COMPUTER NETWORKS

Network Layer: Internet Protocol

1. INTRODUCTION

In the Internet model (**TCP/IP**), the main network protocol is the **Internet Protocol** (IP). In the following section first we will discuss the need of network layer then we will describe the two protocols IPv4 and IPv6 at network layer.

2. INTERNETWORKING

The physical and data link layers of a network operate locally. These two layers are jointly responsible for data delivery on the network from one node to the next, as shown in Figure 20.1. This internetwork is made of five networks: four LANs and one WAN. If host A needs to send a data packet to host D, the packet needs to go first from A to S1 (a switch or router), then from S1 to S3, and finally from S3 to host D. We say that the data packet passes through three links. In each link, two physical and two data link layers are involved.

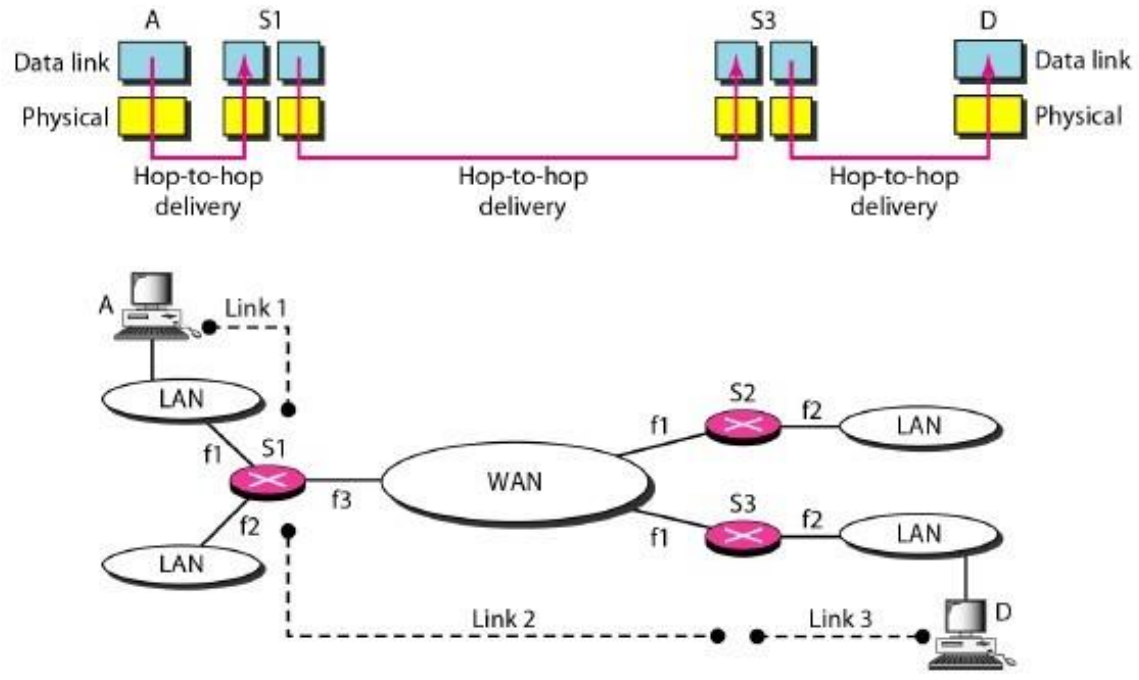


Figure 20.1 Links between two hosts

However, there is a big problem here. When data arrive at interface f1 of S1, how does S1 know that interface f3 is the outgoing interface? There is no provision in the data link (or physical) layer to help S1 make the right decision. The frame does not carry any routing information either. The frame contains the MAC address of A as the source and the MAC address of S1 as the destination. For a LAN or a WAN, delivery means carrying the frame through one link, and not beyond i.e. it is restricted for hop-to-hop delivery.

2.1. Need for Network Layer

To solve the problem of delivery through several links, the network layer (or the internetwork layer, as it is sometimes called) was designed. **The network layer is responsible for host-to-host delivery and for routing the packets through the routers or switches**. Figure 20.2 shows the same internetwork with a network layer added.

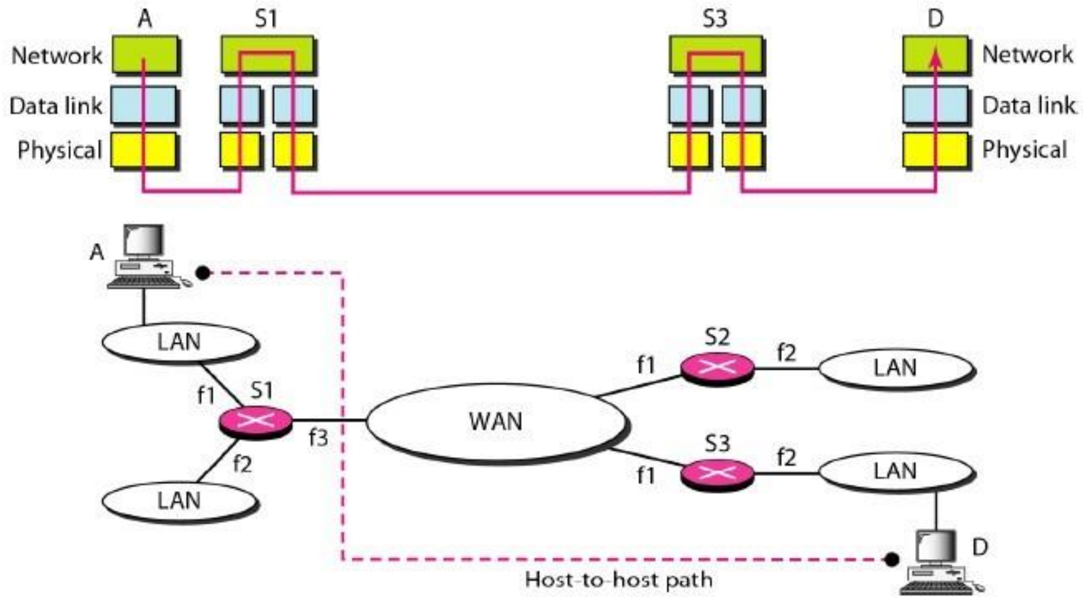


Figure 20.2 Network layer in an internetwork

Figure 20.3 shows the general idea of the functionality of the network layer at a source, at a router, and at the destination:

Responsibilities of Network Layer at Source

- The network layer at the source is responsible for creating a packet from the data coming from another protocol (such as a transport layer protocol or a routing protocol).
- The header of the packet contains, among other information, the logical addresses of the source and destination.
- The network layer is responsible for checking its routing table to find the routing information (such as the outgoing interface of the packet or the physical address of the next node).
- If the packet is too large, the packet is fragmented (fragmentation is discussed later in this chapter).

Responsibilities of Network Layer at Switch/Router

- The network layer at the switch or router is responsible for routing the packet.
- When a packet arrives, the router or switch consults its routing table and finds the interface from which the packet must be sent.
- The packet, after some changes in the header, with the routing information is passed to the data link layer again.

Responsibilities of Network Layer at Source

- The network layer at the destination is responsible for address verification; it makes sure that the destination address on the packet is the same as the address of the host.
- If the packet is a fragment, the network layer waits until all fragments have arrived, and then reassembles them and delivers the reassembled packet to the transport layer.

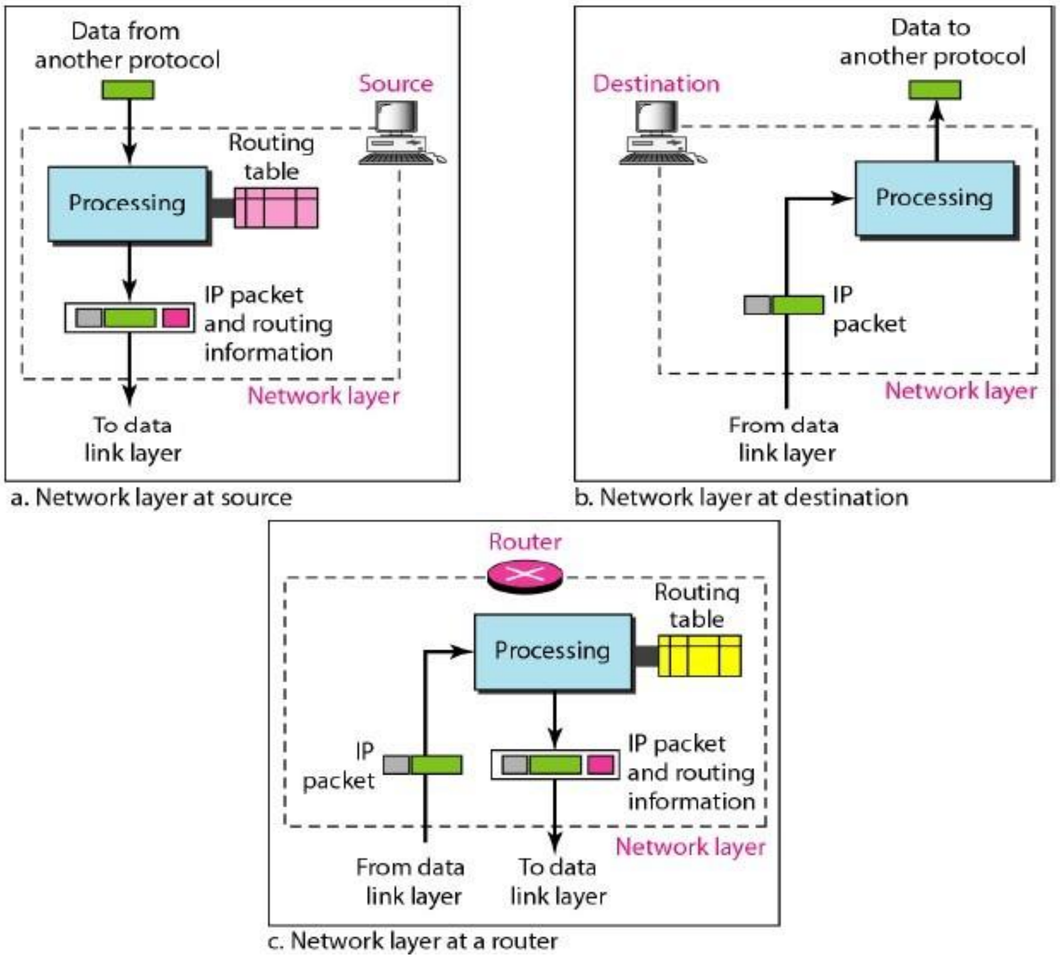


Figure 20.3 Network layer at the source, router, and destination

3. IPv4

The Internet Protocol version 4 (IPv4) is the delivery mechanism used by the TCP/IP protocols. Figure 20.4 shows the position of IPv4 in the suite.

3.1. Characteristics of IPv4 protocol

- **IPv4** is an **unreliable** and **connectionless datagram protocol-a best-effort delivery service**.
- The term ***best-effort* means** that IPv4 provides **no error control or flow control** (except for error detection on the header).
- IPv4 assumes the unreliability of the underlying layers and does its best to get a transmission through to its destination, but with no guarantees.
- If reliability is important, IPv4 must be paired with a reliable protocol such as **TCP**.

An **example** of a more commonly understood **best-effort delivery service** is the **post office**. The post office does its best to deliver the mail but does not always succeed. If an unregistered letter is lost, it is up to the sender or would-be recipient to discover the loss and rectify the problem. The post office itself does not keep track of every letter and cannot notify a sender of loss or damage.

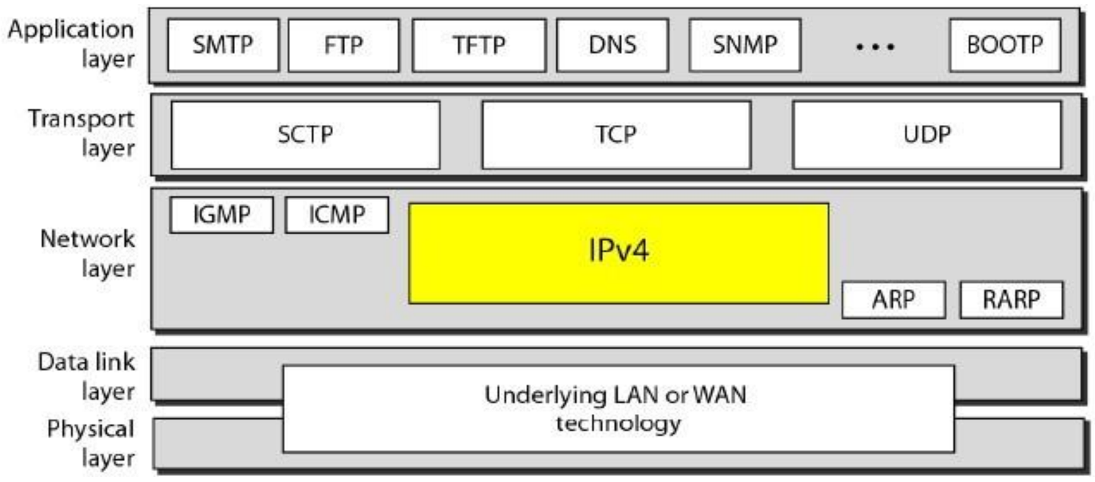


Figure 20.4 Position of IPv4 in TCP/IP protocol suite

- **IPv4** is also a **connectionless packet-switching** network that uses the **datagram approach**.
- This means that **each datagram** is **handled independently**, and each datagram **can follow** a **different route to the destination**.
- This implies that datagrams sent by the same source to the same destination could **arrive out of order**. Also, some could be lost or corrupted during transmission.
- IPv4 relies on a higher-level protocol like **TCP** to take care of all these problems.

3.2. Datagram

Packets in the IPv4 layer are called **datagrams**. Figure 20.5 shows the IPv4 datagram **format**. A datagram is a variable-length packet consisting of **two parts**: **header** and **data**. The **header** is **20 to 60 bytes in length** and contains information essential to routing and delivery. It is customary in *TCP/IP* to show the header in 4-byte sections. A brief description of each field is in order:

- **Version (VER).** This **4-bit** field defines the **version** of the IPv4 protocol. Currently the version is **4**. However, version 6 (or IPv6) may totally replace version 4 in the future. This field tells the IPv4 software running in the processing machine that the datagram has the format of version 4.
- **Header length (HLEN).** This **4-bit** field defines the **total length of the datagram header** in **4-byte words**. This field is needed because the length of the header is variable (between 20 and 60 bytes). When there are no options, the header length is 20 bytes, and the value of this field is 5 (5 x 4 = 20). When the option field is at its maximum size, the value of this field is 15 (15 x 4 = 60).
- **Services.** IETF has changed the interpretation and name of this 8-bit field. This field, previously called **service type**, is now called **differentiated services**. We show both interpretations in Figure 20.6.

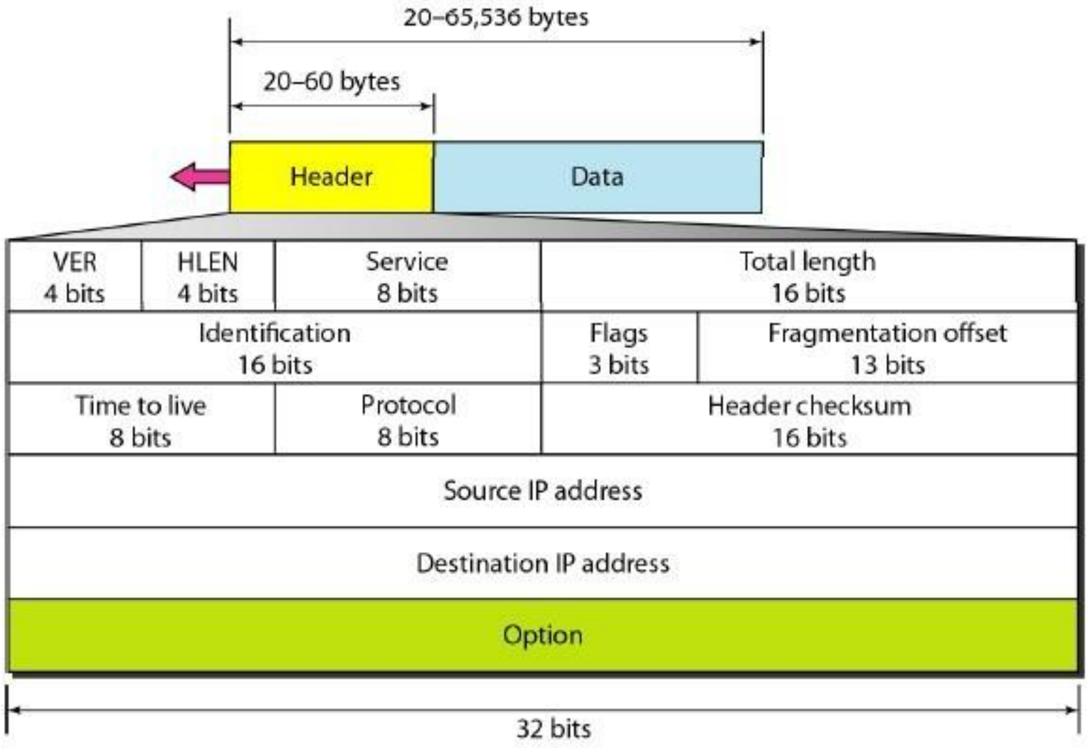


Figure 20.5 IPv4 datagram format

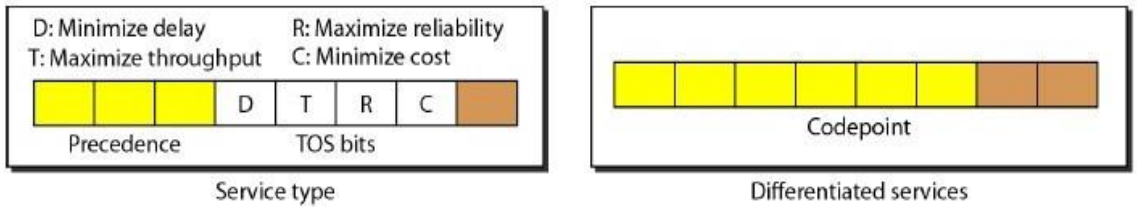


Figure 20.6 Service type or differentiated services

1. Service Type

In this interpretation, the **first 3 bits** are called **precedence bits**. The **next 4 bits** are called **type of service (TOS) bits**, and the **last bit** is **not used**.

a. **Precedence** is a **3-bit** subfield ranging from 0 (**000** in binary) to 7 (**111** in binary). **The precedence defines the priority of the datagram in issues such as congestion**. If a router is congested and needs to discard some datagrams, those **datagrams with lowest precedence are discarded first**. Some datagrams in the Internet are more important than others. For **example**, a datagram used for network management is much more urgent and important than a datagram containing optional information for a group.

b. **TOS bits** is a **4-bit** subfield with **each bit** having a **special meaning**. Although a bit can be either 0 or 1, one and only one of the bits can have the value of 1 in each datagram. The bit patterns and their interpretations are given in Table 20.1. With only 1 bit set at a time, we can have **five different types of services**. Application programs can request a specific type of service. The defaults for some applications are shown in Table 20.2.

Table 20.1 Types of service

TOS Bits	*Description*
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

Table 20.2 Default types of service

Protocol	*TOS Bits*	*Description*
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput

It is clear from Table 20.2 that:

- Interactive activities, activities requiring immediate attention, and activities requiring immediate response need **minimum delay**. e.g. TELNET.
- Those activities that send bulk data require **maximum throughput**. E.g. FTP(data)
- Management activities need **maximum reliability**. e.g. SNMP.
- Background activities need **minimum cost**. e.g. NNTP.

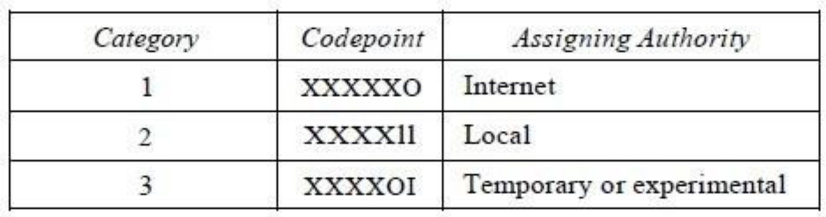
2. Differentiated Services

In this interpretation, the **first 6 bits** make up the **codepoint subfield**, and the **last 2 bits** are **not used**. The codepoint subfield can be used in two different ways:

i. When the **3 rightmost bits are 0s**, the **3 leftmost bits are interpreted the same as the precedence bits** in the service type interpretation. In other words, it is compatible with the old interpretation.

ii. When the **3 rightmost bits are not all 0s**, the **6 bits define 64 services based on the priority assignment** by the Internet or local authorities according to Table 20.3. The first category contains 32 service types; the second and the third each contain 16. The first category (numbers 0, 2,4, ... ,62) is assigned by the Internet authorities (IETF). The second category (3, 7, 11, 15, , 63) can be used by local authorities (organizations). The third category (1, 5, 9, ,61) is temporary and can be used for experimental purposes.

Table 20.3 Values for codepoints



Category	*Codepoint*	*Assigning Authority*
1	XXXXXO	Internet
2	XXXXll	Local
3	XXXXOI	Temporary or experimental

- **Total length.** This is a **16-bit** field that defines the **total length (header plus data)** of the **IPv4 datagram in bytes**. To find the length of the data coming from the upper layer, subtract the header length from the total length.

  Length of data =total length - header length

  Since the field length is 16 bits, the **total length of the IPv4 datagram** is limited to **65,535 (2^{16} - 1) bytes**, of which 20 to 60 bytes are the header and the rest is data from the upper layer. If the size of an IPv4 datagram is less than 46 bytes, some **padding** will be added to meet this requirement. In this case, when a machine decapsulates the datagram, it needs to check the total length field to determine how much is really data and how much is padding (see Figure 20.7).

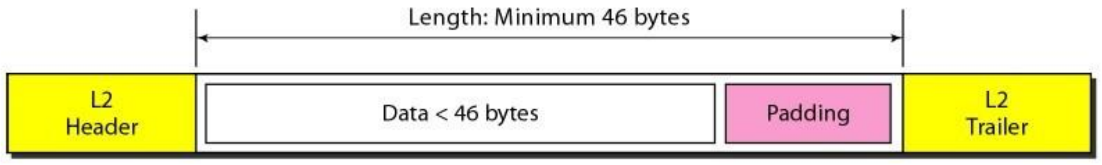


Figure 20.7 Encapsulation of a small datagram in an Ethernet frame

- **Identification.** This field is used in fragmentation (discussed in the next section).
- **Flags.** This field is used in fragmentation (discussed in the next section).
- **Fragmentation offset.** This field is used in fragmentation (discussed in the next section).
- **Time to live(TTL).** A datagram has a **limited lifetime** in its travel through an internet. This field is used mostly to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately **2 times the maximum number of routes between any two hosts**. Each router that processes the datagram **decrements this number by 1**. If this value, after being decremented, is zero, the router discards the datagram.

  Need of TTL: This field is needed because routing tables in the Internet can become corrupted. A datagram may travel between two or more routers for a long time without ever getting delivered to the destination host. This field limits the lifetime of a datagram. Another use of this field is to intentionally limit the journey of the packet. For example, if the source wants to confine the packet to the local network, it can store 1 in this field. When the packet arrives at the first router, this value is decremented to 0, and the datagram is discarded.

- **Protocol.** This **8-bit** field defines the **higher-level protocol** that uses the services of the IPv4 layer. An IPv4 datagram can encapsulate data from several higher-level protocols such as **TCP, UDP, ICMP, and IGMP**. Since the IPv4 protocol carries data from different other protocols, the value of this field helps the receiving network layer know to which protocol the data belong (see Figure 20.8).

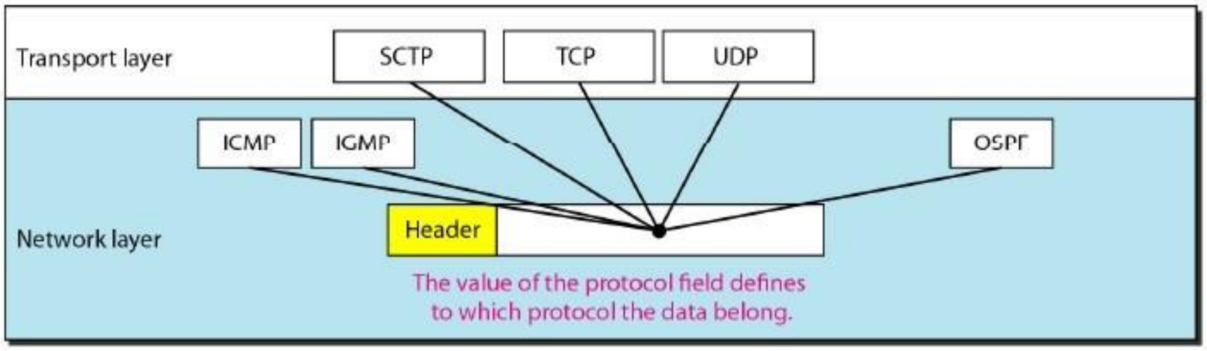

Transport layer
SCTP
TCP
UDP
ICMP
IGMP
OSPF
Network layer
Header
The value of the protocol field defines to which protocol the data belong.

Figure 20.8 Protocol field and encapsulated data

The value of this field for each higher-level protocol is shown in Table 20.4.

Table 20.4 Protocol values

Value	*Protocol*
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

- **Checksum.** The checksum is used to secure the IPv4 **header**.

- **Source address.** This **32-bit field** defines the **IPv4 address of the source**. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

- **Destination address.** This **32-bit field** defines the **IPv4 address of the destination**. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

- **Options.** The header of the IPv4 datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes long and was discussed in the previous section. The variable part comprises the options that can be a maximum of 40 bytes. Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging.

Example 20.1
An IPv4 packet has arrived with the first 8 bits as shown:
01000010
The receiver discards the packet. Why?
Solution
There is an error in this packet. The 4 leftmost bits (0100) show the version, which is correct. The next 4 bits (0010) show an invalid header length (2 x 4 =8). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

Example 20.2
In an IPv4 packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?
Solution
The HLEN value is 8, which means the total number of bytes in the header is 8 x 4, or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

Example 20.3
In an IPv4 packet, the value of HLEN is 5, and the value of the total length field is 0x0028. How many bytes of data are being carried by this packet?
Solution
The HLEN value is 5, which means the total number of bytes in the header is 5 x 4, or 20 bytes (no options). The total length is 40 bytes, which means the packet is carrying 20 bytes of data (40- 20).

Fragmentation

A datagram can travel through different networks. Each router decapsulates the IPv4 datagram from the frame it receives, processes it, and then encapsulates it in another frame. The format and size of the received frame depend on the protocol used by the physical network through which the frame has just traveled. The format and size of the sent frame depend on the protocol used by the physical network through which the frame is going to travel. For example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

Maximum Transfer Unit (MTU)

Each data link layer protocol has its own frame format in most protocols. One of the fields defined in the format is the maximum size of the data field. In other words, when a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network (see Figure 20.9).

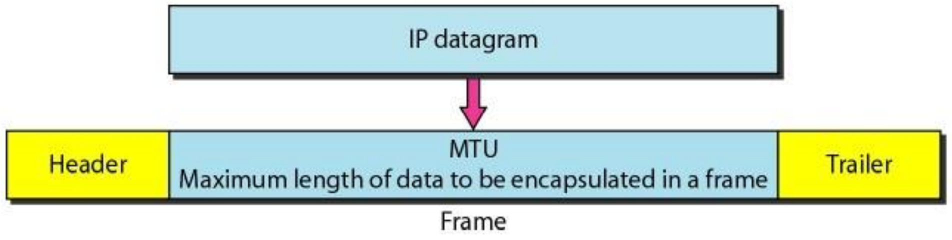


Figure 20.9 Maximum transfer unit (MTU)

The value of the MTU depends on the physical network protocol. Table 20.5 shows the values for some protocols.

Table 20.5 MTUs for some networks

Protocol	*MTU*
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

- To make the IPv4 protocol independent of the physical network, the designers decided to make the maximum length of the IPv4 datagram equal to 65,535 bytes. This makes transmission more efficient if we use a protocol with an MTU of this size. However, for other physical networks, we must divide the datagram to make it possible to pass through these networks. This is called **fragmentation.**
- When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but with some changed. A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU. In other words, a datagram can be fragmented several times before it reaches the final destination.
- In IPv4, a datagram can be **fragmented by the source host or any router in the path** although there is a tendency to limit fragmentation only at the source.
- The **reassembly of the datagram**, however, is done **only by the destination host** because each fragment becomes an independent datagram. Whereas the fragmented datagram can travel through different routes, and we can never control or guarantee which route a fragmented datagram may take, all the fragments belonging to the same datagram should finally arrive at the destination host. So it is logical to do the reassembly at the final destination.
- When a datagram is fragmented, required parts of the header must be copied by all fragments. The option field may or may not be copied.
- The host or router that fragments a datagram must change the values of **three fields**: flags, fragmentation offset, and total length. The rest of the fields must be copied.

Fields Related to Fragmentation

The fields that are related to fragmentation and reassembly of an IPv4 datagram are:

- identification
- flags
- fragmentation offset

- **Identification:** This **16-bit** field identifies a datagram originating from the source host.
  - The combination of the identification and source IPv4 address must uniquely define a datagram as it leaves the source host.

- When the IPv4 protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by 1.
- When a datagram is fragmented, the value in the identification field is copied to all fragments. In other words, all fragments have the same identification number, the same as the original datagram.
- The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value must be assembled into one datagram.

- **Flags.** This is a **3-bit** field.
  - The **first bit is reserved**.
  - The **second bit is called the *do not fragment* bit**. If its **value is 1**, the machine must **not fragment** the datagram. If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host (see Chapter 21). If its **value is 0**, the datagram **can be fragmented** if necessary.
  - The **third bit** is called the ***more fragment* bit**. If its **value is 1**, it means **the datagram is not the last fragment**; there are more fragments after this one. **If its value is 0**, it means this is **the last or only fragments** (see Figure 20.10).

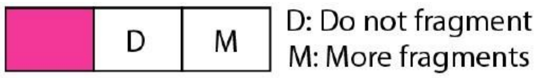


Figure 20.10 Flags used in fragmentation

- **Fragmentation offset.** This **13-bit** field shows the **relative position of this fragment with respect to the whole datagram**. It is the offset of the data in the original datagram measured in **units of 8 bytes**. Figure 20.11 shows a datagram with a data size of 4000 bytes fragmented into three fragments. The bytes in the original datagram are numbered 0 to 3999. The first fragment carries bytes 0 to 1399. The offset for this datagram is 0/8 =0. The second fragment carries bytes 1400 to 2799; the offset value for this fragment is 1400/8 = 175.Finally, the third fragment carries bytes 2800 to 3999. The offset value for this fragment is 2800/8 =350.

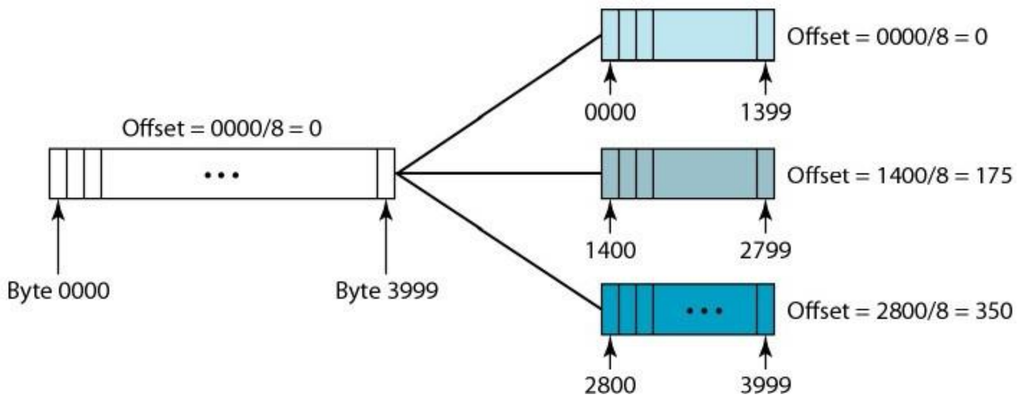


Figure 20.11 Fragmentation example

Figure 20.12 shows an expanded view of the fragments in Figure 20.11. Notice the value of the identification field is the same in all fragments. Notice **the value of the flags field with the *more* bit set for all fragments except the last**. Also, the value of the offset field for each fragment is shown.

- The figure also shows what happens if a fragment itself is fragmented. In this case the value of the offset field is always relative to the original datagram. For **example**, in the figure, the second fragment is itself fragmented later to two fragments of 800 bytes and 600 bytes, but the offset shows the relative position of the fragments to the original data.
- It is obvious that even if each fragment follows a different path and arrives out of order, the final destination host can reassemble the original datagram from the fragments received (if none of them is lost) by using the following strategy:

  1. The first fragment has an offset field value of zero.

  2. Divide the length of the first fragment by 8. The second fragment has an offset value equal to that result.

  3. Divide the total length of the first and second fragments by 8. The third fragment has an offset value equal to that result.

  4. Continue the process. The last fragment has a *more* bit value of 0.

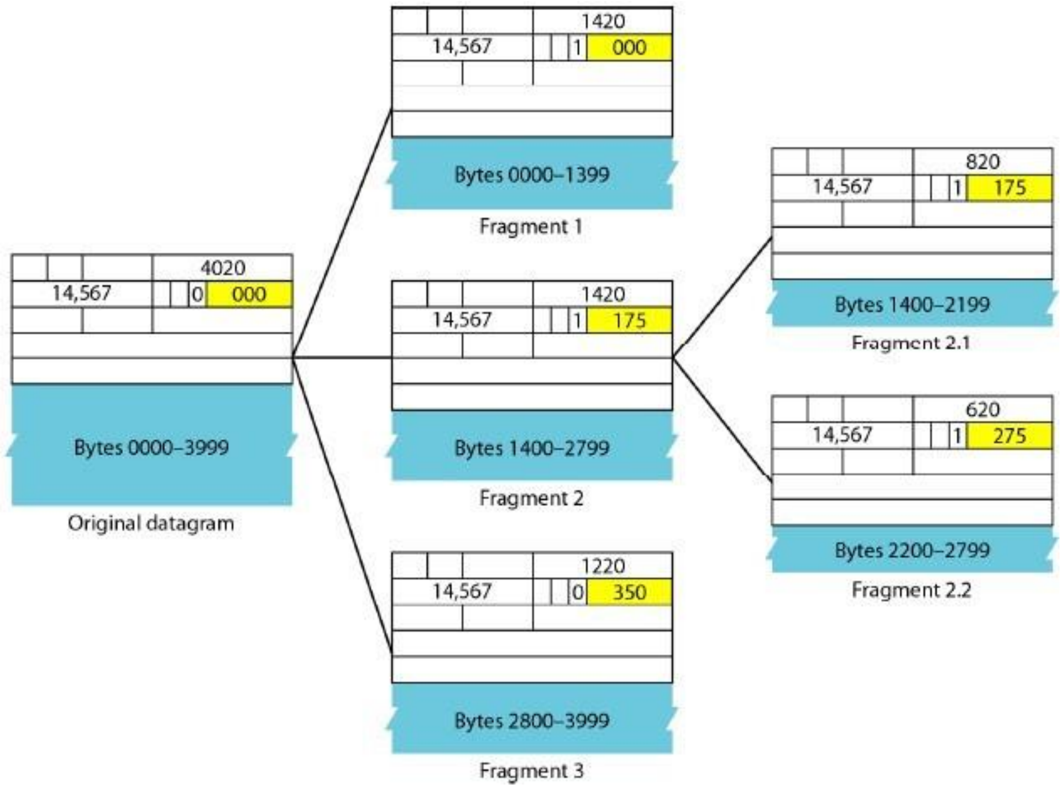


Figure 20.12 Detailed fragmentation example

Example 20.5
A packet has arrived with an *M* bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution
If the *M* bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A non fragmented packet is considered the last fragment.

Example 20.6
A packet has arrived with an *M* bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution
If the *M* bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset). See Example 20.7.

Example 20.7
A packet has arrived with an *M* bit value of 1 and a fragmentation offset value of 0. Is this the first fragment, the last fragment, Or a middle fragment?

Solution
Because the *M* bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

Example 20.8
A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution
To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

Example 20.9
A packet has arrived in which the offset value is 100, the value of HLEN is 5, and the value of the tota1 length field is 100. What are the numbers of the first byte and the last byte?

Solution
The first byte number is 100 x 8 = 800. The total length is 100 bytes, and the header length is 20 bytes (5 x 4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.