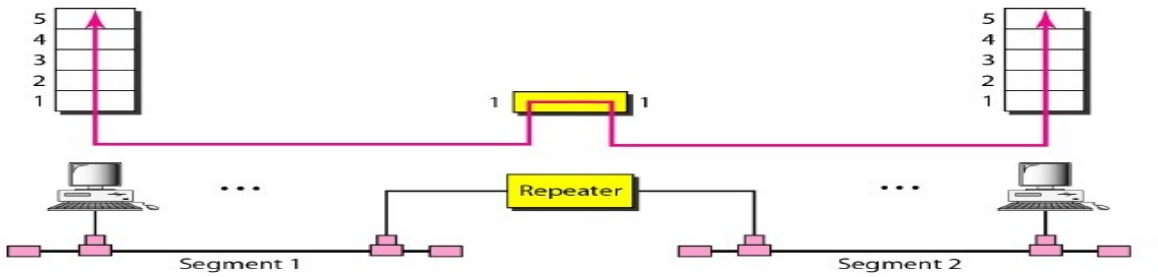
UNIT 4
NETWORK LAYER AND TRANSPORT LAYER
Repeaters, Bridges, gateways and routers

Repeaters

A **repeater** is a device that operates only in the **physical** layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A repeater receives a signal and, before it becomes too weak or corrupted, regenerates the original bit pattern. The repeater then sends the refreshed signal. A repeater can extend the physical length of a LAN, as shown in Figure 15.2.

Figure 15.2 *A repeater connecting two segments of a LAN*



A repeater does not actually connect two LANs; it connects two segments of the same LAN. The segments connected are still part of one single LAN. A repeater is not a device that can connect two LANs of different protocols.

A repeater connects segments of a LAN.

A repeater can overcome the 10Base5 Ethernet length restriction. In this standard, the length of the cable is limited to 500 m. To extend this length, we divide the cable into segments and install repeaters between segments. Note that the whole network is still considered one LAN, but the portions of the network separated by repeaters are called **segments**. The repeater acts as a two-port node, but operates only in the physical layer. When it receives a frame from any of the ports, it regenerates and forwards it to the other port.

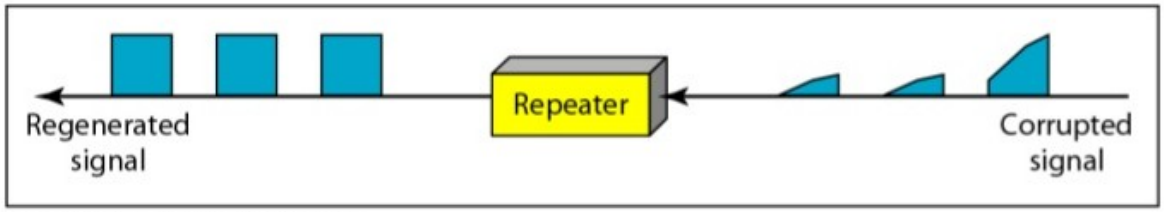
A repeater forwards every frame; it has no filtering capability.

An amplifier cannot discriminate between the intended signal and noise; it amplifies equally everything fed into it. A repeater does not amplify the signal; it regenerates the signal. When it receives a weakened or corrupted signal, it creates a copy, bit for bit, at the original strength.

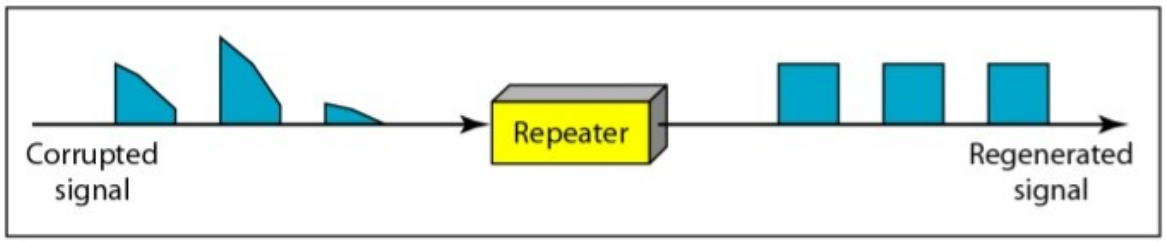
A repeater is a regenerator, not an amplifier.

A repeater must be placed so that a signal reaches it before any noise changes the meaning of any of its bits. A little noise can alter the precision of a bit's voltage without destroying its identity (see Figure 15.3). If the corrupted bit travels much farther, however, accumulated noise can change its meaning completely. At that point, the original voltage is not recoverable, and the error needs to be corrected. A repeater placed on the line before the legibility of the signal becomes lost can still read the signal well enough to determine the intended voltages and replicate them in their original form.

Figure 15.3 *Function of a repeater*



a. Right-to-left transmission.

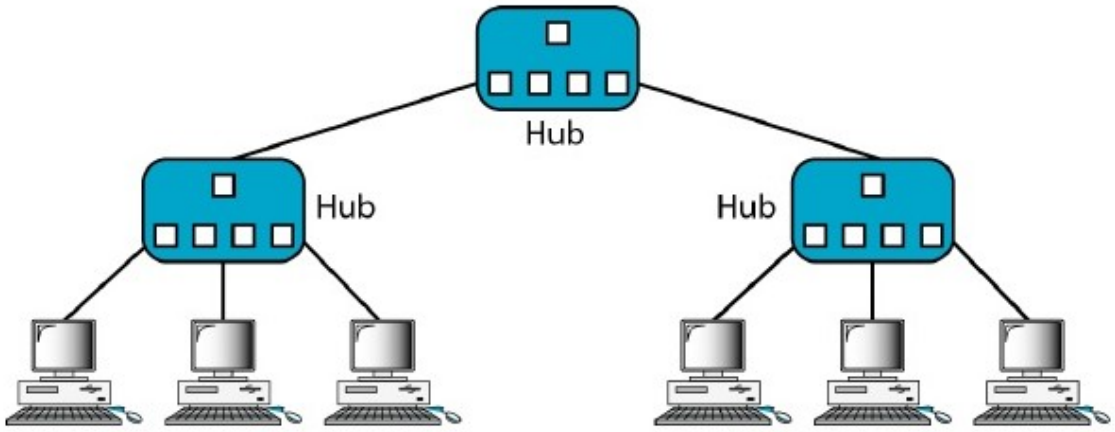


b. Left-to-right transmission.

Bridges

A bridge operates in both the **physical and the data link layer**. As a physical layer device, it regenerates the signal it receives. As a data link layer device, the bridge can check the physical (MAC) addresses (source and destination) contained in the frame.

Figure 15.4 *A hierarchy of hubs*



Filtering

A bridge has filtering capability. It can check the destination address of a frame and decide if the frame should be forwarded or dropped. If the frame is to be forwarded, the decision must specify the port. A bridge has a table that maps addresses to ports. A bridge does not change the physical addresses contained in the frame.

A bridge does not change the physical (MAC) addresses in a frame.

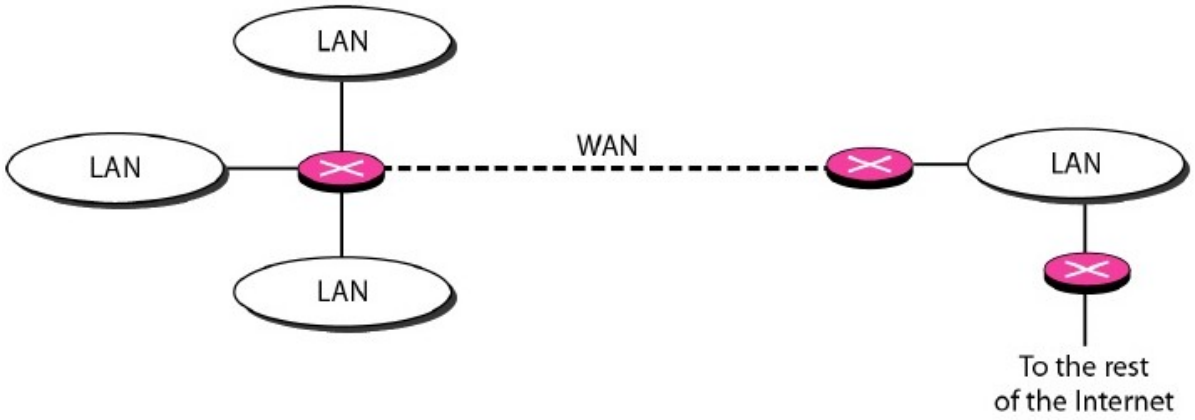
Transparent Bridges

A transparent bridge is a bridge in which the stations are completely unaware of the bridge's existence. If a bridge is added or deleted from the system, reconfiguration of the stations is unnecessary.

Routers

A router is a three-layer device that routes packets based on their logical addresses (host-to-host addressing). A router normally connects LANs and WANs in the Internet and has a routing table that is used for making decisions about the route. The routing tables are normally dynamic and are updated using routing protocols. Figure 15.11 shows a part of the Internet that uses routers to connect LANs and WANs.

Figure 15.11 ***Routers connecting independent LANs and WANs***



Gateway

A gateway is normally a computer that operates in all five layers of the Internet or seven layers of OSI model. A gateway takes an application message, reads it, and interprets it. This means that it can be used as a connecting device between two internetworks that use different models. For example, a network designed to use the OSI model can be connected to another network using the Internet model. The gateway connecting the two systems can take a frame as it arrives from the first system, move it up to the OSI application layer, and remove the message. Gateways can provide security. The gateway is used to filter unwanted application-layer messages.

UNIT 4

LOGICAL ADDRESSING-IPV4 AND IPV6 ADDRESSING, INTERNET PROTOCOL-IPV4 AND IPV6

IPv4 ADDRESSES

An **IPv4** address is a 32-bit address that *uniquely* and *universally* defines the connection of a device (for example, a computer or a router) to the Internet.

IPv4 addresses are unique. They are unique in the sense that each address defines one, and only one, connection to the Internet. Two devices on the Internet can never have the same address at the same time. **The IPv4 addresses are unique and universal.**

Address Space

A protocol such as IPv4 that defines addresses has an address space. An **address space** is the total number of addresses used by the protocol. If a protocol uses *N* bits to define an address, the address space is 2^N because each bit can have two different values (0 or 1) and *N* bits can have 2^N values.

IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than 4 billion). This means that, theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet.

The address space of IPv4 is 2^{32} or 4,294,967,296.

Notations

There are two prevalent notations to show an IPv4 address: binary notation and dotted-decimal notation.

Binary Notation

In binary notation, the IPv4 address is displayed as 32 bits. Each octet is often referred to as a byte. An IPv4 address is referred to as a 32-bit address or a 4-byte address. The following is an example of an IPv4 address in binary notation:

01110101 10010101 00011101 00000010

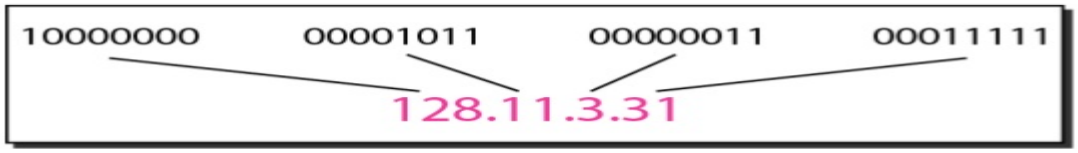
Dotted-Decimal Notation

To make the IPv4 address more compact and easier to read, Internet addresses are usually written in decimal form with a decimal point (dot) separating the bytes. The following is the dotted-decimal notation of the above address:

117.149.29.2

Figure 19.1 shows an IPv4 address in both binary and dotted-decimal notation. Note that because each byte (octet) is 8 bits, each number in dotted-decimal notation is a value ranging from 0 to 255.

Figure 19.1 *Dotted-decimal notation and binary notation for an IPv4 address*

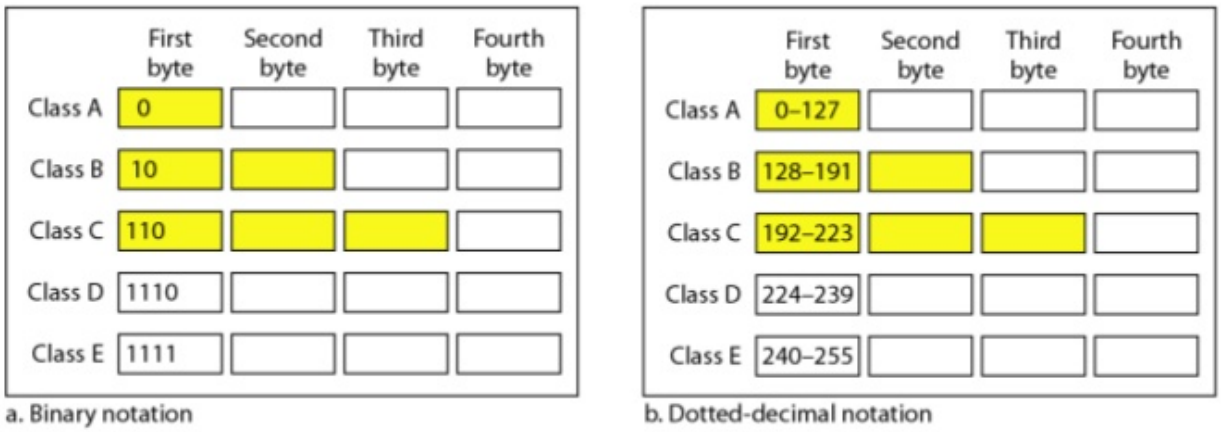


Classful Addressing

IPv4 addressing, at its inception, used the concept of classes. This architecture is called classful addressing. In classful addressing, the address space is divided into five classes: A, B, C, D, and E. Each class occupies some part of the address space.

We can find the class of an address when given the address in binary notation or dotted-decimal notation. If the address is given in **binary notation**, the first few bits can immediately tell us the class of the address. If the address is given in **decimal-dotted notation**, the first byte defines the class. Both methods are shown in Figure 19.2.

Figure 19.2 *Finding the classes in binary and dotted-decimal notation*



Classes and Blocks

Class A addresses were designed for large organizations with a large number of attached hosts or routers. Class B addresses were designed for midsize organizations with tens of thousands of attached hosts or routers. Class C addresses were designed for small organizations with a small number of attached hosts or routers.

A block in class A address is too large for almost any organization. This means most of the addresses in class A were wasted and were not used. A block in class B is also very large, probably too large for many of the organizations that received a class B block. A block in class C is probably too small for many organizations. Class D addresses were designed for multicasting. The class E addresses were reserved for future use.

Netid and Hostid

In classful addressing, an IP address in class A, B, or C is divided into netid and hostid. These parts are of varying lengths, depending on the class of the address. Figure 19.2 shows some netid and hostid bytes. The netid is in color, the hostid is in white. Note that the concept does not apply to classes D and E.

In class A, one byte defines the netid and three bytes define the hostid. In class B, two bytes define the netid and two bytes define the hostid. In class C, three bytes define the netid and one byte defines the hostid.

Mask

Although the length of the netid and hostid (in bits) is predetermined in classful addressing, we can also use a mask (also called the default mask), a 32-bit number made of contiguous Is followed by contiguous as. The masks for classes A, B, and C are shown in Table 19.2. The concept does not apply to classes D and E.

Table 19.2 ***Default masks for classful addressing***

Class	*Binary*	*Dotted-Decimal*	*CIDR*
A	**11111111** 00000000 00000000 00000000	**255**.0.0.0	/8
B	**11111111 11111111** 00000000 00000000	**255.255**.0.0	/16
C	**11111111 11111111 11111111** 00000000	**255.255.255**.0	/24

The mask can help us to find the netid and the hostid. For example, the mask for a class A address has eight 1s, which means the first 8 bits of any address in class A define the netid; the next 24 bits define the hostid.

The last column of Table 19.2 shows the mask in the form */n* where *n* can be 8, 16, or 24 in classful addressing. This notation is also called slash notation or Classless Interdomain Routing (CIDR) notation. The notation is used in classless addressing.

Subnetting

During the era of classful addressing, subnetting was introduced. If an organization was granted a large block in class A or B, it could divide the addresses into several contiguous groups and assign each group to smaller networks (called subnets) or, in rare cases, share part of the addresses with neighbours. Subnetting increases the number of 1s in the mask.

Supernetting

The time came when most of the class A and class B addresses were depleted; however, there was still a huge demand for midsize blocks. The size of a class C block with a maximum number of 256 addresses did not satisfy the needs of most organizations. Even a midsize organization needed more addresses. One solution was supernetting. In supernetting, an organization can combine several class C blocks to create a larger range of addresses. In other words, several networks are combined to create a supernetwork or a super net. An organization can apply for a set of class C blocks instead of just one. For eg, an organization that needs 1000 addresses can be granted 4 contiguous class C blocks. The organization can then use these addresses to create one supernetwork. Supernetting decreases the number of Is in the mask. For example, if an organization is given four class C addresses, the mask changes from /24 to /22.The classless addressing eliminated the need for supernetting.

Classless Addressing

To overcome address depletion and give more organizations access to the Internet, classless addressing was designed and implemented. In this scheme, there are no classes, but the addresses are still granted in blocks.

Address Blocks

In classless addressing, when an entity, small or large, needs to be connected to the Internet, it is granted a block (range) of addresses. The size of the block (the number of addresses) varies based on

the nature and size of the entity. For example, a household may be given only two addresses; a large organization may be given thousands of addresses. An ISP, as the Internet service provider, may be given thousands or hundreds of thousands based on the number of customers it may serve.

Restrictions

To simplify the handling of addresses, the Internet authorities impose three restrictions on classless address blocks:

1. The addresses in a block must be contiguous, one after another.
2. The number of addresses in a block must be a power of 2 (1, 2, 4, 8...).
3. The first address must be evenly divisible by the number of addresses.

Mask

A better way to define a block of addresses is to select any address in the block and the mask. A mask is a 32-bit number in which the *n* leftmost bits are 1s and the 32 - *n* rightmost bits are 0s. However, in classless addressing the mask for a block can take any value from 0 to 32. It is very convenient to give just the value of *n* preceded by a slash (CIDR notation).

In 1Pv4 addressing, a block of addresses can be defined as

x.y.z.t/n

in which x.y.z.t defines one of the addresses and the */n* defines the mask

Network Addresses

A very important concept in IP addressing is the network address. When an organization is given a block of addresses, the organization is free to allocate the addresses to the devices that need to be connected to the Internet.

The first address in the class, however, is normally (not always) treated as a special address. The first address is called the network address and defines the organization network. It defines the organization itself to the rest of the world. The first address is the one that is used by routers to direct the message sent to the organization from the outside.

Network Address Translation (NAT)

The number of home users and small businesses that want to use the Internet is ever increasing. In the beginning, a user was connected to the Internet with a dial-up line, which means that she was connected for a specific period of time. An ISP with a block of addresses could dynamically assign an address to this user. An address was given to a user when it was needed. Home users and small businesses can be connected by an ADSL line or cable modem. In addition, many are not happy with one address; many have created small networks with several hosts and need an IP address for each host.

A quick solution to this problem is called network address translation (NAT**). NAT enables a user to have a large set of addresses internally and one address, or a small set of addresses, externally.** The traffic inside can use the large set; the traffic outside, the small set.

To separate the addresses used inside the home or business and the ones used for the Internet, the Internet authorities have reserved three sets of addresses as private addresses, shown in Table 19.3.

Any organization can use an address out of this set without permission from the Internet authorities. Everyone knows that these reserved addresses are for private networks. They are unique inside the organization, but they are not unique globally. No router will forward a packet that has one of these addresses as the destination address. The site must have only one single connection to the global Internet through a router that runs the NAT software.

Table 19.3 *Addresses for private networks*

Range			Total
10.0.0.0	to	10.255.255.255	2^{24}
172.16.0.0	to	172.31.255.255	2^{20}
192.168.0.0	to	192.168.255.255	2^{16}

IPv6 ADDRESSES

Structure

An IPv6 address consists of 16 bytes (octets); it is 128 bits long.

Hexadecimal Colon Notation

To make addresses more readable, IPv6 specifies hexadecimal colon notation. In this notation, 128 bits is divided into eight sections, each 2 bytes in length. Two bytes in hexadecimal notation requires four hexadecimal digits. Therefore, the address consists of 32 hexadecimal digits, with every four digits separated by a colon.

Although the IP address, even in hexadecimal format, is very long, many of the digits are Zeros. In this case, we can abbreviate the address. The leading zeros of a section (four digits between two colons) can be omitted. Only the leading zeros can be dropped, not the trailing zeros.

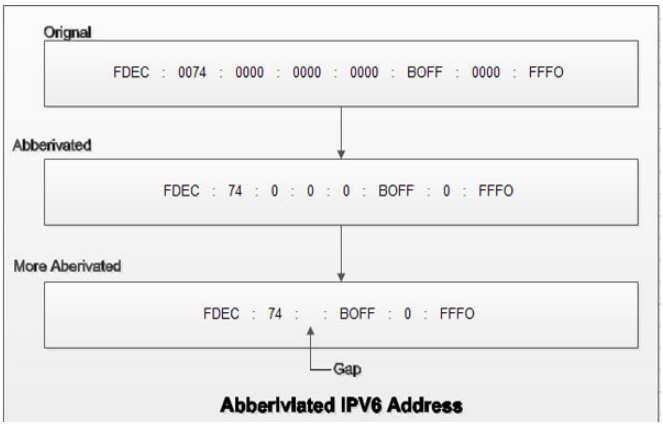


Fig: *Abbreviated IPv6 addresses*

Using this form of abbreviation, 0074 can be written as 74, OOOF as F, and 0000 as O. We can remove the zeros altogether and replace them with a double semicolon. This type of abbreviation is allowed only once per address. If there are two runs of zero sections, only one of them can be abbreviated. Reexpansion of the abbreviated address is very simple: Align the unabbreviated portions and insert zeros to get the original expanded address.

Address Space

IPv6 has a much larger address space; 2^{128} addresses are available. The address can be divided into several categories. A few leftmost bits, called the *type prefix*, in each address define its category. The type prefix is variable in length, but it is designed such that no code is identical to the first part of any other code. Table shows the prefix for each type of address. The third column shows the fraction of each type of address relative to the whole address space.

Type Prefix	Type	Fraction
0000 0000	Reserved	1/256
0000 0001	Unassigned	1/256
0000 001	ISO network addresses	1/128
0000 010	IPX (Novell) network addresses	1/128
0000 011	Unassigned	1/128
0000 1	Unassigned	1/32
0001	Reserved	1/16
001	Reserved	1/8
010	**Provider-based unicast addresses**	**1/8**

Type Prefix	Type	Fraction
010	**Provider-based unicast addresses**	**1/8**
011	Reserved	1/8
100	Geographic unicast addresses	1/8
101	Reserved	1/8
110	Reserved	1/8
1110	Reserved	1/16
1111 0	Reserved	1/32
1111 10	Reserved	1/64
1111 110	Reserved	1/128
1111 1110 0	Reserved	1/512
1111 1110 10	Link local addresses	1/1024
1111 1110 11	Site local addresses	1/1024
1111 1111	Multicast addresses	1/256

Fig: *Type prefixes for 1Pv6 addresses*

Unicast Addresses

A **unicast address** defines a single computer. The packet sent to a unicast address must be delivered to that specific computer.
IPv6 defines two types of unicast addresses:

- Geographically based (left for future definition)
- Provider-based.(used by a normal host)
  The address format of Provider-based is shown in Figure:

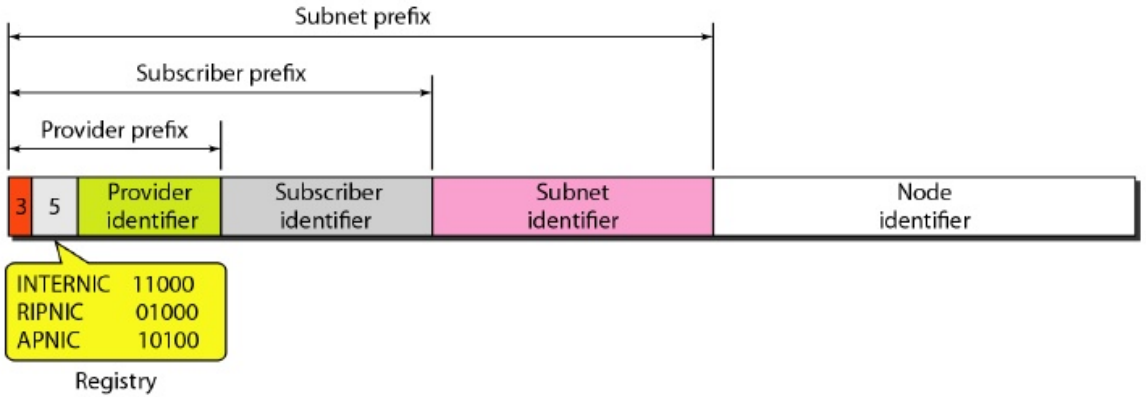


Fig: Prefixes for provider-based unicast address

Fields for the provider-based address are as follows:

Type identifier. This 3-bit field defines the address as a provider-based address.
Registry identifier. This 5-bit field indicates the agency that has registered the address. Currently three registry centers have been defined.

- ✓ INTERNIC (code 11000) is the center for North America;
- ✓ RIPNIC (code 01000) is the center for European registration;
- ✓ APNIC (code 10100) is for Asian and Pacific countries.

Provider identifier. This variable-length field identifies the provider for Internet access (such as an ISP). A 16-bit length is recommended for this field.
Subscriber identifier. When an organization subscribes to the Internet through a provider, it is assigned a subscriber identification. A 24-bit length is recommended for this field.
Subnet identifier. Each subscriber can have many different subnetworks, and each subnetwork can have an identifier. The subnet identifier defines a specific subnetwork under the territory of the subscriber. A 32-bit length is recommended for this field.
Node identifier. The last field defines the identity of the node connected to a subnet. A length of 48 bits is recommended for this field to make it compatible with the 48-bit link (physical) address used by Ethernet.

Multicast Addresses

Multicast addresses are used to define a group of hosts A packet sent to a multicast address must be delivered to each member of the group.
The **second field is a flag** that defines the group address as either permanent or transient.
A *permanent* group address is defined by the Internet authorities and can be accessed at all times.
A *transient* group address, on the other hand, is used only temporarily. Systems engaged in a teleconference, for example, can use a transient group address.
The **third field** defines the scope of the group address.

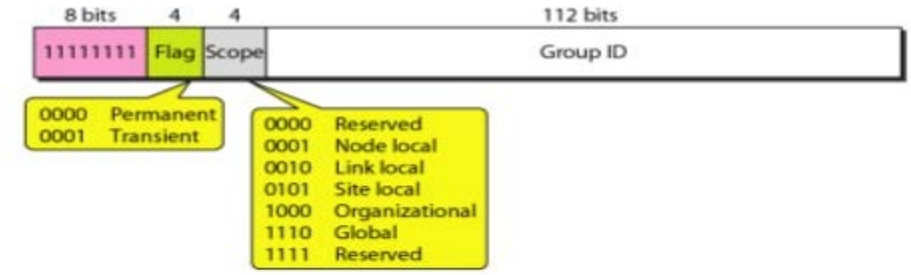


Fig:Multicast address in IPv6

Anycast Addresses

IPv6 also defines anycast addresses. An anycast address, like a multicast address, defines a group of nodes. However, a packet destined for an anycast address is delivered to only one of the members of the anycast group, the nearest one (the one with the shortest route).The routers outside the ISP deliver a packet destined for the ISP to the nearest ISP router. No block is assigned for anycast addresses.

Reserved Addresses

Another category in the address space is the reserved address. These addresses start with eight Os (type prefix is 00000000).

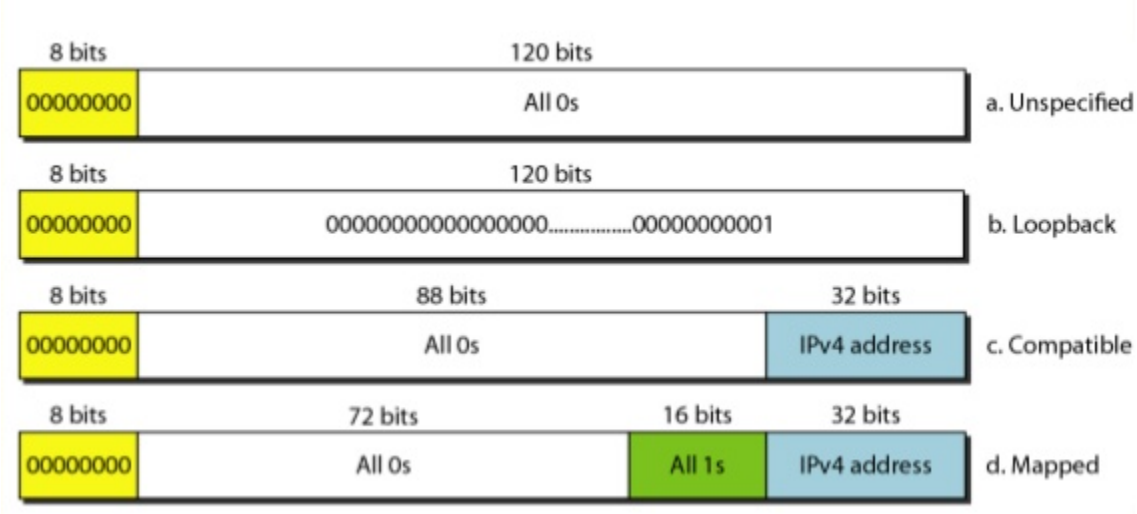


Fig: *Reserved addresses in IPv6*

An **unspecified address** is used when a host does not know its own address and sends an inquiry to find its address.

A loopback address is used by a host to test itself without going into the network.

A **compatible address** is used during the transition from IPv4 to IPv6 .It is used when a computer using IPv6 wants to send a message to another computer using IPv6, but the message needs to pass through a part of the network that still operates in IPv4.
A **mapped address** is also used during transition. However, it is used when a computer that has migrated to IPv6 wants to send a packet to a computer still using IPv4.

Local Addresses
These addresses are used when an organization wants to use IPv6 protocol without being connected to the global Internet. Nobody outside the organization can send a message to the nodes using these addresses. Two types of addresses are defined for this purpose.
A **link local address** is used in an isolated subnet; a **site local address** is used in an isolated site with several subnets.

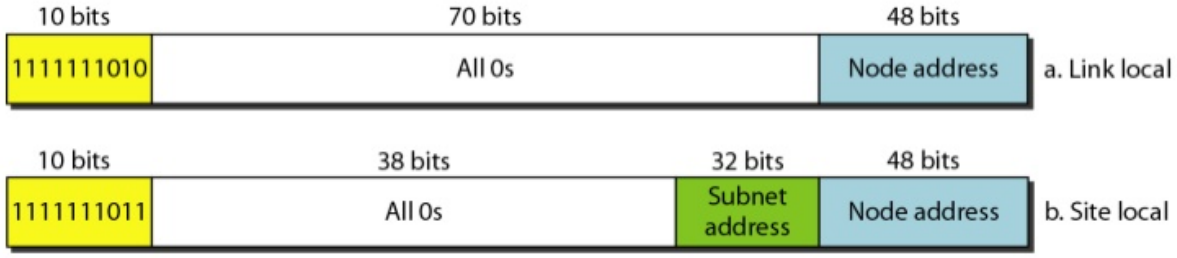


Fig: *Local addresses in IPv6*

IPV4

The Internet Protocol version 4 (IPv4) is the delivery mechanism used by the TCP/IP protocols. IPv4 provides the host-to-host communication betweensystems in the Internet.

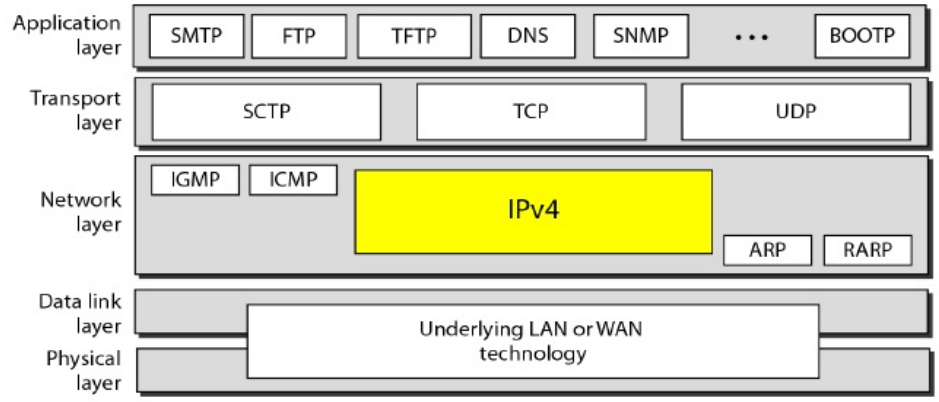


Fig: *Position of IPv4 in TCPIIP protocol suite*

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**.IPv4 assumes the

unreliability of the underlying layers and does its best to get a transmission through to its destination.
IPv4 is also **a connectionless protocol** for a packet-switching network that uses the datagram approach. 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. Again, IPv4 relies on a higher-level protocol to take care of all these problems.

Datagram

Packets in the IPv4 layer are called **datagrams**. Figure shows the IPv4 datagram format.
A datagram is a variable-length packet consisting of two parts:

- header
- 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

- **Version (VER).** This 4-bit field defines the version of the IPv4 protocol. Currently the version is 4.
- **Header length (HLEN).** This 4-bit field defines the total length of the datagram header in 4-byte words. The length of the header is variable (between 20 and 60 bytes).
- **Services**. IETF has changed the interpretation and name of this 8-bit field. This field, previously called service type, is now called differentiated services.

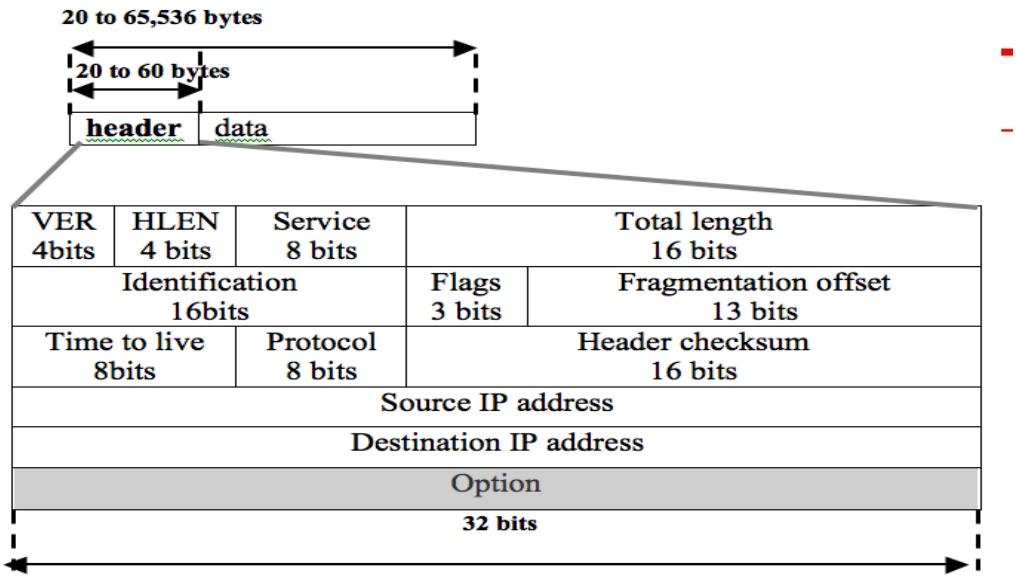
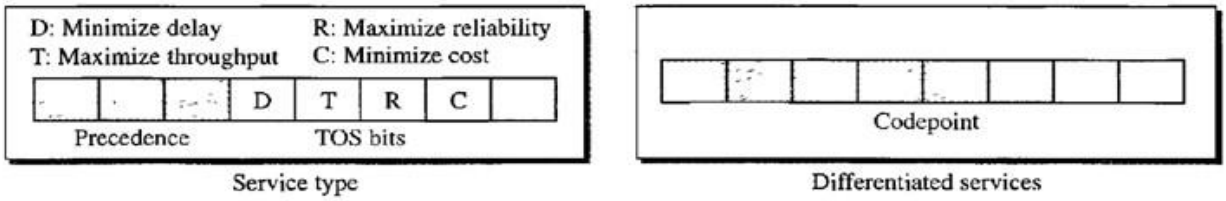


Fig: *IPv4 datagram format*



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 datagram, those datagrams with lowest precedence are discarded first.

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.

With only 1 bit set at a time, we can have five different types of services

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

Fig: Types of service

Application programs can request a specific type of service. The defaults for some applications are shown in Table

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

Note: Activities requiring immediate attention, and activities requiring immediate response need minimum delay. Those activities that send bulk data require maximum throughput. Management activities need maximum reliability. Background activities need minimum cost.

Fig: Default types of service

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.

a. When **the 3 rightmost bits are Os**, the 3 leftmost bits are interpreted the same as the precedence bits in the service type interpretation.
b. When the 3 rightmost bits are not all Os, the 6 bits define 64 services based on the priority assignment by the Internet or local authorities.

Total length. This is a In-bit field that defines the total length (header plus data) of the IPv4 datagram in bytes.

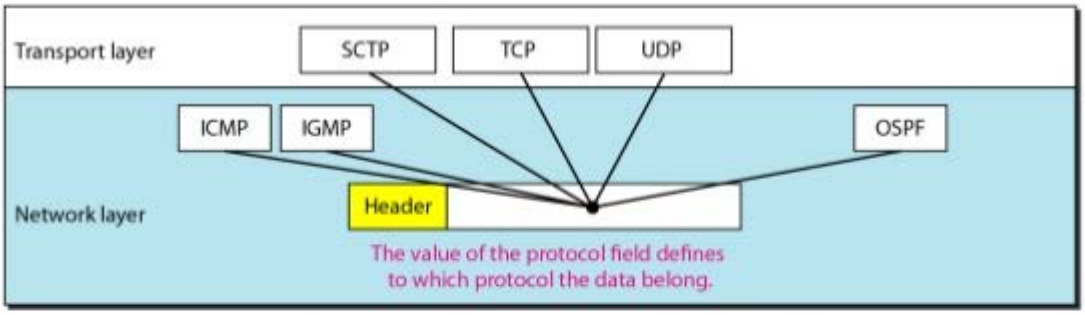
Identification. This field is used in fragmentation

Flags. This field is used in fragmentation

Fragmentation offset. This field is used in fragmentation

Time to live. A datagram has a limited lifetime in its travel through an internet. This field was originally designed to hold a timestamp, which was decremented by each visited router. The datagram was discarded when the value became zero.

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.



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

Fig: Protocol field and encapsulated data

The value of this field for each higher-level protocol is shown in Table above:

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.

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

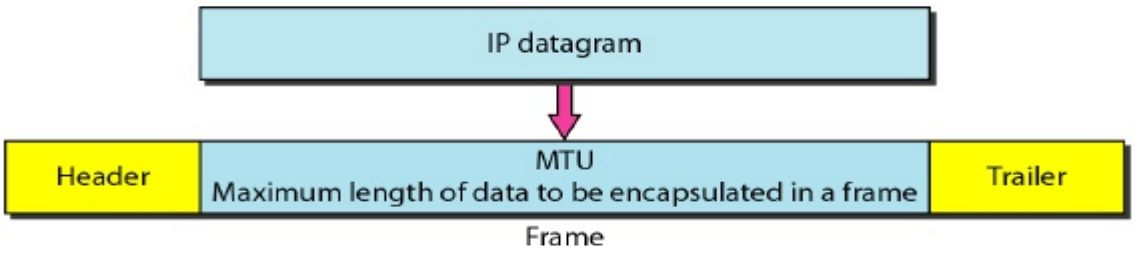


Fig: *Maximum transfer unit (MTU)*

MTUs for some networks are shown in table:

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

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.**

The transport layer segments the data into a size that can be accommodated by IPv4 and the data link layer in use. When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but with some changed. 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. The reassembly of the datagram is done only by the destination host because each fragment becomes an independent datagram .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 the efficiency will be lost.

When a datagram is fragmented, required parts of the header must be copied by all fragments. The option field mayor may not be copied.

The host or router that fragments a datagram must change the values of three fields:

- Flags
- Fragmentation offset
- Total length

Fields Related to Fragmentation

Identification. This 16-bit field identifies a datagram originating from the source host. When a datagram is fragmented, the value in the identification field is copied to all fragments. 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 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 fragment

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.

Checksum

The implementation of the checksum in the IPv4 packet follows the same principles.
First, the value of the checksum field is set to O. Then the entire header is divided into 16-bit sections and added together. The result (sum) is complemented and inserted into the checksum field.

The checksum in the IPv4 packet covers only the header, not the data. There are two reasons for this.

1) All higher-level protocols that encapsulate data in the IPv4 datagram have a checksum field that covers the whole packet. Therefore, the checksum for the IPv4 datagram does not have to check the encapsulated data.

2) The header of the IPv4 packet changes with each visited router, but the data do not. So the checksum includes only the part that has changed. If the data were included, each router must recalculate the checksum for the whole packet, which means an increase in processing time.

Options

The header of the IPv4 datagram is made of two parts:

- a fixed part (20 bytes long)
- a variable part.(maximum of 40 bytes)

Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging. Although options are not a required part of the IPv4 header, option processing is required of the IPv4 software. This means that all implementations must be able to handle options if they are present in the header.

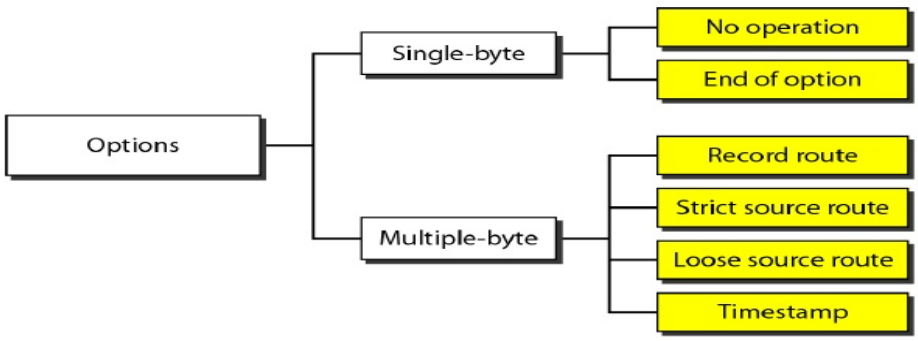


Fig: Taxonomy of options in IPv4

No Operation

A **no-operation option** is a 1-byte option used as filler between options.

End of Option

An end-of-option option is a I-byte option used for padding at the end of the option field. It, however, can only be used as the last option.

Record Route

A record route option is used to record the Internet routers that handle the datagram. It can list up to nine router addresses. It can be used for debugging and management purposes.

Strict Source Route
A strict source route option is used by the source to predetermine a route for the datagram as it travels through the Internet.

Loose Source Route
A loose source route option is similar to the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.

Timestamp
A timestamp option is used to record the time of datagram processing by a router.

IPv6

IPv4 has some deficiencies that make it unsuitable for the fast-growing Internet.

- Despite all short-term solutions, such as subnetting, classless addressing, and NAT, address depletion is still a long-term problem in the Internet.
- The Internet must accommodate real-time audio and video transmission. This type of transmission requires minimum delay strategies and reservation of resources not provided in the IPv4 design.
- The Internet must accommodate encryption and authentication of data for some applications. No encryption or authentication is provided by IPv4.

To overcome these deficiencies, IPv6 (Internetworking Protocol, version 6), also known as **IPng** (Internetworking Protocol, next generation), was proposed and is now a standard. The format and the length of the IP address were changed along with the packet format.

Advantages

The next-generation IP, or IPv6, has some advantages over IPv4.

1. **Larger address space**. An IPv6 address is 128 bits long. Compared with the 32-bit address of IPv4, this is a huge (2^{96}) increase in the address space.

2. **Better header format**. IPv6 uses a new header format in which options are separated from the base header and inserted, when needed, between the base header and the upper-layer data. This simplifies and speeds up the routing process because most of the options do not need to be checked by routers.

3. **New options**. IPv6 has new options to allow for additional functionalities.

4. **Allowance for extension.** IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.

5. **Support for resource allocation**. In IPv6, the type-of-service field has been removed, but a mechanism (called jlow *label)* has been added to enable the source
to request special handling of the packet. This mechanism can be used to support traffic such as real-time audio and video.

6. **Support for more security.** The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

Packet Format

The IPv6 packet is shown in Figure. Each packet is composed of a **mandatory base header followed by the payload**. The base header occupies 40 bytes, whereas payload can be upto 65,535 bytes of information. The payload consists of two parts:

- optional extension headers
- data from an upper layer

Base Header

Figure shows the base header with its eight fields. These fields are as follows:

- **Version**. This 4-bit field defines the version number of the IP. For IPv6, the value is 6.
- **Traffic class.** The 8-bit traffic class field is used to distinguish different payloads with different delivery requirements. It replaces the types of service field in IPv4.

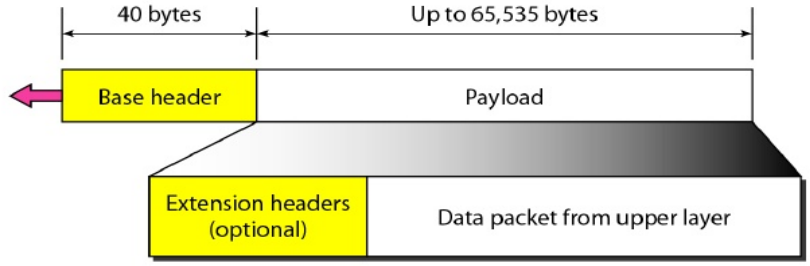


Fig: *IPv6 datagram header and payload*

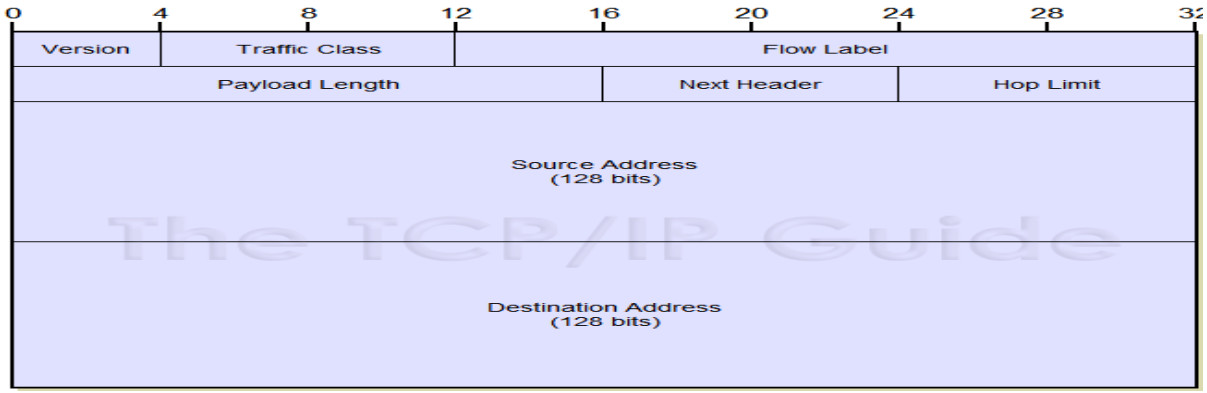


Fig:Base Header

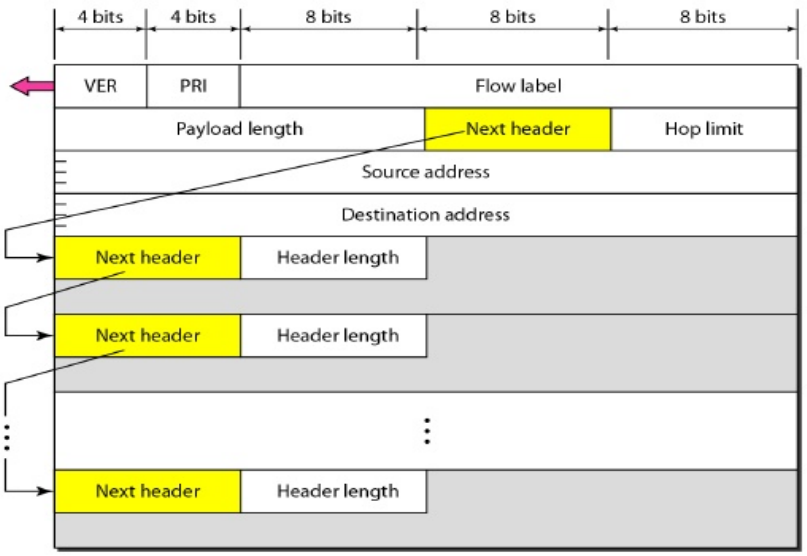


Fig*: Format of an IPv6 datagram*

- **Flow label**. The flow label is a 3-byte (24-bit) field that is designed to provide special handling for a particular flow of data.
- **Payload length**. The 2-byte payload length field defines the length of the IP datagram excluding the base header.
- **Next header.** The next header is an 8-bit field defining the header that follows the base header in the datagram. The next header is either one of the optional extension headers used by IP or the header of an encapsulated packet such as UDP or TCP. Each extension header also contains this field.
  Table shows the values of next headers. This field in version 4 is called the *protocol.*

Code	*Next Header*
0	Hop-by-hop option
2	ICMP
6	TCP
17	UDP
43	Source routing
44	Fragmentation
50	Encrypted security payload
51	Authentication
59	Null (no next header)
60	Destination option

Fig: Next header codes for IPv6

- **Hop limit**. This 8-bit hop limit field serves the same purpose as the TTL field in IPv4.
- **Source address.** The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram.

- **Destination address.** The destination address field is a 16-byte (128-bit) Internet address that usually identifies the final destination of the datagram. If source routing is used, this field contains the address of the next router.

Priority

If one of two consecutive datagrams must be discarded due to congestion, the datagram with the lower **packet priority** will be discarded. IPv6 divides traffic into two broad categories: congestion-controlled and non congestion-controlled.

Congestion-Controlled Traffic

If a source adapts itself to traffic slowdown when there is congestion, the traffic is referred to as **congestion-controlled traffic.** In congestion-controlled traffic, it is understood that packets may arrive delayed, lost, or out of order. Congestion-controlled data are assigned priorities from 0 to 7

Non congestion-Controlled Traffic

This refers to a type of traffic that expects minimum delay. Discarding of packets is not desirable. Retransmission in most cases is impossible. In other words, the source does not adapt itself to congestion. Real-time audio and video are examples of this type of traffic.
Priority numbers from 8 to 15 are assigned to non congestion-controlled traffic.

Table 20.7 *Priorities for congestion-controlled traffic*

Priority	*Meaning*
0	No specific traffic
1	Background data
2	Unattended data traffic
3	Reserved
4	Attended bulk data traffic
5	Reserved
6	Interactive traffic
7	Control traffic

Extension Headers

The length of the base header is fixed at 40 bytes. However, to give greater functionality to the IP datagram, the base header can be followed by up to six extension headers. Many of these headers are options in IPv4. Six types of extension headers have been defined,

Hop-by-Hop Option

The hop-by-hop option is used when the source needs to pass information to all routers visited by the datagram. Three options have been defined:
The **Padl** option is 1 byte long and is designed for alignment purposes.
PadN is similar in concept to PadI. The difference is that PadN is used when 2 or more bytes is needed for alignment.
The **jumbo payload** option is used to define a payload longer than 65,535 bytes.

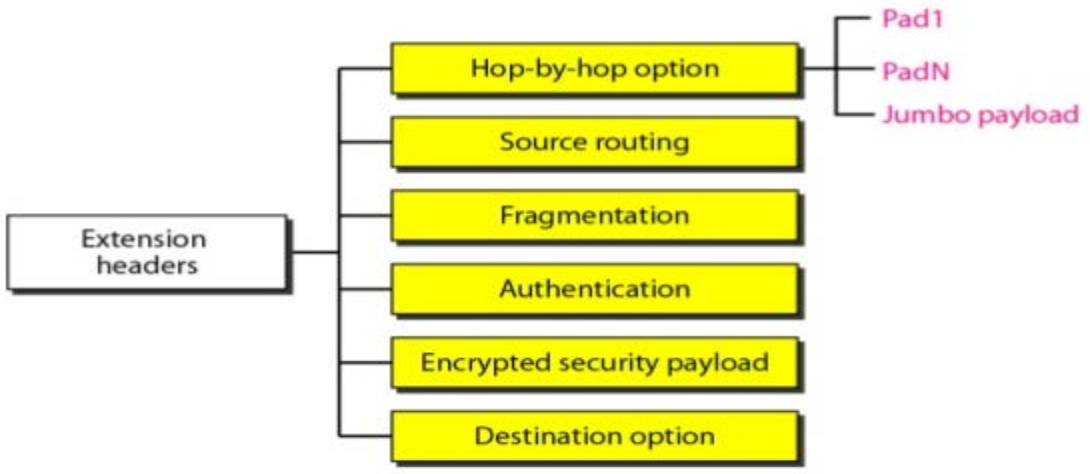


Fig: Extension header types

Source Routing

The source routing extension header combines the concepts of the strict source route and the loose source route options of IPv4.

Fragmentation

The concept of fragmentation is the same as that in IPv4.

Authentication

The authentication extension header has a dual purpose: it validates the message sender and ensures the integrity of data.

Encrypted Security Payload

The encrypted security payload (ESP) is an extension that provides confidentiality and guards against eavesdropping.

Destination Option The destination option is used when the source needs to pass information to the destination only. Intermediate routers are not permitted access to this information.

Comparison between IPv4 and IPv6 Headers

Comparison
1. The header length field is eliminated in IPv6 because the length of the header is fixed in this version.
2. The service type field is eliminated in IPv6. The priority and flow label fields together take over the function of the service type field.
3. The total length field is eliminated in IPv6 and replaced by the payload length field.
4. The identification, flag, and offset fields are eliminated from the base header in IPv6. They are included in the fragmentation extension header.
5. The TTL field is called hop limit in IPv6.
6. The protocol field is replaced by the next header field.
7. The header checksum is eliminated because the checksum is provided by upper layer protocols; it is therefore not needed at this level.
8. The option fields in IPv4 are implemented as extension headers in IPv6.

Comparison between IPv4 and IPv6

IPv4	IPv6
The Adress Space is 32 bits.	The space is 128 bits.
The length of header is 20 bytes	The length of header is 40
4 bytes for each adress in the header	16 bytes for each adressin the header
The number of Header field 12	The number of header field 8
Checksum field, used to measure error in the header,required	Chicksum field eliminated from header as error in the IP header are ot very crucial
Internet Protocol Security (IPSec) with repect to network security is optional	Internet Protocol Secuirty (IPSec) With respect to net work secuirty is mandatory
No identification to the packet flow (Lack of QoS handing).	The flow level field on the header portion identifies the packet flow and directs to router (Efficient QoS handling)
The Fargmentation is done both by sending host and routers	The framentation is done both by sending hoost; there is no role of the routers.

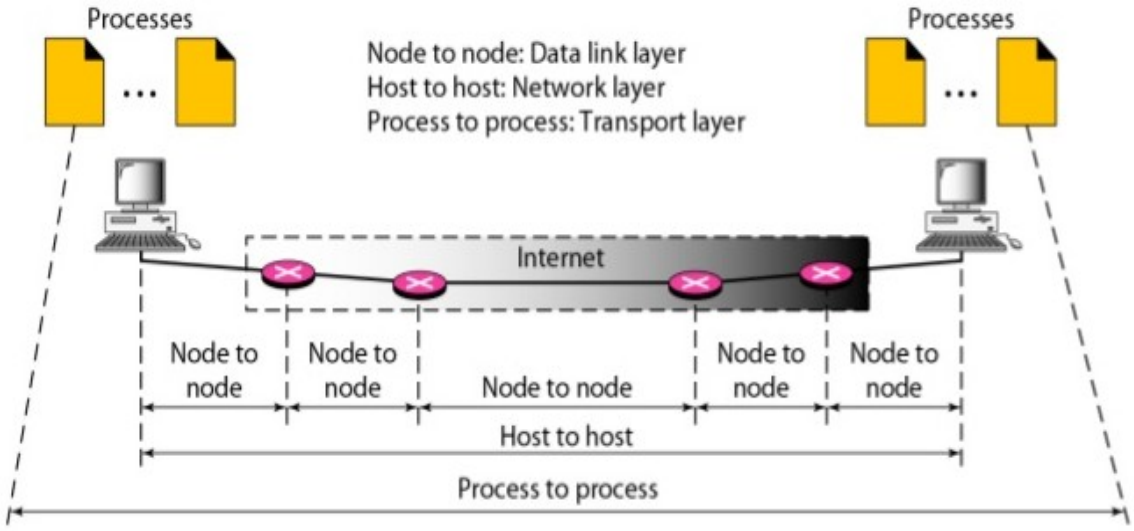
UNIT 4

Connectionless and Connection Oriented Service UDP, TCP. Congestion control, quality of service.

Process-To-Process Delivery

The **data link layer** is responsible for delivery of frames between two neighbouring nodes over a link. This is called ***node-to-node delivery.***
The **network layer** is responsible for delivery of datagrams between two hosts. This is called ***host-to-host delivery***. Communication on the Internet is not defined as the exchange of data between two nodes or between two hosts.
Real communication takes place between two processes (application programs).

The transport layer is responsible for process-to-process delivery-the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship. Figure shows these three types of deliveries and their domains.



Connectionless Versus Connection-Oriented Service

A transport layer protocol can either be connectionless or connection-oriented.

Connectionless Service
In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either. Eg:UDP

Connection-Oriented Service
In a connection-oriented service, a connection is first established between the sender and the receiver. Data are transferred. At the end, the connection is released. Eg: TCP and SCTP

USER DATAGRAM PROTOCOL (UDP)

The User Datagram Protocol (UDP) is called a **connectionless, unreliable transport protocol.** It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication. Also, it performs **very limited error checking.**
UDP is a very simple protocol using a minimum of overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

Well-Known Ports for UDP

Table shows some well-known port numbers used by UDP. Some port numbers can be used by both UDP and TCP.

Well-known Ports used with UDP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Domain	Domain Name Service (DNS)
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

User Datagram

UDP packets, called user datagrams, have a fixed-size header of 8 bytes. Figure shows the format of a user datagram.

- **Source port number**. This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535.
  If the **source host** is the **client** (a client sending a request), the port number is an ephemeral port number requested by the process and chosen by the UDP software running on the source host.
  If the **source host** is the **server** (a server sending a response), the port number is a well-known port number.

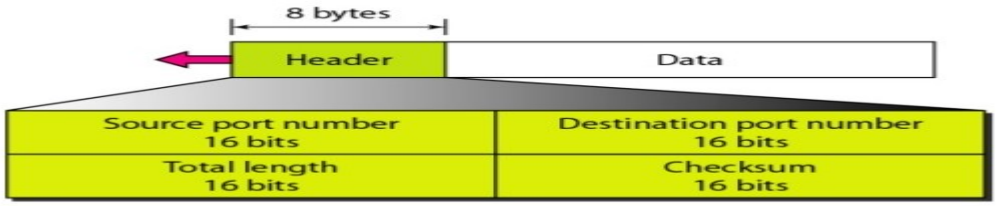


Figure: User *datagram format*

- **Destination port number.** This is the port number used by the process running on the destination host. It is also 16 bits long.
  If the **destination host** is the **server** (a client sending a request), the port number is a well-known port number.
  If the **destination host** is the **client** (a server sending a response), the port number is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

- **Length.** This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. A user datagram is encapsulated in an IP datagram. There is a field in the IP datagram that defines the total length. There is another field in the IP datagram that defines the length of the header.

  UDP length = IP length - IP header's length

- **Checksum.** This field is used to detect errors over the entire user datagram (header plus data).

Checksum

The UDP checksum includes three sections:

- A pseudoheader,
- The UDP header,
- The data coming from the application layer.

The **pseudoheader** is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s.

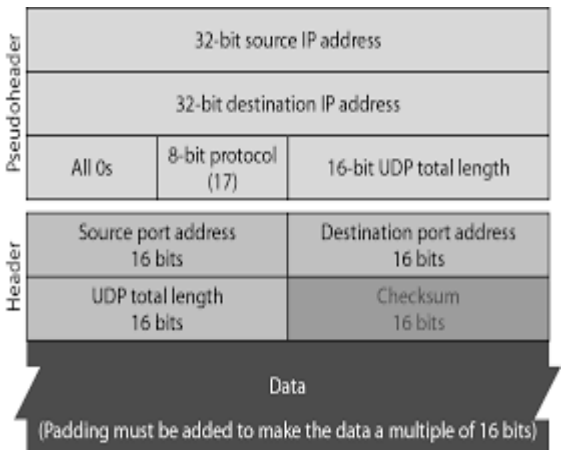


Figure: *Pseudo header for checksum calculation*

If the checksum does not include the pseudoheader, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host. The protocol field is added to ensure that the packet belongs to UDP, and not to other transport-layer protocols.
The value of the protocol field for UDP is 17. If this value is changed during transmission, the checksum calculation at the receiver will detect it and UDP drops the packet. It is not delivered to the wrong protocol.

UDP Operation

UDP uses concepts common to the transport layer.

Connectionless Services

UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program. The user datagrams are not numbered. Also, there is no connection establishment and no connection termination, as is the case for TCP. This means that each user datagram can travel on a different path. One of the ramifications of being connectionless is that the process that uses UDP cannot send a stream of data to UDP and expect UDP to chop them into different related user datagrams. Instead each request must be small enough to fit into one user datagram. Only those processes sending short messages should use UDP.

Flow and Error Control

UDP is a very simple, unreliable transport protocol. There is no flow control and no window mechanism. The receiver may overflow with incoming messages. There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated. When the receiver detects an error through the checksum, the user datagram is silently discarded. The lack of flow control and error control means that the process using UDP should provide these mechanisms.

Encapsulation and Decapsulation

To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.

Queuing

In UDP, queues are associated with ports.

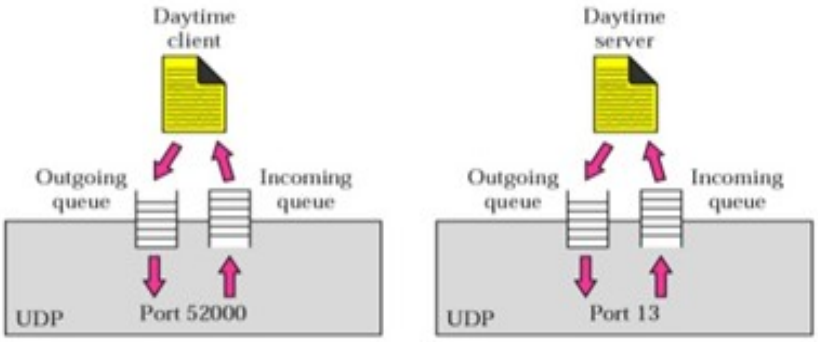


Figure: *Queues in UDP*

At the client site, when a process starts, it requests a port number from the operating system. Even if a process wants to communicate with multiple processes, it obtains only one port number and eventually one outgoing and one incoming queue. The queues opened by the client are identified by ephemeral port numbers.

The queues function as long as the process is running. When the process terminates, the queues are destroyed.

The client process can send messages to the outgoing queue by using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. If an outgoing queue overflows the operating system can ask the client process to wait before sending any more messages.

When a message arrives for a client, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram. If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a *port unreachable* message to the server.

At the server site, the mechanism of creating queues is different. In its simplest form, a server asks for incoming and outgoing queues, using its well-known port, when it starts running. The queues remain open as long as the server is running.

When a message arrives for a server, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram. If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the client.

When a server wants to respond to a client, it sends messages to the outgoing queue, using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP.

Use of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process that needs to send bulk data.
- UDP is suitable for a process with internal flow and error control mechanisms. For eg, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for management processes such as SNMP
- UDP is used for some route updating protocols such as Routing Information Protocol

TCP

- TCP, like UDP, is a process-to-process (program-to-program) protocol.
- TCP, therefore, like UDP, uses port numbers.
- Unlike UDP, TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data.
- TCP uses flow and error control mechanisms at the transport level.
- TCP is called a *connection-oriented, reliable* transport protocol. It adds connection-oriented and reliability features to the services of IP.

TCP Services

1. Process-to-Process Communication

Like UDP, TCP provides process-to-process communication using port numbers. Table lists some well-known port numbers used by TCP.

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Well-known ports used by TCP

2. Stream Delivery Service

TCP is a stream-oriented protocol. It allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.

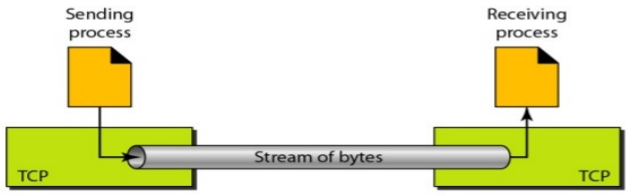


Figure: *Stream delivery*

Sending and Receiving Buffers:

Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction.

Segments:

The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a segment. TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission. The segments are encapsulated in IP datagrams and transmitted. This entire operation is transparent to the receiving process.

3. Full-Duplex Communication

TCP offers full-duplex service, in which data can flow in both directions at the same time. Each TCP then has a sending and receiving buffer, and segments move in both directions.

4. Connection-Oriented Service

TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:

1. The two TCPs establish a connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

This is a virtual connection, not a physical connection. The TCP segment is encapsulated in an IP datagram and can be sent out of order, or lost, or corrupted, and then resent. Each may use a different path to reach the destination. There is no physical connection. TCP creates a stream-oriented environment in which it accepts the responsibility of delivering the bytes in order to the other site.

5. Reliable Service

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

TCP Features

1. *Numbering System*

There is no field for a segment number value in the segment header. Instead, there are two fields called

- The Sequence Number
- The Acknowledgment Number.

These two fields refer to the **byte number** and not the segment number.

Byte Number:

TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them. The numbering does not necessarily start from 0. Instead, TCP generates a random number between 0 and 2^{32} - 1 for the number of the first byte.

Sequence Number:
After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment.
When a segment carries a combination of data and control information (piggybacking), it uses a sequence number. If a segment does not carry user data, it does not logically define a sequence number. The field is there, but the value is not valid.

Acknowledgment Number:
Communication in TCP is full duplex; when a connection is established, both parties can send and receive data at the same time. Each party numbers the bytes, usually with a different starting byte number. The sequence number in each direction shows the number of the first byte carried by the segment. The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.

2. *Flow Control*

TCP, unlike UDP, provides *flow control.* The receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

3. *Error Control*

To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.

4. *Congestion Control*

TCP, unlike UDP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network.

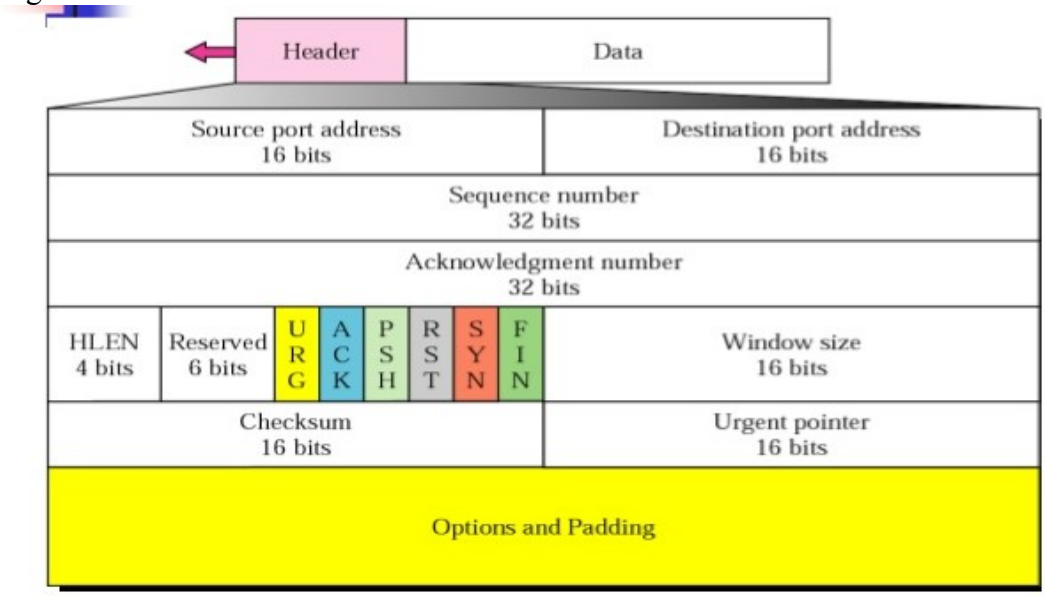
Segment

A packet in TCP is called a segment. The format of a segment is shown in Figure.
The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

- **Source port address**. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment. This serves the same purpose as the source port address in the UDP header.
- **Destination port address**. This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment. This serves the same purpose as the destination port address in the UDP header.
- **Sequence number.** This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence

number tells the destination which byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.

- **Acknowledgment number**. This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number *x* from the other party, it defines *x* + 1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.



- **Header length**. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes.
- **Reserved**. This is a 6-bit field reserved for future use.
- **Control.** This field defines 6 different control bits or flags as shown in Figure. One or more of these bits can be set at a time.

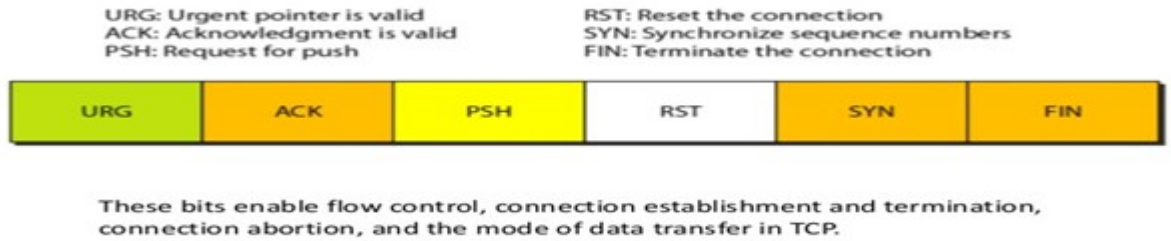


Figure: Control field

These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.

Flag	*Description*
URG	The value of the urgent pointer field is valid
ACK	The value of the acknowledgment field is valid
PSH	Push the data
RST	The connection must be reset
SYN	Synchronize sequence numbers during connection
FIN	Terminate the connection

Figure: *Description of flags in the control field*

- **Window size**. This field defines the size of the window, in bytes, that the other party must maintain. T the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver.
- **Checksum.** This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.
- **Urgent pointer.** This l6-bit field, which is valid, only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.
- **Options**. There can be up to 40 bytes of optional information in the TCP header.

A TCP Connection

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. All the segments belonging to a message are sent over this virtual path. Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.

A TCP connection is virtual, not physical. TCP operates at a higher level. TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself. If a segment is lost or corrupted, it is retransmitted. Unlike TCP, IP is unaware of this retransmission. If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering.

In TCP, connection-oriented transmission requires three phases:

1. connection establishment,
2. data transfer
3. Connection termination.

Connection Establishment

TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously. This implies that each party must initialize communication and get approval from the other party before any data are transferred.

Three-Way Handshaking:

The connection establishment in TCP is called three way handshaking. The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a *passive open.* Although the server TCP is ready to accept any connection from any machine in the world, it cannot make the connection itself.

The client program issues a request for an *active open.* A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process as shown in Figure

The **three steps** in this phase are

1. The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1.

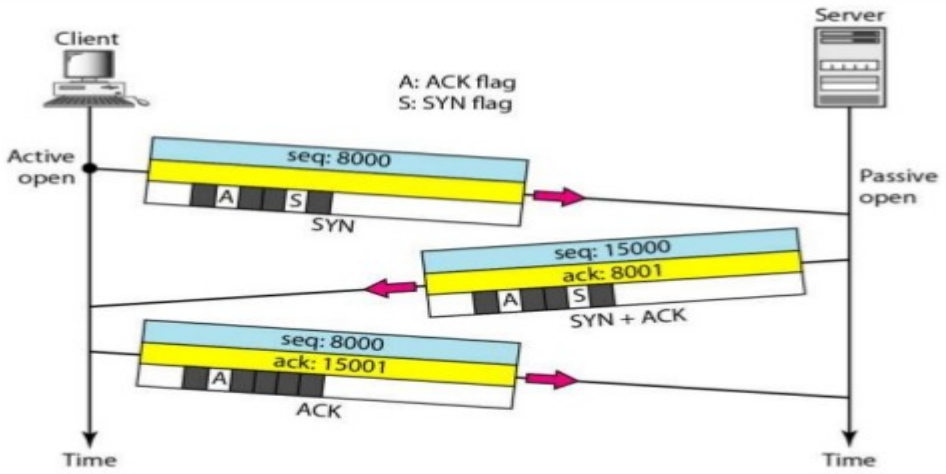
A SYN segment cannot carry data, but it consumes one sequence number.

2. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

A SYN +ACK segment cannot carry data, but does consume one sequence number.

3. The client sends the third segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. The sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

An ACK segment, if carrying no data, consumes no sequence number.



Simultaneous Open: A rare situation, called a simultaneous open, may occur when both processes issue an active open. In this case, both TCPs transmit a SYN + ACK segment to each other, and one single connection is established between them.

SYN Flooding Attack: This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is corning from a different client by faking the source IP addresses in the datagrams. The server, assuming that the clients are issuing an active open, allocates the necessary resources, such as creating communication tables and setting timers. The TCP server then sends the SYN +ACK segments to the fake clients, which are lost.

Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments. The acknowledgment is piggybacked with the data. Figure shows an example. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.

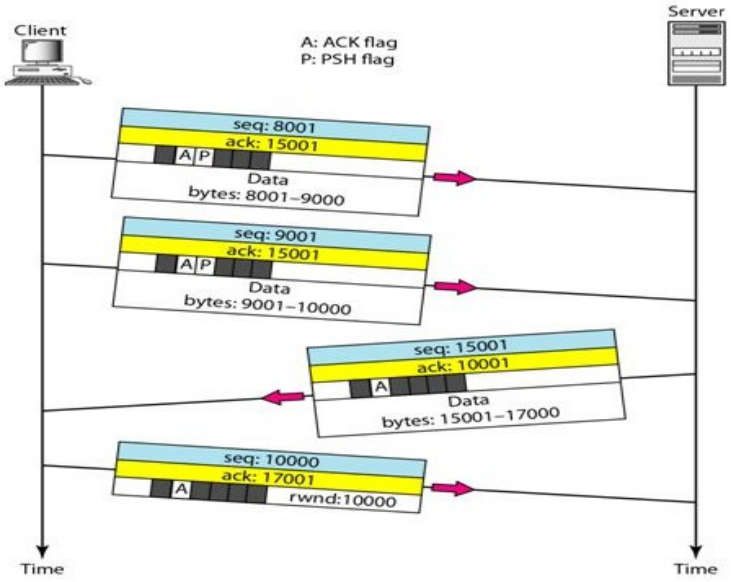


Figure: Data transfer

In this example, after connection is established, the client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent. The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received. The segment from the server, on the other hand, does not set the push flag. Most TCP implementations have the option to set or not set this flag.

Pushing Data: The sending TCP uses a buffer to store the stream of data coming from the sending application program. The sending TCP can select the segment size. The receiving TCP also buffers the data when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP. This type of flexibility increases the efficiency of TCP.

The application program at the sending site can request a *push* operation. This means that the sending TCP must not wait for the window to be filled. It must create a segment and send it immediately. The sending TCP must also set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come.

Urgent Data: TCP is a stream-oriented protocol. This means that the data are presented from the application program to TCP as a stream of bytes. Each byte of data has a position in the stream. If the sending application program wants a piece of data to be read out of order by the receiving application program. That is, an application program needs to send *urgent* Bytes then send a segment with the URG bit set. The sending application program tells the sending TCP that the piece of data is urgent. The sending TCP creates a segment and inserts the

urgent data at the beginning of the segment. The rest of the segment can contain normal data from the buffer. The urgent pointer field in the header defines the end of the urgent data and the start of normal data.

When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment, using the value of the urgent pointer, and delivers them, out of order, to the receiving application program.

Connection Termination

Two options for connection termination: three-way handshaking and four-way handshaking with a half-close option.

Three-Way Handshaking

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. A FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in Figure. If it is only a control segment, it consumes only one sequence number.

The FIN segment consumes one sequence number if it does not carry data.

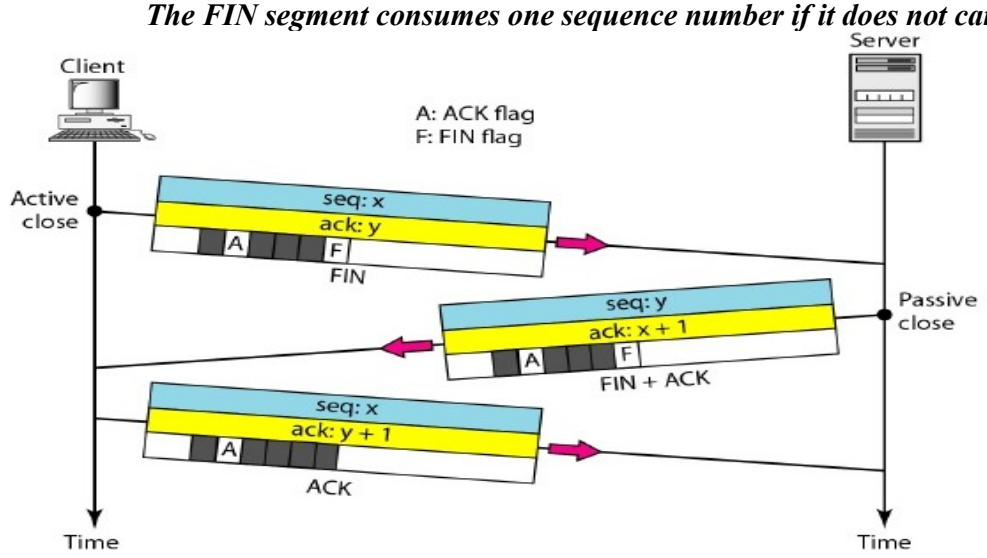


Figure: *Connection termination using three-way handshaking*

2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN +ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

   The FIN +ACK segment consumes one sequence number if it does not carry data.

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment

number, which is 1 plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

Half-Close: In TCP, one end can stop sending data while still receiving data. This is called a half-close. Although either end can issue a half-close, it is normally initiated by the client. It can occur when the server needs all the data before processing can begin.

After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server. The client cannot send any more data to the server.

Flow Control

TCP uses a sliding window to handle flow control. The sliding window protocol used by TCP is something between the *Go-Back-N* and Selective Repeat sliding window. The sliding window protocol in TCP looks like the Go-Back-N protocol because it does not use NAKs; it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.

Difference between sliding window and the one we used at the data link layer.

1, The sliding window of TCP is byte-oriented; the one in the data link layer is frame oriented.
2, The TCP's sliding window is of variable size; the one in the data link layer was of fixed size.

The window is *opened, closed,* or *shrunk.* These three activities are in the control of the receiver (and depend on congestion in the network), not the sender. The sender must obey the commands of the receiver. Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending. Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore.
Shrinking the window means moving the right wall to the left. This is strongly discouraged and not allowed in some implementations because it means revoking the eligibility of some bytes for sending.

The size of the window at one end is determined by two values:

- *receiver window (rwnd)* or *congestion window (cwnd).*

The *receiver window* is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded.
The *congestion window* is a value determined by the network to avoid congestion.

Some points about TCP sliding windows:
o The size of the window is the lesser of *rwnd* and *cwnd.* The source does not have to send a full window's worth of data.
o The window can be opened or closed by the receiver, but should not be shrunk.

o The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
o The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.

Error Control

TCP is a reliable transport layer protocol. This means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, without error, and without any part lost or duplicated.

TCP provides reliability using error control. Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments. Error control also includes a mechanism for correcting errors after they are detected. Error detection and correction in TCP is achieved through the use of three simple tools:
Checksum, acknowledgment and time-out.

Checksum
Each segment includes a checksum field which is used to check for a corrupted segment. If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment.

Acknowledgment
TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged.

ACK segments do not consume sequence numbers and are not acknowledged

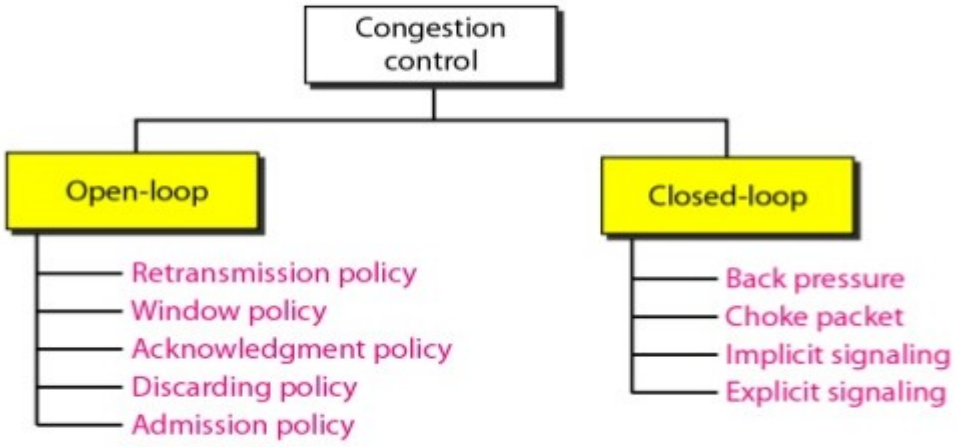
Congestion

Congestion in a network may occur if the **load** on the network-the number of packets sent to the network-is greater than the *capacity* of the network-the number of packets a network can handle. **Congestion control** refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

Congestion control

Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. Two broad categories of congestion control mechanism are:

- open-loop congestion control (prevention)
- closed-loop congestion control (removal)



Open-Loop Congestion Control

In open-loop congestion control, policies are applied to prevent congestion before it happens. In these mechanisms, congestion control is handled by either the source or the destination.

Retransmission Policy

Retransmission is sometimes unavoidable. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. Retransmission may increase congestion in the network. A good retransmission policy can prevent congestion. The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion.

Window Policy

The type of window at the sender may also affect congestion. The Selective Repeat window is better than the Go-Back-N window for congestion control. In the *Go-Back-N* window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse. The Selective Repeat window tries to send the specific packets that have been lost or corrupted.

Acknowledgment Policy

The acknowledgment policy imposed by the receiver may also affect congestion. If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.. A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires. A receiver may decide to acknowledge only *N* packets at a time. Sending fewer acknowledgments means imposing less load on the network.

Discarding Policy

A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission. For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented.

Admission Policy

An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks. Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual circuit connection if there is congestion in the network or if there is a possibility of future congestion.

Closed-Loop Congestion Control

Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols.

Backpressure

The technique of *backpressure* refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes.
Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source. The backpressure technique can be applied only to **virtual circuit networks**, in which each node knows the upstream node from which a flow of data is coming.

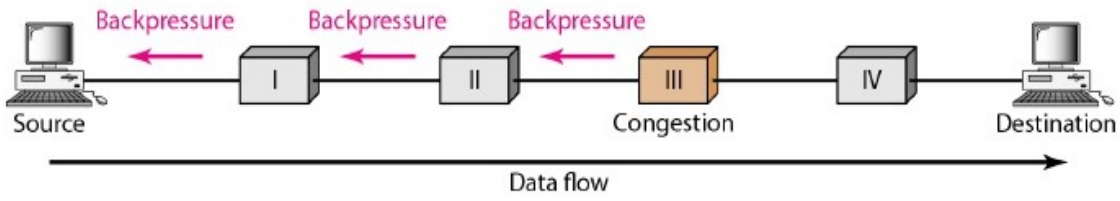


Figure: Backpressure method

Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down. Node II, in turn, may be congested because it is slowing down the output flow of data. If node II is congested, it informs node I to slow down, which in turn may create congestion. If so, node I inform the source of data to slow down. This, in time, alleviates the congestion. Note that the *pressure* on node III is moved backward to the source to remove the congestion.

Choke Packet

A choke packet is a packet sent by a node to the source to inform it of congestion. In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly. The intermediate nodes through which the packet has travelled are not warned.

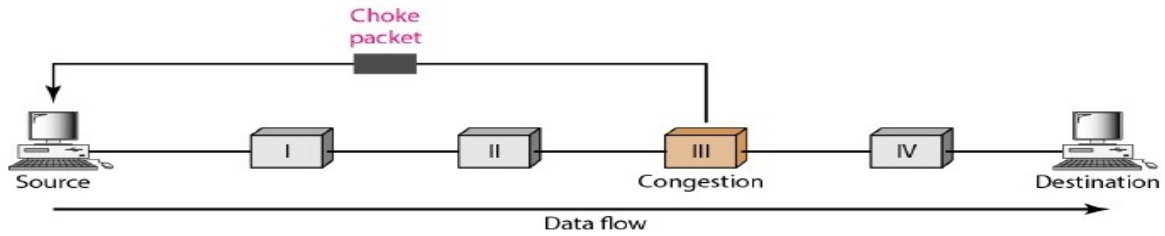


Figure: *Choke packet*

Implicit Signaling

In implicit signaling, there is no communication between the congested node or nodes and the source. The source guesses that there is congestion somewhere in the network from other symptoms. For e.g.:, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested. The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down.

Explicit Signaling

The node that experiences congestion can explicitly send a signal to the source or destination. The explicit signaling method is different from the choke packet method. In the choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data.

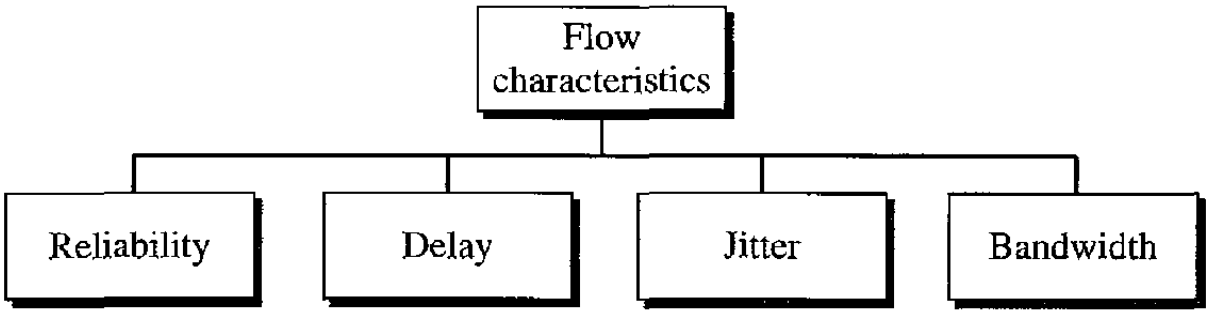
Backward Signaling: A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.
Forward Signaling A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

Quality of Service

Quality of service (QoS) is an internetworking issue.

Flow Characteristics

Four types of characteristics are attributed to a flow: reliability, delay, jitter, and bandwidth



Reliability
Reliability is a characteristic that a flow needs. Lack of reliability means losing a packet or acknowledgment, which entails retransmission. The sensitivity of application programs to

reliability is not the same. For example, it is more important that electronic mail, file transfer, and Internet access have reliable transmissions than telephony or audio conferencing.

Delay
Source-to-destination delay is another flow characteristic. Applications can tolerate delay in different degrees. In this case, telephony, audio conferencing, video conferencing, and remote log-in need minimum delay, while delay in file transfer or e-mail is less important.

Jitter
Jitter is the variation in delay for packets belonging to the same flow. For eg if 4 packets depart at times 0, 1, 2, 3 and arrive at 20, 21, 22, 23, all have the same delay, 20 units of time. On the other hand, if the above four packets arrive at 21, 23, 21, and 28, they will have different delays: 21, 22, 19, and 24.
For applications such as audio and video, the first case is completely acceptable; the second case is not. For these applications, it does not matter if the packets arrive with a short or long delay as long as the delay is the same for all packets. For this application, the second case is not acceptable.
Jitter is defined as the variation in the packet delay. High jitter means the difference between delays is large; low jitter means the variation is small.

Bandwidth
Different applications need different bandwidths. In video conferencing we need to send millions of bits per second to refresh a colour screen while the total number of bits in an e-mail may not reach even a million.

TECHNIQUES TO IMPROVE QoS

Four common methods:

- Scheduling
- Traffic shaping
- Admission control
- Resource reservation

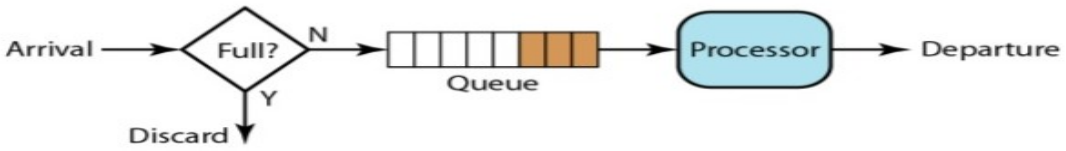
Scheduling

Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Several scheduling techniques are designed to improve the quality of service. Three of them are

- FIFO queuing
- priority queuing
- Weighted fair queuing.

FIFO Queuing
In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded. A FIFO queue is familiar to those who have had to wait for a bus at a bus stop. Figure 24.16 shows a conceptual view of a FIFO queue.

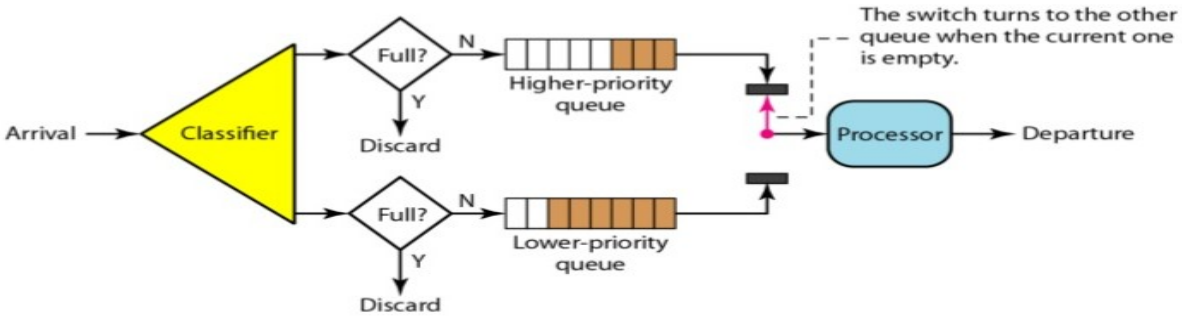
Figure 24.16 *FIFO queue*



Priority Queuing

In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue. The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last. The system does not stop serving a queue until it is empty. Figure 24.17 shows priority queuing with two priority levels (for simplicity).

Figure 24.17 *Priority queuing*

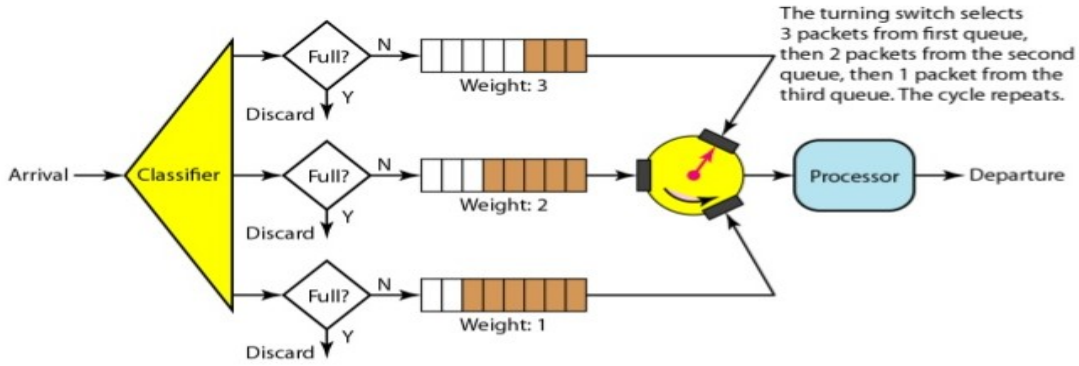


A priority queue can provide better QoS than the FIFO queue because higher priority traffic, such as multimedia, can reach the destination with less delay. However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called *starvation.*

Weighted Fair Queuing

A better scheduling method is weighted fair queuing. In this technique, the packets are still assigned to different classes and admitted to different queues. The queues are weighted based on the priority of the queues; higher priority means a higher weight. The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight. For eg, if the weights are 3, 2, and 1, three packets are processed from the first queue, two from the second queue, and one from the third queue. If the system does not impose priority on the classes, all weights can be equal. In this way, we have fair queuing with priority. Figure 24.18 shows the technique with three classes.

Figure 24.18 *Weighted fair queuing*



Traffic Shaping

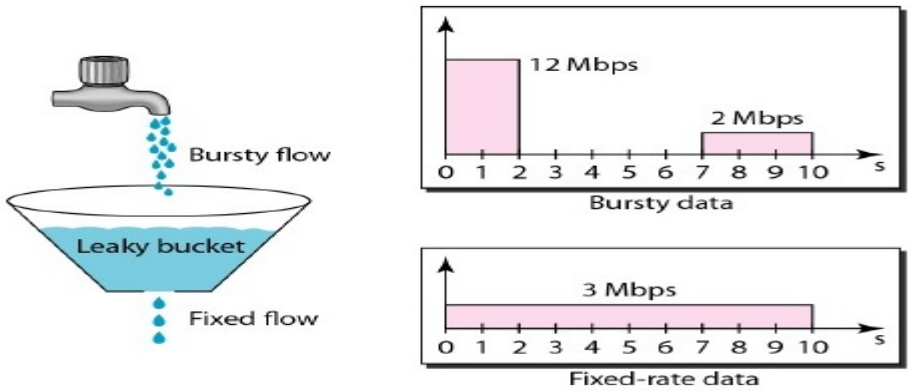
Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic:

- Leaky bucket
- Token bucket.

Leaky Bucket

If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate. Figure 24.19 shows a leaky bucket and its effects.

Figure 24.19 *Leaky bucket*



In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure 24.19 the host sends a burst of data at a rate of 12 Mbps for 2s, for a total of 24 Mbits of data. The host is silent for 5s and then sends data at a rate of 2 Mbps for 3s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in l0s. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 s. Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion.

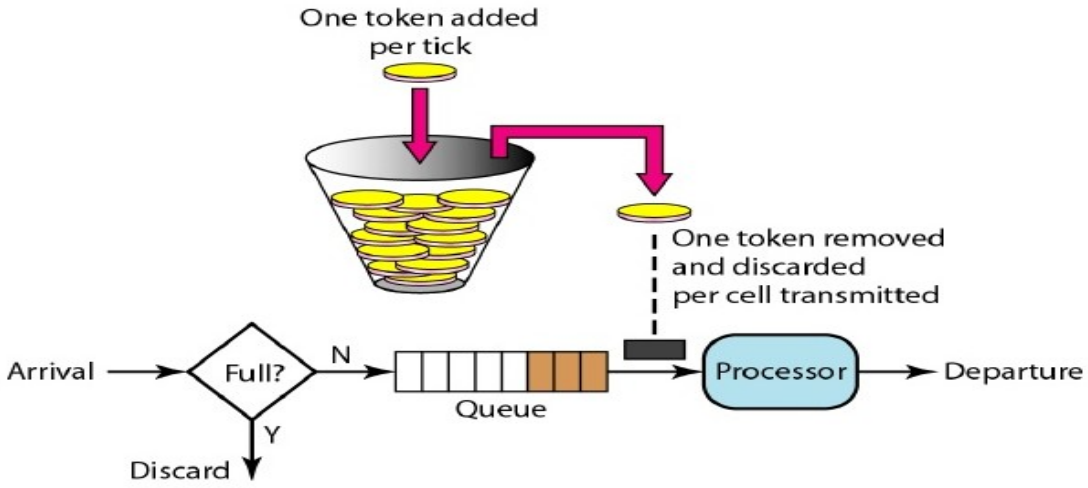
As an analogy, consider the freeway during rush hour (bursty traffic). If, instead, commuters could stagger their working hours, congestion on our freeways could be avoided.

Token Bucket

The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens. For each tick of the clock, the system sends *n* tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if *n* is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens. Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick. In other words, the host can send bursty data as long as the bucket is not empty. Figure 24.21 shows the idea.

The token bucket can easily be implemented with a counter. The token is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is 0, the host cannot send data.

Figure 24.21 ***Token bucket***



Resource Reservation

A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on. The quality of service is improved if these resources are reserved beforehand. The QoS model called Integrated Services depends heavily on resource reservation to improve the quality of service.

Admission Control

Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications. Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.