnet number. The bit positions containing this extended network number are indicated by the address mask. The use of the address mask allows the host to determine whether an outgoing datagram is destined for a host on the same LAN (send directly) or another LAN (send datagram to router). It is assumed that some other means (e.g., manual configuration) are used to create address masks and make them known to the local routers.

Table 18.3a shows the calculations involved in the use of a subnet mask. Note that the effect of the subnet mask is to erase the portion of the host field that refers to an actual host on a subnet. What remains is the network number and the subnet number. Figure 18.8 shows an example of the use of subnetting. The figure shows a local complex consisting of three LANs and two routers. To the rest of the internet, this complex is a single network with a Class C address of the form 192.228.17.*x*, where the leftmost three octets are the network number and the rightmost octet contains a host number *x*. Both routers R1 and R2 are configured with a subnet

Table 18.3 IP Addresses and Subnet Masks [STEI95]

(a) Dotted decimal and binary representations of IP address and subnet masks

	Binary Representation	Dotted Decimal
IP address	11000000.11100100.00010001.00111001	192.228.17.57
Subnet mask	11111111.11111111.11111111.11100000	255.255.255.224
Bitwise AND of address and mask (resultant network/subnet number)	11000000.11100100.00010001.00100000	192.228.17.32
Subnet number	11000000.11100100.00010001.001	1
Host number	00000000.00000000.00000000.00011001	25

(b) Default subnet masks

	Binary Representation	Dotted Decimal
Class A default mask	11111111.00000000.00000000.00000000	255.0.0.0
Example Class A mask	11111111.11000000.00000000.00000000	255.192.0.0
Class B default mask	11111111.11111111.00000000.00000000	255.255.0.0
Example Class B mask	11111111.11111111.11111000.00000000	255.255.248.0
Class C default mask	11111111.11111111.11111111.00000000	255. 255. 255.0
Example Class C mask	11111111.11111111.11111111.11111100	255. 255. 255.252

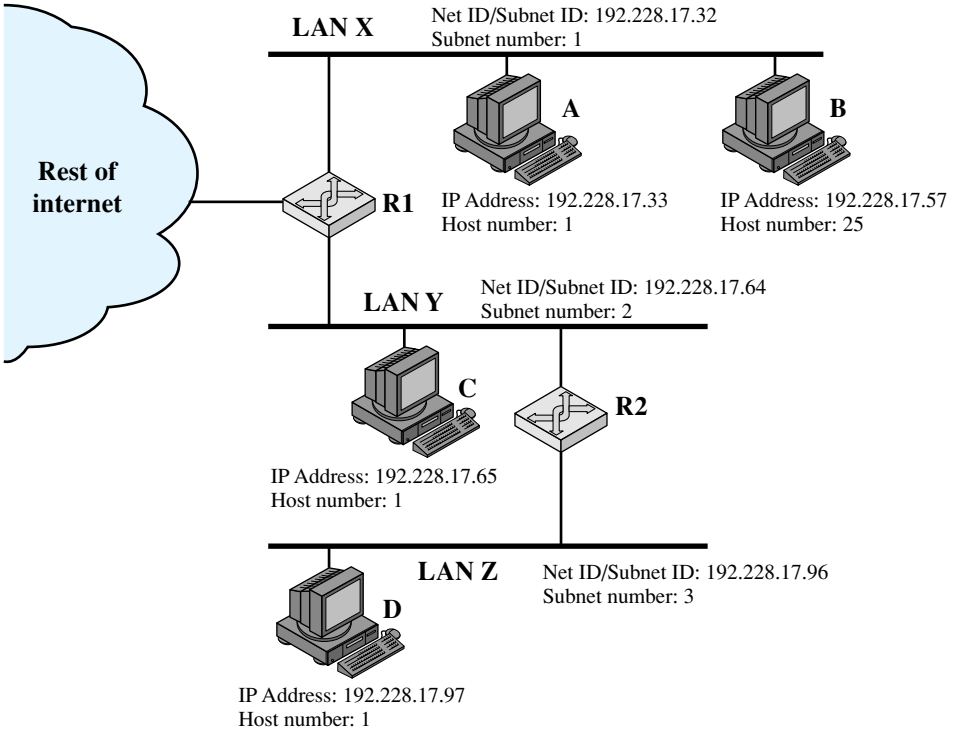


Figure 18.8 Example of Subnetworking

mask with the value 255.255.255.224 (see Table 18.3a). For example, if a datagram with the destination address 192.228.17.57 arrives at R1 from either the rest of the internet or from LAN Y, R1 applies the subnet mask to determine that this address refers to subnet 1, which is LAN X, and so forwards the datagram to LAN X. Similarly, if a datagram with that destination address arrives at R2 from LAN Z, R2 applies the mask and then determines from its forwarding database that datagrams destined for subnet 1 should be forwarded to R1. Hosts must also employ a subnet mask to make routing decisions.

The default subnet mask for a given class of addresses is a null mask (Table 18.3b), which yields the same network and host number as the non-subnetted address.

Internet Control Message Protocol (ICMP)

The IP standard specifies that a compliant implementation must also implement ICMP (RFC 792). ICMP provides a means for transferring messages from routers and other hosts to a host. In essence, ICMP provides feedback about problems in the communication environment. Examples of its use are when a datagram cannot reach its destination, when the router does not have the buffering capacity to forward a datagram, and when the router can direct the station to send traffic on a shorter route. In most cases, an ICMP message is sent in response to a datagram, either by a router along the datagram's path or by the intended destination host.

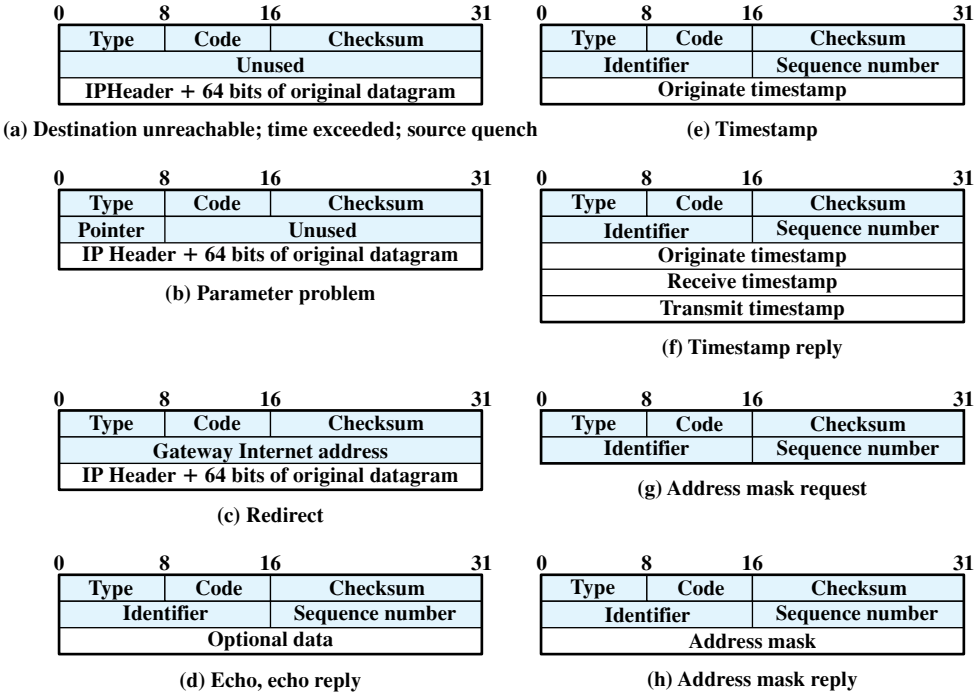


Figure 18.9 ICMP Message Formats

Although ICMP is, in effect, at the same level as IP in the TCP/IP architecture, it is a user of IP. An ICMP message is constructed and then passed down to IP, which encapsulates the message with an IP header and then transmits the resulting datagram in the usual fashion. Because ICMP messages are transmitted in IP datagrams, their delivery is not guaranteed and their use cannot be considered reliable.

Figure 18.9 shows the format of the various ICMP message types. An ICMP message starts with a 64-bit header consisting of the following:

- **Type (8 bits):** Specifies the type of ICMP message.
- **Code (8 bits):** Used to specify parameters of the message that can be encoded in one or a few bits.
- **Checksum (16 bits):** Checksum of the entire ICMP message. This is the same checksum algorithm used for IP.
- **Parameters (32 bits):** Used to specify more lengthy parameters.

These fields are generally followed by additional information fields that further specify the content of the message.

In those cases in which the ICMP message refers to a prior datagram, the information field includes the entire IP header plus the first 64 bits of the data field of the original datagram. This enables the source host to match the incoming ICMP message with the prior datagram. The reason for including the first 64 bits of the data field is that this will enable the IP module in the host to determine which

upper-level protocol or protocols were involved. In particular, the first 64 bits would include a portion of the TCP header or other transport-level header.

The **destination unreachable** message covers a number of contingencies. A router may return this message if it does not know how to reach the destination network. In some networks, an attached router may be able to determine if a particular host is unreachable and returns the message. The destination host itself may return this message if the user protocol or some higher-level service access point is unreachable. This could happen if the corresponding field in the IP header was set incorrectly. If the datagram specifies a source route that is unusable, a message is returned. Finally, if a router must fragment a datagram but the Don't Fragment flag is set, the datagram is discarded and a message is returned.

A router will return a **time exceeded** message if the lifetime of the datagram expires. A host will send this message if it cannot complete reassembly within a time limit.

A syntactic or semantic error in an IP header will cause a **parameter problem** message to be returned by a router or host. For example, an incorrect argument may be provided with an option. The Parameter field contains a pointer to the octet in the original header where the error was detected.

The **source quench** message provides a rudimentary form of flow control. Either a router or a destination host may send this message to a source host, requesting that it reduce the rate at which it is sending traffic to the internet destination. On receipt of a source quench message, the source host should cut back the rate at which it is sending traffic to the specified destination until it no longer receives source quench messages. The source quench message can be used by a router or host that must discard datagrams because of a full buffer. In that case, the router or host will issue a source quench message for every datagram that it discards. In addition, a system may anticipate congestion and issue source quench messages when its buffers approach capacity. In that case, the datagram referred to in the source quench message may well be delivered. Thus, receipt of a source quench message does not imply delivery or nondelivery of the corresponding datagram.

A router sends a **redirect** message to a host on a directly connected router to advise the host of a better route to a particular destination. The following is an example, using Figure 18.8. Router R1 receives a datagram from host C on network Y, to which R1 is attached. R1 checks its routing table and obtains the address for the next router, R2, on the route to the datagram's internet destination network, Z. Because R2 and the host identified by the internet source address of the datagram are on the same network, R1 sends a redirect message to C. The redirect message advises the host to send its traffic for network Z directly to router R2, because this is a shorter path to the destination. The router forwards the original datagram to its internet destination (via R2). The address of R2 is contained in the parameter field of the redirect message.

The **echo** and **echo reply** messages provide a mechanism for testing that communication is possible between entities. The recipient of an echo message is obligated to return the message in an echo reply message. An identifier and sequence number are associated with the echo message to be matched in the echo reply message. The identifier might be used like a service access point to

identify a particular session, and the sequence number might be incremented on each echo request sent.

The **timestamp** and **timestamp reply** messages provide a mechanism for sampling the delay characteristics of the internet. The sender of a timestamp message may include an identifier and sequence number in the parameters field and include the time that the message is sent (originate timestamp). The receiver records the time it received the message and the time that it transmits the reply message in the timestamp reply message. If the timestamp message is sent using strict source routing, then the delay characteristics of a particular route can be measured.

The **address mask request** and **address mask reply** messages are useful in an environment that includes subnets. The address mask request and reply messages allow a host to learn the address mask for the LAN to which it connects. The host broadcasts an address mask request message on the LAN. The router on the LAN responds with an address mask reply message that contains the address mask.

Address Resolution Protocol (ARP)

Earlier in this chapter, we referred to the concepts of a global address (IP address) and an address that conforms to the addressing scheme of the network to which a host is attached (subnetwork address). For a local area network, the latter address is a MAC address, which provides a physical address for a host port attached to the LAN. Clearly, to deliver an IP datagram to a destination host, a mapping must be made from the IP address to the subnetwork address for that last hop. If a datagram traverses one or more routers between source and destination hosts, then the mapping must be done in the final router, which is attached to the same subnetwork as the destination host. If a datagram is sent from one host to another on the same subnetwork, then the source host must do the mapping. In the following discussion, we use the term *system* to refer to the entity that does the mapping.

For mapping from an IP address to a subnetwork address, a number of approaches are possible, including

- Each system can maintain a local table of IP addresses and matching subnetwork addresses for possible correspondents. This approach does not accommodate easy and automatic additions of new hosts to the subnetwork.
- The subnetwork address can be a subset of the network portion of the IP address. However, the entire internet address is 32 bits long and for most subnetwork types (e.g., Ethernet) the Host Address field is longer than 32 bits.
- A centralized directory can be maintained on each subnetwork that contains the IP-subnet address mappings. This is a reasonable solution for many networks.
- An address resolution protocol can be used. This is a simpler approach than the use of a centralized directory and is well suited to LANs.

RFC 826 defines an Address Resolution Protocol (ARP), which allows dynamic distribution of the information needed to build tables to translate an IP address A into a 48-bit Ethernet address; the protocol can be used for any broadcast

network. ARP exploits the broadcast property of a LAN; namely, that a transmission from any device on the network is received by all other devices on the network. ARP works as follows:

1. Each system on the LAN maintains a table of known IP-subnetwork address mappings.
2. When a subnetwork address is needed for an IP address, and the mapping is not found in the system's table, the system uses ARP directly on top of the LAN protocol (e.g., IEEE 802) to broadcast a request. The broadcast message contains the IP address for which a subnetwork address is needed.
3. Other hosts on the subnetwork listen for ARP messages and reply when a match occurs. The reply includes both the IP and subnetwork addresses of the replying host.
4. The original request includes the requesting host's IP address and subnetwork address. Any interested host can copy this information into its local table, avoiding the need for later ARP messages.
5. The ARP message can also be used simply to broadcast a host's IP address and subnetwork address, for the benefit of others on the subnetwork.

18.5 IPv6

The Internet Protocol (IP) has been the foundation of the Internet and virtually all multivendor private internetworks. This protocol is reaching the end of its useful life and a new protocol, known as IPv6 (IP version 6), has been defined to ultimately replace IP.[6]

We first look at the motivation for developing a new version of IP and then examine some of its details.

IP Next Generation

The driving motivation for the adoption of a new version of IP was the limitation imposed by the 32-bit address field in IPv4. With a 32-bit address field, it is possible in principle to assign 2^{32} different addresses, which is over 4 billion possible addresses. One might think that this number of addresses was more than adequate to meet addressing needs on the Internet. However, in the late 1980s it was perceived that there would be a problem, and this problem began to manifest itself in the early 1990s. Reasons for the inadequacy of 32-bit addresses include the following:

- The two-level structure of the IP address (network number, host number) is convenient but wasteful of the address space. Once a network number is assigned to a network, all of the host-number addresses for that network number are assigned to that network. The address space for that network may

[6]The currently deployed version of IP is IP version 4; previous versions of IP (1 through 3) were successively defined and replaced to reach IPv4. Version 5 is the number assigned to the Stream Protocol, a connection-oriented internet-layer protocol; hence the use of the label version 6.

be sparsely used, but as far as the effective IP address space is concerned, if a network number is used, then all addresses within the network are used.

- The IP addressing model generally requires that a unique network number be assigned to each IP network whether or not it is actually connected to the Internet.
- Networks are proliferating rapidly. Most organizations boast multiple LANs, not just a single LAN system. Wireless networks have rapidly assumed a major role. The Internet itself has grown explosively for years.
- Growth of TCP/IP usage into new areas will result in a rapid growth in the demand for unique IP addresses. Examples include using TCP/IP to interconnect electronic point-of-sale terminals and for cable television receivers.
- Typically, a single IP address is assigned to each host. A more flexible arrangement is to allow multiple IP addresses per host. This, of course, increases the demand for IP addresses.

So the need for an increased address space dictated that a new version of IP was needed. In addition, IP is a very old protocol, and new requirements in the areas of address configuration, routing flexibility, and traffic support had been defined.

In response to these needs, the Internet Engineering Task Force (IETF) issued a call for proposals for a next generation IP (IPng) in July of 1992. A number of proposals were received, and by 1994 the final design for IPng emerged. A major milestone was reached with the publication of RFC 1752, "The Recommendation for the IP Next Generation Protocol," issued in January 1995. RFC 1752 outlines the requirements for IPng, specifies the PDU formats, and highlights the IPng approach in the areas of addressing, routing, and security. A number of other Internet documents defined details of the protocol, now officially called IPv6; these include an overall specification of IPv6 (RFC 2460), an RFC dealing with addressing structure of IPv6 (RFC 2373), and numerous others.

IPv6 includes the following enhancements over IPv4:

- **Expanded address space:** IPv6 uses 128-bit addresses instead of the 32-bit addresses of IPv4. This is an increase of address space by a factor of 2^{96}. It has been pointed out [HIND95] that this allows on the order of 6×10^{23} unique addresses per square meter of the surface of the earth. Even if addresses are very inefficiently allocated, this address space seems inexhaustible.
- **Improved option mechanism:** IPv6 options are placed in separate optional headers that are located between the IPv6 header and the transport-layer header. Most of these optional headers are not examined or processed by any router on the packet's path. This simplifies and speeds up router processing of IPv6 packets compared to IPv4 datagrams.[7] It also makes it easier to add additional options.

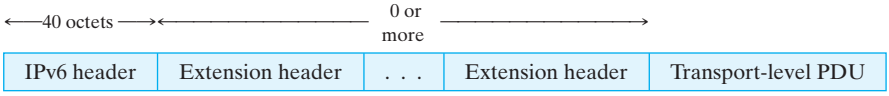
[7]The protocol data unit for IPv6 is referred to as a packet rather than a datagram, which is the term used for IPv4 PDUs.

- **Address autoconfiguration:** This capability provides for dynamic assignment of IPv6 addresses.
- **Increased addressing flexibility:** IPv6 includes the concept of an anycast address, for which a packet is delivered to just one of a set of nodes. The scalability of multicast routing is improved by adding a scope field to multicast addresses.
- **Support for resource allocation:** IPv6 enables the labeling of packets belonging to a particular traffic flow for which the sender requests special handling. This aids in the support of specialized traffic such as real-time video.

All of these features are explored in the remainder of this section.

IPv6 Structure

An IPv6 protocol data unit (known as a packet) has the following general form:



The only header that is required is referred to simply as the IPv6 header. This is of fixed size with a length of 40 octets, compared to 20 octets for the mandatory portion of the IPv4 header (Figure 18.6). The following extension headers have been defined:

- **Hop-by-Hop Options header:** Defines special options that require hop-by-hop processing
- **Routing header:** Provides extended routing, similar to IPv4 source routing
- **Fragment header:** Contains fragmentation and reassembly information
- **Authentication header:** Provides packet integrity and authentication
- **Encapsulating Security Payload header:** Provides privacy
- **Destination Options header:** Contains optional information to be examined by the destination node

The IPv6 standard recommends that, when multiple extension headers are used, the IPv6 headers appear in the following order:

1. IPv6 header: Mandatory, must always appear first
2. Hop-by-Hop Options header
3. Destination Options header: For options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header
4. Routing header
5. Fragment header
6. Authentication header
7. Encapsulating Security Payload header

8. Destination Options header: For options to be processed only by the final destination of the packet

Figure 18.10 shows an example of an IPv6 packet that includes an instance of each header, except those related to security. Note that the IPv6 header and each extension header include a Next Header field. This field identifies the type of the immediately following header. If the next header is an extension header, then this field contains the type identifier of that header. Otherwise, this field contains the protocol identifier of the upper-layer protocol using IPv6 (typically a transport-level protocol), using the same values as the IPv4 Protocol field. In Figure 18.10, the upper-layer protocol is TCP; thus, the upper-layer data carried by the IPv6 packet consist of a TCP header followed by a block of application data.

We first look at the main IPv6 header and then examine each of the extensions in turn.

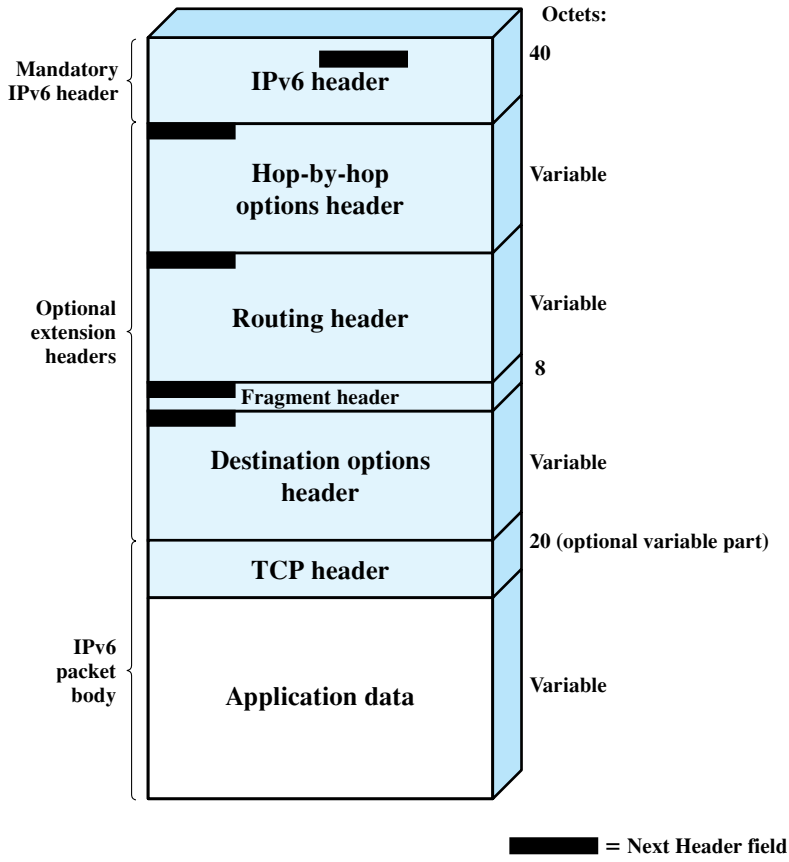


Figure 18.10 IPv6 Packet with Extension Headers (containing a TCP Segment)

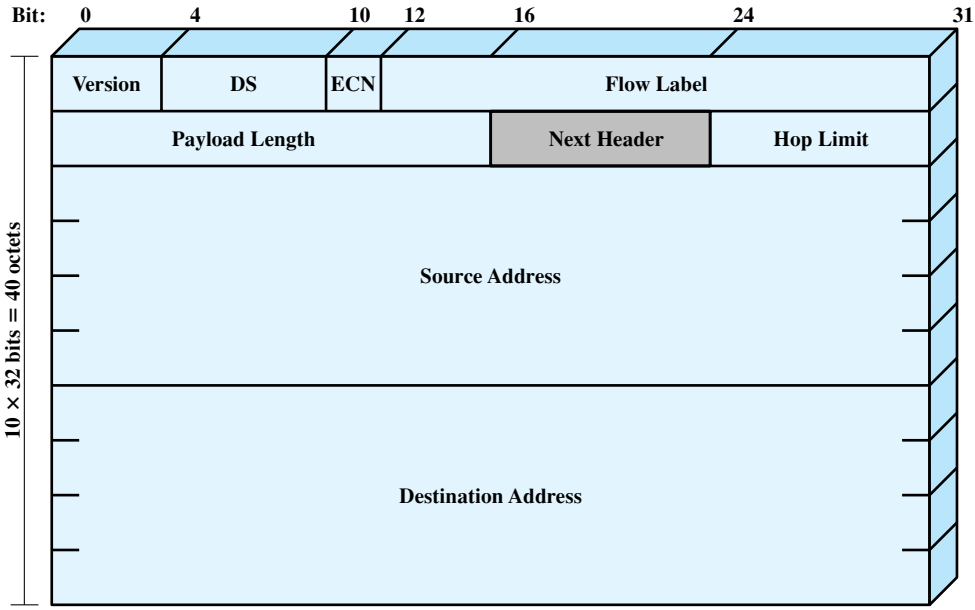


Figure 18.11 IPv6 Header

IPv6 Header

The IPv6 header has a fixed length of 40 octets, consisting of the following fields (Figure 18.11):

- **Version (4 bits):** Internet protocol version number; the value is 6.
- **DS/ECN (8 bits):** Available for use by originating nodes and/or forwarding routers for differentiated services and congestion functions, as described for the IPv4 DS/ECN field.
- **Flow Label (20 bits):** May be used by a host to label those packets for which it is requesting special handling by routers within a network; discussed subsequently.
- **Payload Length (16 bits):** Length of the remainder of the IPv6 packet following the header, in octets. In other words, this is the total length of all of the extension headers plus the transport-level PDU.
- **Next Header (8 bits):** Identifies the type of header immediately following the IPv6 header; this will either be an IPv6 extension header or a higher-layer header, such as TCP or UDP.
- **Hop Limit (8 bits):** The remaining number of allowable hops for this packet. The hop limit is set to some desired maximum value by the source and decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero. This is a simplification over the processing required for the Time to Live field of IPv4. The consensus was that the extra effort in accounting for time intervals in IPv4 added no significant value to the protocol. In fact, IPv4 routers, as a general rule, treat the Time to Live field as a hop limit field.
- **Source Address (128 bits):** The address of the originator of the packet.

- **Destination Address (128 bits):** The address of the intended recipient of the packet. This may not in fact be the intended ultimate destination if a Routing header is present, as explained subsequently.

Although the IPv6 header is longer than the mandatory portion of the IPv4 header (40 octets versus 20 octets), it contains fewer fields (8 versus 12). Thus, routers have less processing to do per header, which should speed up routing.

Flow Label RFC 3967 defines a flow as a sequence of packets sent from a particular source to a particular (unicast, anycast, or multicast) destination for which the source desires special handling by the intervening routers. A flow is uniquely identified by the combination of a source address, destination address, and a nonzero 20-bit flow label. Thus, all packets that are to be part of the same flow are assigned the same flow label by the source.

From the source's point of view, a flow typically will be a sequence of packets that are generated from a single application instance at the source and that have the same transfer service requirements. A flow may comprise a single TCP connection or even multiple TCP connections; an example of the latter is a file transfer application, which could have one control connection and multiple data connections. A single application may generate a single flow or multiple flows. An example of the latter is multimedia conferencing, which might have one flow for audio and one for graphic windows, each with different transfer requirements in terms of data rate, delay, and delay variation.

From the router's point of view, a flow is a sequence of packets that share attributes that affect how these packets are handled by the router. These include path, resource allocation, discard requirements, accounting, and security attributes. The router may treat packets from different flows differently in a number of ways, including allocating different buffer sizes, giving different precedence in terms of forwarding, and requesting different quality of service from networks.

There is no special significance to any particular flow label. Instead the special handling to be provided for a packet flow must be declared in some other way. For example, a source might negotiate or request special handling ahead of time from routers by means of a control protocol, or at transmission time by information in one of the extension headers in the packet, such as the Hop-by-Hop Options header. Examples of special handling that might be requested include some sort of nondefault quality of service and some form of real-time service.

In principle, all of a user's requirements for a particular flow could be defined in an extension header and included with each packet. If we wish to leave the concept of flow open to include a wide variety of requirements, this design approach could result in very large packet headers. The alternative, adopted for IPv6, is the flow label, in which the flow requirements are defined prior to flow commencement and a unique flow label is assigned to the flow. In this case, the router must save flow requirement information about each flow.

The following rules apply to the flow label:

1. Hosts or routers that do not support the Flow Label field must set the field to zero when originating a packet, pass the field unchanged when forwarding a packet, and ignore the field when receiving a packet.

2. All packets originating from a given source with the same nonzero Flow Label must have the same Destination Address, Source Address, Hop-by-Hop Options header contents (if this header is present), and Routing header contents (if this header is present). The intent is that a router can decide how to route and process the packet by simply looking up the flow label in a table and without examining the rest of the header.
3. The source assigns a flow label to a flow. New flow labels must be chosen (pseudo-) randomly and uniformly in the range 1 to $2^{20} - 1$, subject to the restriction that a source must not reuse a flow label for a new flow within the lifetime of the existing flow. The zero flow label is reserved to indicate that no flow label is being used.

This last point requires some elaboration. The router must maintain information about the characteristics of each active flow that may pass through it, presumably in some sort of table. To forward packets efficiently and rapidly, table lookup must be efficient. One alternative is to have a table with 2^{20} (about 1 million) entries, one for each possible flow label; this imposes an unnecessary memory burden on the router. Another alternative is to have one entry in the table per active flow, include the flow label with each entry, and require the router to search the entire table each time a packet is encountered. This imposes an unnecessary processing burden on the router. Instead, most router designs are likely to use some sort of hash table approach. With this approach a moderate-sized table is used, and each flow entry is mapped into the table using a hashing function on the flow label. The hashing function might simply be the low-order few bits (say 8 or 10) of the flow label or some simple calculation on the 20 bits of the flow label. In any case, the efficiency of the hash approach typically depends on the flow labels being uniformly distributed over their possible range. Hence requirement number 3 in the preceding list.

IPv6 Addresses

IPv6 addresses are 128 bits in length. Addresses are assigned to individual interfaces on nodes, not to the nodes themselves.[8] A single interface may have multiple unique unicast addresses. Any of the unicast addresses associated with a node's interface may be used to uniquely identify that node.

The combination of long addresses and multiple addresses per interface enables improved routing efficiency over IPv4. In IPv4, addresses generally do not have a structure that assists routing, and therefore a router may need to maintain huge table of routing paths. Longer internet addresses allow for aggregating addresses by hierarchies of network, access provider, geography, corporation, and so on. Such aggregation should make for smaller routing tables and faster table lookups. The allowance for multiple addresses per interface would allow a subscriber that uses multiple access providers across the same interface to have separate addresses aggregated under each provider's address space.

IPv6 allows three types of addresses:

- **Unicast:** An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.

[8] In IPv6, a *node* is any device that implements IPv6; this includes hosts and routers.

- **Anycast:** An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance).
- **Multicast:** An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

Hop-by-Hop Options Header

The Hop-by-Hop Options header carries optional information that, if present, must be examined by every router along the path. This header consists of (Figure 18.12a):

- **Next Header (8 bits):** Identifies the type of header immediately following this header.
- **Header Extension Length (8 bits):** Length of this header in 64-bit units, not including the first 64 bits.
- **Options:** A variable-length field consisting of one or more option definitions. Each definition is in the form of three subfields: Option Type (8 bits), which identifies the option; Length (8 bits), which specifies the length of the Option Data field in octets; and Option Data, which is a variable-length specification of the option.

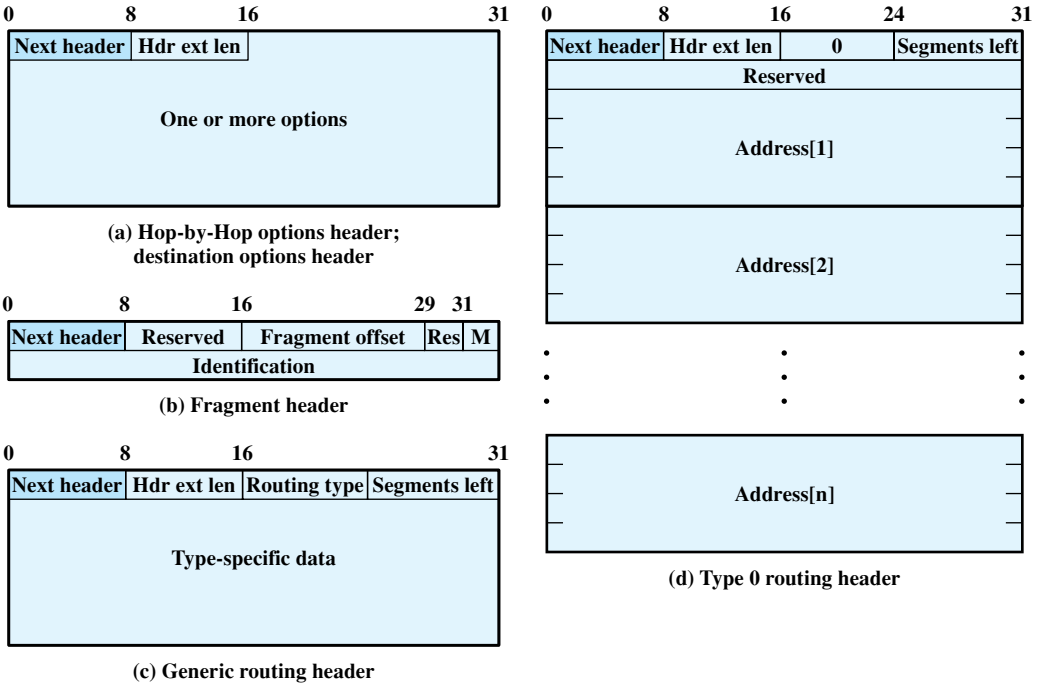


Figure 18.12 IPv6 Extension Headers

It is actually the lowest-order five bits of the Option Type field that are used to specify a particular option. The high-order two bits indicate that action to be taken by a node that does not recognize this option type, as follows:

- 00—Skip over this option and continue processing the header.
- 01—Discard the packet.
- 10—Discard the packet and send an ICMP Parameter Problem message to the packet's Source Address, pointing to the unrecognized Option Type.
- 11—Discard the packet and, only if the packet's Destination Address is not a multicast address, send an ICMP Parameter Problem message to the packet's Source Address, pointing to the unrecognized Option Type.

The third highest-order bit specifies whether the Option Data field does not change (0) or may change (1) en route from source to destination. Data that may change must be excluded from authentication calculations, as discussed in Chapter 21.

These conventions for the Option Type field also apply to the Destination Options header.

Four hop-by-hop options have been specified so far:

- **Pad1:** Used to insert one byte of padding into the Options area of the header.
- **PadN:** Used to insert N bytes ($N \geq 2$) of padding into the Options area of the header. The two padding options ensure that the header is a multiple of 8 bytes in length.
- **Jumbo payload:** Used to send IPv6 packets with payloads longer than 65,535 octets. The Option Data field of this option is 32 bits long and gives the length of the packet in octets, excluding the IPv6 header. For such packets, the Payload Length field in the IPv6 header must be set to zero, and there must be no Fragment header. With this option, IPv6 supports packet sizes up to more than 4 billion octets. This facilitates the transmission of large video packets and enables IPv6 to make the best use of available capacity over any transmission medium.
- **Router alert:** Informs the router that the contents of this packet is of interest to the router and to handle any control data accordingly. The absence of this option in an IPv6 datagram informs the router that the packet does not contain information needed by the router and hence can be safely routed without further packet parsing. Hosts originating IPv6 packets are required to include this option in certain circumstances. The purpose of this option is to provide efficient support for protocols such as RSVP (Chapter 19) that generate packets that need to be examined by intermediate routers for purposes of traffic control. Rather than requiring the intermediate routers to look in detail at the extension headers of a packet, this option alerts the router when such attention is required.

Fragment Header

In IPv6, fragmentation may only be performed by source nodes, not by routers along a packet's delivery path. To take full advantage of the internetworking environment, a node must perform a path discovery algorithm that enables it to learn the smallest maximum transmission unit (MTU) supported by any network on the path. With this knowledge, the source node will fragment, as required, for each given

destination address. Otherwise the source must limit all packets to 1280 octets, which is the minimum MTU that must be supported by each network.

The fragment header consists of the following (Figure 18.12b):

- **Next Header (8 bits):** Identifies the type of header immediately following this header.
- **Reserved (8 bits):** For future use.
- **Fragment Offset (13 bits):** Indicates where in the original packet the payload of this fragment belongs, measured in 64-bit units. This implies that fragments (other than the last fragment) must contain a data field that is a multiple of 64 bits long.
- **Res (2 bits):** Reserved for future use.
- **M Flag (1 bit):** 1 = more fragments; 0 = last fragment.
- **Identification (32 bits):** Intended to uniquely identify the original packet. The identifier must be unique for the packet's source address and destination address for the time during which the packet will remain in the internet. All fragments with the same identifier, source address, and destination address are reassembled to form the original packet.

  The fragmentation algorithm is the same as that described in Section 18.3.

Routing Header

The Routing header contains a list of one or more intermediate nodes to be visited on the way to a packet's destination. All routing headers start with a 32-bit block consisting of four 8-bit fields, followed by routing data specific to a given routing type (Figure 18.12c). The four 8-bit fields are as follows:

- **Next Header:** Identifies the type of header immediately following this header.
- **Header Extension Length:** Length of this header in 64-bit units, not including the first 64 bits.
- **Routing Type:** Identifies a particular Routing header variant. If a router does not recognize the Routing Type value, it must discard the packet.
- **Segments Left:** Number of route segments remaining; that is, the number of explicitly listed intermediate nodes still to be visited before reaching the final destination.

The only specific routing header format defined in RFC 2460 is the Type 0 Routing header (Figure 18.12d). When using the Type 0 Routing header, the source node does not place the ultimate destination address in the IPv6 header. Instead, that address is the last address listed in the Routing header (Address[n] in Figure 18.12d), and the IPv6 header contains the destination address of the first desired router on the path. The Routing header will not be examined until the packet reaches the node identified in the IPv6 header. At that point, the IPv6 and Routing header contents are updated and the packet is forwarded. The update consists of placing the next address to be visited in the IPv6 header and decrementing the Segments Left field in the Routing header.

Destination Options Header

The Destination Options header carries optional information that, if present, is examined only by the packet's destination node. The format of this header is the same as that of the Hop-by-Hop Options header (Figure 18.12a).

18.6 VIRTUAL PRIVATE NETWORKS AND IP SECURITY

In today's distributed computing environment, the **virtual private network** (VPN) offers an attractive solution to network managers. In essence, a VPN consists of a set of computers that interconnect by means of a relatively unsecure network and that make use of encryption and special protocols to provide security. At each corporate site, workstations, servers, and databases are linked by one or more local area networks (LANs). The LANs are under the control of the network manager and can be configured and tuned for cost-effective performance. The Internet or some other public network can be used to interconnect sites, providing a cost savings over the use of a private network and offloading the wide area network management task to the public network provider. That same public network provides an access path for telecommuters and other mobile employees to log on to corporate systems from remote sites.

But the manager faces a fundamental requirement: security. Use of a public network exposes corporate traffic to eavesdropping and provides an entry point for unauthorized users. To counter this problem, the manager may choose from a variety of encryption and authentication packages and products. Proprietary solutions raise a number of problems. First, how secure is the solution? If proprietary encryption or authentication schemes are used, there may be little reassurance in the technical literature as to the level of security provided. Second is the question of compatibility. No manager wants to be limited in the choice of workstations, servers, routers, firewalls, and so on by a need for compatibility with the security facility. This is the motivation for the IP Security (IPSec) set of Internet standards.

IPSec

In 1994, the Internet Architecture Board (IAB) issued a report titled "Security in the Internet Architecture" (RFC 1636). The report stated the general consensus that the Internet needs more and better security and identified key areas for security mechanisms. Among these were the need to secure the network infrastructure from unauthorized monitoring and control of network traffic and the need to secure end-user-to-end-user traffic using authentication and encryption mechanisms.

To provide security, the IAB included authentication and encryption as necessary security features in the next-generation IP, which has been issued as IPv6. Fortunately, these security capabilities were designed to be usable both with the current IPv4 and the future IPv6. This means that vendors can begin offering these features now, and many vendors do now have some IPSec capability in their products. The IPSec specification now exists as a set of Internet standards.

Applications of IPSec

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include the following:

- **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.

- **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet service provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.
- **Establishing extranet and intranet connectivity with partners:** IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security. IPSec guarantees that all traffic designated by the network administrator is both encrypted and authenticated, adding an additional layer of security to whatever is provided at the application layer.

The principal feature of IPSec that enables it to support these varied applications is that it can encrypt and/or authenticate *all* traffic at the IP level. Thus, all distributed applications, including remote logon, client/server, e-mail, file transfer, Web access, and so on, can be secured.

Figure 18.13 is a typical scenario of IPSec usage. An organization maintains LANs at dispersed locations. Nonsecure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPSec protocols are used. These protocols operate in networking devices, such as a router or firewall, that connect each LAN to the outside world. The IPSec networking device will typically encrypt and compress all traffic going into the WAN, and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPSec protocols to provide security.

Benefits of IPSec

Some of the benefits of IPSec are as follows:

- When IPSec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.
- IPSec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPSec is implemented in the firewall or router. Even if IPSec is implemented in end systems, upper-layer software, including applications, is not affected.
- IPSec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.
- IPSec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual subnetwork within an organization for sensitive applications.

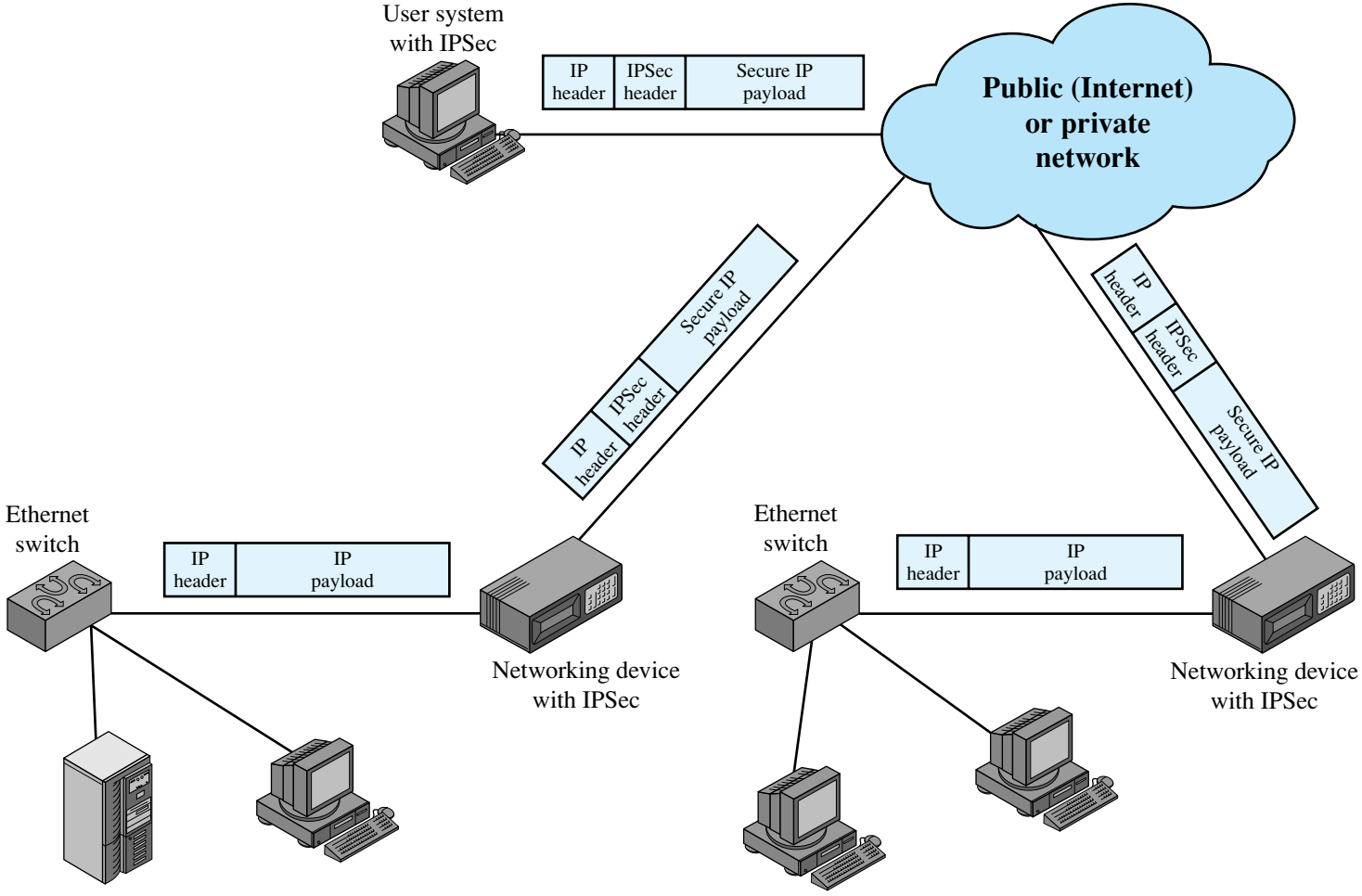


Figure 18.13 An IP Security Scenario

IPSec Functions

IPSec provides three main facilities: an authentication-only function referred to as Authentication Header (AH), a combined authentication/encryption function called Encapsulating Security Payload (ESP), and a key exchange function. For VPNs, both authentication and encryption are generally desired, because it is important both to (1) assure that unauthorized users do not penetrate the virtual private network and (2) assure that eavesdroppers on the Internet cannot read messages sent over the virtual private network. Because both features are generally desirable, most implementations are likely to use ESP rather than AH. The key exchange function allows for manual exchange of keys as well as an automated scheme.

IPSec is explored in Chapter 21.

18.7 RECOMMENDED READING AND WEB SITES

[RODR02] provides clear coverage of all of the topics in this chapter. Good coverage of internetworking and IPv4 can be found in [COME06] and [STEV94]. [SHAN02] and [KENT87] provide useful discussions of fragmentation. [LEE05] is a thorough technical description IPv6. [KESH98] provides an instructive look at present and future router functionality. [METZ02] and [DOI94] describe the IPv6 anycast feature. For the reader interested in a more in-depth discussion of IP addressing, [SPOR03] offers a wealth of detail.

COME06 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture*. Upper Saddle River, NJ: Prentice Hall, 2006.

DOI04 Doi, S., et al. "IPv6 Anycast for Simple and Effective Communications." *IEEE Communications Magazine*, May 2004.

HUIT98 Huitema, C. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.

KENT87 Kent, C., and Mogul, J. "Fragmentation Considered Harmful." *ACM Computer Communication Review*, October 1987.

KESH98 Keshav, S., and Sharma, R. "Issues and Trends in Router Design." *IEEE Communications Magazine*, May 1998.

LEE05 Lee, H. *Understanding IPv6*. New York: Springer-Verlag, 2005.

METZ02 Metz C. "IP Anycast." *IEEE Internet Computing*, March 2002.

RODR02 Rodriguez, A., et al. *TCP/IP Tutorial and Technical Overview*. Upper Saddle River: NJ: Prentice Hall, 2002.

SHAN02 Shannon, C.; Moore, D.; and Claffy, K. "Beyond Folklore: Observations on Fragmented Traffic." *IEEE/ACM Transactions on Networking*, December 2002.

SPOR03 Sportack, M. *IP Addressing Fundamentals*. Indianapolis, IN: Cisco Press, 2003.

STEV94 Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, MA: Addison-Wesley, 1994.

Recommended Web sites:

- **IPv6:** Information about IPv6 and related topics.
- **IPv6 Working Group:** Chartered by IETF to develop standards related to IPv6. The Web site includes all relevant RFCs and Internet drafts.
- **IPv6 Forum:** An industry consortium that promotes IPv6-related products. Includes a number of white papers and articles.

18.8 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

broadcast	Internet Protocol (IP)	segmentation
datagram lifetime	internetworking	subnet
end system	intranet	subnet mask
fragmentation	IPv4	subnetwork
intermediate system	IPv6	traffic class
Internet	multicast	unicast
Internet Control Message Protocol (ICMP)	reassembly	
	router	

Review Questions

18.1. Give some reasons for using fragmentation and reassembly.
18.2. List the requirements for an internetworking facility.
18.3. What are the pros and cons of limiting reassembly to the endpoint as compared to allowing en route reassembly?
18.4. Explain the function of the three flags in the IPv4 header.
18.5. How is the IPv4 header checksum calculated?
18.6. What is the difference between the traffic class and flow label fields in the IPv6 header?
18.7. Briefly explain the three types of IPv6 addresses.
18.8. What is the purpose of each of the IPv6 header types?

Problems

18.1 Although not explicitly stated, the Internet Protocol (IP) specification, RFC 791, defines the minimum packet size a network technology must support to allow IP to run over it.

a. Read Section 3.2 of RFC 791 to find out that value. What is it?
b. Discuss the reasons for adopting that specific value.

18.2 In the discussion of IP, it was mentioned that the *identifier, don't fragment identifier*, and *time-to-live* parameters are present in the Send primitive but not in the Deliver primitive because they are only of concern to IP. For each of these parameters, indicate whether it is of concern to the IP entity in the source, the IP entities in any intermediate routers, and the IP entity in the destination end systems. Justify your answer.

18.3 What is the header overhead in the IP protocol?

18.4 Describe some circumstances where it might be desirable to use source routing rather than let the routers make the routing decision.

18.5 Because of fragmentation, an IP datagram can arrive in several pieces, not necessarily in the correct order. The IP entity at the receiving end system must accumulate these fragments until the original datagram is reconstituted.

- **a.** Consider that the IP entity creates a buffer for assembling the data field in the original datagram. As assembly proceeds, the buffer will contain blocks of data and "holes" between the data blocks. Describe an algorithm for reassembly based on this concept.
- **b.** For the algorithm in part (a), it is necessary to keep track of the holes. Describe a simple mechanism for doing this.

18.6 A 4480-octet datagram is to be transmitted and needs to be fragmented because it will pass through an Ethernet with a maximum payload of 1500 octets. Show the Total Length, More Flag, and Fragment Offset values in each of the resulting fragments.

18.7 Consider a header that consists of 10 octets, with the checksum in the last two octets (this does not correspond to any actual header format) with the following content (in hexadecimal): 01 00 F6 F7 F4 F5 F2 03 00 00

- **a.** Calculate the checksum. Show your calculation.
- **b.** Show the resulting packet.
- **c.** Verify the checksum.

18.8 The IP checksum needs to be recalculated at routers because of changes to the IP header, such as the lifetime field. It is possible to recalculate the checksum from scratch. Suggest a procedure that involves less calculation. *Hint*: Suppose that the value in octet k is changed by $Z = \text{new\_value} - \text{old\_value}$; consider the effect of this change on the checksum.

18.9 An IP datagram is to be fragmented. Which options in the option field need to be copied into the header of each fragment, and which need only be retained in the first fragment? Justify the handling of each option.

18.10 A transport-layer message consisting of 1500 bits of data and 160 bits of header is sent to an internet layer, which appends another 160 bits of header. This is then transmitted through two networks, each of which uses a 24-bit packet header. The destination network has a maximum packet size of 800 bits. How many bits, including headers, are delivered to the network-layer protocol at the destination?

18.11 The architecture suggested by Figure 18.2 is to be used. What functions could be added to the routers to alleviate some of the problems caused by the mismatched local and long-haul networks?

18.12 Should internetworking be concerned with a network's internal routing? Why or why not?

18.13 Provide the following parameter values for each of the network classes A, B, and C. Be sure to consider any special or reserved addresses in your calculations.

- **a.** Number of bits in network portion of address
- **b.** Number of bits in host portion of address
- **c.** Number of distinct networks allowed
- **d.** Number of distinct hosts per network allowed
- **e.** Integer range of first octet

18.14 What percentage of the total IP address space does each of the network classes represent?

18.15 What is the difference between the subnet mask for a Class A address with 16 bits for the subnet ID and a class B address with 8 bits for the subnet ID?

18.16 Is the subnet mask 255.255.0.255 valid for a Class A address?

18.17 Given a network address of 192.168.100.0 and a subnet mask of 255.255.255.192,

- **a.** How many subnets are created?
- **b.** How many hosts are there per subnet?

18.18 Given a company with six individual departments and each department having ten computers or networked devices, what mask could be applied to the company network to provide the subnetting necessary to divide up the network equally?

18.19 In contemporary routing and addressing, the notation commonly used is called classless interdomain routing or CIDR. With CIDR, the number of bits in the mask is indicated in the following fashion: 192.168.100.0/24. This corresponds to a mask of 255.255.255.0. If this example would provide for 256 host addresses on the network, how many addresses are provided with the following?

a. 192.168.100.0/23
b. 192.168.100.0/25

18.20 Find out about your network. Using the command "ipconfig", "ifconfig", or "winipcfg", we can learn not only our IP address but other network parameters as well. Can you determine your mask, gateway, and the number of addresses available on your network?

18.21 Using your IP address and your mask, what is your network address? This is determined by converting the IP address and the mask to binary and then proceeding with a bitwise logical AND operation. For example, given the address 172.16.45.0 and the mask 255.255.224.0, we would discover that the network address would be 172.16.32.0.

18.22 Compare the individual fields of the IPv4 header with the IPv6 header. Account for the functionality provided by each IPv4 field by showing how the same functionality is provided in IPv6.

18.23 Justify the recommended order in which IPv6 extension headers appear (i.e., why is the Hop-by-Hop Options header first, why is the Routing header before the Fragment header, and so on).

18.24 The IPv6 standard states that if a packet with a nonzero flow label arrives at a router and the router has no information for that flow label, the router should ignore the flow label and forward the packet.

a. What are the disadvantages of treating this event as an error, discarding the packet, and sending an ICMP message?
b. Are there situations in which routing the packet as if its flow label were zero will cause the wrong result? Explain.

18.25 The IPv6 flow mechanism assumes that the state associated with a given flow label is stored in routers, so they know how to handle packets that carry that flow label. A design requirement is to flush flow labels that are no longer being used (stale flow label) from routers.

a. Assume that a source always send a control message to all affected routers deleting a flow label when the source finishes with that flow. In that case, how could a stale flow label persist?
b. Suggest router and source mechanisms to overcome the problem of stale flow labels.

18.26 The question arises as to which packets generated by a source should carry nonzero IPv6 flow labels. For some applications, the answer is obvious. Small exchanges of data should have a zero flow label because it is not worth creating a flow for a few packets. Real-time flows should have a flow label; such flows are a primary reason flow labels were created. A more difficult issue is what to do with peers sending large amounts of best-effort traffic (e.g., TCP connections). Make a case for assigning a unique flow label to each long-term TCP connection. Make a case for not doing this.

18.27 The original IPv6 specifications combined the Traffic Class and Flow Label fields into a single 28-bit Flow Label field. This allowed flows to redefine the interpretation of different values of priority. Suggest reasons why the final specification includes the Priority field as a distinct field.

18.28 For Type 0 IPv6 routing, specify the algorithm for updating the IPv6 and Routing headers by intermediate nodes.