TRANSPORT LAYER

The transport layer in the TCP/IP suite is located between the application layer and the network layer. It provides services to the application layer and receives services from the network layer.

The transport layer acts as a liaison between a client program and a server program, a process-to-process connection. The transport layer is the heart of the TCP/IP protocol suite; it is the end-to- end logical vehicle for transferring data from one point to another in the Internet.

Introduction:

The transport layer is located between the application layer and the network layer. It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host. Communication is provided using a logical connection, which means that the two application layers, which can be located in different parts of the globe, assume that there is an imaginary direct connection through which they can send and receive messages.

SERVICES PROVIDED TO THE UPPER LAYERS

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services provided by the network layer. The software and/or hardware within the transport layer that does the work is called the **transport entity**.

The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card. The first two options are most common on the Internet. The (logical) relationship of the network, transport, and application layers is illustrated in Fig. 4.1. *Just as there are two types of network service,* ***connection-oriented*** *and* ***connectionless****, there are also two types of transport service*. The ***connection-oriented transport service*** is similar to the connection-oriented network service in many ways. In both cases, connections have three phases: establishment, data transfer, and release. Addressing and flow control are also similar in both layers.

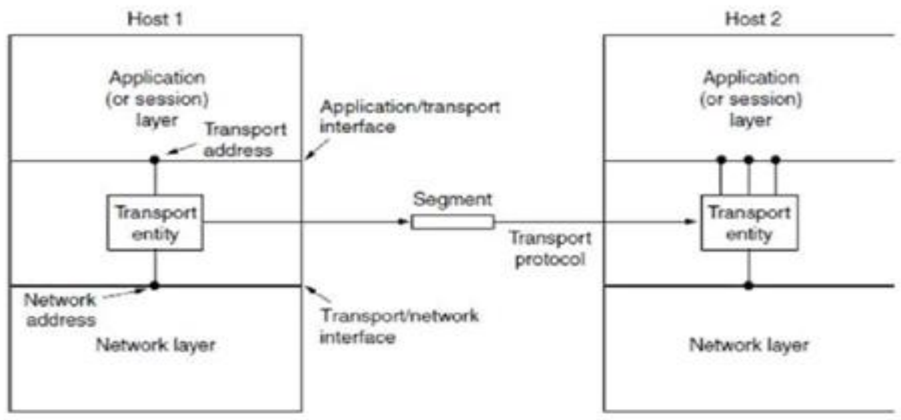


Figure 4.1: The network, transport, and application layers

Furthermore, the ***connectionless transport service*** is also very similar to the connectionless network service. However, note that it can be difficult to provide a connectionless transport service on top of a connection-oriented network service, since it is inefficient to set up a connection to send a single packet and then tear (*meaning run/rip/rush*) it down immediately afterwards.

ELEMENTS OF TRANSPORT PROTOCOLS

The transport service is implemented by a **transport protocol** used between the two transport entities. In some ways, transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control, among other issues. However, significant differences between the two also exist. These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig. 4.3.

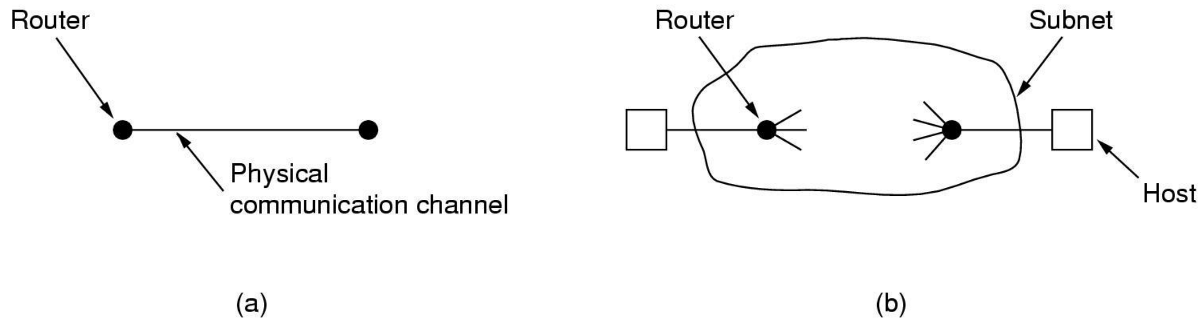


Figure 4.3: Environment of the (a) data link layer (b) transport layer

At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network.

For one thing, over point-to-point links such as wires or optical fiber, it is usually not necessary for a router to specify which router it wants to talk to—each outgoing line leads directly to a particular router. In the transport layer, explicit addressing of destinations is required.

For another thing, the process of establishing a connection over the wire of Fig. 4.3(a) is simple: the other end is always there (unless it has crashed, in which case it is not there). Either way, there is not much to do.

Even on wireless links, the process is not much different. Just sending a message is sufficient to have it reach all other destinations. If the message is not acknowledged due to an error, it can be resent. In the transport layer, initial connection establishment is complicated.

ADDRESSING

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. (Connectionless transport has the same problem: to whom should each message be sent?) The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called **ports**.

We will use the generic term **TSAP** (**Transport Service Access Point**) to mean a specific endpoint in the transport layer. The analogous endpoints in the network layer (i.e., network layer addresses) are naturally called **NSAPs** (**Network Service Access Points**). IP addresses are examples of NSAPs.

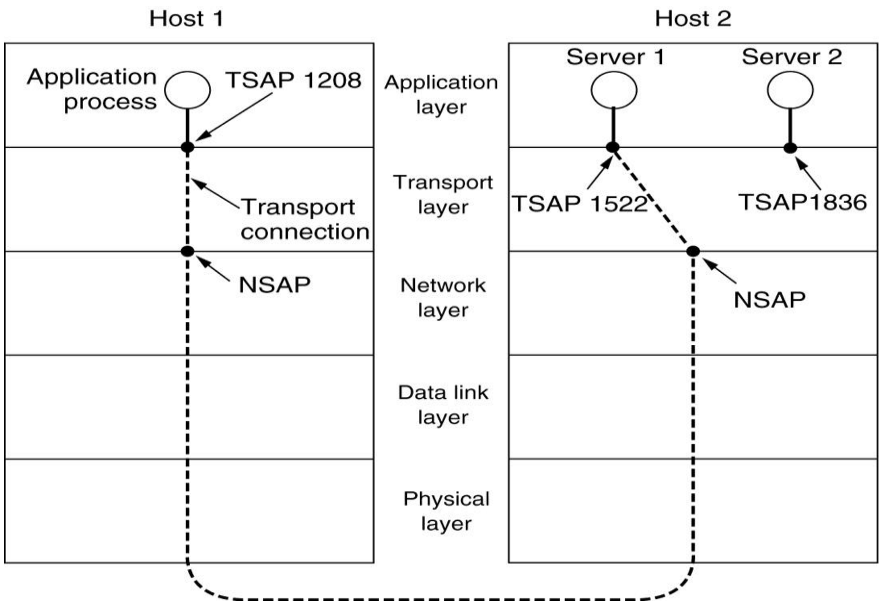


Figure 4.4: TSAPs, NSAPs, and Transport connections

Figure 4.4 illustrates the relationship between the NSAPs, the TSAPs, and a transport connection.

Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host.

A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.
3. The application process sends over the mail message.
4. The mail server responds to say that it will deliver the message.
5. The transport connection is released.

CONNECTION ESTABLISHMENT

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, delay, corrupt, and duplicate packets. This behavior causes serious complications. Imagine a network that is so congested that acknowledgements hardly ever get back in time and each packet times out and is retransmitted two or three times. Suppose that the network uses datagrams inside and that every packet follows a different route.

Some of the packets might get stuck in a traffic jam inside the network and take a long time to arrive. That is, they may be delayed in the network and pop out much later, when the sender thought that they had been lost. <u>The worst possible nightmare is as follows.</u> A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person. Unfortunately, the packets decide to take the scenic route to the destination and go off exploring a remote corner of the network.

The sender then times out and sends them all again. This time the packets take the shortest route and are delivered quickly so the sender releases the connection.

Unfortunately, eventually the initial batch of packets finally come out of hiding and arrive at the destination in order, asking the bank to establish a new connection and transfer money (again).

The bank has no way of telling that these are duplicates. It must assume that this is a second, independent transaction, and transfers the money again.

The crux (*meaning root*) of the problem is that the delayed duplicates are thought to be new packets. We cannot prevent packets from being duplicated and delayed. But if and when this happens, the packets must be rejected as duplicates and not processed as fresh packets. The problem can be attacked in various ways, none of them very satisfactory. One way is to use throwaway transport addresses. In this approach, each time a transport address is needed, a new one is generated. When a connection is released, the address is discarded and never used again. Delayed duplicate packets then never find their way to a transport process and can do no damage.

Note: However, this approach makes it more difficult to connect with a process in the first place.

Another possibility is to give each connection a unique identifier (i.e., a sequence number incremented for each connection established) chosen by the initiating party and put in each segment, including the one requesting the connection. After each connection is released, each transport entity can update a table listing obsolete connections as (peer transport entity, connection identifier) pairs. Whenever a connection request comes in, it can be checked against the table to see if it belongs to a previously released connection.

Unfortunately, this scheme has a basic flaw: it requires each transport entity to maintain a certain amount of history information indefinitely. This history must persist at both the source and destination machines. Otherwise, if a machine crashes and loses its memory, it will no longer know which connection identifiers have already been used by its peers.

Instead, we need to take a different tack to simplify the problem. Rather than allowing packets to live forever within the network, we devise a mechanism to kill off aged packets that are still hobbling about.

Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.
2. Putting a hop counter in each packet.
3. Timestamping each packet.

TCP uses three-way handshake to establish connections in the presence of delayed duplicate control segments as shown in figure 4.5.

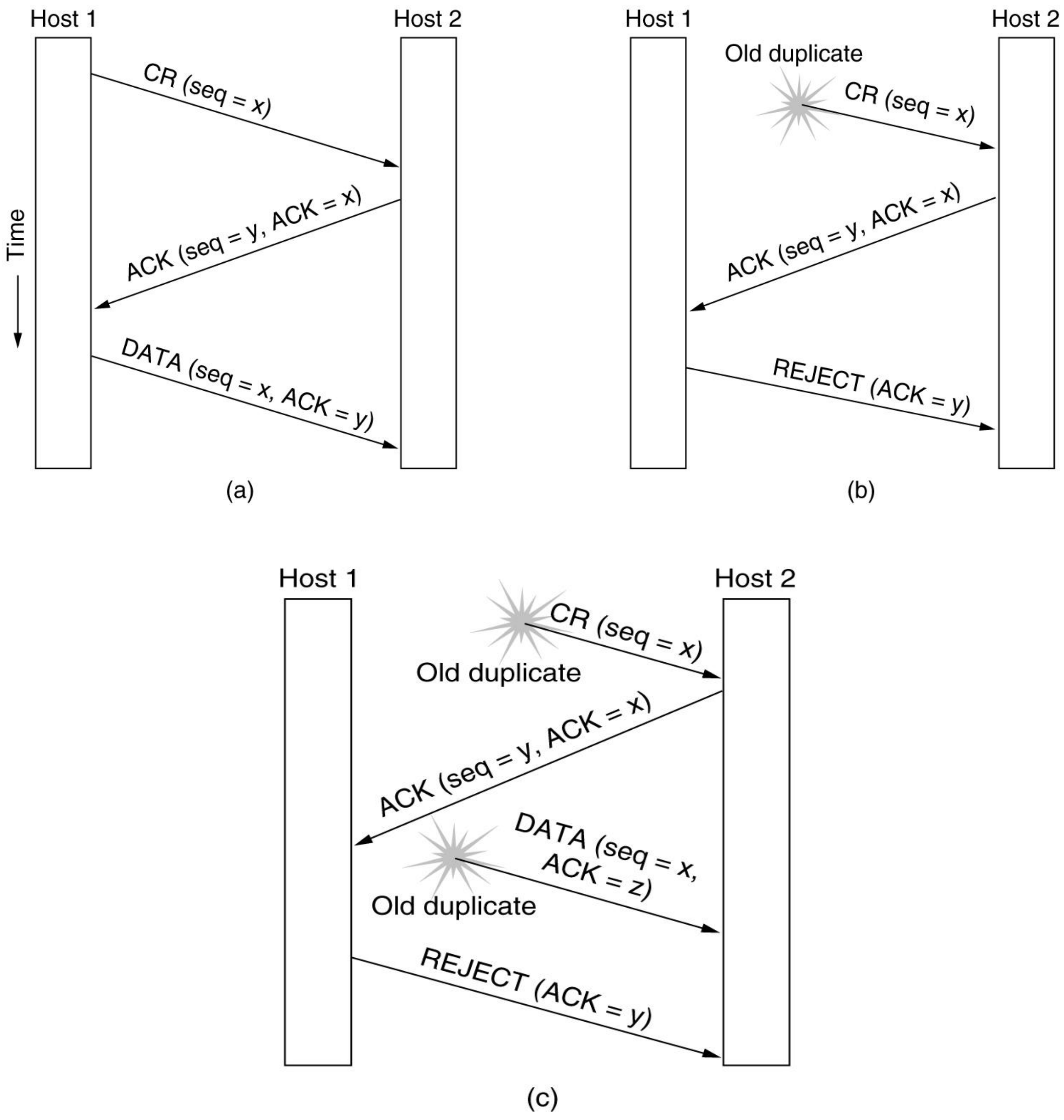


Figure 4.5: Three protocol scenarios for establishing a connection using a three - way handshake. CR denotes Connection Request.

(a) Normal operation.

(b) Old duplicate connection request appearing out of nowhere.

(c) Duplicate connection request and duplicate ack.

CONNECTION RELEASE:

Releasing a connection is easier than establishing one. There are two styles of terminating a connection: ***asymmetric release*** and ***symmetric release***. ***Asymmetric release*** is the way the telephone system works: when one party hangs up, the connection is broken. ***Symmetric release*** treats the connection as two separate unidirectional connections and requires each one to be released separately.

Asymmetric release is abrupt and may result in data loss. Consider the scenario of Fig. 4.6. After the connection is established, host 1 sends a segment that arrives properly at host 2. Then host 1 sends another segment.

Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.

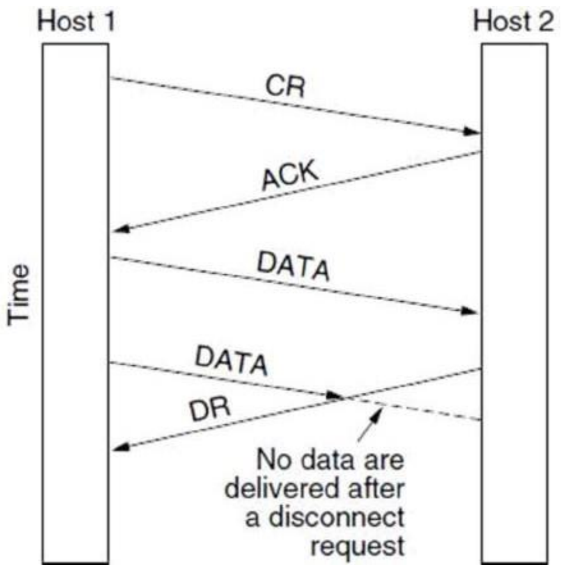
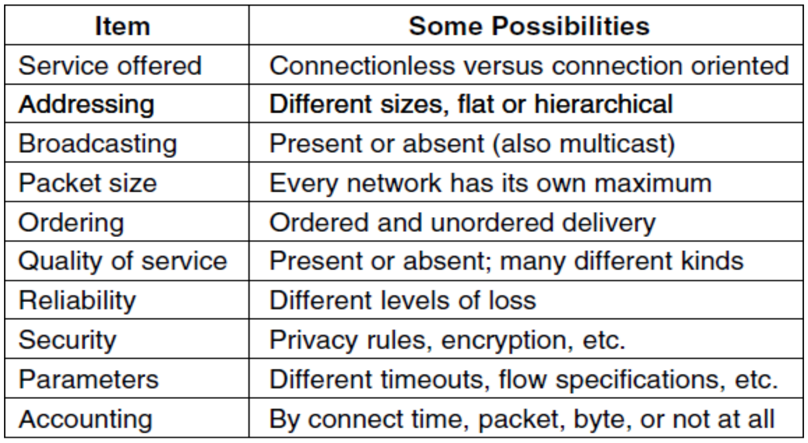


Figure 4.6: Abrupt disconnection with loss of data

INTERNETWORKING

How Networks Differ:

Networks can differ in many ways. Some of the differences, such as different modulation techniques or frame formats, are internal to the physical and data link layers. These differences will not concern us here. Instead, in Fig. 5-38 we list some of the differences that can be exposed to the network layer. It is papering over these differences that makes internetworking more difficult than operating within a single network.



Item	Some Possibilities
Service offered	Connectionless versus connection oriented
Addressing	Different sizes, flat or hierarchical
Broadcasting	Present or absent (also multicast)
Packet size	Every network has its own maximum
Ordering	Ordered and unordered delivery
Quality of service	Present or absent; many different kinds
Reliability	Different levels of loss
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, packet, byte, or not at all

Figure 5-38. Some of the many ways networks can differ.

How Networks Can Be Connected

There are two basic choices for connecting different networks: we can build devices that translate or convert packets from each kind of network into packets for each other network, or, like good computer scientists, we can try to solve the problem by adding a layer of indirection and building a common layer on top of the different networks. In either case, the devices are placed at the boundariesbetween networks.

Let us first explore at a high level how interconnection with a common network layer can be used to interconnect dissimilar networks. An internet comprised of 802.11, MPLS, and Ethernet networks is shown in Fig. 5-39(a). Suppose that the source machine on the 802.11 network wants to send a packet to the destination machine on the Ethernet network. Since these technologies are different, and they are further separated by another kind of network (MPLS), some added processing is needed at the boundaries between the networks.

Because different networks may, in general, have different forms of addressing, the packet carries a network layer address that can identify any host across the three networks. The first boundarythe packet reaches is when it transitions from an 802.11 network to an MPLS network. 802.11 provide a connectionless service, but MPLS provides a connection-oriented service. This means that a virtual circuit must be set up to cross that network.

Once the packet has traveled along the virtual circuit, it will reach the Ethernet network. At this boundary, the packet may be too large to be carried, since 802.11 can work with larger frames than Ethernet. To handle this problem, the packet is divided into fragments, and each fragment is sent separately. When the fragments reach the destination, they are reassembled. Then the packet has completed its journey.

The protocol processing for this journey is shown in Fig. 5-39(b). The source accepts data from the transport layer and generates a packet with the common network layer header, which is IP in this example. The network header contains the ultimate destination address, which is used to determine that the packet should be sent via the first router. So the packet is encapsulated in an 802.11 frame whose destination is the first router and transmitted. At the router, the packet is removed from the frame's data field and the 802.11 frame header is discarded. The router now examines the IP address in the packet and looks up this address in its routing table. Based on this address, it decides to send thepacket to the second router next. For this part of the path, an MPLS virtual circuit must be establishedto the second router and the packet must be encapsulated with MPLS headers that travel this circuit. At the far end, the MPLS header is discarded and the network address is again consulted to find the next network layer hop. It is the destination itself. Since the packet is too long to be sent over Ethernet, it is split into two portions. Each of these portions is put into the data field of an Ethernet frame and sent to the Ethernet address of the destination. At the destination, the Ethernet header is stripped fromeach of the frames, and the contents are reassembled. The packet has finally reached its destination.

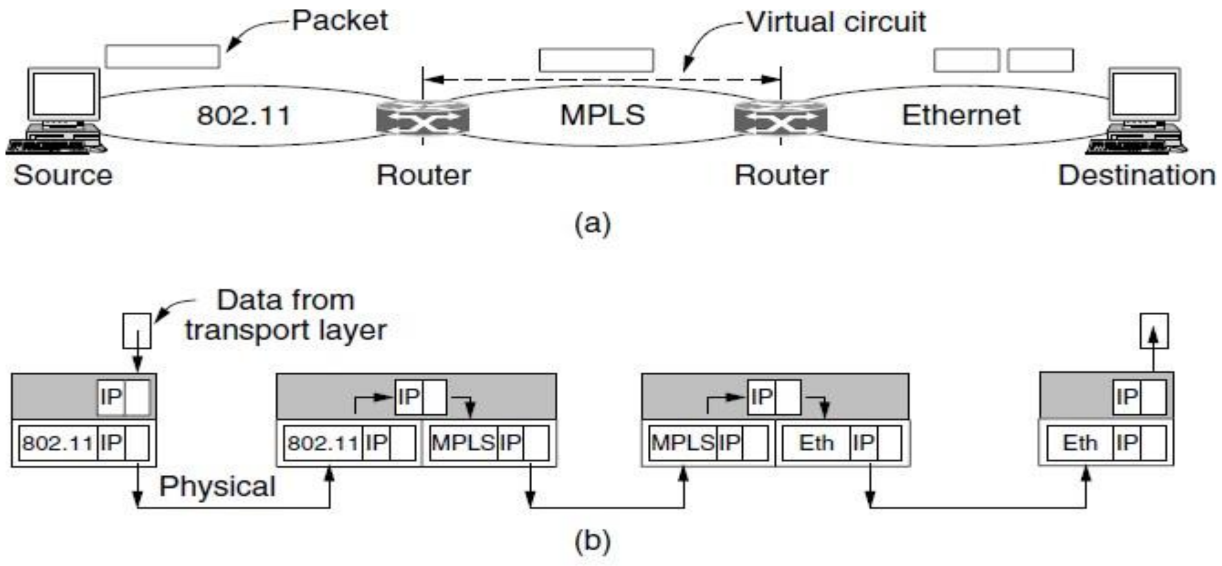


Figure 5-39. (a) A packet crossing different networks. (b) Network and link layer protocol processing.

TUNNELING

1. Handling the general case of making two different networks interwork is exceedingly difficult. However, there is a common special case that is manageable.
2. This case is where the source and destination hosts are on the same type of network, butthere is a different network in between.
3. As an example, think of an international bank with a TCP/IP-based Ethernet in Paris, a TCP/IP-based Ethernet in London, and a non-IP wide area network (e.g., ATM) in between,as shown in Fig. 5-47.

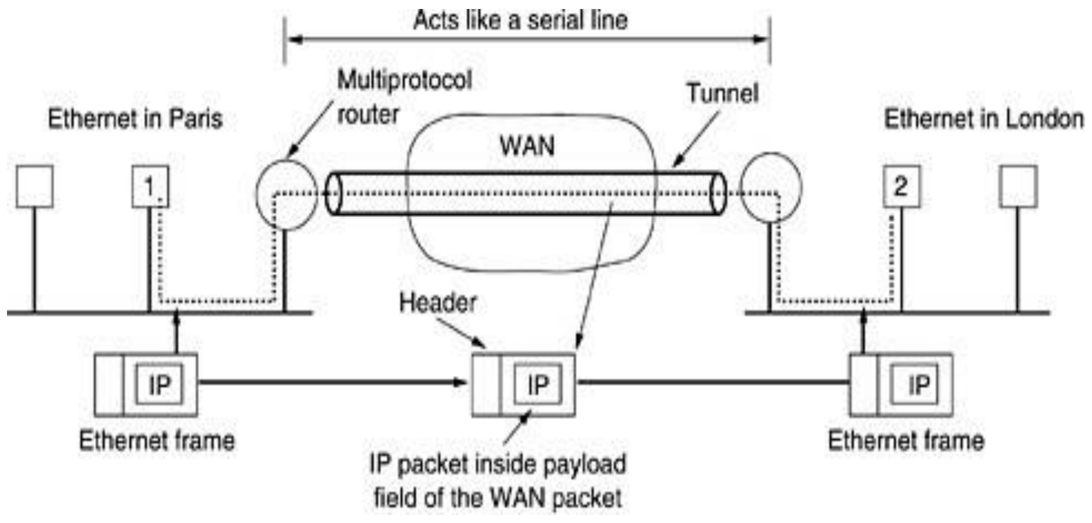


Figure 5-47. Tunneling a packet from Paris to London.

- The solution to this problem is a technique called **tunneling**.
- To send an IP packet to host 2, host 1 constructs the packet containing the IP address of host 2, inserts it into an Ethernet frame addressed to the Paris multiprotocol router, and puts it on the Ethernet. When the multiprotocol router gets the frame, it removes the IP packet, inserts it in the payload field of the WAN network layer packet, and addresses thelatter to the WAN address of the London multiprotocol router. When it gets there, the London router removes the IP packet and sends it to host 2 inside an Ethernet frame.
- The WAN can be seen as a big tunnel extending from one multiprotocol router to the other. The IP packet just travels from one end of the tunnel to the other, snug in its nice box. Neither do the hosts on either Ethernet. Only the multiprotocol router has to understand IP and WAN packets. In effect, the entire distance from the middle of onemultiprotocol router to the middle of the other acts like a serial line.

- Effectively, the car is being carried as freight, as depicted in Fig. 5-48. At the far end, the car is let loose on the English roads and once again continues to move under its own power. Tunneling of packets through a foreign network works the sameway.

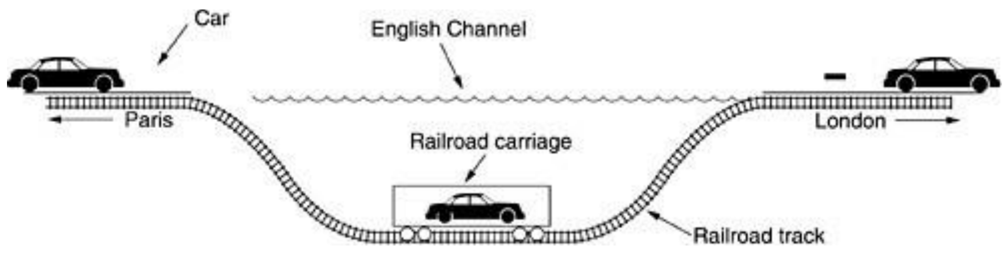


Figure 5-48. Tunneling a car from France to England.

<u>**INTERNETWORK ROUTING**</u>

1. Routing through an internetwork is similar to routing within a single subnet, but with someadded complications.
2. Consider, for example, the internetwork of Fig. 5-49(a) in which five networks are connected by six (possibly multiprotocol) routers. Making a graph model of this situationis complicated by the fact that every router can directly access (i.e., send packets to) everyother router connected to any network to which it is connected. For example, *B* in Fig. 5-49(a) can directly access *A* and *C* via network 2 and also *D* via network 3. This leads to thegraph of Fig. 5-49(b).

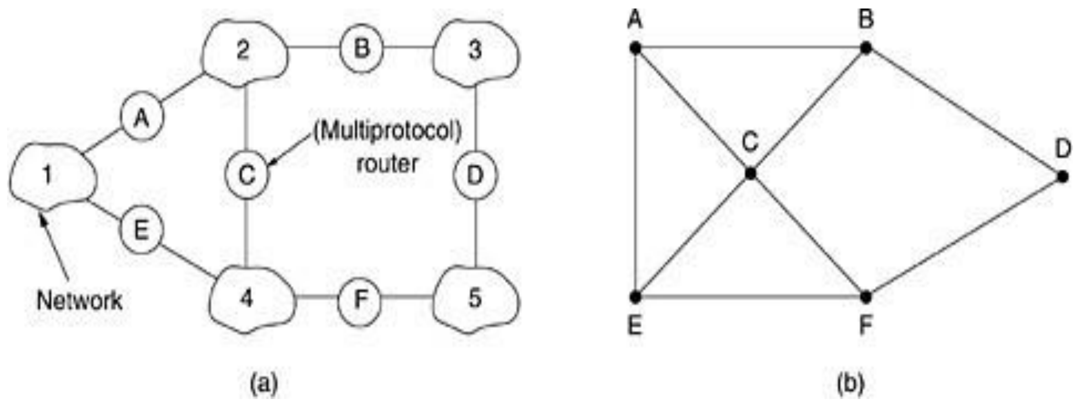


Figure 5-49. (a) An internetwork. (b) A graph of the internetwork.

3. Once the graph has been constructed, known routing algorithms, such as the distance vector and link state algorithms, can be applied to the set of multiprotocol routers.
4. This gives a two-level routing algorithm: within each network an **interior gateway protocol** is used, but between the networks, an **exterior gateway protocol** is used("gateway" is an older term for "router").
5. Network in an internetwork is independent of all the others, it is often referred to as an **Autonomous System** (**AS**).
6. A typical internet packet starts out on its LAN addressed to the local multiprotocol router(in the MAC layer header). After it gets there, the network layer code decides which multiprotocol router to forward the packet to, using its own routing tables. If that router can be reached using the packet's native network protocol, the packet is forwarded theredirectly. Otherwise it is tunneled there, encapsulated in the protocol required by the intervening network. This process is repeated until the packet reaches the destination network.
7. One of the differences between internetwork routing and intranet work routing is that internetwork routing may require crossing international boundaries.

FRAGMENTATION

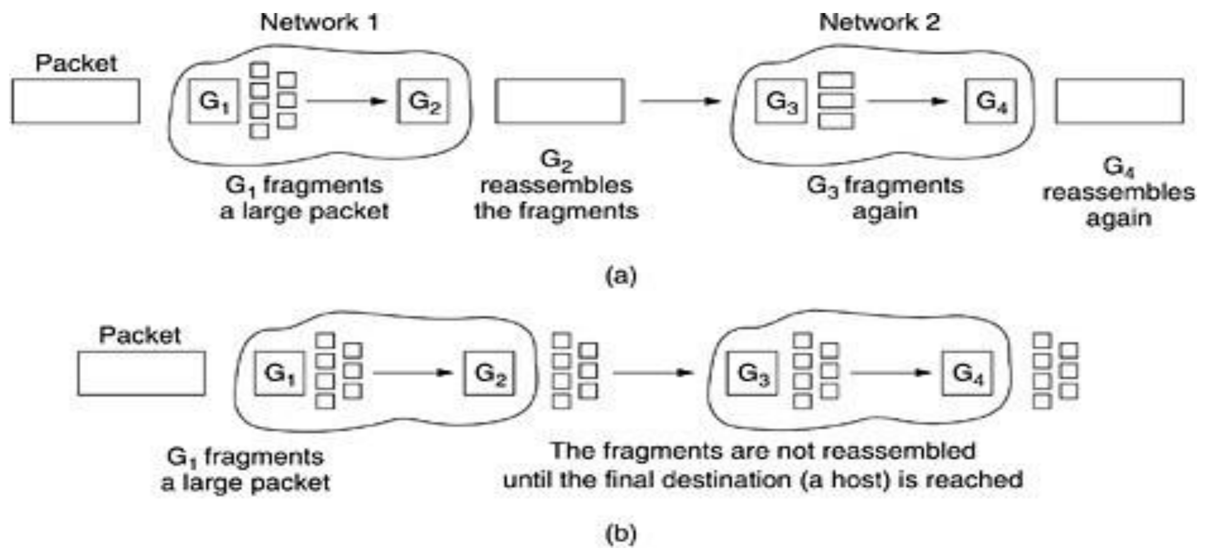
Each network imposes some maximum size on its packets. These limits have various causes, among them:

1. Hardware (e.g., the size of an Ethernet frame).
2. Operating system (e.g., all buffers are 512 bytes).
3. Protocols (e.g., the number of bits in the packet length field).
4. Compliance with some (inter)national standard.
5. Desire to reduce error-induced retransmissions to some level.
6. Desire to prevent one packet from occupying the channel too long.

- From the above factors maximum payloads range from 48 bytes (ATM cells) to 65,515 bytes (IP packets), although the payload size in higher layers is often larger.
- If the original source packet is too large to be handled by the destination network? The routing algorithm can hardly bypass the destination.

- The only solution to the problem is to allow gateways to break up packets into **fragments**,sending each fragment as a separate internet packet. Packet-switching networks, too, have trouble putting the fragments back together again.
- **Two opposing strategies** exist for recombining the fragments back into the original packet.

The **first strategy** is to make fragmentation caused by a "small-packet" network transparent to any subsequent networks through which the packet must pass on its way to the ultimate destination. This option is shown in Fig. 5-50(a). In this approach, the small-packet network has gateways that interface to other networks. When an oversized packet arrives at a gateway, the gateway breaks it up into fragments. Each fragmentis addressed to the same exit gateway, where the pieces are recombined. In this way passage through the small-packet network has been made transparent. Subsequent networks are not even aware that fragmentation has occurred.


Network 1
Network 2
Packet
G_1
G_2
G_3
G_4
G_1 fragments a large packet
G_2 reassembles the fragments
G_3 fragments again
G_4 reassembles again
(a)
Packet
G_1 fragments a large packet
The fragments are not reassembled until the final destination (a host) is reached
(b)
Figure 5-50. (a) Transparent fragmentation. (b) Nontransparent fragmentation.

1. Transparent fragmentation is straightforward but has some problems.
2. For one thing, the exit gateway must know when it has received all the pieces, so either acount field or an "end of packet" bit must be provided.
3. For another thing, all packets must exit via the same gateway. By not allowing some fragments to follow one route to the ultimate destination and other fragments a disjoint route, some performance may be lost.
4. A last problem is the overhead required to repeatedly reassemble and then refragment alarge packet passing through a series of small-packet networks. ATM requires transparentfragmentation.

The **other fragmentation strategy** is to refrain from recombining fragments at any intermediate gateways. Once a packet has been fragmented, each fragment is treated as thoughit were an original packet. All fragments are passed through the exit gateway (or gateways), asshown in Fig. 5-50(b). Recombination occurs only at the destination host. IP works this way.

1. Nontransparent fragmentation also has some problems.
2. For example, it requires *every* host to be able to do reassembly.
3. Yet another problem is that when a large packet is fragmented, the total overhead increases because each fragment must have a header. Whereas in the first method this overhead disappears as soon as the small-packet network is exited, in this method the overhead remains for the rest of the journey.
4. An advantage of nontransparent fragmentation, however, is that multiple exit gateways can now be used and higher performance can be achieved
5. When a packet is fragmented, the fragments must be numbered in such a way that the original data stream can be reconstructed.
6. One way of numbering the fragments is to use a tree. If packet 0 must be split up, the pieces are called 0.0, 0.1, 0.2, etc. If these fragments themselves must be fragmented lateron, the pieces are numbered 0.0.0, 0.0.1, 0.0.2, . . . , 0.1.0, 0.1.1, 0.1.2, etc.
7. No duplicates are generated anywhere, this scheme is sufficient to ensure that all the pieces can be correctly reassembled at the destination, no matter what order they arrivein.
8. However, if even one network loses or discards packets, end-to-end retransmissions are needed, with unfortunate effects for the numbering system.
9. Suppose that a 1024-bit packet is initially fragmented into four equal-sized fragments, 0.0,0.1, 0.2, and 0.3. Fragment 0.1 is lost, but the other parts arrive at the destination. Eventually, the source time out and retransmits the original packet again.
10. When a packet is fragmented, all the pieces are equal to the elementary fragment size except the last one, which may be shorter. An internet packet may contain several fragments, for efficiency reasons.
11. The internet header must provide the original packet number and the number of the (first)elementary fragment contained in the packet.
12. A bit indicating that the last elementary fragment contained within the internet packet isthe last one of the original packet.

This approach requires **two sequence fields** in the **internet header**:

a) **The original packet number**

b) **The fragment number**.

There is clearly a trade-off between the size of the elementary fragment and the number of bits in the fragment number. Because the elementary fragment size is presumed to be acceptable to every network, subsequent fragmentation of an internet packet containing several fragments causes no problem. The ultimate limit here is to have the elementary fragment be a singlebit or byte, with the fragment number then being the bit or byte offset within the originalpacket, as shown in Fig. 5-51.

Some internet protocols take this method even further and consider the entire transmission on a virtual circuit to be one giant packet, so that each fragment contains the absolute byte numberof the first byte within the fragment.

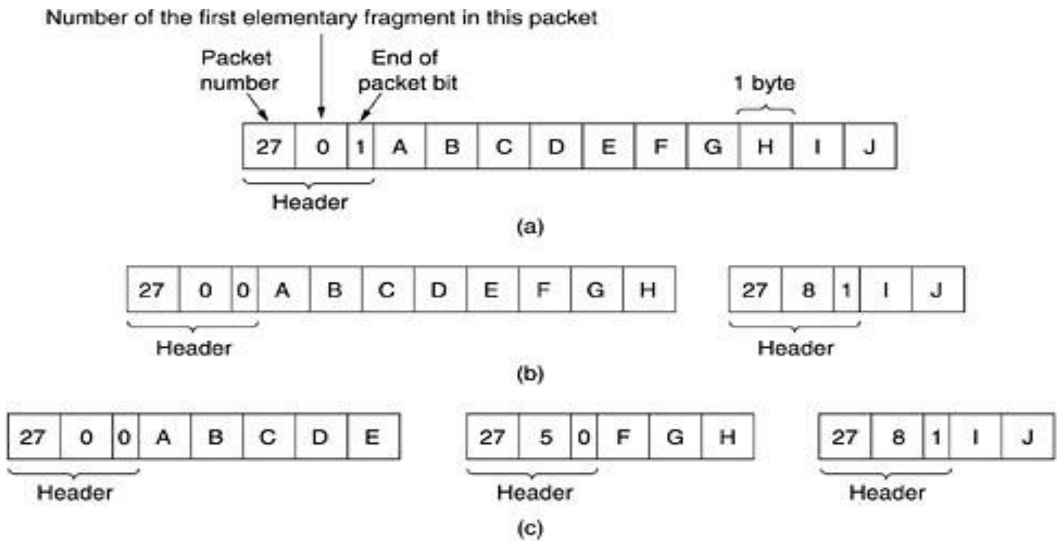


Figure 5-51. Fragmentation when the elementary data size is 1 byte.

(a) Original packet, containing 10 data bytes.

(b) Fragments after passing through a network with maximumpacket size of 8 payload bytes plus header.

(c) Fragments after passing through a size 5 gateway.

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.

Characteristics of IPv4 protocol

1. **IPv4** is an **unreliable** and **connectionless datagram protocol-a best-effort delivery service**.
2. The term ***best-effort*** **means** that IPv4 provides **no error control or flow control** (except for error detection on the header).
3. IPv4 assumes the unreliability of the underlying layers and does its best to get a transmission through to its destination, but with no guarantees.
4. 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.

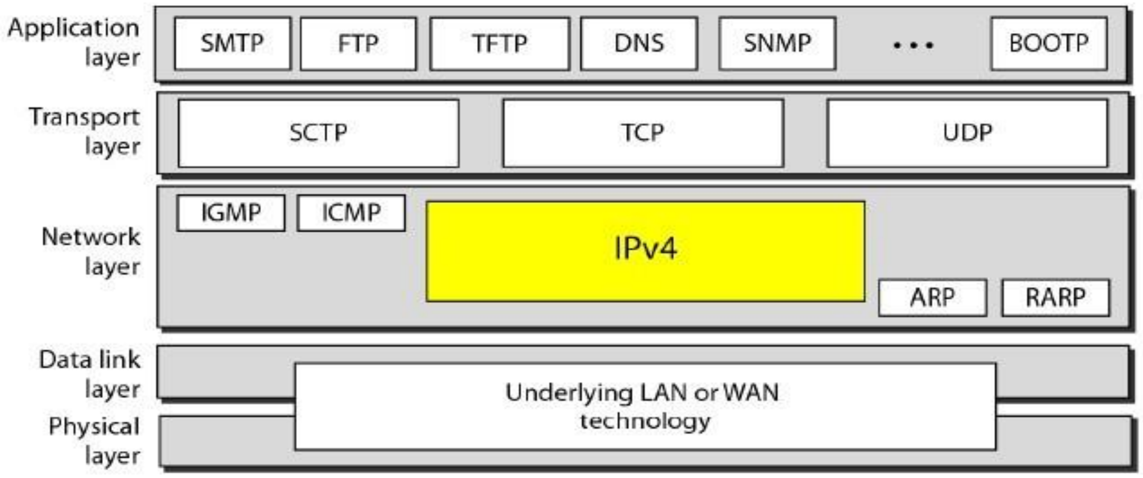


Figure 20.4 Position of IPv4 in TCP/IP protocol suite

1. **IPv4** is also a **connectionless packet-switching** network that uses the datagram **approach**.
2. This means that **each datagram** is **handled independently**, and each datagram can **follow** a **different route to the destination**.
3. This implies that datagram 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.

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:

1. **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.
2. **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).
3. **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.

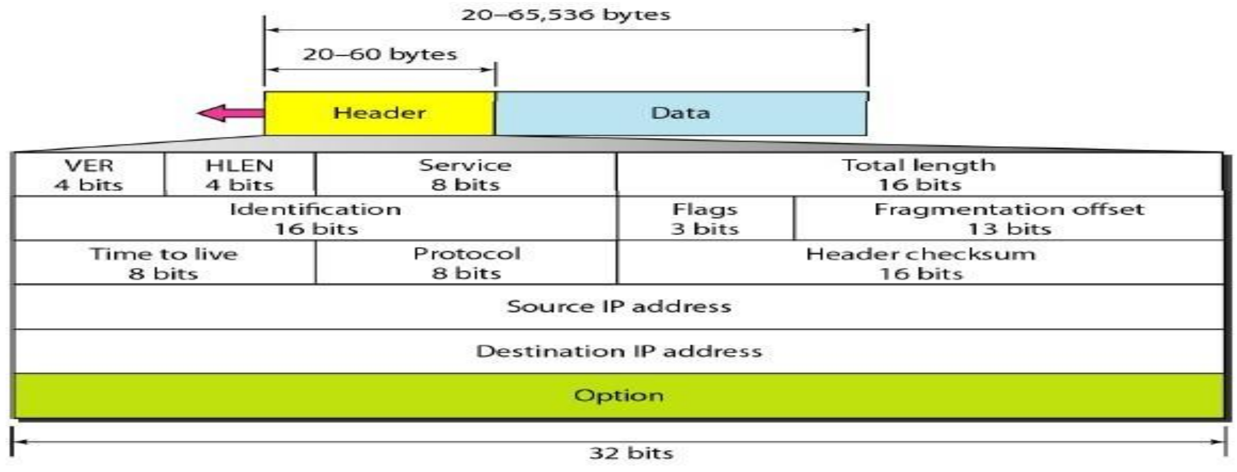


Figure 20.5 IPv4 datagram format

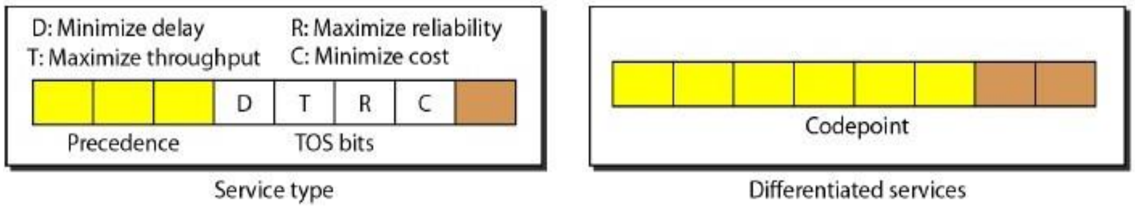


Figure 20.6 Service type or differentiated services

1. **Service Type**

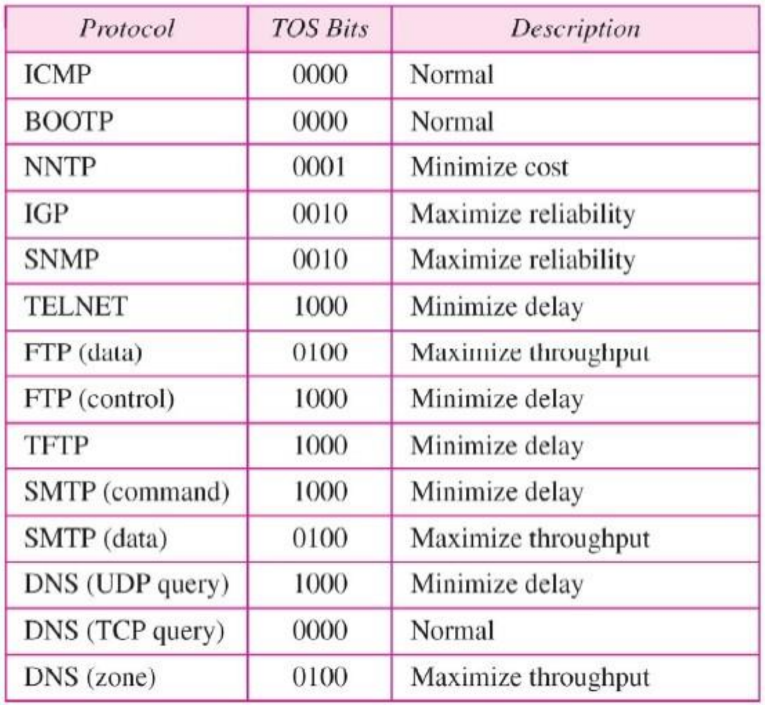
In this interpretation, the **first 3 bits** are called **precedence bits**. The **next 4 bits** are called **type of service (TOS) bits**, and the **last bit** is **not used**.

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.

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

Table 20.1 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

Table 20.2 Default types of service

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.

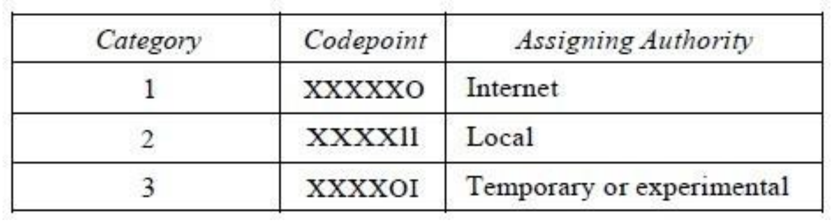
2. Differentiated Services

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:

1. 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.
2. 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	XXXXX0	Internet
2	XXXX11	Local
3	XXXX01	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).

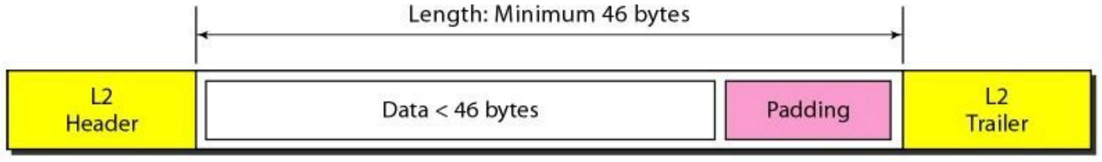


Figure 20.7 Encapsulation of a small datagram in an Ethernet frame

Identification. This field is used in fragmentation.

Flags. This field is used in fragmentation.

Fragmentation offset. This field is used in fragmentation.

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.

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.

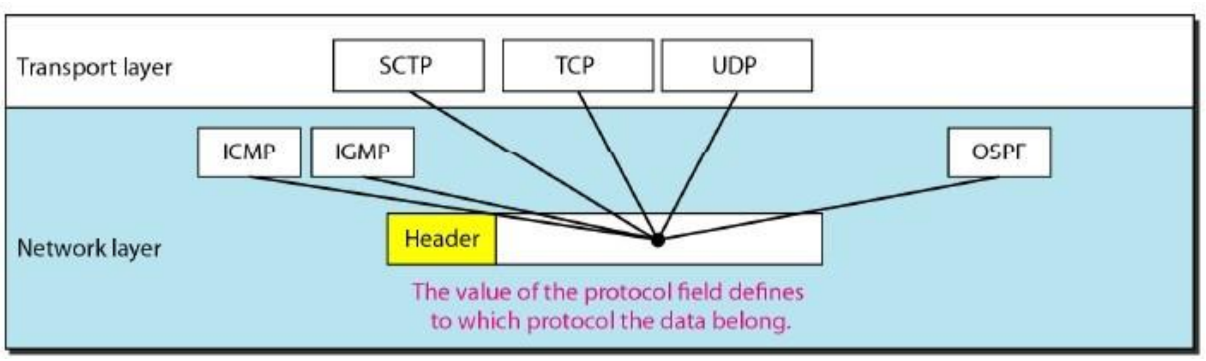

Transport layer
SCTP
TCP
UDP
ICMP
IGMP
OSPF
Network layer
Header
The value of the protocol field defines to which protocol the data belong.

Figure 20.8 Protocol field and encapsulated data

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

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

Table 20.4 Protocol values

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

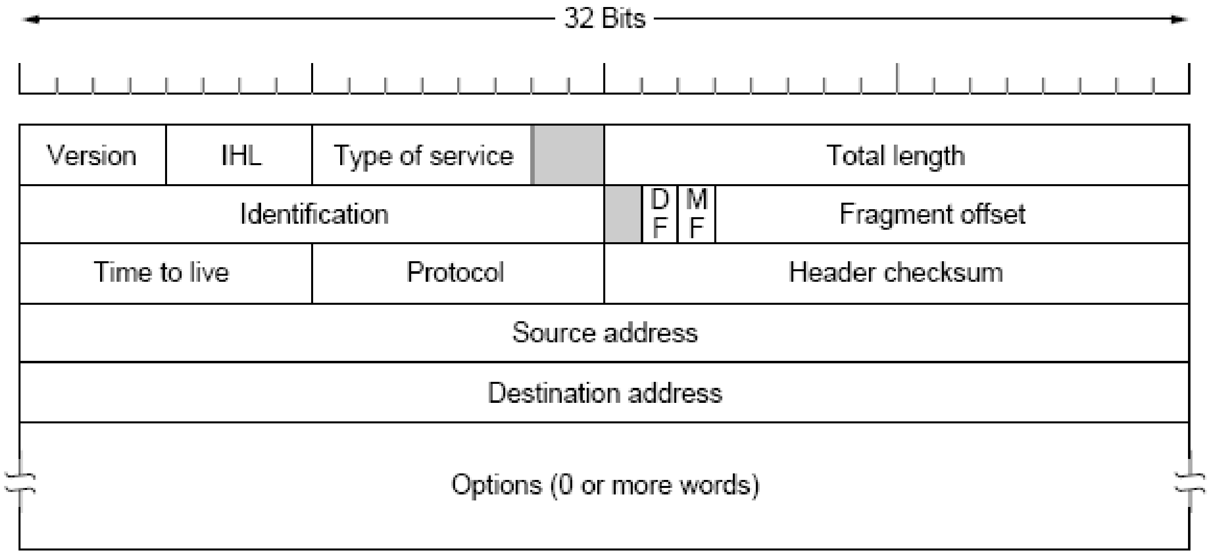
Source address. This **32-bit field** defines the **IPv4 address of the source**. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

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.

THE IP PROTOCOL

An IP datagram consists of a header part and a text part. The header hasa 20-byte fixed part and a variable length optional part.



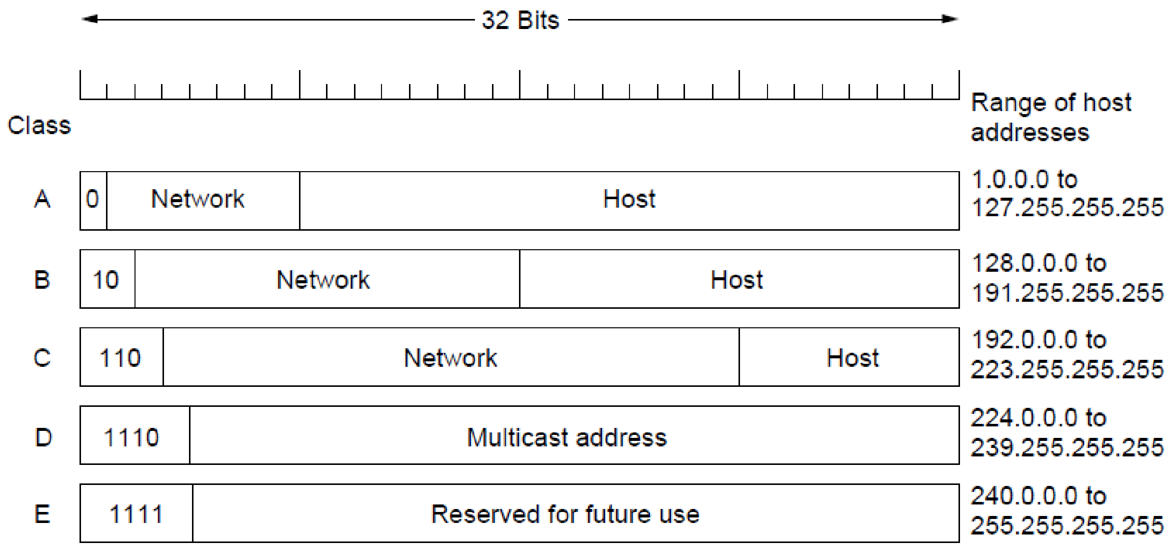
1. The ***Version*** field keeps track of which version of the protocol the datagram belongs to.
2. The header length is not constant, a field in the header, ***IHL***, is provided to tell howlong the header is, in 32-bit words.
3. The ***Type of service*** field is one of the few fields that have changed its meaning (slightly) over the years. It was and is still intended to distinguish between different classes of service.

4. The ***Total length*** includes everything in the datagram—both header and data. Themaximum length is 65,535 bytes.
5. The ***Identification*** field is needed to allow the destination host to determine whichdatagram a newly arrived fragment belongs to. All the fragments of a datagram contain the same *Identification* value. Next comes an unused bit and then two 1- bit fields.
6. ***DF*** stands for Don't Fragment. It is an order to the routers not to fragment the datagram because the destination is incapable of putting the pieces back togetheragain.
7. ***MF*** stands for More Fragments. All fragments except the last one have this bit set.It is needed to know when all fragments of a datagram have arrived.
8. The ***Fragment offset*** tells where in the current datagram this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, theelementary fragment unit. Since 13 bits are provided, there is a maximum of 8192fragments per datagram, giving a maximum datagram length of 65,536 bytes, onemore than the ***Total length*** **field.**
9. The ***Time to live*** field is a counter used to limit packet lifetimes. It is supposed to count time in seconds, allowing a maximum lifetime of 255 sec.
10. When the network layer has assembled a complete datagram, it needs to know what to do with it. The ***Protocol*** field tells it which transport process to give it to. TCP is one possibility, but so are UDP and some others. The numbering of protocolsis global across the entire Internet.
11. The ***Header checksum*** verifies the header only.
12. The ***Source address*** **and** ***Destination address*** indicate the network number and host number.
13. The ***Options*** field was designed to provide an escape to allow subsequent versionsof the protocol to include information not present in the original design, to permitexperimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed.
14. The options are variable length.

IP ADDRESS

IP addresses were divided into the five categories listed in Fig. This allocation has come to be called classful addressing. It is no longer used, but references to it in the literature are still common.

The class A, B, C, and D formats allow for up to 128 networks with 16 million hosts each, 16,384 networks with up to 64K hosts, and 2 million networks (e.g., LANs) with up to 256 hosts each (although a few of these are special). Also supported is multicast, in which a datagram is directed to multiple hosts. Addresses beginning with 1111 are reserved for futureuse.


32 Bits
Class
Range of host addresses
A
0
Network
Host
1.0.0.0 to 127.255.255.255
B
10
Network
Host
128.0.0.0 to 191.255.255.255
C
110
Network
Host
192.0.0.0 to 223.255.255.255
D
1110
Multicast address
224.0.0.0 to 239.255.255.255
E
1111
Reserved for future use
240.0.0.0 to 255.255.255.255

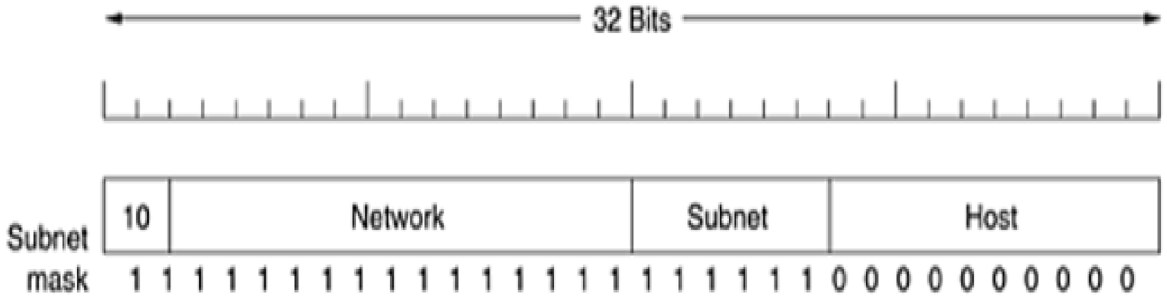
Network addresses, which are 32-bit numbers, are usually written in dotted decimal notation. In this format, each of the 4 bytes is written in decimal, from 0 to 255. For example, the 32-bit hexadecimal address C0290614 is written as 192.41.6.20. The lowest IP address is
0.0.0.0 and the highest is 255.255.255.255.

Subnets:

As we have seen, all the hosts in a network must have the same network number. This property of IP addressing can cause problems as networks grow. The solution is to allow a network to be split into several parts for internal use but still act like a single network to the outside world.

In the Internet literature, the parts of the network (in this case, Ethernets) are called **subnets.**

To implement subnetting, the main router needs a **subnet mask** that indicates the split between network + subnet number and host, as shown in Fig. Subnet masks are also written in dotted decimal notation, with the addition of a slash followed by the number of bits in the network + subnet part. For the example of Fig. the subnet mask can be written as 255.255.252.0. An alternative notation is /22 to indicate that the subnet mask is 22 bits long.



IP VERSION 6 (IPV6)

Internet Protocol version 6 is a new addressing protocol designed to incorporate all the possible requirements of future Internet. This protocol as its predecessor IPv4, works on the Network Layer (Layer-3). Along with its offering of an enormous amount of logical address space, this protocol has ample features to address the shortcoming of IPv4.

The major goals of IPV6 are:

1. Support billions of hosts, even with inefficient address allocation.
2. Reduce the size of the routing tables.
3. Simplify the protocol, to allow routers to process packets faster.
4. Provide better security (authentication and privacy).
5. Pay more attention to the type of service, particularly for real-time data.
6. Aid multicasting by allowing scopes to be specified.
7. Make it possible for a host to roam without changing its address.
8. Allow the protocol to evolve in the future.
9. Permit the old and new protocols to coexist for years.

First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long, which solves the problem that IPv6 set out to solve: providing an effectively unlimited supply of Internet addresses.

The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay.

The third major improvement is better support for options. This change was essential withthe new header because fields that previously were required are now optional (because they arenot used so often). In addition, the way options are represented is different, making it simple for routers to skip over options not intended for them. This feature speeds up packet processingtime.

A fourth area in which IPv6 represents a big advance is in security. Authentication and privacy are key features of the new IP.

The Main IPv6 Header

The IPv6 header is shown in Fig. 5-56. The Version field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routerswill be able to examine this field to tell what kind of packet they have.

The ***Differentiated services*** field (originally called *Traffic class*) is used to distinguish the classof service for packets with different real-time delivery requirements. It is used with the differentiated service architecture for quality of service in the same manner as the field of thesame name in the IPv4 packet.

The ***Flow label*** field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forminga pseudo connection. The flow can be set up in advance and given an identifier.

The ***Payload length*** field tells how many bytes follow the 40-byte header of Fig. 5-56. The name was changed from the IPv4 *Total length* field because the meaning was changed slightly:the 40 header bytes are no longer counted as part of the length (as they used to be). This changemeans the payload can now be 65,535 bytes instead of a mere 65,515 bytes.

The ***Next header*** field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers. This field tells which of the (currently) six extension headers, if any, follow this one.

The ***Hop limit* field** is used to keep packets from living forever. It is, in practice, the same as the *Time to live* field in IPv4, namely, a field that is decremented on each hop.

Next the ***Source address* and *Destination address* fields** which contains 1228 bit address.

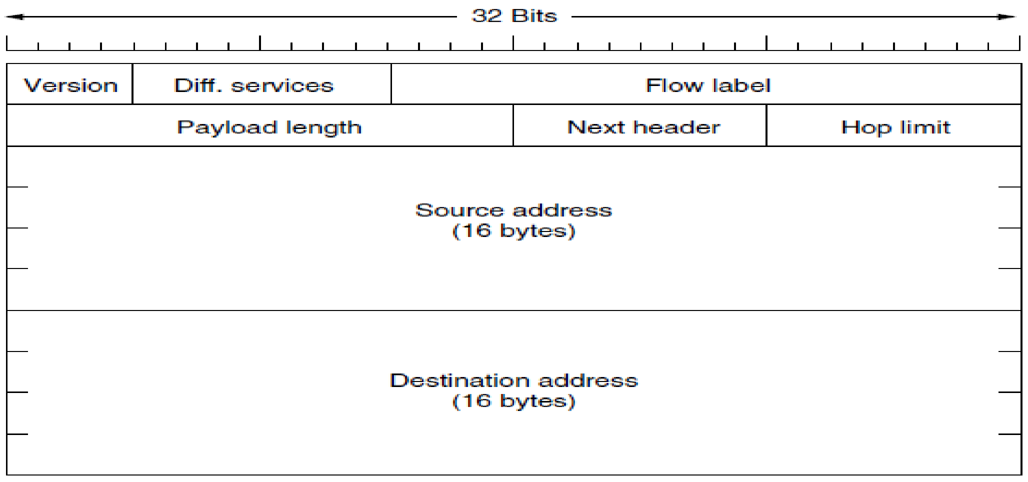


Figure 5-56. The IPv6 fixed header (required).

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this:

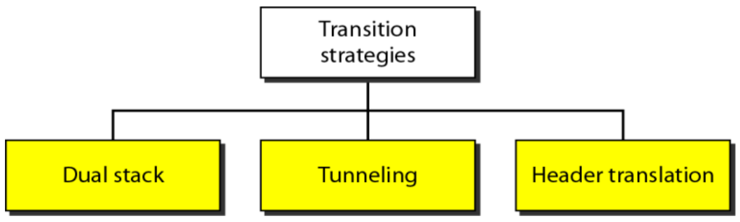
8000:0000:0000:0000:0123:4567:89AB:CDEF

Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted, so 0123 can be written as 123. Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the aboveaddress now becomes

8000::123:4567:89AB:CDEF

<u>Transition from IPV4 to IPV6:</u>

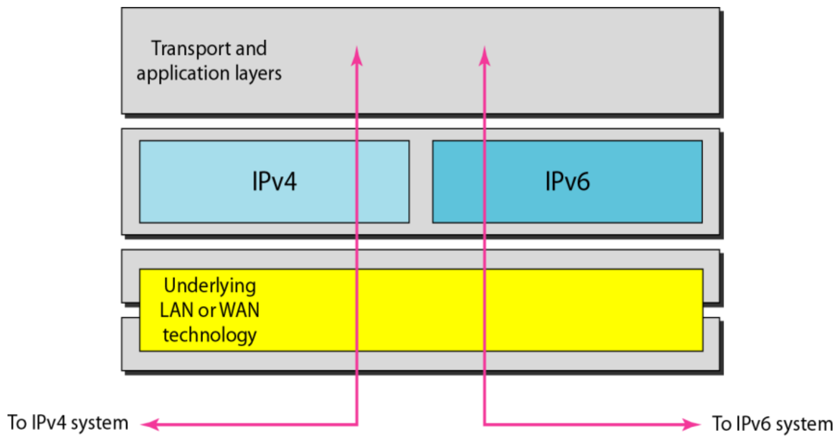
Because of the huge number of systems on the Internet, the transition from IPv4 to IPv6 cannothappen suddenly. It takes a considerable amount of time before every system in the Internet can move from IPv4 to IPv6. The transition must be smooth to prevent any problems betweenIPv4 and IPv6 systems. Three strategies have been devised by the IETF to help the transition.



1. Dual Stack

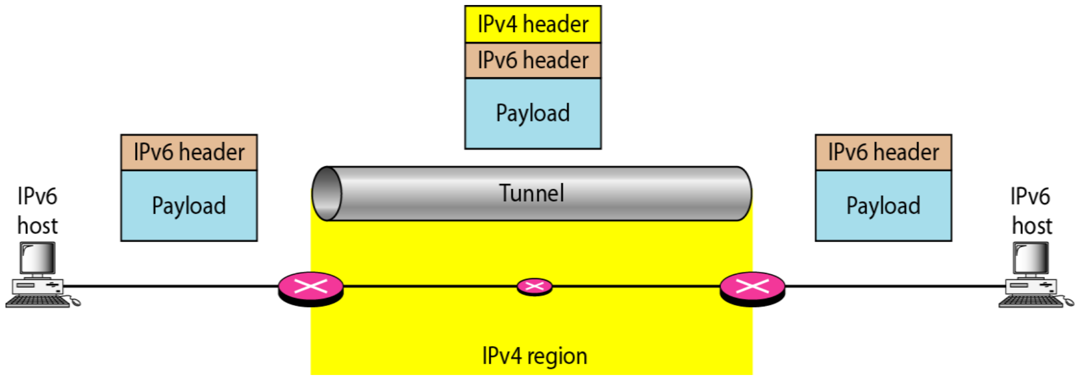
It is recommended that all hosts, before migrating completely to version 6, have a dual stack of protocols. In other words, a station must run IPv4 and IPv6 simultaneously until all the Internet uses IPv6.

To determine which version to use when sending a packet to a destination, the source host queries the DNS. If the DNS returns an IPv4 address, the source host sends an IPv4 packet. If the DNS returns an IPv6 address, the source host sends an IPv6 packet.


Transport and application layers
IPv4
IPv6
Underlying LAN or WAN technology
To IPv4 system
To IPv6 system

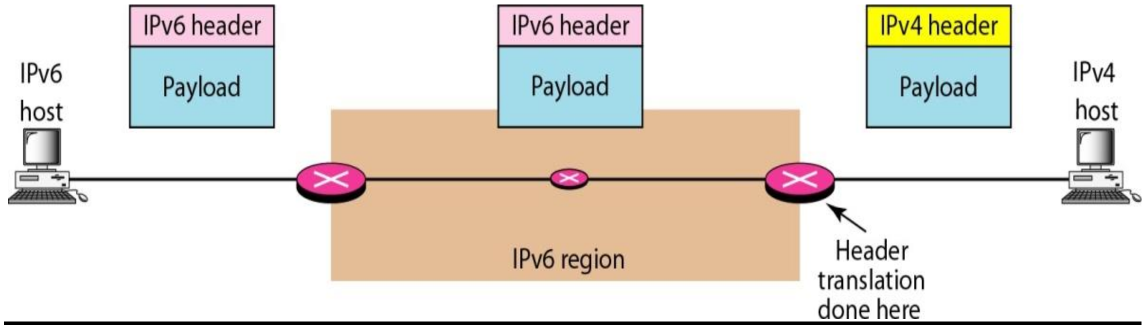
2. Tunneling

Tunneling is a strategy used when two computers using IPv6 want to communicate with each other and the packet must pass through a region that uses IPv4. To pass through this region, the packet must have an IPv4 address. So the IPv6 packet is encapsulated in an IPv4packet when it enters the region, and it leaves its capsule when it exits the region. It seems as if the IPv6 packet goes through a tunnel at one end and emerges at the other end. To makeit clear that the IPv4 packet is carrying an IPv6 packet as data.



3. Header Translation

Header translation is necessary when the majority of the Internet has moved to IPv6 but some systems still use IPv4. The sender wants to use IPv6, but the receiver does not understand IPv6. Tunneling does not work in this situation because the packet must be in the IPv4 format to be understood by the receiver.



Comparison of IPV4 & IPV6

IPv4 Address	**IPv6 Address**
Address Length – 32 bits	**128 bits**
Address Representation - decimal	**hexadecimal**
Internet address classes	**Not applicable in IPv6**
Multicast addresses (224.0.0.0/4)	**IPv6 multicast addresses (FF00::/8)**
Broadcast addresses	**Not applicable in IPv6**
Unspecified address is 0.0.0.0	**Unspecified address is ::**
Loopback address is 127.0.0.1	**Loopback address is ::1**
Public IP addresses	**Global unicast addresses**
Private IP addresses (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16)	**Site-local addresses (FEC0::/10)**
Autoconfigured addresses (169.254.0.0/16)	**Link-local addresses (FE80::/64)**

CIDR—CLASSLESS INTER DOMAIN ROUTING:

The basic idea behind CIDR, which is described in RFC 1519, is to allocate the remaining IP addresses in **variable-sized blocks**, without regard to the classes. If a site needs, say, 2000 addresses, it is given a block of 2048 addresses on a 2048-byte boundary.

Using CIDR, each IP address has a **network prefix** that identifies either one or several network gateways. The length of the network prefix in IPv4 CIDR is also specified as part of the IP address and varies depending on the number of bits needed, rather than any arbitrary class assignment structure. A destination IP address or route that describes many possible destinations has a shorter prefix and is said to be less specific. A longer prefix describes a destination gateway more specifically.

CIDR notation:

$$\boxed{\text{a.b.c.d/n}}$$

Where n = number of network IDsHost ID = 32-n

IP = $2^{(32-n)}$

Example: 20.10.20.100/20

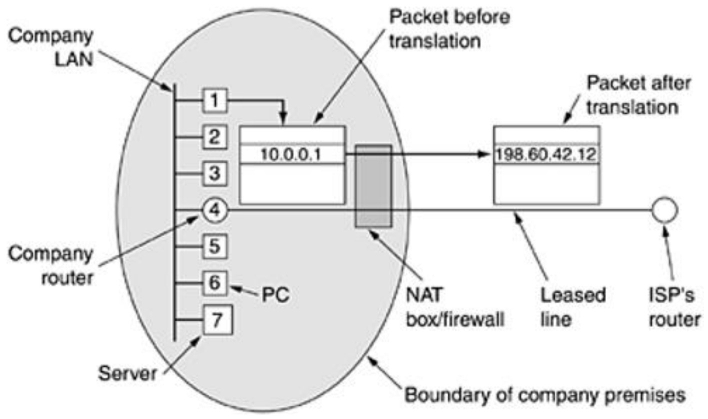
Network ID = 20. Host ID = 32-20 = 12.

IP = $2^{(32-20)}$ = 4096.

NAT—NETWORK ADDRESS TRANSLATION:

IP addresses are scarce. An ISP might have a /16 (formerly class B) address, giving it 65,534 host numbers. If it has more customers than that, it has a problem. This quick fix camein the form of **NAT (Network Address Translation),** which is described in RFC 3022 and which we will summarize below.

The basic idea behind NAT is to assign each company a single IP address (or at most, a small number of them) for Internet traffic. Within the company, every computer gets a unique IP address, which is used for routing intramural traffic. However, when a packet exits the company and goes to the ISP, an address translation takes place. To make this scheme possible, three ranges of IP addresses have been declared as private.



10.0.0.0	–	10.255.255.255/8	(16,777,216 hosts)
172.16.0.0	–	172.31.255.255/12	(1,048,576 hosts)
192.168.0.0	–	192.168.255.255/16	(65,536 hosts)

The operation of NAT is shown in Fig. Within the company premises, every machine has a unique address of the form 10.x.y.z. However, when a packet leaves the company premises, it passes through a NAT box that converts the internal IP source address, 10.0.0.1 in the figure, to the company's true IP address, 198.60.42.12 in this example.

INTERNET CONTROL PROTOCOLS:

In addition to IP, which is used for data transfer, the Internet has several companion controlprotocols that are used in the network layer. They include **ICMP, ARP,** and **DHCP**.

IMCP—THE INTERNET CONTROL MESSAGE PROTOCOL

The operation of the Internet is monitored closely by the routers. When something unexpected occurs during packet processing at a router, the event is reported to the sender by the **ICMP** (**Internet Control Message Protocol**). ICMP is also used to test the Internet. About a dozen types of ICMP messages are defined. Each

ICMP message type is carried encapsulated in an IP packet. The most important ones are listed in Fig. 5-60.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

Figure 5-60. The principal ICMP message types.

The **DESTINATION UNREACHABLE message** is used when the router cannot locate thedestination or when a packet with the *DF* bit cannot be delivered because a ‘‘small-packet’’ network stands in the way.

The **TIME EXCEEDED message** is sent when a packet is dropped because it's *TTL (Time to live)* counter has reached zero. This event is a symptom that packets are looping, or that the counter values are being set too low.

The **PARAMETER PROBLEM message** indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host's IP software or possibly in thesoftware of a router transited.

The **SOURCE QUENCH message** was long ago used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down. It is rarely used anymore because when congestion occurs, these packets tend to add more fuel to the fire andit is unclear how to respond to them.

The **REDIRECT message** is used when a router notices that a packet seems to be routed incorrectly. It is used by the router to tell the sending host to update to a better route.

The **ECHO and ECHO REPLY** messages are sent by hosts to see if a given destination is reachable and currently alive. Upon receiving the ECHO message, the destination is expected to send back an ECHO REPLY message. These messages are used in the **ping** utility that checks if ahost is up and on the Internet.

The **TIMESTAMP REQUEST and TIMESTAMP REPLY** messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility can be used to measure network performance.

The **ROUTER ADVERTISEMENT and ROUTER SOLICITATION messages** are used to let hosts find nearby routers. A host needs to learn the IP address of at least one router tobe able to send packets off the local network.

ARP—THE ADDRESS RESOLUTION PROTOCOL

The address resolution Protocol associates an ip address with physical address. On a typical physical network, such as a LAN, each device on a link is identified by a physical or station address usually imprinted on the network interface card.(NIC)

Physical address have local jurisdiction and can be changed easily. For example, if the NIC on a particular machine fails, the physical address changes. The IP address, on the other hand ,have universal jurisdiction and cannot be changed. ARP is used to find the physical address of thenode when its Internet address is known.

Anytime a host or a router needs to find the physical address of another host on its network, it formats an ARP query packet that includes the IP address and broadcast it over the network. Every host on the network receives and processes the ARP packet, but only the intended recipient recognizes its internet address and sends back its physical address. The host both to its cache memory and to the datagram header, then sends the datagram on its way.

REVERSE ADDRESS RESOLUTION PROTOCOL (RARP)

The RARP allows a host to discover its internet address when it knows only its physical address. The question here is, why do we need RARP? A host is supposed to have its internet address stored on its hard disk.

RARP works much like ARP. The host wishing to retrieve its internet address broadcasts an RARP query packet that contains its physical address to every host on its physical network. A server on the network recognizes the RARP packet and returns the host‘s internet address.

DHCP—THE DYNAMIC HOST CONFIGURATION PROTOCOL

With DHCP, every network must have a DHCP server that is responsible for configuration. When a computer is started, it has a built-in Ethernet or other link layer address embedded in the NIC, but no IP address. Much like ARP, the computer broadcasts a request for an IP address on its network. It does this by using a DHCP DISCOVER packet.

This packet must reach the DHCP server. If that server is not directly attached to the network, the router will be configured to receiveDHCP broadcasts and relay them to the DHCP server, wherever it is located.

When the server receives the request, it allocates a free IP address and sends it to the host in a DHCP OFFER packet (which again may be relayed via the router). To be able to do this workeven when hosts do not have IP addresses, the server identifies a host using its Ethernet address (which is carried in the DHCP DISCOVER packet)

An issue that arises with automatic assignment of IP addresses from a pool is for how longan IP address should be allocated. If a host leaves the network and does not return its IP address to the DHCP server, that address will be permanently lost. After a period of time, many addresses may be lost. To prevent that from happening, IP address assignment may be for a fixed period of time, a technique called **leasing**. Just before the lease expires, the host must ask for a DHCP renewal. If it fails to make a request or the request is denied, the host may no longer use the IP address it was given earlier.