UNIT-1

IP (Internet Protocol):

Internet Protocols are a set of rules that governs the communication and exchange of data over the internet. Both the sender and receiver should follow the same protocols to communicate the data.

The main task of IP is to deliver the packets from source to the destination based on the IP addresses available in the packet headers. IP defines the packet structure that hides the data which is to be delivered as well as the addressing method that labels the datagram with a source and destination information.

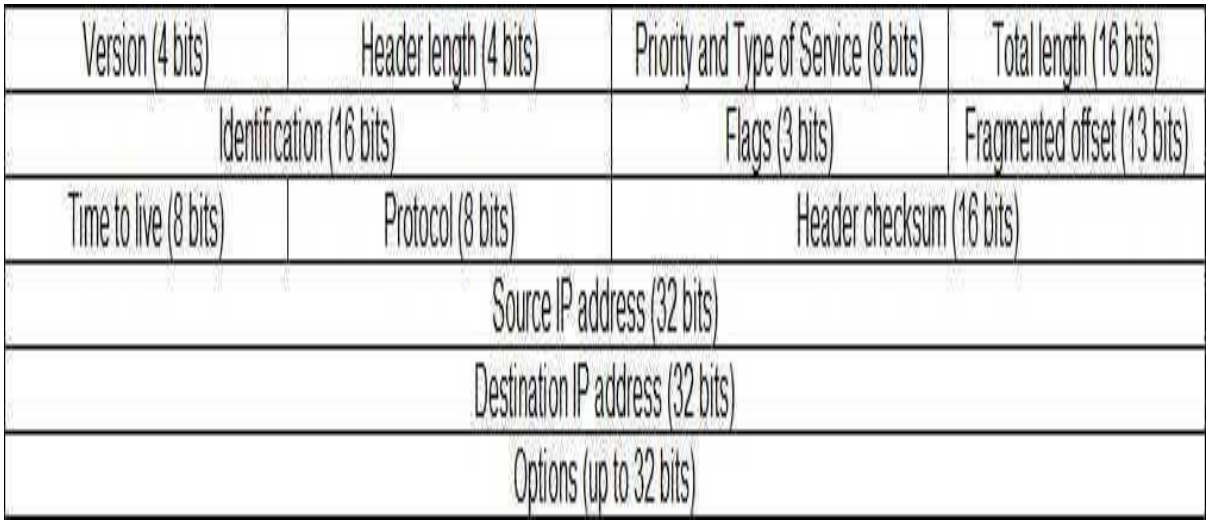
An IP protocol provides the connectionless service, which is accompanied by two transport protocols, i.e., TCP/IP and UDP/IP, so internet protocol is also known as TCP/IP or UDP/IP.

The first version of IP (Internet Protocol) was IPv4. After IPv4, IPv6 came into the market, which has been increasingly used on the public internet since 2006.

IP header

An **IP header** is a prefix to an IP packet that contains information about the IP version, length of the packet, source, and destination IP addresses, etc.

IPv4: It consists of the following fields:

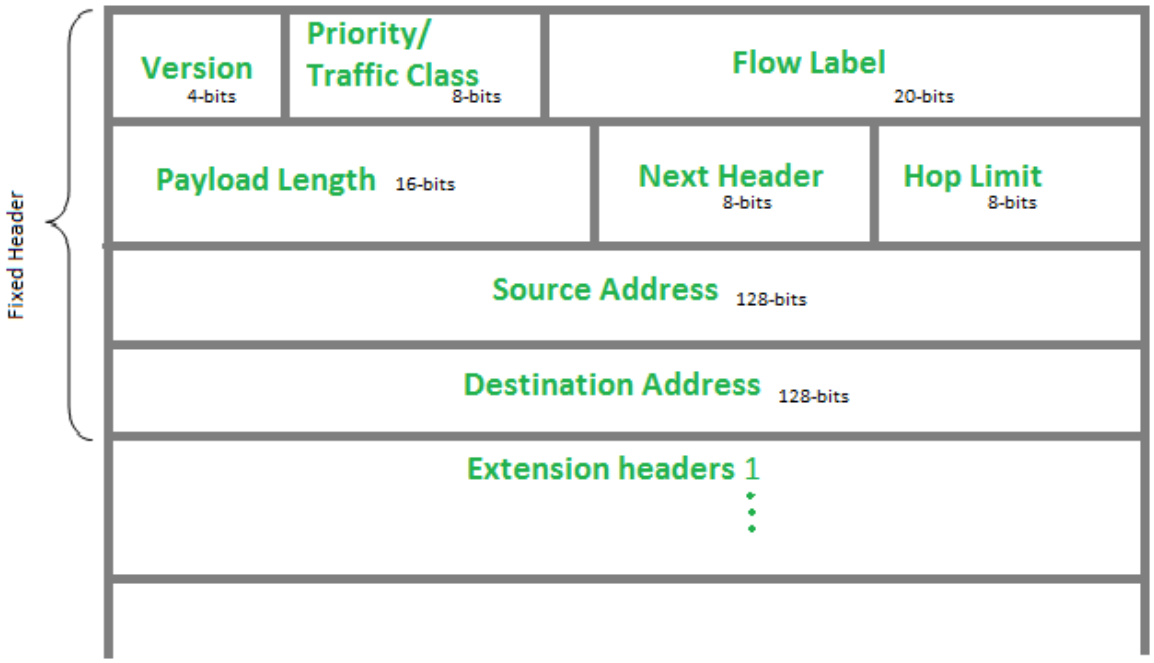


- **Version** – The version of the IP protocol. For IPv4, this field has a value of 4 which contains bit sequence 0110.
- **Header length** – the length of the header in 32-bit words. The minimum value is 20 bytes, and the maximum value is 60 bytes.
- **Priority and Type of Service** – specifies how the datagram should be handled. The first 3 bits are the priority bits.
- **Total length** – the length of the entire packet (header + data). The minimum length is 20 bytes, and the maximum is 65,535 bytes.

- **Identification** – used to differentiate fragmented packets from different datagrams.
- **Flags** – used to control or identify fragments.
- **Fragmented offset** – used for fragmentation and reassembly if the packet is too large to put in a frame.
- **Time to live** – limits a datagram's lifetime. If the packet doesn't get to its destination before the TTL expires, it is discarded.
- **Protocol** – defines the protocol used in the data portion of the IP datagram. For example, TCP is represented by the number 6 and UDP by 17.
- **Header checksum** – used for error-checking of the header. If a packet arrives at a router and the router calculates a different checksum than the one specified in this field, the packet will be discarded.
- **Source IP address** – the IP address of the host that sent the packet.
- **Destination IP address** – the IP address of the host that should receive the packet.
- **Options** – used for network testing, debugging, security, and more. This field is usually empty.

IPv6 Header:

IP version 6 is the new version of Internet Protocol, which is way better than IP version 4 in terms of complexity and efficiency.



- **Version (4-bits):** Indicates version of Internet Protocol has a value of 6 which contains bit sequence 0110.

- **Traffic Class (8-bits):** The Traffic Class field indicates class or priority of IPv6 packet which is similar to *Service Field* in IPv4 packet. It helps routers to handle the traffic based on the priority of the packet. If congestion occurs on the router, then packets with the least priority will be discarded.

Priority assignment of 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

- **Flow Label (20-bits):** Flow Label field is used by a source to label the packets belonging to the same flow to request special handling by intermediate IPv6 routers. In order to distinguish the flow, an intermediate router can use the source address, a destination address, and flow label of the packets. Between a source and destination, multiple flows may exist because many processes might be running at the same time. Routers or Host that does not support the functionality of flow label field and for default router handling, flow label field is set to 0. While setting up the flow label, the source is also supposed to specify the lifetime of the flow.

- **Payload Length (16-bits):** It is a 16-bit (unsigned integer) field, indicates the total size of the payload which tells routers about the amount of information a particular packet contains in its payload. The payload Length field includes extension headers (if any) and an upper-layer packet.

- **Next Header (8-bits):** Next Header indicates the type of extension header (if present) immediately following the IPv6 header. Whereas in some cases it indicates the protocols contained within upper-layer packets, such as TCP, UDP.

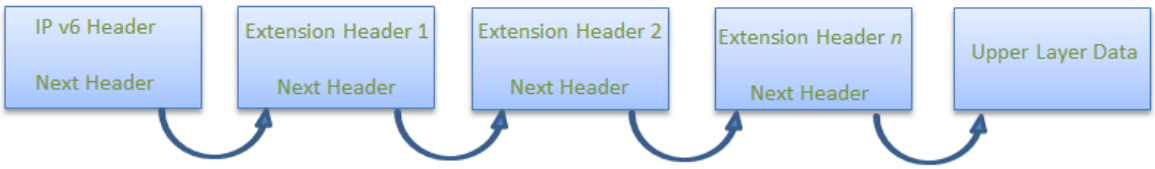
- **Hop Limit (8-bits):** Hop Limit field is the same as TTL in IPv4 packets. It indicates the maximum number of intermediate nodes IPv6 packet is allowed to travel. Its value gets decremented by one, by each node that forwards the packet, and the packet is discarded if the value decrements to 0. This is used to discard the packets that are stuck in an infinite loop because of some routing error.

- **Source Address (128-bits):** Source Address is the 128-bit IPv6 address of the original source of the packet.

- **Destination Address (128-bits):** The destination Address field indicates the IPv6 address of the final destination. All the intermediate nodes can use this information in order to correctly route the packet.

- **Extension Headers:** In order to rectify the limitations of the *IPv4 Option Field*, Extension Headers are introduced in IP version 6. The extension header mechanism

is a very important part of the IPv6 architecture. The next Header field of IPv6 fixed header points to the first Extension Header and this first extension header points to the second extension header and so on.



IP Fragmentation:

Fragmentation is done by the network layer when the maximum size of datagram is greater than maximum size of data. When a host sends an IP packet onto the network it cannot be larger than the maximum size supported by that local network. This size is determined by the network's data link and IP Maximum Transmission Units (MTUs). Since there are 16 bits for total length in IP header so, the maximum size of IP datagram = $2^{16} - 1$ = 65, 535 bytes.

However, packets that are initially transmitted over a network supporting one MTU may need be routed across networks. In these cases, if the packet size exceeds the lower MTU the data in the packet must be fragmented. This means it is broken into pieces carried within new packets (fragments) that are equal to or smaller than the lower MTU. This is called Fragmentation and the data in these fragments is then typically reassembled when they reach their destination.

Fragmentation has a number of drawbacks which result in its use being avoided where possible, primarily:

- The loss of a single fragment results in all the fragments having to be resent.
- Only the first fragment contains the high layer headers which can cause issues with firewalls, middle-boxes and routers that rely on inspecting those headers.
- Fragmentation may result in out of order packet delivery and the need for reordering.
- Overhead at the network layer is present due to the extra header introduced due to fragmentation.

The IPv4 Header Fields Used

The processes of fragmentation and reassembly involve a number of IP header fields being set in the fragments:

Fragmentation's operation relies upon three IP header fields (32 bits in total):

- **Identification (16 bits):** use to identify fragments of the same frame with the **identification (16 bits)** field in the IP header. Each fragment of a frame has the same identification number.

- **Flag (3 bits):** the first, reserved bit of the **Flags** field (3 bits) will be 0 (unset) and the second bit, **Don't Fragment (DF)**, and the third bit of the field, **More Fragments (MF)**.

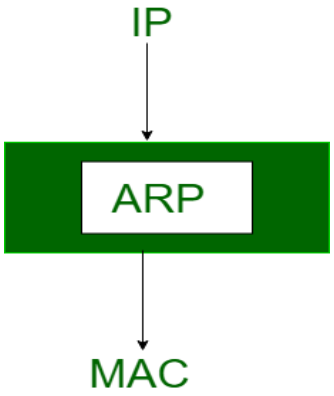
    - **More fragments (MF = 1 bit) –** if MF = 1, more fragments are ahead of this fragment and if MF = 0, it is the last fragment.
    - **Don't fragment (DF = 1 bit) –** if we don't want the packet to be fragmented then DF is set i.e. DF = 1.
- **Fragment offset (13 bits):** use to identify the sequence of fragments in the frame. It generally indicates a number of data bytes preceding or ahead of the fragment.

ARP: MAC address is a physical address of computer system. It is also known as the data link layer, which establishes and terminates a connection between two physically connected devices so that data transfer can take place. An IP address is a unique address that identifies a device on the internet or a local network. The IP address is also referred to as the network layer or the layer responsible for forwarding packets of data through different routers.

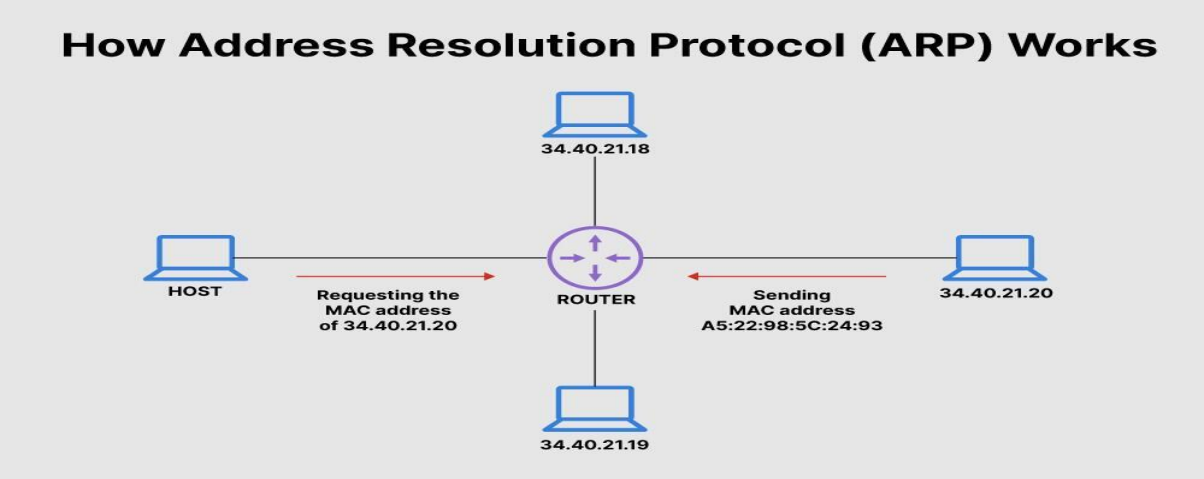
Position of ARP/RARP/ICMP/IGMP in the network layer

The ICMP/IGMP/ARP/RARP resides in the IP layer, as shown in the below diagram.

Most of the computer programs/applications use **logical address (IP address)** to send/receive messages, however, the actual communication happens over the **physical address (MAC address)**. This is where ARP comes into the picture, its functionality is to translate IP address to physical addresses.



This mapping procedure is important because the lengths of the IP and MAC addresses differ, and a translation is needed so that the systems can recognize one another. ARP is necessary because the software address (IP address) of the host or computer connected to the network needs to be translated to a hardware address (MAC address). Without ARP, a host would not be able to figure out the hardware address of another host. The LAN keeps a table or directory that maps IP addresses to MAC addresses of the different devices, including both endpoints and routers on that network. This table or directory is not maintained by users or even by IT administrators.
Instead, the ARP protocol creates entries on the fly. If a user's device does not know the hardware address of the destination host, the device will send a message to every host on the network asking for this address. When the proper destination host learns of the request, it will reply back with its hardware address, which will then be stored in the ARP directory or table.



What Does ARP Do and How Does It Work?

When a new computer joins a local area network (LAN), it will receive a unique IP address to use for identification and communication. Packets of data arrive at a gateway, destined for a particular host machine. The gateway, or the piece of hardware on a network that allows data to flow from one network to another, asks the ARP program to find a MAC address that matches the IP address. The ARP cache keeps a list of each IP address and its matching MAC address. The ARP cache is dynamic, but users on a network can also configure a static ARP table containing IP addresses and MAC addresses.

Every time a device requests a MAC address to send data to another device connected to the LAN, the device verifies its ARP cache to see if the IP-to-MAC-address connection has already been completed. If it exists, then a new request is unnecessary. However, if the translation has not yet been carried out, then the request for network addresses is sent, and ARP is performed.

An ARP cache size is limited by design, and addresses tend to stay in the cache for only a few minutes. It is purged regularly to free up space. This design is also intended for privacy and security to prevent IP addresses from being stolen or spoofed by cyber attackers. While MAC addresses are fixed, IP addresses are constantly updated.

ARP Packet Format:

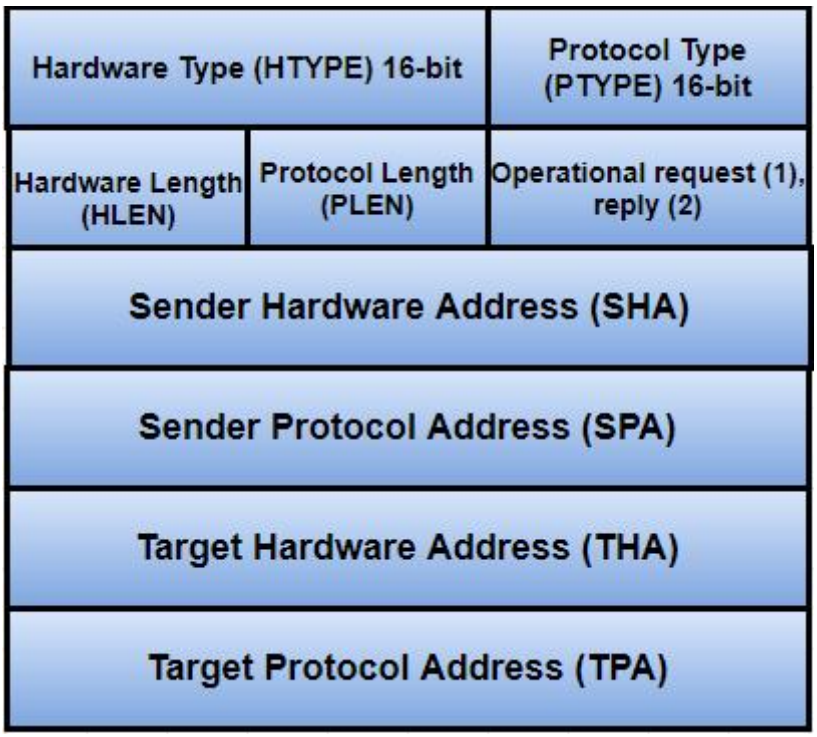


Figure shows the format of an ARP packet. The fields are as follows:

❑ **Hardware type:** This is a 16-bit field defining the type of the network on which ARP is running. Each LAN has been assigned an integer based on its type. For example, Ethernet is given the type 1. ARP can be used on any physical network.

❑ **Protocol type:** This is a 16-bit field defining the protocol. For example, the value of this field for the IPv4 protocol is 080016. ARP can be used with any higher-level protocol.

❑ **Hardware length:** This is an 8-bit field defining the length of the physical address in bytes. For example, for Ethernet the value is 6.

❑ **Protocol length:** This is an 8-bit field defining the length of the logical address in bytes. For example, for the IPv4 protocol the value is 4.

❑ **Operation:** This is a 16-bit field defining the type of packet. Two packet types are defined: ARP request (1), ARP reply (2).

❑ **Sender hardware address:** This is a variable-length field defining the physical address of the sender. For example, for Ethernet this field is 6 bytes long.

❑ **Sender protocol address:** This is a variable-length field defining the logical (for example, IP) address of the sender. For the IP protocol, this field is 4 bytes long.

❑ **Target hardware address:** This is a variable-length field defining the physical address of the target. For example, for Ethernet this field is 6 bytes long. For an ARP request message, this field is all 0s because the sender does not know the physical address of the target.

❑ **Target protocol address:** This is a variable-length field defining the logical (for example, IP) address of the target. For the IPv4 protocol, this field is 4 bytes long.

Types of ARP:

1. Proxy ARP: Proxy ARP is a technique by which a proxy device on a given network answers the ARP request for an IP address that is not on that network. The proxy is aware of the location of the traffic's destination and offers its own MAC address as the destination.

2. Gratuitous ARP: Gratuitous ARP is almost like an administrative procedure, carried out as a way for a host on a network to simply announce or update its IP-to-MAC address. Gratuitous ARP is not prompted by an ARP request to translate an IP address to a MAC address.

3. Reverse ARP (RARP): Host machines that do not know their own IP address can use the Reverse Address Resolution Protocol (RARP) for discovery.

4. Inverse ARP (IARP): Whereas ARP uses an IP address to find a MAC address, IARP uses a MAC address to find an IP address.

Reverse Address Resolution Protocol (RARP): It is a protocol a physical machine in a local area network (LAN) can use to request its IP address. It does this by sending the device's physical address to a specialized RARP server that is on the same LAN and is actively listening for RARP requests.

A network administrator creates a table in a RARP server that maps the physical interface or media access control (MAC) addresses to corresponding IP addresses. When a new RARP-enabled device first connects to the network, its RARP client program sends its physical MAC address to the RARP server for the purpose of receiving an IP address in return that the device can use to communicate with other devices on the IP network. The RARP request is sent in the form of broadcast.

The general RARP process flow follows these steps:

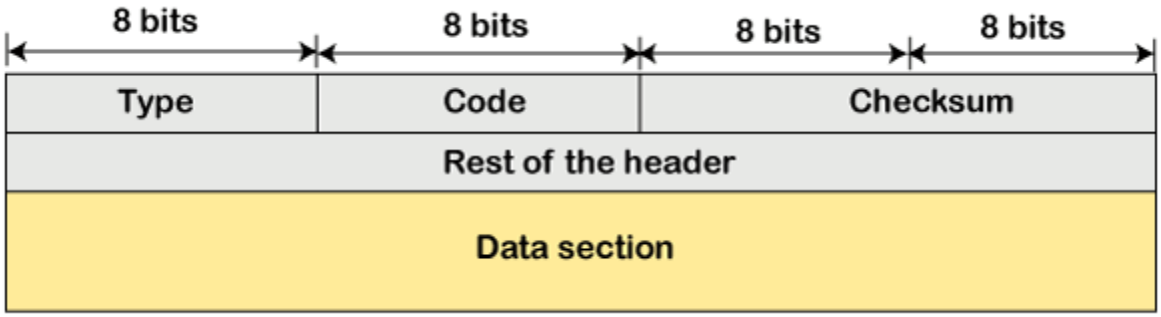
1. Device 1 connects to the local network and sends an RARP broadcast to all devices on the subnet. In the RARP broadcast, the device sends its physical MAC address and requests an IP address it can use.
2. Because a broadcast is sent, device 2 receives the broadcast request. However, since it is not a RARP server, device 2 ignores the request.
3. The broadcast message also reaches the RARP server. The server processes the packet and attempts to find device 1's MAC address in the RARP lookup table. If one is found, the RARP server returns the IP address assigned to the device.

Difference between ARP and RARP:

RARP	ARP
RARP stands for Reverse Address Resolution Protocol	ARP stands for Address Resolution Protocol
In RARP, we find our own IP address	In ARP, we find the IP address of a remote machine
The MAC address is known and the IP address is requested	The IP address is known, and the MAC address is being requested
It uses the value 3 for requests and 4 for responses	It uses the value 1 for requests and 2 for responses

Internet Control Message Protocol (ICMP): Since IP does not have an inbuilt mechanism for sending error and control messages. It depends on Internet Control Message Protocol (ICMP) to provide an error control. It is used for reporting errors and management queries. It is a supporting protocol and is used by networks devices like routers for sending error messages and operations information.

ICMP Message Format: The ICMP message contains the following fields:



- **Type:** It is an 8-bit field. It defines the ICMP message type.
- **Code:** It is an 8-bit field that defines the subtype of the ICMP message
- **Checksum:** It is a 16-bit field to detect whether the error exists in the message or not.

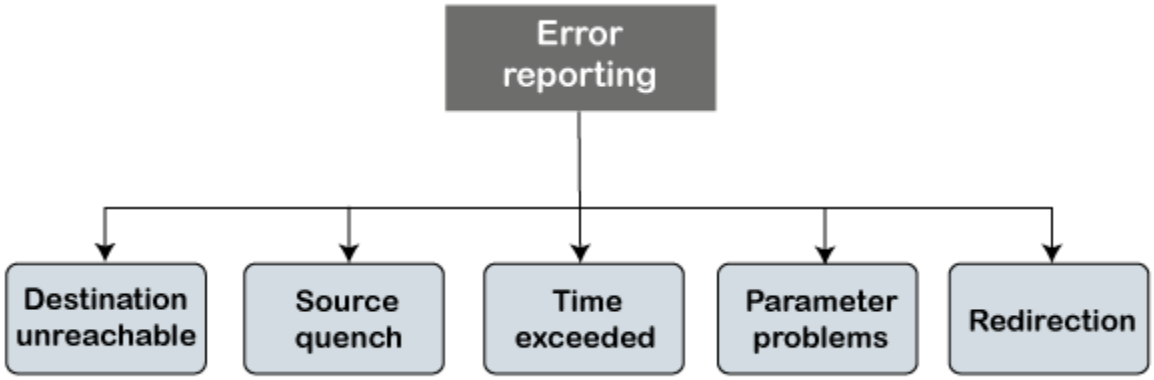
Types of ICMP Messages: The ICMP messages are usually divided into two categories:

- **Error-reporting messages:** The error-reporting message means that the router encounters a problem when it processes an IP packet then it reports a message.

- **Query messages:** The query messages are those messages that help the host to get the specific information of another host. For example, suppose there are a client and a

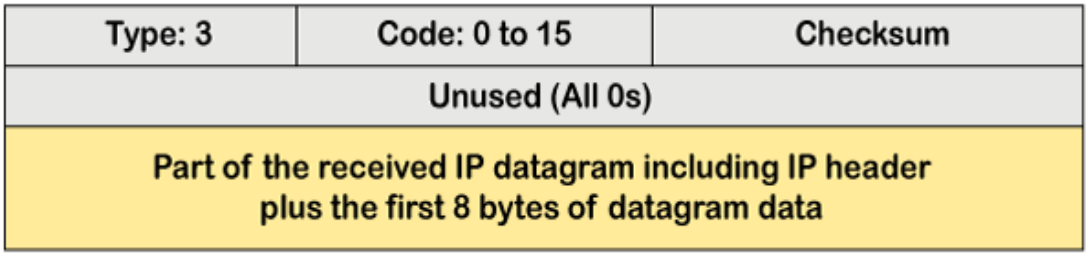
server, and the client wants to know whether the server is live or not, then it sends the ICMP message to the server.

Types of Error Reporting messages: The error reporting messages are broadly classified into the following categories:



1. **Destination unreachable:** The destination unreachable error occurs when the packet does not reach the destination. Suppose the sender sends the message, but the message does not reach the destination, then the intermediate router reports to the sender that the destination is unreachable.

   Message format of the destination unreachable message:



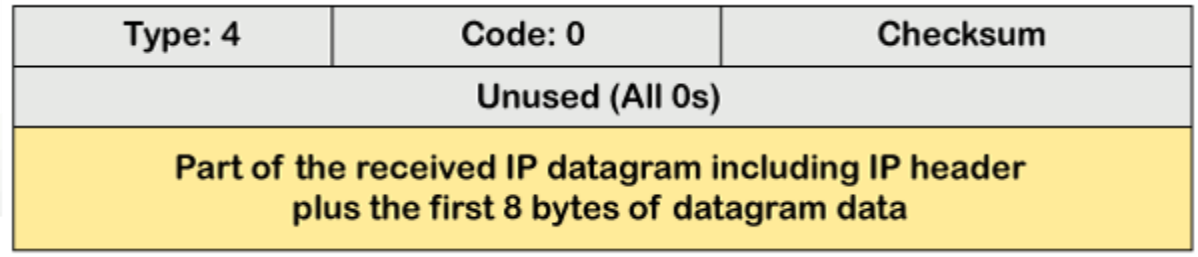
In the message format:

Type: It defines the type of message. The number 3 specifies that the destination is unreachable.

Code (0 to 15): It is a 4-bit number which identifies whether the message comes from some intermediate router or the destination itself.

2. **Source quench:** There is no flow control or congestion control mechanism in the network layer or the IP protocol. The sender is concerned with only sending the packets, and the sender does not think whether the receiver is ready to receive those packets or is there any congestion occurs in the network layer so that the sender can send a lesser number of packets, so there is no flow control or congestion control mechanism. In this case, ICMP provides feedback, i.e., source quench. Suppose the

sender resends the packet at a higher rate, and the router is not able to handle the high data rate. To overcome such a situation, the router sends a source quench message to tell the sender to send the packet at a lower rate.



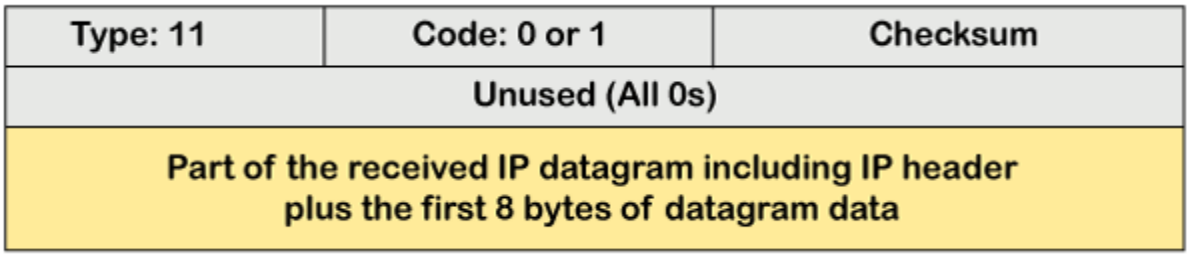
The above diagram shows the message format of the source quench message. It is a type 4 message, and code is zero.

A source quench message informs the sender that the datagram has been discarded due to the congestion occurs in the network layer. So, the sender must either stop or slow down the sending of datagrams until the congestion is reduced. The router sends one source-quench message for each datagram that is discarded due to the congestion in the network layer.

3. **Time exceeded:** Sometimes the situation arises when there are many routers that exist between the sender and the receiver. When the sender sends the packet, then it moves in a routing loop. The time exceeded is based on the time-to-live value. When the packet traverses through the router, then each router decreases the value of TTL by one. Whenever a router decreases a datagram with a time-to-live value to zero, then the router discards a datagram and sends the time exceeded message to the original source.

   In the case of fragmentation, if all the fragments are not reached to the destination in a set time, they discard all the received fragments and send a time-exceeded message to the original source. The code will be different as compared to TTL.

The message format of time exceeded:



The above message format shows that the type of time-exceeded is 11, and the code can be either 0 or 1. The code 0 represents TTL, while code 1 represents fragmentation.

In a time-exceeded message, the code 0 is used by the routers to show that the time-to-live value is reached to zero.

The code 1 is used by the destination to show that all the fragments do not reach within a set time.

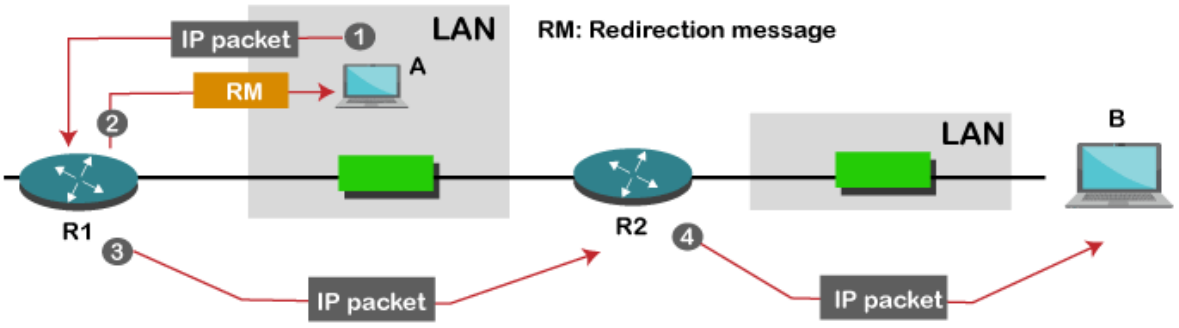
4. **Parameter problems:** The router and the destination host can send a parameter problem message. This message conveys that some parameters are not properly set.

The message format of the parameter problem:

Type: 12	Code: 0 or 1	Checksum
Pointer	Unused (All 0s)	
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

In this, type of message is 12, and the code can be 0 or 1.

5. **Redirection:** A redirection message is sent from the router to the host on the same network**.**



When the packet is sent, then the routing table is gradually augmented and updated. The tool used to achieve this is the redirection message. For example, A wants to send the packet to B, and there are two routers exist between A and B and there is a direct way from the host to router R1 and R2 both. First, A sends the data to the router 1. The router 1 sends the IP packet to router 2 and redirection message to A so that A can update its routing table.

Types of ICMP Query Messages: The ICMP Query message is used for error handling or debugging the internet. This message is commonly used to ping a message.

1. **Echo-request and echo-reply message:** A router or a host can send an echo-request message. It is used to ping a message to another host that "Are you alive". If the other host is alive, then it sends the echo-reply message. An echo-reply message is sent by the router or the host that receives an echo-request message.

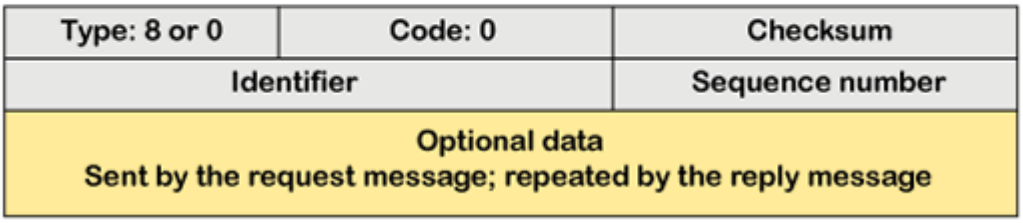
Key points of Query messages

I. The echo-request message and echo-reply message can be used by the network managers to check the operation of the IP protocol. Suppose two hosts, i.e., A and B, exist, and A wants to communicate with host B. The A host can communicate to host B if the link is not broken between A and B, and B is still alive.

II. The echo-request message and echo-reply message check the host's reachability, and it can be done by invoking the ping command.

The message format of echo-request and echo-reply message

Type 8: Echo request
Type 0: Echo reply

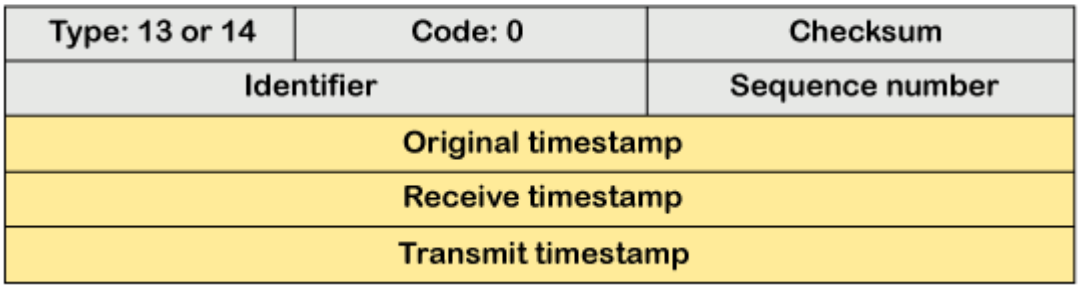


The above diagram shows the message format of the echo-request and echo-reply message. The type of echo-request is 8, and the request of echo-reply is 0. The code of this message is 0.

2. **Timestamp-request and timestamp-reply message:** The timestamp-request and timestamp-reply messages are also a type of query messages. Suppose the computer A wants to know the time on computer B, so it sends the timestamp-request message to computer B. The computer B responds with a timestamp-reply message.

Message format of timestamp-request and timestamp-reply

Type 13: request
Type 14: reply



The type of timestamp-request is 13, and the type of timestamp-reply is 14. The code of this type of message is 0.

Key points related to timestamp-request and timestamp-reply message

- It can be used to calculate the round-trip time between the source and the destination, even if the clocks are not synchronized.
- It can also be used to synchronize the clocks in two different machines if the exact transit time is known.

If the sender knows the exact transit time, then it can synchronize the clock. The sender asks the time on the receiver's clock, and then it adds the time and propagation delay. Suppose the time is 1:00 clock and propagation delay is 100 ms, then time would be 1:00 clock plus 100 ms.

Debugging tools: There are several tools used for debugging. Here we will learn two tools that use ICMP for debugging. The two tools are **ping** and **traceroute**.

1. **ping:** By ping tool we can send echo-request and echo-reply messages that check whether the host or a router is alive or running.

2. traceroute: Traceroute is a tool that tracks the route taken by a packet on an IP network from source to destination. It records the time taken by the packet on each hop during its route from source to destination. Traceroute uses ICMP messages and TTL values. The TTL value is calculated; if the TTL value reaches zero, the packet gets discarded. Traceroute uses small TTL values as they get quickly expired. If the TTL value is 1 then the message is produced by router 1; if the TTL value is 2 then the message is produced by router 2, and so on.

Example: Suppose A and B are two different hosts, and A wants to send the packet to the host B. Between A and B, 3 routers exist. To determine the location of the routers, we use the traceroute tool.

TTL value =1: First, host A sends the packet to router 1 with TTL value 1, and when the packet reaches to router 1 then router reduces the value of TTL by one and TTL values becomes 0. In this case, router 1 generates the time-exceeded message and host A gets to know that router 1 is the first router in a path.

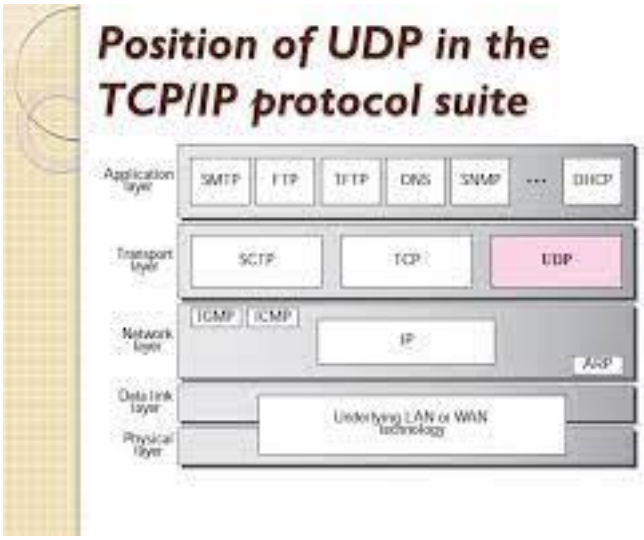
TTL value=2: When host A sends the packet to router 1 with TTL value 2, and when the packet reaches to router 1 then the TTL value gets decremented by 1 and the TTL value becomes 1. Then router 1 sends the packet to router 2, and the TTL value becomes 0, so the router generates a time-exceeded message. The host A gets to know that router 2 is the second router on the path.

TTL value=3: When host A sends the packet to router 1 with TTL value 3, then the router decrements its value by one, and the TTL value becomes 2. Then, router 1 sends the packet to router 2, and the TTL value becomes 1. Then, router 2 sends the packet to router 3, and the TTL value becomes 0. As TTL value becomes 0, router 3 generates a time-exceeded message. In this way, host A is the third router on a path.

UDP (User Datagram Protocol): UDP stands for User Datagram Protocol. The David P. Reed developed the UDP protocol in 1980. The UDP protocol allows the computer applications to send the messages in the form of datagrams from one machine to another machine over the Internet Protocol (IP) network. The UDP is an alternative communication protocol to the TCP protocol (transmission control protocol). Like TCP, UDP provides a set of rules that governs how the data should be exchanged over the internet. The UDP works by encapsulating the data into the packet and providing its own header information to the packet. Then, this UDP packet is encapsulated to the IP packet and sent off to its destination. Both the TCP and UDP protocols send the data over the internet protocol network, so it is also known as TCP/IP and UDP/IP.

It is said to be an unreliable transport protocol but it uses IP services which provides best effort delivery mechanism. In UDP, the receiver does not generate an acknowledgement of packet received and in turn, the sender does not wait for any acknowledgement of packet sent. This shortcoming makes this protocol unreliable as well as easier on processing.

Position of UDP in the TCP/IP protocol suite: Figure shows the relationship of the **User Datagram Protocol** (UDP) to the other protocols and layers of the TCP/IP protocol suite: UDP is located between the application layer and the IP layer, and serves as the intermediary between the application programs and the network operations.



UDP Features:

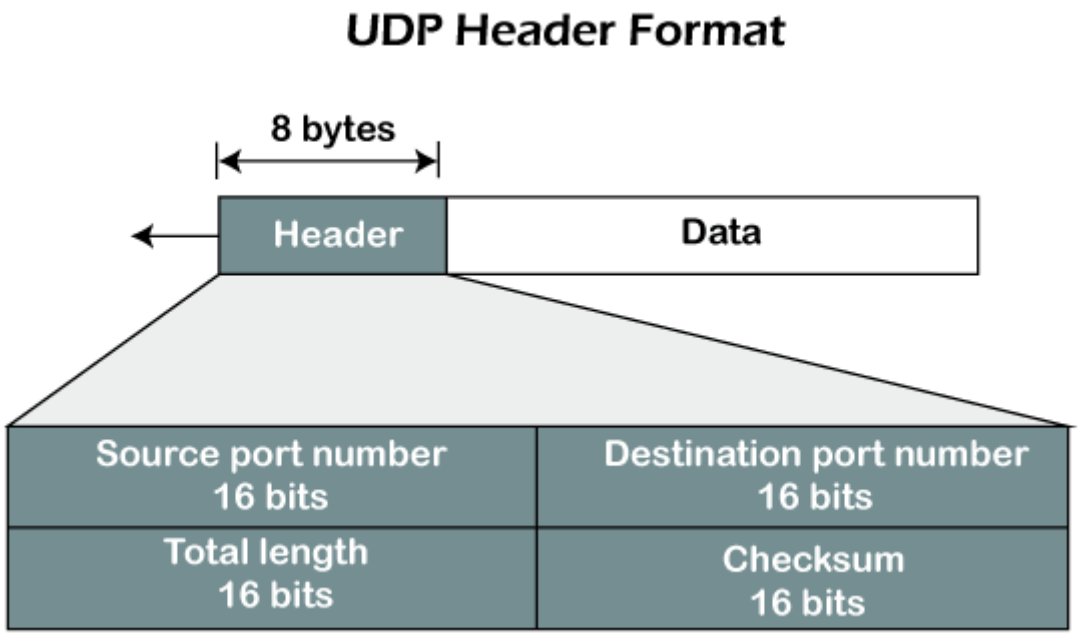
1. **Transport layer protocol:** UDP is the simplest transport layer communication protocol. It contains a minimum amount of communication mechanisms.
2. **Connectionless:** The UDP is a connectionless protocol as it does not create a virtual path to transfer the data. It does not use the virtual path, so packets are sent in different paths between the sender and the receiver, which leads to the loss of packets or received out of order.
3. **Ordered delivery of data is not guaranteed: T**he datagrams are sent in some order will be received in the same order is not guaranteed as the datagrams are not numbered.

4. **Ports:** The UDP protocol uses different port numbers so that the data can be sent to the correct destination. The port numbers are defined between 0 and 1023.
5. **Faster transmission:** UDP enables faster transmission as it is a connectionless protocol, i.e., no virtual path is required to transfer the data.
6. **Acknowledgment mechanism:** The UDP does have any acknowledgment mechanism, i.e., there is no handshaking between the UDP sender and UDP receiver. UDP is used when acknowledgement of data does not hold any significance.
7. **Segments are handled independently:** Each UDP segment is handled individually of others as each segment takes different path to reach the destination. The UDP segments can be lost or delivered out of order to reach the destination as there is no connection setup between the sender and the receiver.
8. **Stateless:** It is a stateless protocol that means that the sender does not get the acknowledgement for the packet which has been sent.
9. UDP is good protocol for data flowing in one direction.
10. UDP does not provide congestion control mechanism.

Why do we require the UDP protocol?

As we know that the UDP is an unreliable protocol, but we still require a UDP protocol in some cases. The UDP is deployed where the packets require a large amount of bandwidth along with the actual data. For example, in video streaming, acknowledging thousands of packets is troublesome and wastes a lot of bandwidth. In the case of video streaming, the loss of some packets couldn't create a problem, and it can also be ignored.

UDP Header Format:



The UDP header contains four fields:

- **Source port number:** It is 16-bit information that identifies which port is going to send the packet.
- **Destination port number:** It identifies which port is going to accept the information.
- **Length:** It is 16-bit field that specifies the entire length of the UDP packet that includes the header also. The minimum value would be 8-byte as the size of the header is 8 bytes.
- **Checksum:** It is a 16-bits field, and it is an optional field. This checksum field checks whether the information is accurate or not as there is the possibility that the information can be corrupted while transmission. It is an optional field, which means that it depends upon the application, whether it wants to write the checksum or not. If it does not want to write the checksum, then all the 16 bits are zero. In UDP, the checksum field is applied to the entire packet, i.e., header as well as data part whereas, in IP, the checksum field is applied to only the header field.

UDP SERVICES:

1. **Process-to-Process Communication:** UDP provides process-to-process communication
2. **Connectionless Services:** UDP provides a *connectionless service.* This means that each
3. 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.
4. **Flow Control:** UDP is a very simple protocol. There is no *flow control*, and hence no window mechanism. The receiver may overflow with incoming messages.
5. **Error Control:** 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.
6. **Checksum:** UDP checksum calculation is different from the one for IP. Here the checksum includes three sections: a pseudo header, the UDP header, and the data coming from the application layer.
7. **Congestion Control:** Since UDP is a connectionless protocol, it does not provide congestion control. UDP assumes that the packets sent are small and sporadic, and cannot create congestion in the network. This assumption may or may not be true today when UDP is used for real-time transfer of audio and video.
8. **Encapsulation and Decapsulation:** To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages.
9. **Queuing:** In UDP, queues are associated with ports. At the client site, when a process starts, it requests a port number from the operating system. Some implementations create both an incoming and an outgoing queue associated with each process. Other implementations create only an incoming queue associated with each process.
10. **Multiplexing and Demultiplexing:** In a host running a TCP/IP protocol suite, there is only one UDP but possibly several processes that may want to use the services of UDP. To handle this situation, UDP multiplexes and multiplexes

- **Multiplexing:** At the sender site, there may be several processes that need to send user datagrams. However, there is only one UDP. This is a many-to-one relationship and requires multiplexing. UDP accepts messages from different processes,

differentiated by their assigned port numbers. After adding the header, UDP passes the user datagram to IP.

- **Demultiplexing:** At the receiver site, there is only one UDP. However, we may have many processes that a receive user datagrams. This is a one-to-many relationship and requires demultiplexing.

.

Limitations:

- It provides an unreliable connection delivery service. It does not provide any services of IP except that it provides process-to-process communication.
- The UDP message can be lost, delayed, duplicated, or can be out of order.
- It does not provide a reliable transport delivery service. It does not provide any acknowledgment or flow control mechanism. However, it does provide error control to some extent.

Advantages:

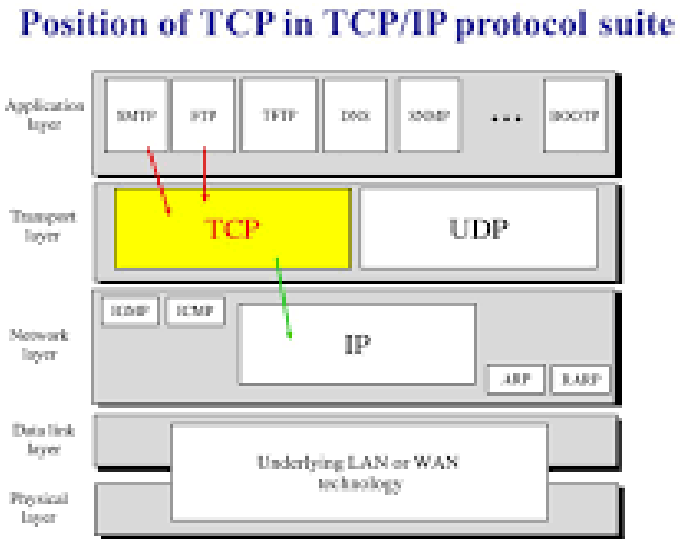
- It produces a minimal number of overheads.

UDP Applications:

- Used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control.
- It is a suitable protocol for multicasting as UDP supports packet switching.
- UDP is used for some routing update protocols like RIP (Routing Information Protocol).
- Normally used for real-time applications which cannot tolerate uneven delays between sections of a received message?
- Following implementations uses UDP as a transport layer protocol:
    - NTP (Network Time Protocol)
    - DNS (Domain Name Service)
    - BOOTP, DHCP.
    - NNP (Network News Protocol)
    - Quote of the day protocol
    - Trivial File Transfer Protocol
    - Routing Information Protocol
    - Kerberos
- The application layer can do some of the tasks through UDP-
    - Trace Route
    - Record Route
    - Timestamp
- UDP takes a datagram from Network Layer, attaches its header, and sends it to the user. So, it works fast.
- Actually, UDP is a null protocol if you remove the checksum field.
    - Reduce the requirement of computer resources.
    - When using the Multicast or Broadcast to transfer.
    - The transmission of Real-time packets, mainly in multimedia applications

TCP (Transmission Control Protocol): TCP is one of the basic standards that define the rules of the internet and is included within the standards defined by the Internet Engineering Task Force (IETF). It is one of the most commonly used protocols within digital network communications and ensures end-to-end data delivery. TCP (Transmission Control Protocol) is a reliable transport protocol as it establishes a connection before sending any data and everything that it sends is acknowledged by the receiver.

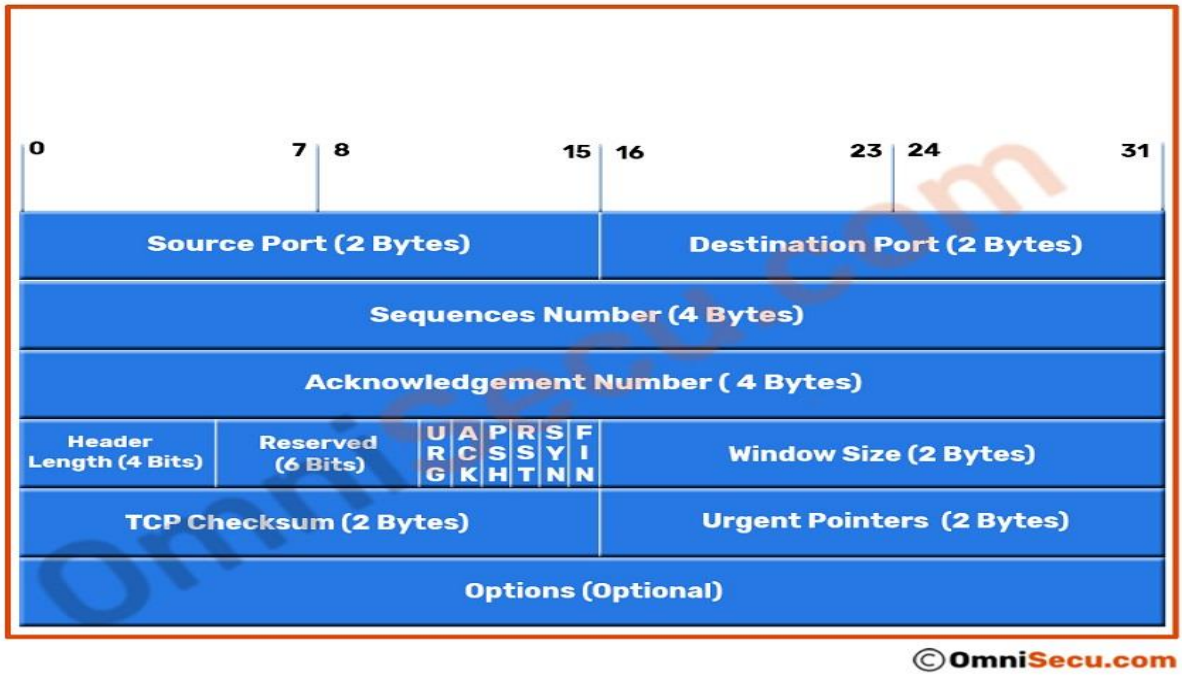
Position of TCP in the TCP/IP protocol suite: Figure shows the relationship of the **Transmission Control Protocol** (TCP) to the other protocols and layers of the TCP/IP protocol suite.



Layers of TCP: The transmission process comprises four layers:

1. **Application layer:** The **application layer** performs the function similar to the top three layers (application, presentation, and session) of the OSI model and control user-interface specifications. The user interacts with the application layer of the TCP model, such as messaging and email systems
2. **Transport layer:** The **transport layer** provides a reliable and error-free data connection. It divides the data received from the application layer into packets, which helps in creating ordered sequence
3. **Internet layer**: The **internet layer** controls the routing of packet and ensures the delivery of a packet at the destination
4. **Data link layer:** The data link layer performs the function similar to the bottom two layers (data link and physical) of the OSI model. It is responsible for transmitting the data between the applications or devices in the network.

TCP Header:



Source port : 16 Bit number which identifies the Source Port number (Sending Computer's TCP Port).

Destination port : 16 Bit number which identifies the Destination Port number (Receiving Port).

Sequence number: the sequence number is a 32 bit field that indicates how much data is sent during the TCP session. When you establish a new TCP connection (3 way handshakes) then the initial sequence number is a random 32 bit value. The receiver will use this sequence number and sends back an acknowledgment

Acknowledgment Number: this 32 bit field is used by the receiver to request the next TCP segment. This value will be the sequence number incremented by 1.

Header Length: 4 Bit field which shows the number of 32 Bit words in the header. Also known as the Data Offset field. The minimum size header is 5 words (binary pattern is 0101).

Reserved: Always set to 0 (Size 6 bits).

Control Bit Flags: We have seen before that TCP is a Connection Oriented Protocol. The meaning of Connection Oriented Protocol is that, before any data can be transmitted, a reliable connection must be obtained and acknowledged. Control Bits govern the entire process of connection establishment, data transmissions and connection termination. The control bits are listed as follows: They are:

URG: Urgent Pointer.

ACK: Acknowledgement.

PSH: This flag means Push function. Using this flag, TCP allows a sending application to specify that the data must be pushed immediately. When an application requests the TCP to push data, the TCP should send the data that has accumulated without waiting to fill the segment.

RST: Reset the connection. The RST bit is used to RESET the TCP connection due to unrecoverable errors.

SYN: This flag means synchronize sequence numbers. Source is beginning a new counting sequence

FIN: No more data from the sender. To close a TCP connection gracefully, applications use the FIN flag.

Window: Indicates the size of the receive window, which specifies the number of bytes beyond the sequence number in the acknowledgment field that the receiver is currently willing to receive.

Checksum: The 16-bit checksum field is used for error-checking of the header and data.

Urgent Pointer: Shows the end of the urgent data so that interrupted data streams can continue. When the URG bit is set, the data is given priority over other data streams (Size 16 bits).

Services and Segment structure in TCP: The Transmission Control Protocol is the most common transport layer protocol. It works together with IP and provides a reliable transport service between processes using the network layer service provided by the IP protocol.
The various services provided by the TCP to the application layer are as follows:

1. **Process-to-Process Communication:** TCP provides a process to process communication, i.e., the transfer of data that takes place between individual processes executing on end systems. This is done using port numbers or port addresses.

2. **Stream oriented –** This means that the data is sent and received as a stream of bytes (unlike UDP or IP that divides the bits into datagrams or packets). TCP groups a number of bytes together into a *segment* and adds a header to each of these segments and then delivers these segments to the network layer. At the network layer, each of these segments is encapsulated in an IP packet for transmission. The TCP header has information that is required for control purposes which will be discussed along with the segment structure.

3. **Full-duplex service –** This means that the communication can take place in both directions at the same time.

4. **Connection-oriented service –** TCP provides a connection-oriented service. It defines 3 different phases:
   - Connection establishment
   - Data transfer

- Connection termination

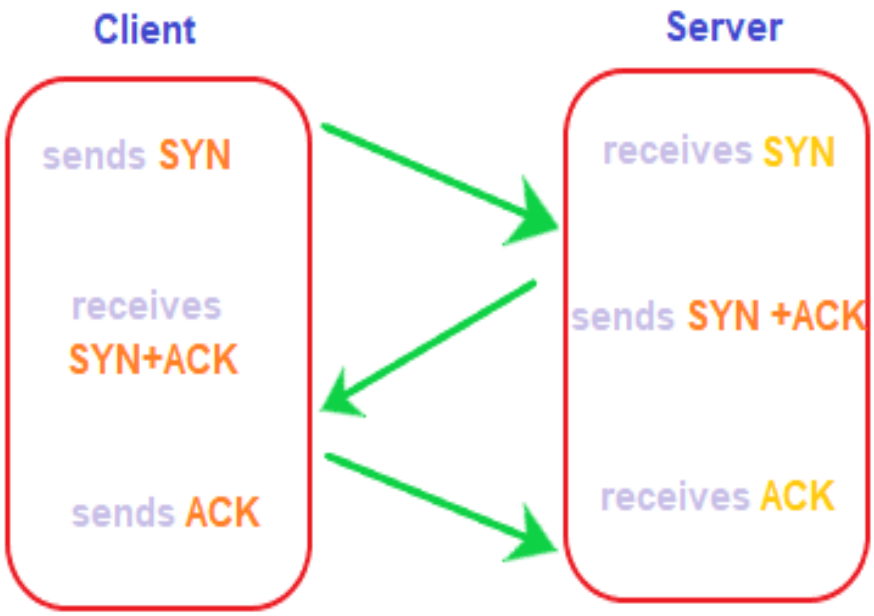
5. **Reliability –** TCP is reliable as it uses checksum for error detection, attempts to recover lost or corrupted packets by re-transmission, acknowledgement policy and timers. It uses features like byte number and sequence number and acknowledgement number so as to ensure reliability. Also, it uses congestion control mechanisms.

6. **Multiplexing and Demultiplexing –** TCP does multiplexing and de-multiplexing at the sender and receiver ends respectively as a number of logical connections can be established between port numbers over a physical connection.

TCP Connection: TCP is a connection-oriented protocol and every connection-oriented protocol needs to establish a connection in order to reserve resources at both the communicating ends. After both the connections are verified, it begins transmitting packets. The connection establishment phase of TCP makes use of 3 packets, it is also known as 3-way Handshaking **(SYN, SYN + ACK, ACK)**. SYN means **synchronize Sequence Number** and ACK means **acknowledgment**. Each step is a type of handshake between the sender and the receiver.

The reliable communication in TCP is termed as **PAR** (Positive Acknowledgement Re-transmission). When a sender sends the data to the receiver, it requires a positive acknowledgement from the receiver confirming the arrival of data. If the acknowledgement has not reached the sender, it needs to resend that data. The positive acknowledgement from the receiver establishes a successful connection.

The diagram of a successful TCP connection showing the three handshakes is shown below:



The three handshakes are discussed in the below steps:

Step 1: SYN: SYN is a segment sent by the client to the server. It acts as a **connection request** between the client and server. It informs the server that the client wants to establish a connection.

Step 2: SYN-ACK: It is an SYN-ACK segment or an SYN + ACK segment sent by the server. The ACK segment informs the client that the server has received the connection request and it is ready to build the connection. The SYN segment informs the sequence number with which the server is ready to start with the segments.

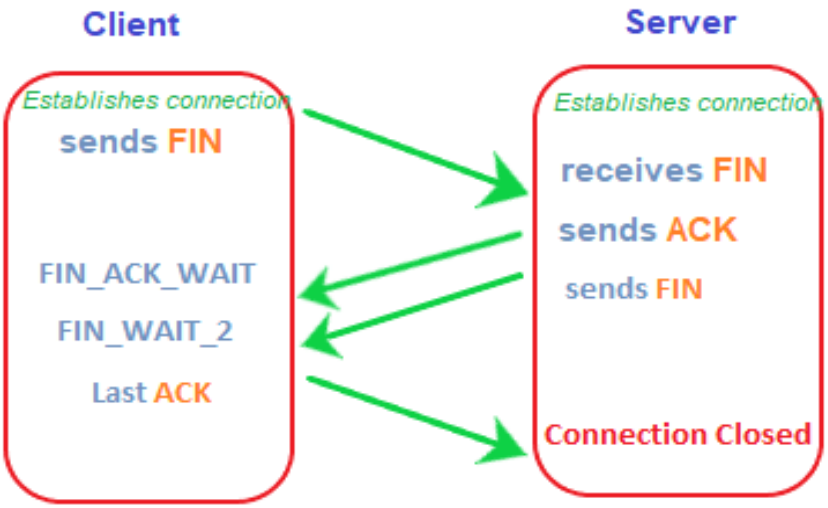
Step 3: ACK: ACK (Acknowledgment) is the last step before establishing a successful TCP connection between the client and server. The ACK segment is sent by the client as the response of the received ACK and SYN from the server. It results in the establishment of a reliable data connection.

After these three steps, the client and server are ready for the data communication process..

TCP Termination (A 4-way handshake): Any device establishes a connection before proceeding with the termination. TCP requires 3-way handshake to establish a connection between the client and server before sending the data. Similarly, to terminate or stop the data transmission, it requires a 4-way handshake. The segments required for TCP termination are similar to the segments to build a TCP connection (ACK and SYN) except the FIN segment. The FIN segment specifies a termination request sent by one device to the other.

Consider the below TCP termination diagram that shows the exchange of segments between the client and server.

The diagram of a successful TCP termination showing the four handshakes is shown below:



The TCP termination process of six steps that includes the sent requests and the waiting states are as follows:

Step 1: FIN: FIN refers to the **termination request** sent by the client to the server. The first FIN termination request is sent by the client to the server. It depicts the start of the termination process between the client and server.

Step 2: FIN_ACK_WAIT: The client waits for the ACK of the FIN termination request from the server. It is a **waiting state** for the client.

Step 3: ACK: The server sends the ACK (Acknowledgement) segment when it receives the FIN termination request. It depicts that the server is ready to close and terminate the connection.

Step 4: FIN _WAIT_2: The client waits for the FIN segment from the server. It is a type of approved signal sent by the server that shows that the server is ready to terminate the connection.

Step 5: FIN: The FIN segment is now sent by the server to the client. It is a confirmation signal that the server sends to the client. It depicts the successful approval for the termination.

Step 6: ACK: The client now sends the ACK (Acknowledgement) segment to the server that it has received the FIN signal, which is a signal from the server to terminate the connection. As soon as the server receives the ACK segment, it terminates the connection.

TCP Retransmission- When sender discovers that the segment sent by it is lost; it retransmits the same segment to the receiver. This is called as **TCP retransmission**.

Sender discovers that the TCP segment is lost when-

1. Either Time Out Timer expires
2. Or it receives three duplicate acknowledgements

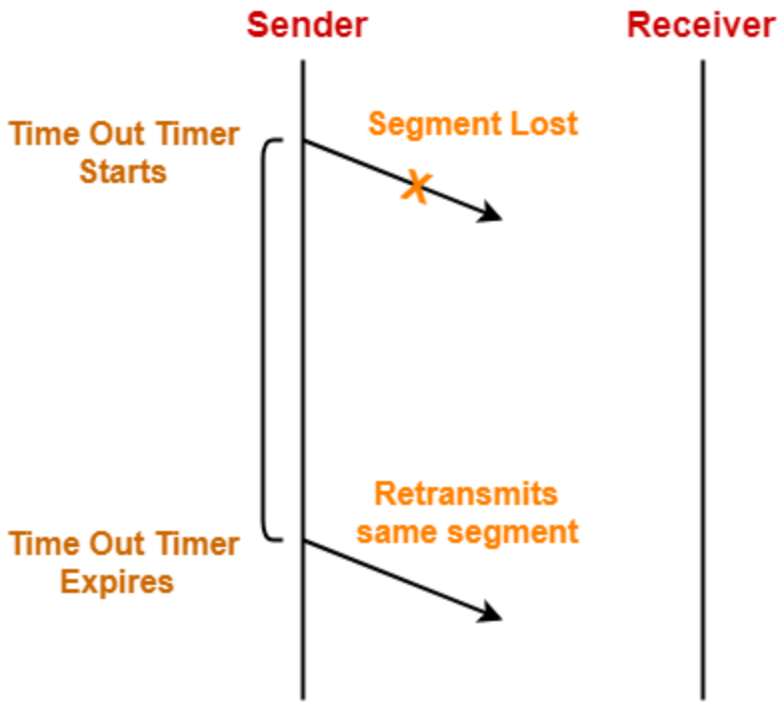
1. Retransmission After Time Out Timer Expiry- Each time sender transmits a TCP segment to the receiver, it starts a Time Out Timer. Now, following two cases are possible-

Case-01:

- Sender receives an acknowledgement for the sent segment before the timer goes off.
- In this case, sender stops the timer.

Case-02:

- Sender does not receives any acknowledgement for the sent segment and the timer goes off.
- In this case, sender assumes that the sent segment is lost.
- Sender retransmits the same segment to the receiver and resets the timer.



Retransmission after Time Out Timer Expiry

2. Retransmission After Receiving 3 Duplicate Acknowledgements-

- Consider sender receives three duplicate acknowledgements for a TCP segment sent by it.
- Then, sender assumes that the corresponding segment is lost.
- So, sender retransmits the same segment without waiting for its time out timer to expire.
- This is known as **Early retransmission** or **Fast retransmission**.

Consider-

- Sender sends 5 TCP segments to the receiver.
- The second TCP segment gets lost before reaching the receiver.

The sequence of steps taking place are-

- On receiving segment-1, receiver sends acknowledgement asking for segment-2 next.

(Original ACK)

- On receiving segment-3, receiver sends acknowledgement asking for segment-2 next.

(1st duplicate ACK)

- On receiving segment-4, receiver sends acknowledgement asking for segment-2 next.

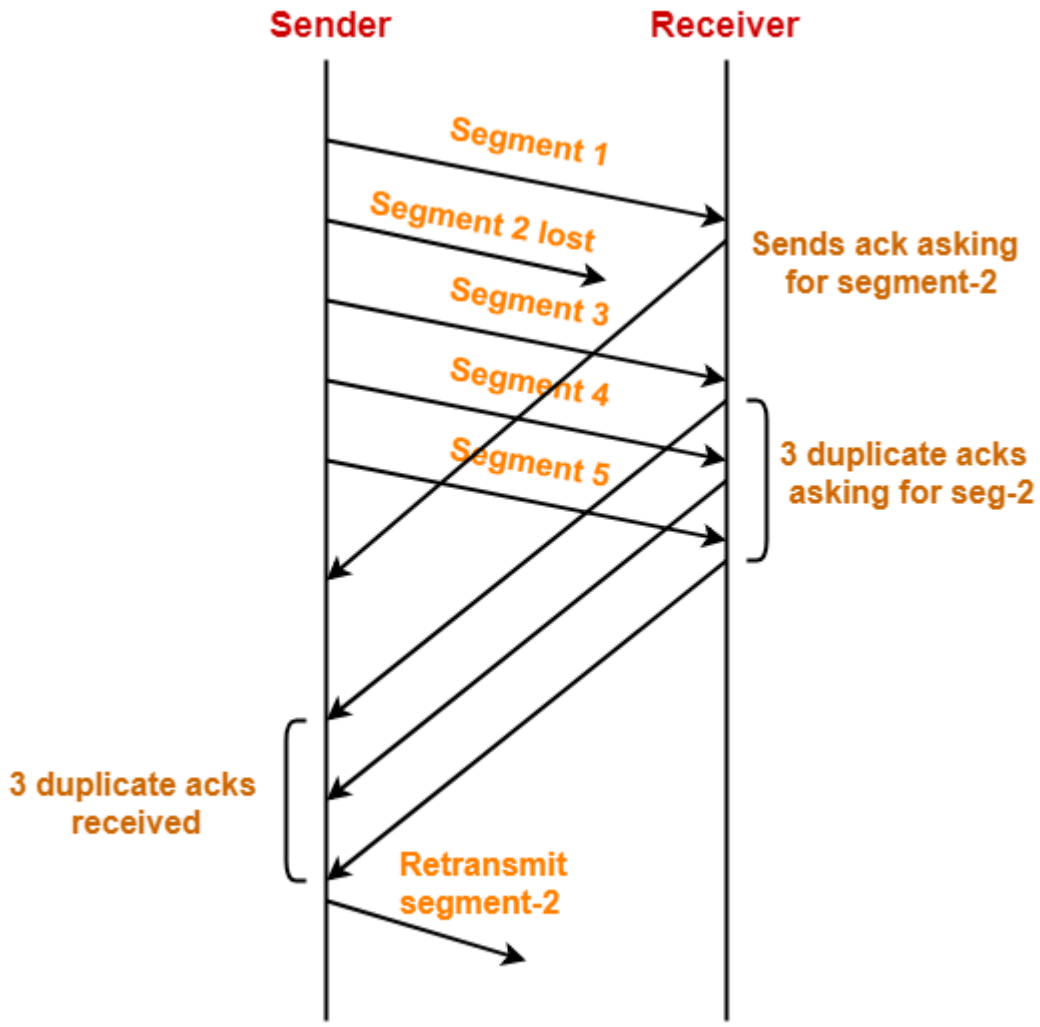
(2nd duplicate ACK)

- On receiving segment-5, receiver sends acknowledgement asking for segment-2 next.

(3rd duplicate ACK)

Now,

- Sender receives 3 duplicate acknowledgements for segment-2 in total.
- So, sender assumes that the segment-2 is lost.
- So, it retransmits segment-2 without waiting for its timer to go off.



Retransmission after receiving 3 duplicate acks

After receiving the retransmitted segment-2,

- Receiver does not send the acknowledgement asking for segment-3 or 4 or 5.
- Receiver sends the acknowledgement asking for segment-6 directly from the sender.

This is because previous segments have been already received and acknowledgements for them have been already sent (although wasted in asking for segment-2).

TCP Error control: Error control in TCP is mainly done through the use of **three simple techniques**:

1. **Checksum –** Every segment contains a checksum field which is used to find corrupted segments. If the segment is corrupted, then that segment is discarded by the destination TCP and is considered lost.
2. **Acknowledgement –** TCP has another mechanism called acknowledgement to affirm that the data segments have been delivered. Control segments that contain no data but have sequence numbers will be acknowledged as well but ACK segments are not acknowledged.
3. **Retransmission –** When a segment is missing, delayed to deliver to a receiver, corrupted when it is checked by the receiver then that segment is retransmitted again. Segments are retransmitted only during two events: when the sender receives three duplicate acknowledgements (ACK) or when a retransmission timer expires.
    - **Retransmission after RTO:** TCP always preserves one retransmission time-out (RTO) timer for all sent but not acknowledged segments. When the timer runs out of time, the earliest segment is retransmitted. Here no timer is set for acknowledgement. In TCP, the RTO value is dynamic in nature and it is updated using the round trip time (RTT) of segments. RTT is the time duration needed for a segment to reach the receiver and an acknowledgement to be received by the sender.
    - **Retransmission after Three duplicate ACK segments:** RTO method works well when the value of RTO is small. If it is large, more time is needed to get confirmation about whether a segment has been delivered or not. Sometimes one segment is lost and the receiver receives so many out-of-order segments that they cannot be saved. In order to solve this situation, three duplicate acknowledgement method is used and missing segment is retransmitted immediately instead of retransmitting already delivered segment. This is a fast retransmission because it makes it possible to quickly retransmit lost segments instead of waiting for timer to end.

TCP Congestion Control: Congestion occurs if the load offered to any network is more than its capacity. It is a network state where - The message traffic becomes so heavy that it slows down the network response time. Congestion is an important issue that can arise in **Packet Switched Network**. It leads to the loss of packets in transit. So, it is necessary to control the congestion in network. TCP uses a congestion window and a congestion policy that avoid congestion.

Congestion Control- Congestion control refers to techniques and mechanisms that can-

- Either prevent congestion before it happens
- Or remove congestion after it has happened

TCP control congestion by means of window Mechanism. TCP reacts to congestion by reducing the sender window size. The size of the sender window is determined by the following two factors-

1. Receiver window size
2. Congestion window size

1. Receiver Window Size- Receiver window size is an advertisement of-

"How much data (in bytes) the receiver can receive without acknowledgement?"

- Sender should not send data greater than receiver window size.
- Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
- So, sender should always send data less than or equal to receiver window size.
- Receiver dictates its window size to the sender through TCP Header.

2. Congestion Window-

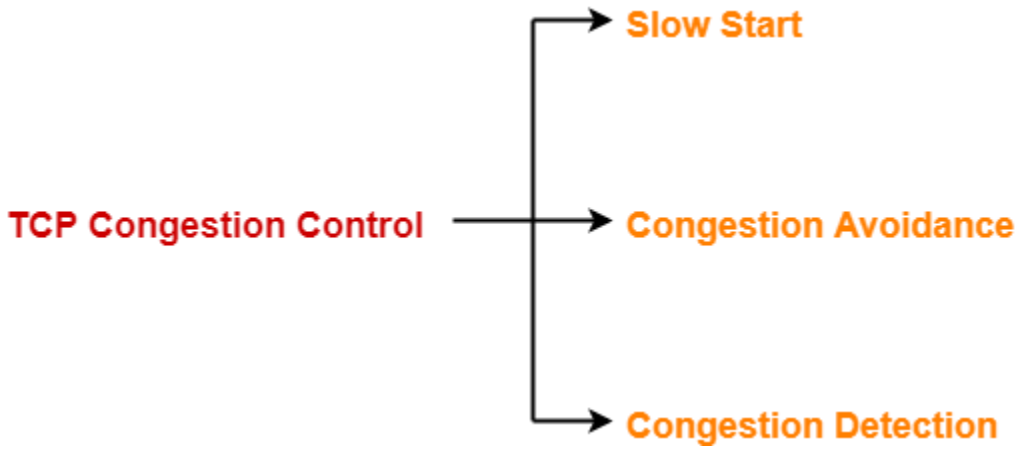
- Sender should not send data greater than congestion window size.
- Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
- So, sender should always send data less than or equal to congestion window size.
- Different variants of TCP use different approaches to calculate the size of congestion window.
- Congestion window is known only to the sender and is not sent over the links

The sender has two pieces of information: the receiver-advertised window size and the congestion window size. The actual size of the window is the minimum of these two.

i.e. Sender window size = Minimum (Receiver window size, Congestion window size)

TCP Congestion Policy-

TCP's general policy for handling congestion consists of following three phases-

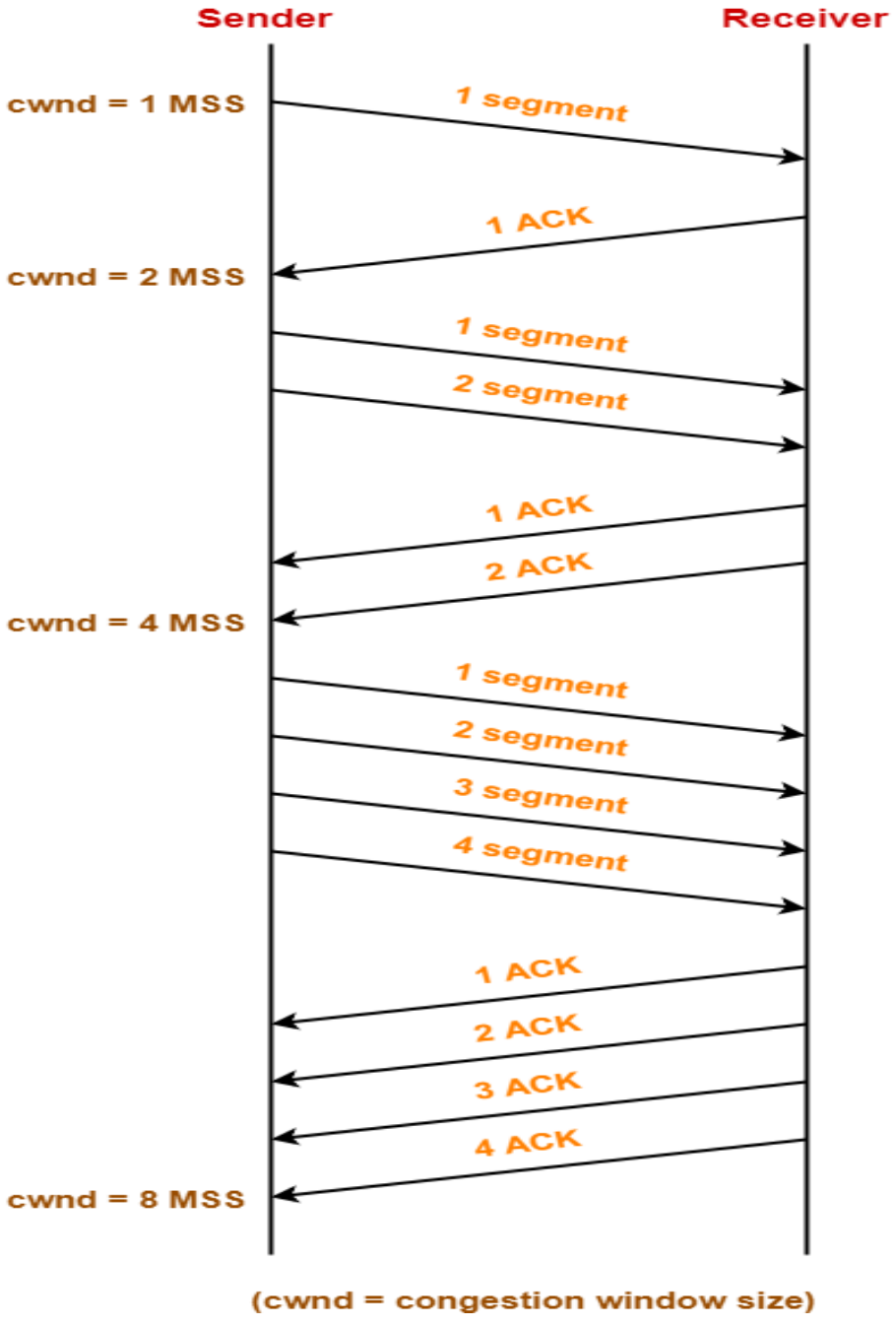


1. Slow Start
2. Congestion Avoidance
3. Congestion Detection

1. Slow Start Phase (*Exponential Increase):* In the slow start algorithm, the size of the congestion window increases exponentially until it reaches a threshold**.** Initially, sender sets congestion window size = Maximum Segment Size (1 MSS). After receiving each acknowledgment, sender increases the congestion window size by 1 exponentially.

If we look at the size of the cwnd in terms of round-trip times (RTTs), we find that the growth rate is exponential as shown below:

- After 1 round trip time, congestion window size = $(2)^1$ = 2 MSS
- After 2 round trip time, congestion window size = $(2)^2$ = 4 MSS
- After 3 round trip time, congestion window size = $(2)^3$ = 8 MSS and so on.



Slow start cannot continue indefinitely. There must be a threshold to stop this phase. The sender keeps track of a variable named *ssthresh* (slow start threshold). When the size of window in bytes reaches this threshold, slow start stops and the next phase start.

Threshold = Maximum number of TCP segments that receiver window can accommodate / 2

= (Receiver window size / Maximum Segment Size) / 2

2. Congestion Avoidance Phase- ***(Additive Increase) :*** To avoid the congestion before it happens it is necessary to slow down the exponential growth. TCP defines another algorithm called **congestion avoidance,** which increases the cwnd additively instead of exponentially

After reaching the threshold,

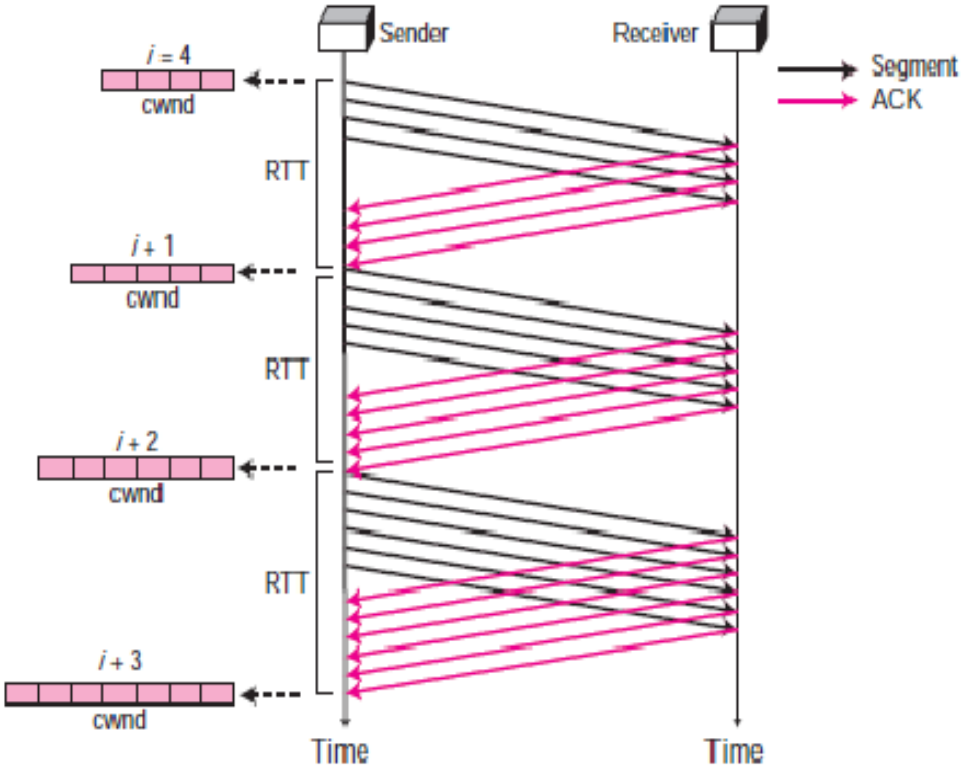
- Sender increases the congestion window size linearly to avoid the congestion.
- On receiving each acknowledgement, sender increments the congestion window size by 1.

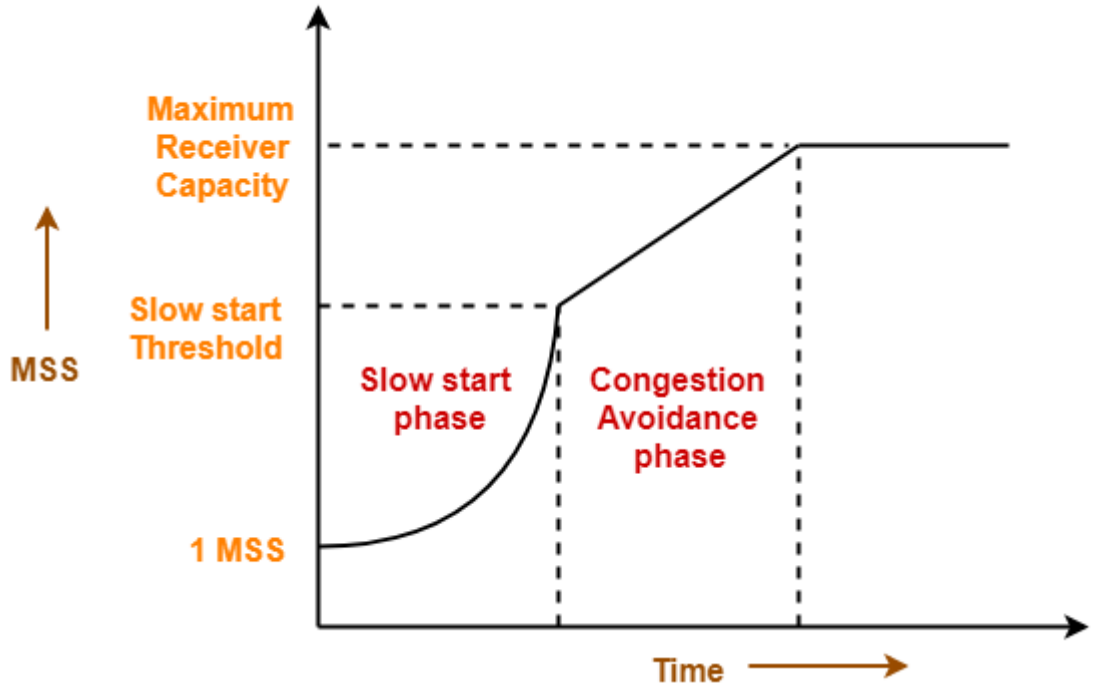
The followed formula is-

Congestion window size = Congestion window size + 1

This phase continues until the congestion window size becomes equal to the receiver window size.

Start → **cwnd** = ***i***
After 1 RTT → **cwnd** = ***i*** + **1**
After 2 RTT → **cwnd** = ***i*** + **2**
After 3 RTT → **cwnd** = ***i*** + **3**





3. **Congestion Detection Phase- (*Multiplicative Decrease*):** If congestion occurs, the congestion window size must be decreased. The only way a sender can guess that congestion has occurred is the need to retransmit a segment. However, retransmission can occur in one of two cases:

- when the RTO timer times out
- or when three duplicate ACKs are received.

In both cases, the size of the threshold is dropped to one half of the previous window size. Most TCP implementations have two reactions:

1. If a time-out occurs, there is a stronger possibility of congestion; a segment has probably been dropped in the network and there is no news about the following sent segments. In this case TCP reacts strongly:

a. It sets the value of the threshold to half of the current window size.
b. It reduces cwnd back to one segment.
c. It starts the slow start phase again.

2. If three duplicate ACKs are received, there is a weaker possibility of congestion; a segment may have been dropped but some segments after that have arrived safely since three duplicate ACKs are received. This is called fast transmission and fast recovery. In this case, TCP has a weaker reaction as shown below:

a. It sets the value of the threshold to half of the current window size.
b. It sets cwnd to the value of the threshold
c. It starts the congestion avoidance phase.

TCP Flow Control: Flow Control balances the rate a producer creates data with the rate a consumer can use the data. Flow control ensures that sender won't overflow receiver's buffer by transmitting too much, too fast. Transmission Control Protocol (TCP) uses a *sliding window* for flow control.

Transmission Control Protocol (TCP) uses a sliding window for flow control.

The TCP sliding window determines the number of unacknowledged bytes, x, that one system can send to another. **Two factors determine the value of x:**

- The size of the send buffer on the sending system
- The size and available space in the receive buffer on the receiving system

The sending system cannot send more bytes than space that is available in the receive buffer on the receiving system. TCP on the sending system must wait to send more data until all bytes in the current send buffer are acknowledged by TCP on the receiving system.

On the receiving system, TCP stores received data in a receive buffer. TCP acknowledges receipt of the data, and advertises (communicates) a new receive window to the sending system. The receive window represents the number of bytes that are available in the receive buffer. If the receive buffer is full, the receiving system advertises a receive window size of zero, and the sending system must wait to send more data. After the receiving application retrieves data from the receive buffer, the receiving system can then advertise a receive window size that is equal to the amount of data that was read. Then, TCP on the sending system can resume sending data.

The available space in the receive buffer depends on how quickly data is read from the buffer by the receiving application. TCP keeps the data in its receive buffer until the receiving application reads it from that buffer. After the receiving application reads the data, that space in the buffer is available for new data. The amount of free space in the buffer is advertised to the sending system, as described in the previous paragraph.

Ensure that you understand the TCP window size when you use sliding window for flow control. The window size is the amount of data that can be managed. You might need to adjust the window size if the receive buffer receives more data than it can communicate.

Working Principle

In these protocols, the sender has a buffer called the sending window and the receiver has buffer called the receiving window.

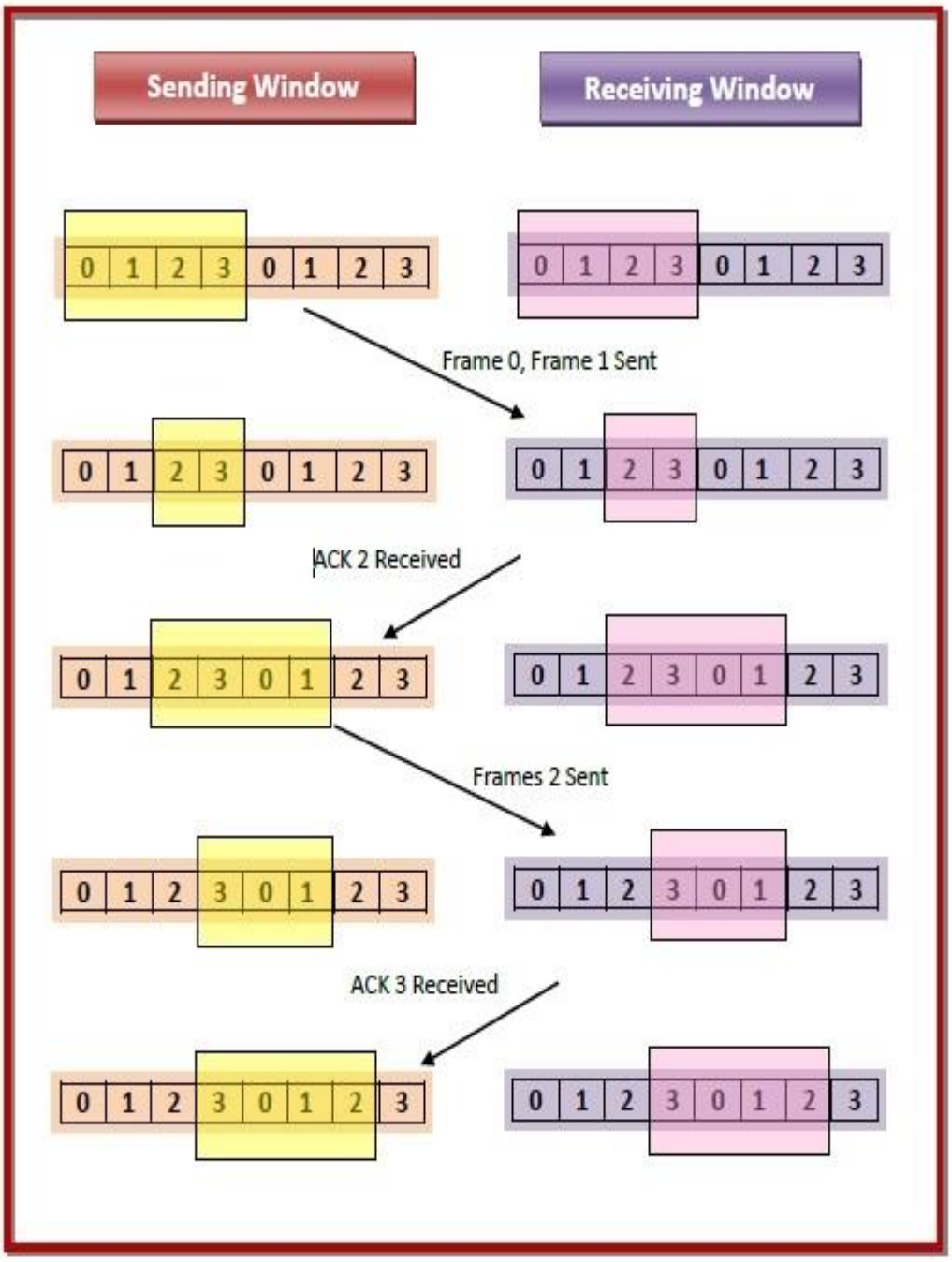
The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an n-bit field, then the range of sequence numbers that can be assigned is 0 to 2^n-1. Consequently, the size of the sending window is 2^n-1. Thus in order to accommodate a sending window size of 2^n-1, a n-bit sequence number is chosen.

The sequence numbers are numbered as modulo-n. For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.

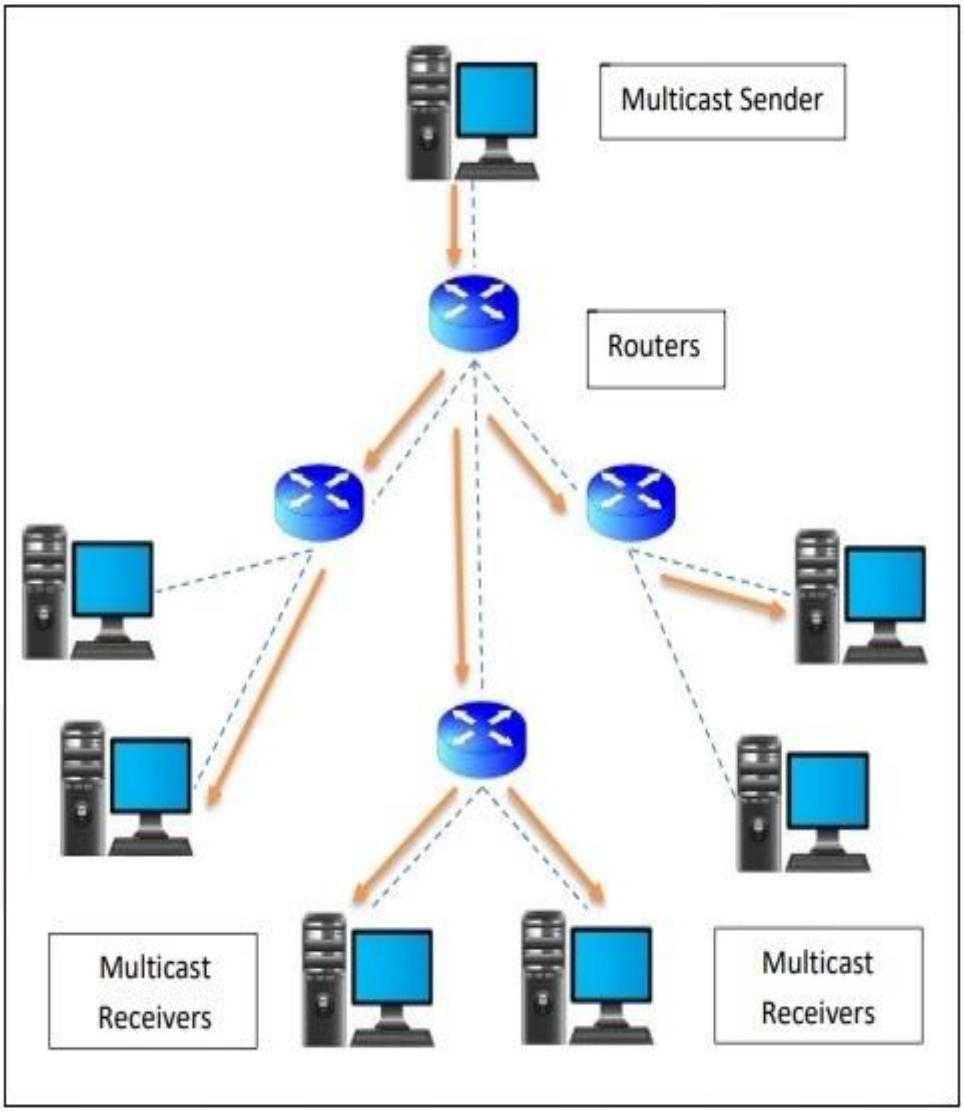
The size of the receiving window is the maximum number of frames that the receiver can accept at a time. It determines the maximum number of frames that the sender can send before receiving acknowledgment.

Example

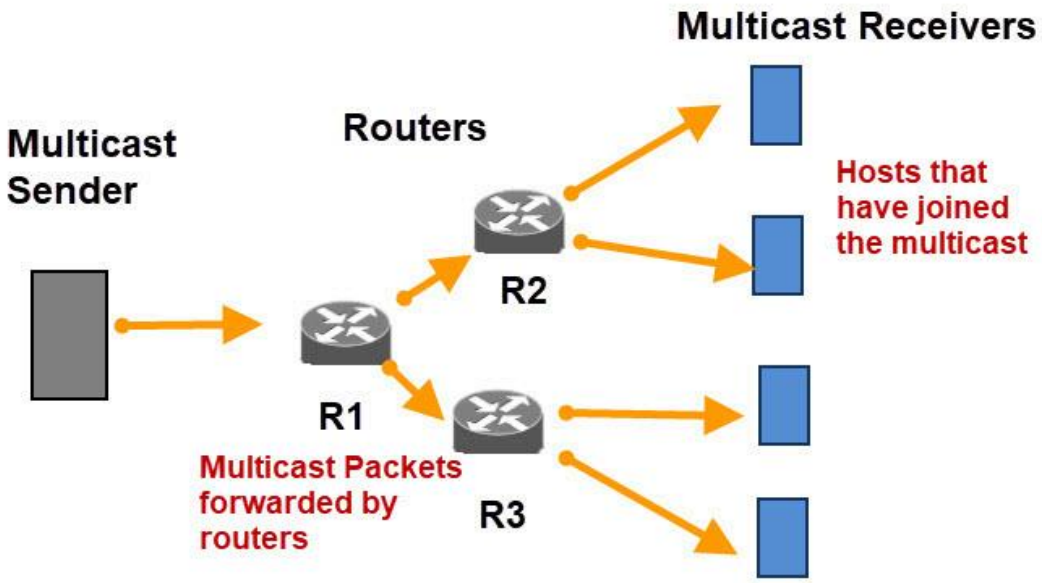
Suppose that we have sender window and receiver window each of size 4. So the sequence numbering of both the windows will be 0,1,2,3,0,1,2 and so on. The following diagram shows the positions of the windows after sending the frames and receiving acknowledgments.



Multicasting: Multicasting is a type of one-to-many and many-to-many communication as it allows sender or senders to send data packets to multiple receivers at once across LANs or WANs. **Multicasting** works in similar to Broadcasting, but in Multicasting, the information is sent to the targeted or specific members of the network.



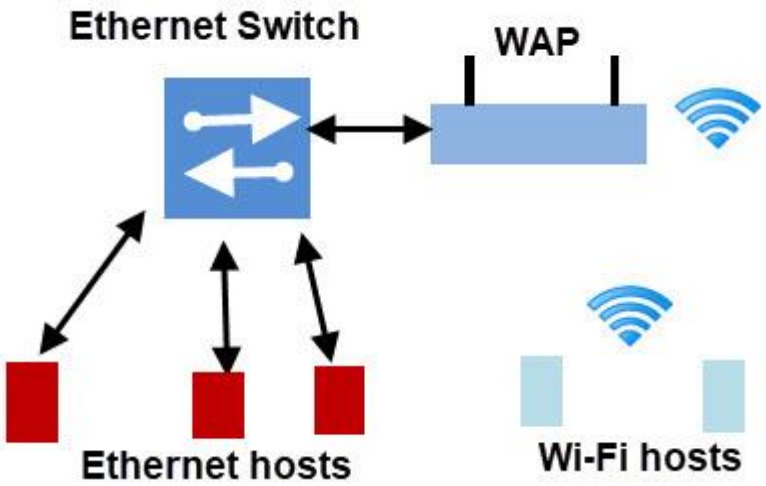
IP Multicast: IP multicasting allows a host to send a single packet to thousands of hosts across a routed network i.e. The Internet. The destination nodes send join and leave messages that informs the routers whether they are correct recipients of the messages. The sender sends the data packet only once irrespective of the number of users. The routers in the network perform necessary replications so that the packet can reach multiple receivers simultaneously. IPv4 Multicast addresses use the reserved **class D**. Multicast uses **UDP** and are sent through switches and hubs. To receive a multicast message a host must be configured to receive on that multicast address. On the Internet multicast packets need to be **forwarded** by routers. Multicast routers **do not keep track** of which hosts are parts of a group, but only need to know if any hosts on that subnet are part of a group. If a router receives a multicast datagram from another network and has no members for that group address on any of its subnets it drops the packet.



Multicasting on the Internet

Ethernet Multicast: Ethernet multicast constitutes multicasting at the data link layer of the OSI model for Ethernet networks. It is used mainly for audio (radio) and video distribution. On a small home or office network any host can send and receive multicast datagrams.

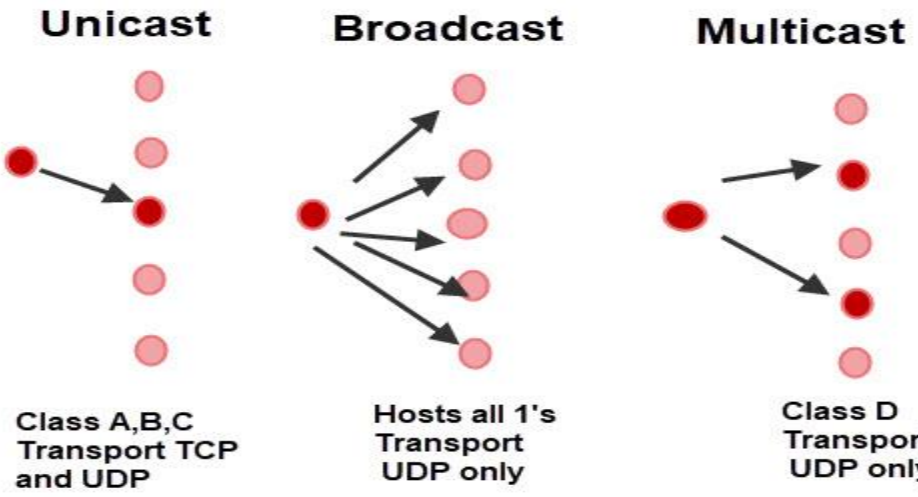
Multicasting on a Home Network



A multicast datagram sent from any host can be received by any host on the network configured to receive messages on that multicast address. A host can send and receive on multiple multicast addresses.

In Networking a packet can be sent to:

- A single host –**Unicast = (TCP and UDP)**
- All hosts **-Broadcast – (UDP only)**
- A group of hosts – **Multicast -(UDP only)**



Unicast,Broadcast and Multicast IP Addressing

Broadcasts vs. Multicasts: Multicasting is different from IP broadcasting as:

- Broadcasting uses a single IP address. Host bits set to all 1's. There are a range of multicast addresses
- Broadcast messages are not sent through routers but multicast messages are.
- All hosts will receive broadcasts by default
- A host must be configured to receive multicast messages.