UNIT 5

Transport layer: UDP, TCP

Introduction to transport layer

The transport layer is part of the TCP/IP networking model, sometimes called the **networking architecture**. It contains a comprehensive set of documents that describes everything required for a computer network to function.

The transport layer is responsible for the logical communication between applications running on different hosts, thus providing services to application layer protocols on a higher layer of the TCP/IP network model.

Even though many transport layer protocols exist, the two most commonly used protocols are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

These protocols provide different functionalities for different application requirements.

A few of the most important functionalities are:

- Tracking of individual conversation.
- Ordered data transfer and data segmentation.
- Multiplexing conversation using port numbers.

TRANSMISSION CONTROL PROTOCOL (TCP)

In terms of the OSI model, TCP is a transport-layer protocol. It provides a reliable virtual-circuit connection between applications; that is, a connection is established before data transmission begins. Data is sent without errors or duplication and is received in the same order as it is sent. No boundaries are imposed on the data; TCP treats the data as a stream of bytes.

According to this article, Transmission Control Protocol (TCP) can be defined as a standard that defines how to establish and maintain a network conversation through which application programs can exchange data.

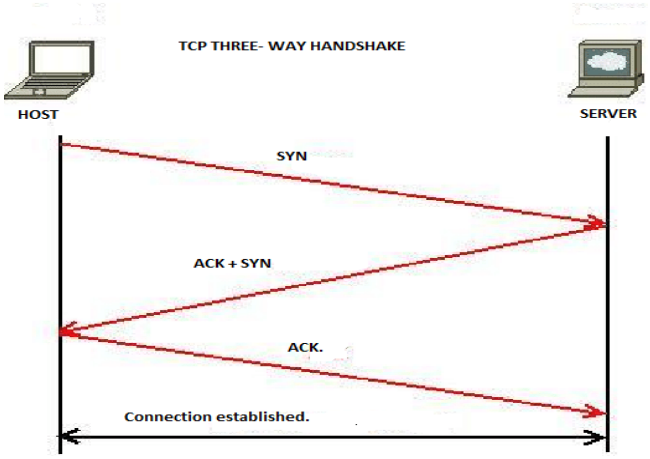
TCP is analogous to sending a package with a tracker that tracks the package from its source to its destination.

<table>
<tr><td colspan="3">Source Port (16 bits)</td><td>Destination Port (16 bits)</td></tr>
<tr><td colspan="4">Sequence Number (32 bits)</td></tr>
<tr><td colspan="4">Acknowledgement Number (32 bits)</td></tr>
<tr><td>Header (4 bits)</td><td>Reserved (6 bits)</td><td>Code Bits (6 bits)</td><td>Window (16bits)</td></tr>
<tr><td colspan="3">Checksum (16bits)</td><td>Urgent (16bits)</td></tr>
<tr><td colspan="4">Options (0 to 32 bits)</td></tr>
</table>

As defined in Request For Comment (RFC) 7913, TCP has the following features:

- Connection establishment and termination.
- Multiplexing using ports.
- Flow control using windowing.
- Error recovery.
- Ordered data transfer and data segmentation.

When the server receives the SYN flag from the host, it sends back another SYN and an ACK flag. This contains a source port number (the port number used as the destination port number on the SYN flag sent by the host) and a destination port number (the port number that the host used as source port number). The host acknowledges those flags' reception with an ACK flag, and a connection is established, thus forming a **three-way handshake.**



The type of transport layer protocol an application chooses to use depends on the application requirement.

User Datagram Protocol (UDP)

UDP is also a transport-layer protocol and is an alternative to TCP. It provides an unreliable datagram connection between applications. Data is transmitted link by link; there is no end-to-end connection. The service provides no guarantees. Data can be lost or duplicated, and datagrams can arrive out of order.

Internet Protocol (IP)

In terms of the OSI model, IP is a network-layer protocol. It provides a datagram service between applications, supporting both TCP and UDP.

USER DATAGRAM PROTOCOL(UDP)

User datagram protocol (UDP) is considered as a best-effort transport protocol because it is a light-weighted transport protocol. UDP is a connectionless protocol, meaning it provides no reliability or reordering of the data segment and flow control like TCP. Because of this, UDP is faster than TCP in transporting data.

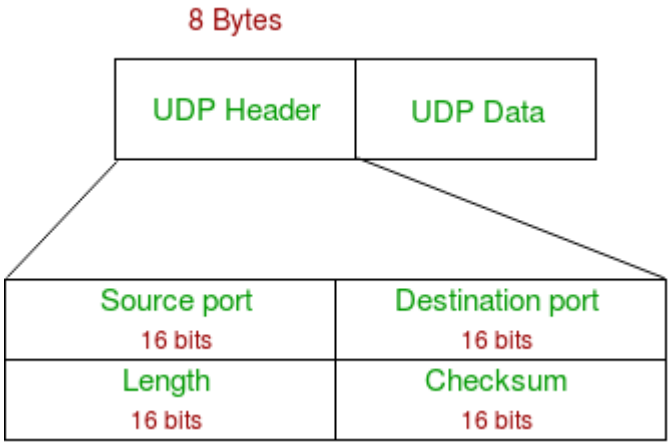
For example, TCP's requirement will make it difficult to stream live video, as all packets must be sent and acknowledged, which will consume many resources and can cause severe delay.

The significant difference between TCP and UDP is that TCP offers a wide range of services to applications, while UDP does not, this does not make UDP inferior to TCP, but by providing fewer services, UDP has fewer bytes in its header, and this makes UDP is faster when transporting data.

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an **unreliable and connectionless protocol.** So, there is no need to establish a connection prior to data transfer.
Through Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of the Internet services; provides assured delivery, reliability, and much more but all these services cost us additional overhead and latency. Here, UDP comes into the picture. For real-time services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also saves bandwidth. User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.

UDP Header –
UDP header is **8-bytes** fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. First 8 Bytes contains all necessary header information and remaining part consist of data. UDP port number fields are each 16

bits long, therefore range for port numbers defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or process.



1. **Source Port :** Source Port is 2 Byte long field used to identify port number of source.
2. **Destination Port :** It is 2 Byte long field, used to identify the port of destined packet.
3. **Length :** Length is the length of UDP including header and the data. It is 16-bits field.
4. **Checksum :** Checksum is 2 Bytes long field. It is the 16-bit one's complement of the one's complement sum of the UDP header, pseudo header of information from the IP header and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Applications of UDP:

- Used for simple request response communication when size of data is less and hence there is lesser concern about flow and error control.
- It is 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 can not 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
    - TFTP, RTSP, RIP.
- Application layer can do some of the tasks through UDP-
    - Trace Route
    - Record Route
    - Time stamp

CONNECTION ESTABLISHMENT AND TERMINATION

TCP Connection Termination

In TCP 3-way Handshake Process we studied that how connection establish between client and server in Transmission Control Protocol (TCP) using **SYN** bit segments. In this article we will study about how TCP close connection between Client and Server. Here we will also need to send bit segments to server which **FIN** bit is set to 1.
TCP supports two types of connection releases like most connection-oriented transport protocols:

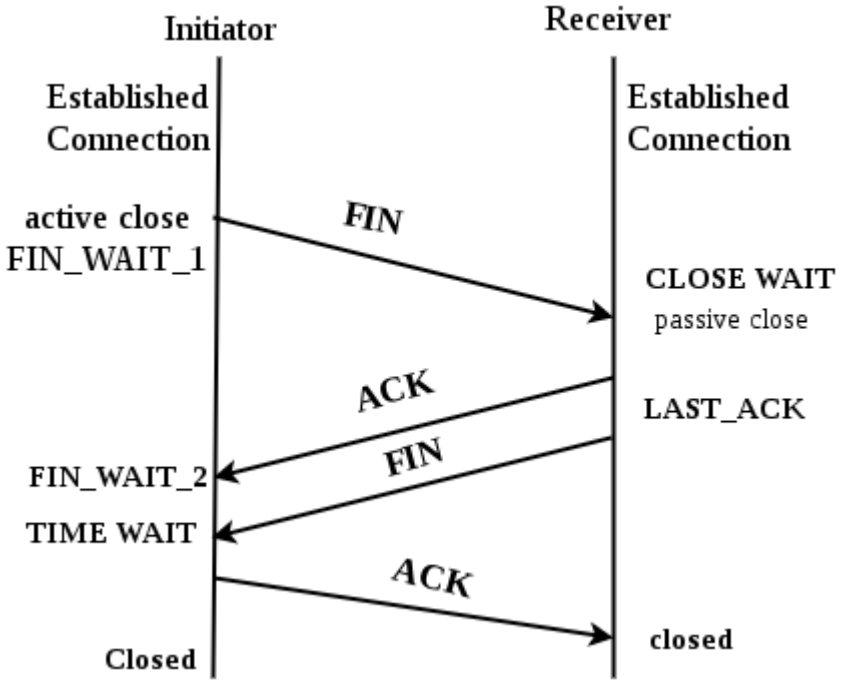
1. **Graceful connection release –**
   In Graceful connection release, the connection is open until both parties have closed their sides of the connection.
2. **Abrupt connection release –**
   In Abrupt connection release, either one TCP entity is forced to close the connection or one user closes both directions of data transfer.

Abrupt connection release :
An abrupt connection release is carried out when a RST segment is sent. A RST segment can be sent for the below reasons:

1. When a non-SYN segment was received for a non-existing TCP connection.
2. In an open connection, some TCP implementations send a RST segment when a segment with an invalid header is received. This will prevent attacks by closing the corresponding connection.
3. When some implementations need to close an existing TCP connection, they send a RST segment. They will close an existing TCP connection for the following reasons:
   - Lack of resources to support the connection
   - The remote host is now unreachable and has stopped responding.

When a TCP entity sends a RST segment, it should contain 00 if it does not belong to any existing connection else it should contain the current value of the sequence number for the connection and the acknowledgment number should be set to the next expected in-sequence sequence number on this connection.

Graceful Connection Release :
The common way of terminating a TCP connection is by using the TCP header's FIN flag. This mechanism allows each host to release its own side of the connection individually.



How mechanism works In TCP :

1. **Step 1 (FIN From Client) –**
   Suppose that the client application decides it wants to close the connection. (Note that the server could also choose to close the connection). This causes the client send a TCP segment with the **FIN** bit set to **1** to server and to enter the **FIN_WAIT_1** state. While in the **FIN_WAIT_1** state, the client waits for a TCP segment from the server with an acknowledgment (ACK).
2. **Step 2 (ACK From Server) –**
   When Server received FIN bit segment from Sender (Client), Server Immediately send acknowledgement (ACK) segment to the Sender (Client).
3. **Step 3 (Client waiting) –**
   While in the **FIN_WAIT_1** state, the client waits for a TCP segment from the server with an acknowledgment. When it receives this segment, the client enters the **FIN_WAIT_2** state. While in the **FIN_WAIT_2** state, the client waits for another segment from the server with the FIN bit set to 1.
4. **Step 4 (FIN from Server) –**
   Server sends FIN bit segment to the Sender(Client) after some time when Server send the ACK segment (because of some closing process in the Server).
5. **Step 5 (ACK from Client) –**
   When Client receive FIN bit segment from the Server, the client acknowledges the server's segment and enters the **TIME_WAIT** state. The **TIME_WAIT** state lets the client resend the final acknowledgment in case the **ACK** is lost. The time spent by client in the **TIME_WAIT** state is depend on their implementation, but their typical values are 30 seconds, 1 minute, and 2 minutes. After the wait, the connection formally closes and all resources on the client side (including port numbers and buffer data) are released.

In the below Figures illustrates the series of states visited by the server-side and also Client-side, assuming the client begins connection tear-down. In these two state-transition figures, we have only shown how a TCP connection is normally established and shut-down.

TCP states visited by Client Side –

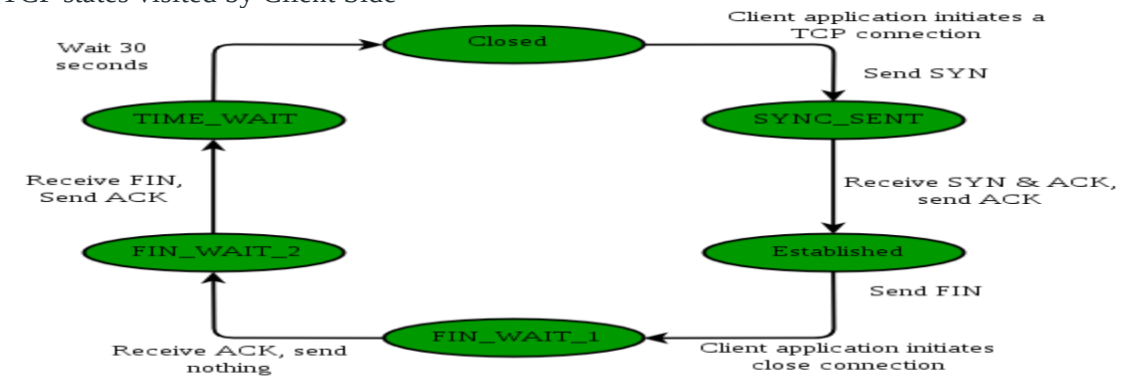


Fig : TCP states visited by a client TCP

TCP states visited by Server Side –

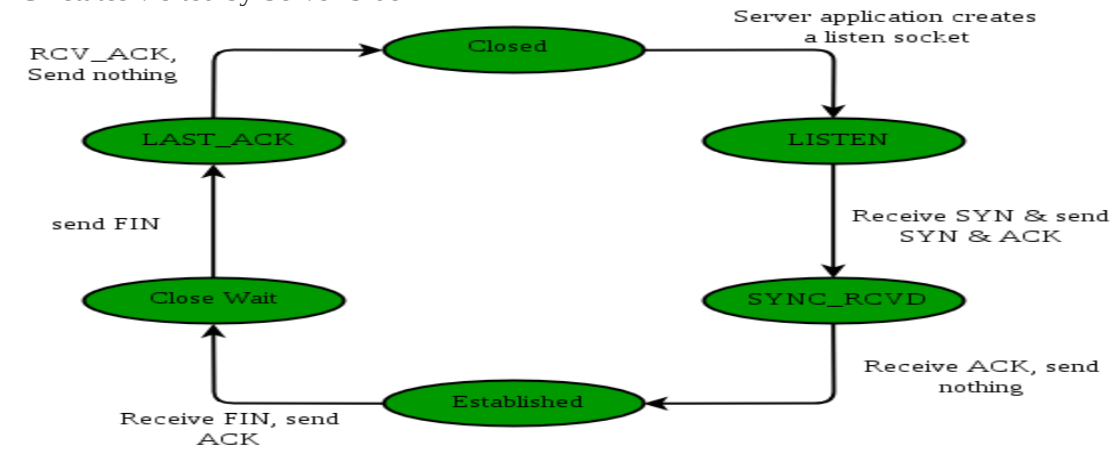


Fig : TCP states visited by a client TCP

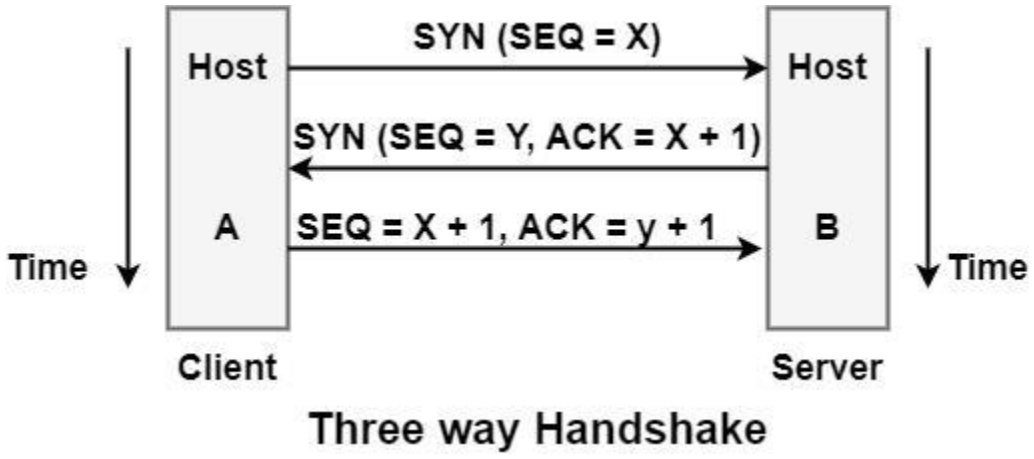
.

TCP Connection Establishment

the transport services reliable, TCP hosts must establish a connection-oriented session with one another. Connection establishment is performed by using the three-way handshake mechanism. A three-way handshake synchronizes both ends of a network by enabling both sides to agree upon original sequence numbers.

This mechanism also provides that both sides are ready to transmit data and learn that the other side is available to communicate. This is essential so that packets are not shared or retransmitted during session establishment or after session termination. Each host randomly selects a sequence number used to track bytes within the stream it is sending and receiving.

The three-way handshake proceeds in the manner shown in the figure below −



Three way Handshake

The requesting end (Host A) sends an SYN segment determining the server's port number that the client needs to connect to and its initial sequence number (x).

The server (Host B) acknowledges its own SYN segment, including the servers initial sequence number (y). The server also responds to the client SYN by accepting the sender's SYN plus one (X + 1).

An SYN consumes one sequence number. The client should acknowledge this SYN from the server by accepting the server's SEQ plus one (SEQ = x + 1, ACK = y + 1). This is how a TCP connection is settled.

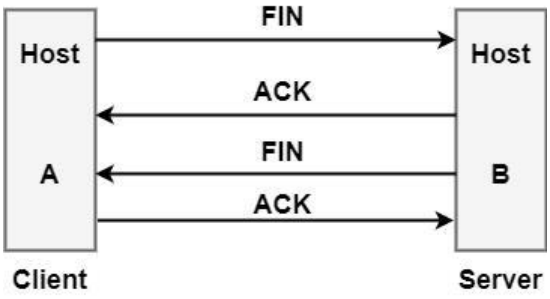
Connection Termination Protocol (Connection Release)

While it creates three segments to establish a connection, it takes four segments to terminate a connection. During a TCP connection is full-duplex (that is, data flows in each direction independently of the other direction), each direction should be shut down alone.

The termination procedure for each host is shown in the figure. The rule is that either end can share a FIN when it has finished sending data.

When a TCP receives a FIN, it should notify the application that the other end has terminated that data flow direction. The sending of a FIN is usually the result of the application issuing a close.

The receipt of a FIN only means that there will be no more data flowing in that direction. A TCP can send data after receiving a FIN. The end that first issues the close (example, send the first FIN) executes the active close. The other end (that receives this FIN) manages the passive close.



TCP Termination

SLIDING WINDOW PROTOCOL

Sliding window protocols are data link layer protocols for reliable and sequential delivery of data frames. The sliding window is also used in Transmission Control Protocol.

In this protocol, multiple frames can be sent by a sender at a time before receiving an acknowledgment from the receiver. The term sliding window refers to the imaginary boxes to hold frames. Sliding window method is also known as windowing.

Working Principle

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

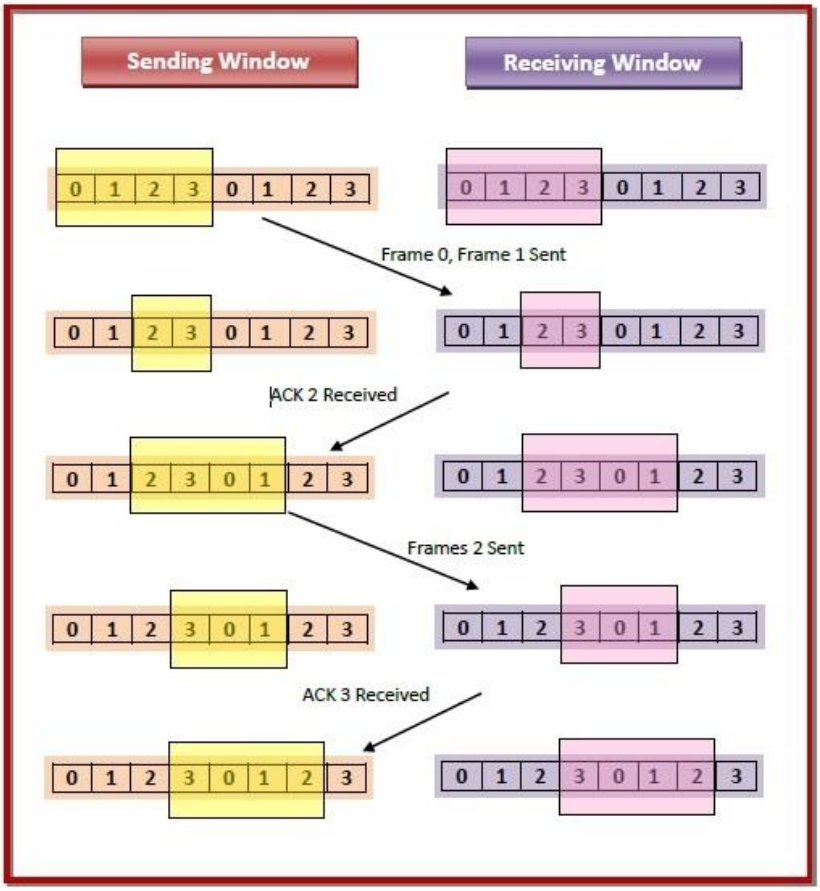
The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an n-bit field, then the range of sequence numbers that can be assigned is 0 to 2^n-1. Consequently, the size of the sending window is 2^n-1. Thus in order to accommodate a sending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



Types of Sliding Window Protocols

The Sliding Window ARQ (Automatic Repeat reQuest) protocols are of two categories −

- **Go – Back – *N* ARQ**

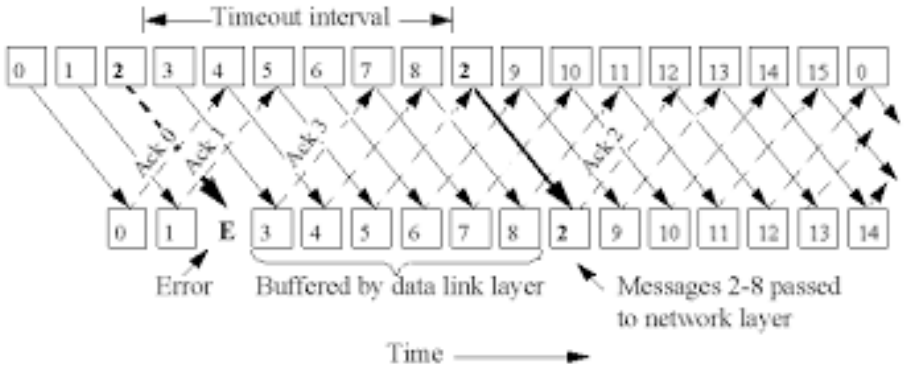
  Go – Back – *N* ARQ provides for sending multiple frames before receiving the acknowledgment for the first frame. It uses the concept of sliding window, and so is also called sliding window protocol. The frames are sequentially numbered and a finite number of frames are sent. If the acknowledgment of a frame is not received within the time period, all frames starting from that frame are retransmitted.

- **Selective Repeat ARQ**

  This protocol also provides for sending multiple frames before receiving the acknowledgment for the first frame. However, here only the erroneous or lost frames are retransmitted, while the good frames are received and buffered.

Data packets are numbered sequentially so they can be tracked when data is being transmitted from the sender to the receiver. During the transmission process, the data packets pass through one of four stages:

1. Sent and acknowledged by the receiver.
2. Sent but not acknowledged by the receiver.
3. Not sent but the receiver is ready accept them.
4. Not sent and the receiver is not ready to accept them.



FLOW AND CONGESTION CONTROL

Congestion Control

congestion

A state occurring in network layer when the message traffic is so heavy that it slows down network response time.

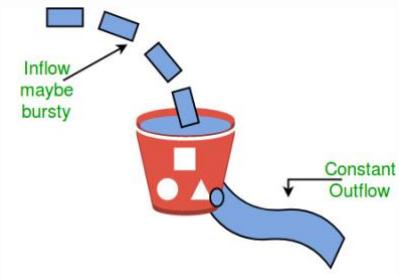
Effects of Congestion

- As delay increases, performance decreases.
    - If delay increases, retransmission occurs, making situation worse.

Congestion control algorithms

Leaky Bucket Algorithm
Let us consider an example to understand

Imagine a bucket with a small hole in the bottom. No matter at what rate water enters the bucket, the outflow is at constant rate. When the bucket is full with water additional water entering spills over the sides and is lost.



Network interface contains a leaky bucket and the following **steps** are involved in leaky bucket algorithm:
1.When host wants to send packet, packet is thrown into the bucket.
2. The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
3. Bursty traffic is converted to a uniform traffic by the leaky bucket.
4.In practice the bucket is a finite queue that outputs at a finite rate.

Token bucket Algorithm

Need of token bucket Algorithm:-

The leaky bucket algorithm enforces output pattern at the average rate, no matter how bursty the traffic is. So in order to deal with the bursty traffic we need a flexible algorithm so that the data is not lost. One such algorithm is token bucket algorithm.

Steps of this algorithm can be described as follows:

1.In regular intervals tokens are thrown into the bucket. *f*
2.The bucket has a maximum capacity. *f*
3.If there is a ready packet, a token is removed from the bucket, and the packet is sent.
4.If there is no token in the bucket, the packet cannot be sent.

Let's understand with an example,

In figure (A) we see a bucket holding three tokens, with five packets waiting to be transmitted. For a packet to be transmitted, it must capture and destroy one token. In figure (B) We see that three of the five packets have gotten through, but the other two are stuck waiting for more tokens to be generated.

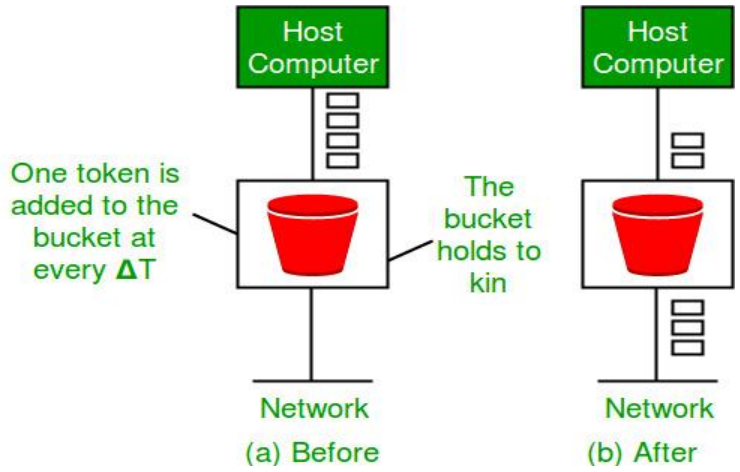
Waysinwhichtokenbucketissuperiortoleakybucket:

The leaky bucket algorithm controls the ate at which the packets are introduced in the network, but it is very conservative in nature. Some flexibility is introduced in the token bucket algorithm. In the token bucket, algorithm tokens are generated at each tick (up to a certain limit). For an incoming packet to be transmitted, it must capture a token and the transmission takes place at the same rate. Hence some of the busty packets are transmitted at the same rate if tokens are available and thus introduces some amount of flexibility in the system.

Formula: $M*s=C+\rho*s$

whereS–is time taken
M – Maximum output rate
ρ – Token arrival rate
C – Capacity of the token bucket in byte

Let's understand with an example,

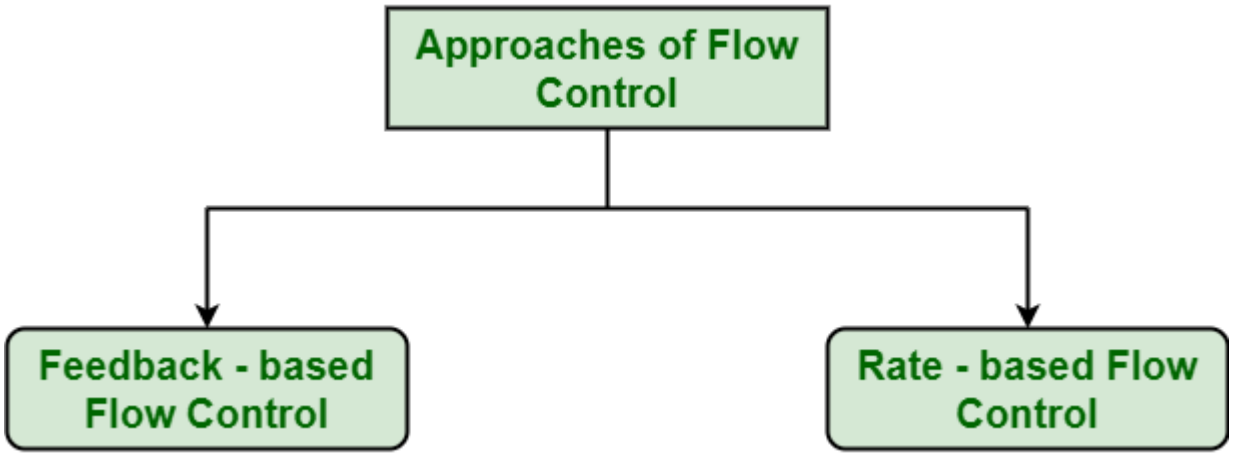


Flow Control

Flow control is used to prevent the sender from overwhelming the receiver. If the receiver is overloaded with too much data, then the receiver discards the packets and asking for the retransmission of packets. This increases network congestion and thus, reducing the system performance. The transport layer is responsible for flow control. It uses the sliding window protocol that makes the data transmission more efficient as well as it controls the flow of data so that the receiver does not become overwhelmed. Sliding window protocol is byte oriented rather than frame oriented.

Sr. No.	Key	Flow Control	Congestion Control
1	Definition	In Flow Control, Traffic is controlled and Traffic represents flow from sender to receiver.	In Congestion Control also, Traffic is controlled and Traffic represents flow entering into the network.
2	Layers	Data link and Transport layers handles flow control.	Network and Transport layers handles congestion control.
3	Prime Focus	Receiver is prevented from being overwhelmed.	Network is prevented from being congested.
4	Responsibility	Only sender is responsible for the traffic.	Transport layer is responsible for the traffic.
5	Way	Traffic is prevented by slowing down the sender.	Traffic is prevented by slowing down the transport layer.

Approaches to Flow Control :
Flow Control is classified into two categories –



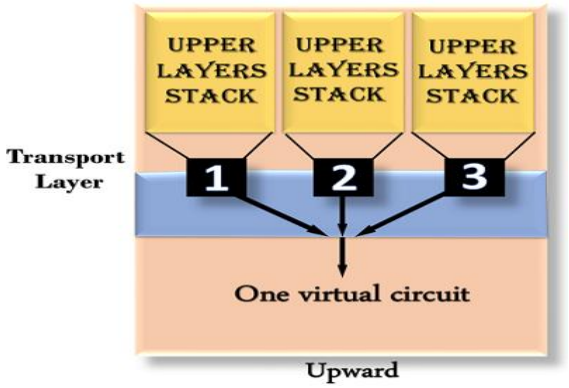
- **Feedback – based Flow Control :**
  In this control technique, sender simply transmits data or information or frame to receiver, then receiver transmits data back to sender and also allows sender to transmit more amount of data or tell sender about how receiver is processing or doing. This simply means that sender transmits data or frames after it has received acknowledgments from user.
- **Rate – based Flow Control :**
  In this control technique, usually when sender sends or transfer data at faster speed to receiver and receiver is not being able to receive data at the speed, then mechanism known as built-in mechanism in protocol will just limit or restricts overall rate at which data or information is being transferred or transmitted by sender without any feedback or acknowledgment from receiver.

Multiplexing

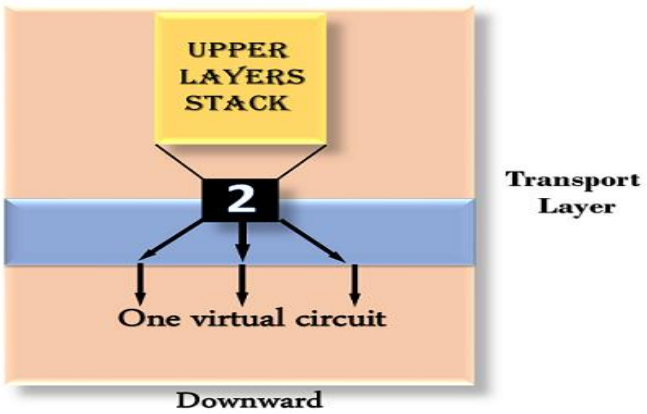
The transport layer uses the multiplexing to improve transmission efficiency.

Multiplexing can occur in two ways:

- **Upward multiplexing:** Upward multiplexing means multiple transport layer connections use the same network connection. To make more cost-effective, the transport layer sends several transmissions bound for the same destination along the same path; this is achieved through upward multiplexing.



- **Downward multiplexing:** Downward multiplexing means one transport layer connection uses the multiple network connections. Downward multiplexing allows the transport layer to split a connection among several paths to improve the throughput. This type of multiplexing is used when networks have a low or slow capacity.



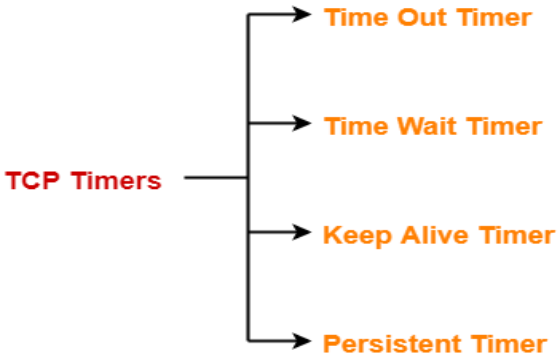
TIMERS

TCP Timers

- Retransmission Timer – To retransmit lost segments, TCP uses retransmission timeout (RTO). ...
- Persistent Timer – To deal with a zero-window-size deadlock situation, TCP uses a persistence timer. ...
- Keep Alive Timer – A keep alive timer is used to prevent a long idle connection between two TCPs.

TCP uses several timers to ensure that excessive delays are not encountered during communications. Several of these timers are elegant, handling problems that are not immediately obvious at first analysis. Each of the timers used by TCP is examined in the following sections, which reveal its role in ensuring data is properly sent from one connection to another.

TCP implementation uses four timers –
Retransmission Timer – To retransmit lost segments, TCP uses retransmission timeout (RTO). When TCP sends a segment the timer starts and stops when the acknowledgment is received. If the timer expires timeout occurs and the segment is retransmitted. RTO (retransmission timeout is for 1 RTT) to calculate retransmission timeout we first need to calculate the RTT(round trip time).



Time Out Timer-

TCP uses a time out timer for retransmission of lost segments.

- Sender starts a time out timer after transmitting a TCP segment to the receiver.
- If sender receives an acknowledgement before the timer goes off, it stops the timer.
- If sender does not receives any acknowledgement and the timer goes off, then **TCP Retransmission** occurs.
- Sender retransmits the same segment and resets the timer.
- The value of time out timer is dynamic and changes with the amount of traffic in the network.
- Time out timer is also called as **Retransmission Timer**.

Time Wait Timer-

TCP uses a time wait timer during connection termination.

- Sender starts the time wait timer after sending the ACK for the second FIN segment.
- It allows to resend the final acknowledgement if it gets lost.
- It prevents the just closed port from reopening again quickly to some other application.
- It ensures that all the segments heading towards the just closed port are discarded.
- The value of time wait timer is usually set to twice the lifetime of a TCP segment.

Keep Alive Timer-

TCP uses a keep alive timer to prevent long idle TCP connections.

- Each time server hears from the client, it resets the keep alive timer to 2 hours.
- If server does not hear from the client for 2 hours, it sends 10 probe segments to the client.
- These probe segments are sent at a gap of 75 seconds.
- If server receives no response after sending 10 probe segments, it assumes that the client is down.
- Then, server terminates the connection automatically.

Persistent Timer-

- TCP uses a persistent timer to deal with a zero-widow-size deadlock situation.
- It keeps the window size information flowing even if the other end closes its receiver window.

Explanation

Consider the following situation-

- Sender receives an acknowledgment from the receiver with zero window size.
- This indicates the sender to wait.
- Later, receiver updates the window size and and sends the segment with the update to the sender.
- This segment gets lost.
- Now, both sender and receiver keeps waiting for each other to do something.

To deal with such a situation, TCP uses a persistent timer.

- Sender starts the persistent timer on receiving an ACK from the receiver with a zero window size.
- When persistent timer goes off, sender sends a special segment to the receiver.
- This special segment is called as probe segment and contains only 1 byte of new data.
- Response sent by the receiver to the probe segment gives the updated window size.
- If the updated window size is non-zero, it means data can be sent now.
- If the updated window size is still zero, the persistent timer is set again and the cycle repeats.

RETRANSMISSION

Retransmission, essentially identical with Automatic repeat request (ARQ), is the resending of packets which have been either damaged or lost. Retransmission is one of the basic mechanisms used by protocols operating over a packet switched computer network to provide reliable communication (such as that provided by a reliable byte stream, for example TCP).

Such networks are usually "unreliable", meaning they offer no guarantees that they will not delay, damage, or lose packets, or deliver them out of order. Protocols which provide reliable communication over such networks use a combination of acknowledgments (i.e. an explicit receipt from the destination of the data), retransmission of missing or damaged packets (usually initiated by a time-out), and checksums to provide that reliability.

Retransmission is a very simple concept. Whenever one party sends something to the other party, it retains a copy of the data it sent until the recipient has acknowledged that it received it. In a variety of circumstances the sender automatically retransmits the data using the retained copy.

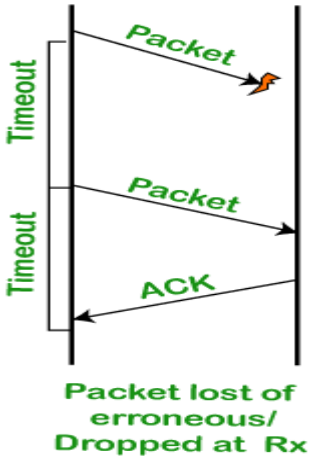
Reasons for resending include:

- if no such acknowledgment is forthcoming within a reasonable time, the time-out
- the sender discovers, often through some out of band means, that the transmission was unsuccessful
- if the receiver knows that expected data has not arrived, and so notifies the sender
- if the receiver knows that the data has arrived, but in a damaged condition, and indicates that to the sender

Retransmission mechanism

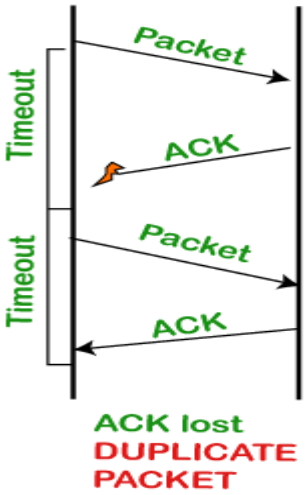
Here, retransmission means the data packets have been lost, which leads to a lack of acknowledgment. This lack of acknowledgment triggers a timer to timeout, which leads to the retransmission of data packets. Here, the timer means that if no acknowledgment is received before the timer expires, the data packet is retransmitted.

Scenario 1: When the data packet is lost or erroneous.



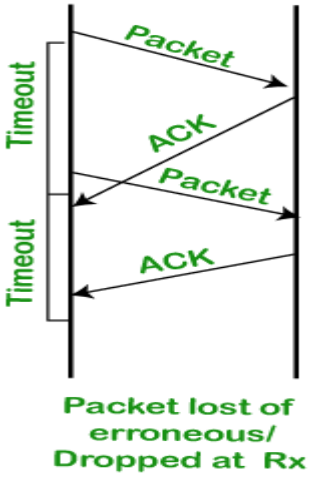
In this scenario, the packet is sent to the receiver, but no acknowledgment is received within that timeout period. When the timeout period expires, then the packet is resent again. When the packet is retransmitted, the acknowledgment is received. Once the acknowledgment is received, retransmission will not occur again.

Scenario 2: When the packet is received but the acknowledgment is lost.



In this scenario, the packet is received on the other side, but the acknowledgment is lost, i.e., the ACK is not received on the sender side. Once the timeout period expires, the packet is resent. There are two copies of the packets on the other side; though the packet is received correctly, the acknowledgment is not received, so the sender retransmits the packet. In this case, the retransmission could have been avoided, but due to the loss of the ACK, the packet is retransmitted.

Scenario 3: When the early timeout occurs.



In this scenario, the packet is sent, but due to the delay in acknowledgment or timeout has occurred before the actual timeout, the packet is retransmitted. In this case, the packet has been sent again unnecessarily due to the delay in acknowledgment or the timeout has been set earlier than the actual timeout.

The sender sets the timeout period for an ACK. The timeout period can be of two types:

- **Too short:** If the timeout period is too short, then the retransmissions will be wasted.
- **Too long:** If the timeout period is too long, then there will be an excessive delay when the packet is lost.

In order to overcome the above two situations, TCP

sets the timeout as a function of the RTT (round trip time) where round trip time is the time required for the packet to travel from the source to the destination and then come back again.

We will use the **original algorithm** to measure the RTT.

Step 1: First, we measure the **SampleRTT** for each segment or ACK pair. When the sender sends the packet, then we know the timer at which the packet is sent, and also, we know the timer at which acknowledgment is received. Calculate the time between these two, and that becomes the **SampleRTT**.

Step 2: We will not take only one sample. We will keep on taking different samples and calculate the weighted average of these samples, and this becomes the EstRTT (Estimated RTT).

where, $\alpha + \beta = 1$

α lies between 0.8 and 0.9

β lies between 0.1 and 0.2

Step 3: The timeout is set based on EstRTT.

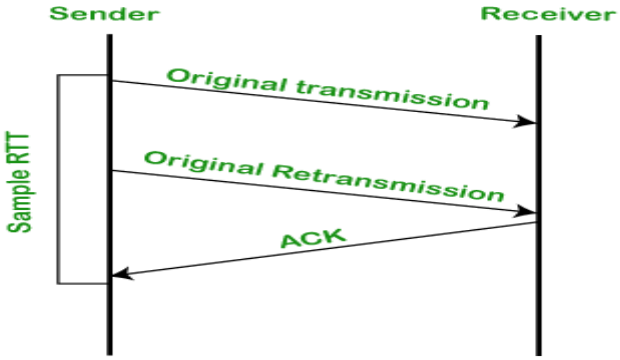
timeout = 2 * EstRTT.

The timeout is set to be twice the estimated RTT. This is how the actual timeout factor is calculated.

A Flaw in this approach

There is a flaw in the original algorithm. Let's consider two scenarios.

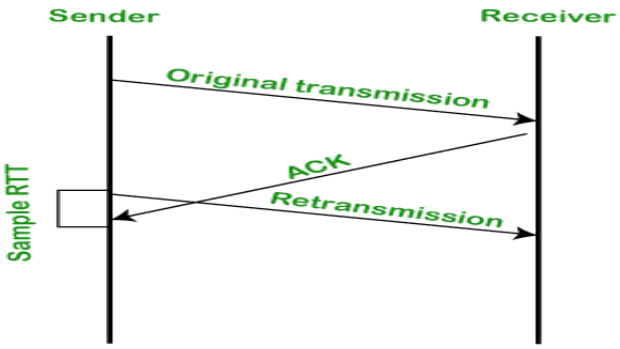
Scenario 1.



Do not sample RTT when retransmitting

The above diagram shows that the sender sends the data, which is said to be an original transmission. Within the timeout period, no acknowledgment is received. So, the sender retransmits the data. After retransmitting the data, the acknowledgment is received. Let's assume that acknowledgment is received for the original transmission, not for the retransmission. Since we get the acknowledgment of the original transmission, so **SampleRTT** is calculated between the time of the original transmission and the time at which the acknowledgment is received. But actually, the **SampleRTT** should have been between the time of the retransmission and time of the acknowledgment.

Scenario 2.



Double timeout after each retransmission

The above diagram shows that the sender sends the original data packet for which we get the acknowledgment also. But the acknowledgment is received after retransmitting the data. If we assume that acknowledgment belongs to the retransmission, then **SampleRTT** is calculated between the time of the retransmission and the time of the acknowledgment.

In the above both the scenarios, there is an ambiguity of not knowing whether the acknowledgment is for the original transmission or for the retransmission.

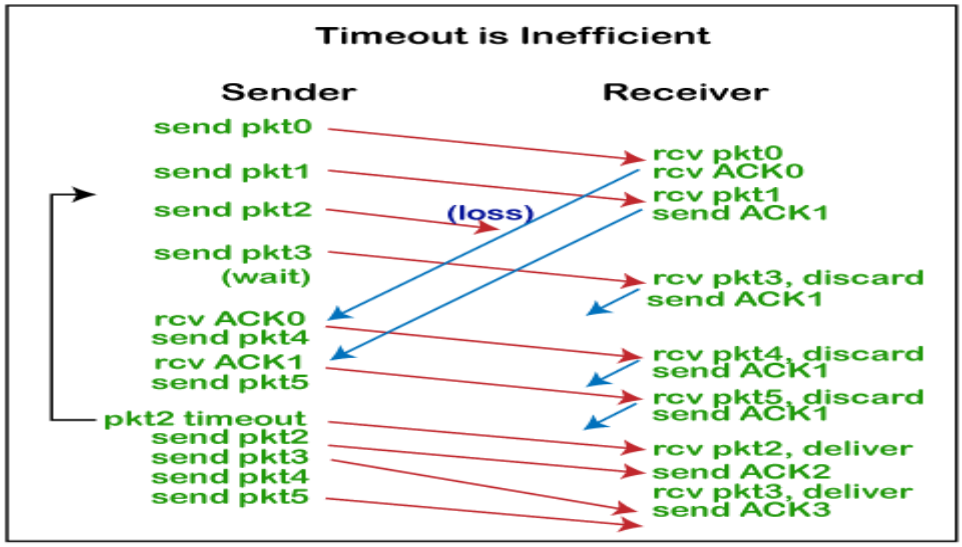
Conclusion of the above algorithm.

- Here, ACK does not mean to acknowledge a transmission, but actually, it acknowledges a receipt of the data.
- If we consider the first scenario, the retransmission is done for the lost packet. In this case, we are assuming that ACK belongs to the original transmission due to which the SampleRTT is coming out to be very large.
- If we consider the second scenario, two same packets are sent so duplicity occurs in this case. In this case, we are assuming that ACK belongs to the retransmission due to which the SampleRTT is coming to be very small.

To overcome the above problems, a simple solution is given by the Karn/Partridge algorithm. This algorithm gave a simple solution that collects the samples sent at one time and does not consider the samples at the retransmission time for calculating the estimated RTT.

Fast Retransmission

The timeout-based strategy for retransmission is inefficient. TCP is a sliding-window kind of protocol, so whenever the retransmission occurs, it starts sending it from the lost packet onward.



Suppose I transmit the packets 0, 1, 2, and 3. Since packet 0 and packet 1 are received on the other side, packet 2 is lost in a network. I have received the acknowledgment of packet 0 and packet 1, so I send two more packets, i.e., packet 4 and packet 5. When packets 3, 4, and 5 are sent, then I will get the acknowledgment of packet 1 as TCP acknowledgments are cumulative, so it acknowledges up to the packet that it has received in order. I have not received the acknowledgment of packet 2, 3,4, and 5 within the timeout period, so I retransmit the packets 2, 3, 4, and 5. Since packet 2 is lost, but other packets, i.e., 3, 4,5 are received on the other side, they are still retransmitted because of this timeout mechanism.

How can this timeout inefficiency be removed?

The better solution under a sliding window:

Suppose n packet has been lost, but still, the packets n+1, n+2, and so on have been received. The receiver is continuously receiving the packets and sending the ACK packets saying that the receiver is still awaiting the nth packet. The receiver is sending repeated or duplicate acknowledgments. In the above case, ACK of packet 1 is sent three-time as packet 2 has been lost. This duplicate ACK packet is an indication that the nth packet is missing, but the later packets are received.

The above situation can be solved in the following ways:

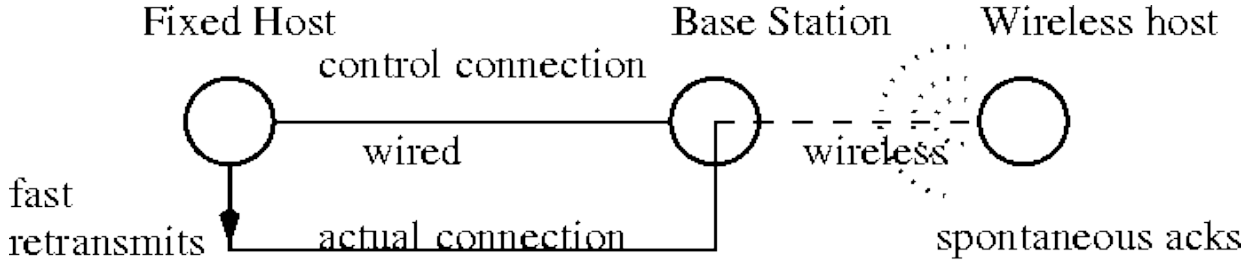
- The sender can take the "duplicate ACKs" as an early hint that the nth packet has been lost so that the sender can do the retransmission as early as possible, i.e., the sender should not wait until the timeout occurs.
- The sender can implement a fast transmission strategy in TCP. In a fast transmission strategy, the sender should consider the triple duplicate ACKs as a trigger and retransmit it.

TCP EXTENSIONS

In TCP for the purpose of better throughput when connected to mobile hosts. The changes at the source side is to ensure that the source can distinguish between congestion in the fixed network and noise in the wireless link as the cause for packet loss. The changes in the receiving side (the wireless host) is to ensure that in case packets are dropped due to

noise in the wireless channel, the source starts to re-transmit packets earlier than its timeout deadline, for which it normally waits.

When a TCP connection is to be opened between a fixed host on a network and a wireless host, it opens two independent connections - one to the base station and the other to the mobile host. The call setup functionality is left unchanged as is in original TCP. The connection between the source and the base station basically serves as a control, to estimate the network congestion that occurs between the two ends. With the assumption that the mobile host receives all packets through the base station, we expect that packets on both these connections should be routed by almost the same route. and would see the same network congestion.



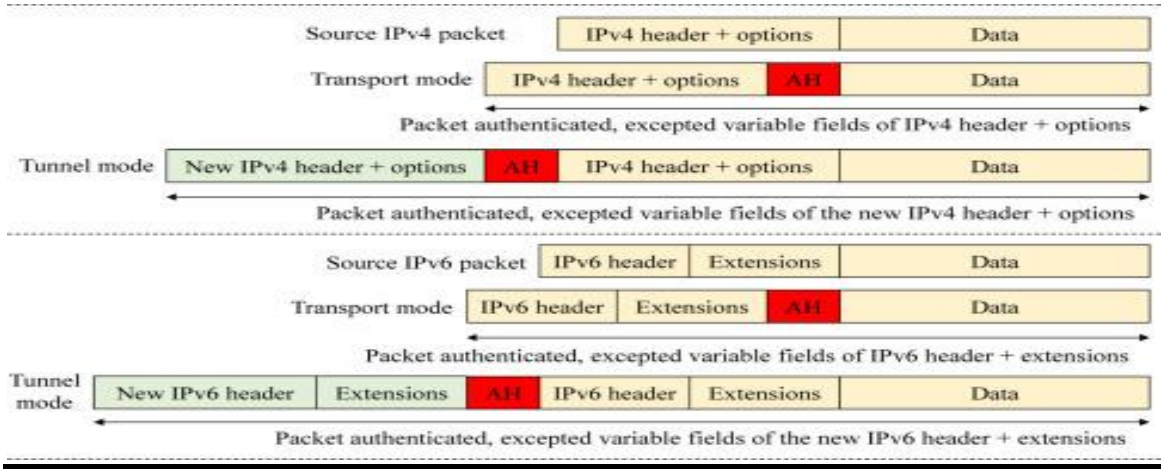
Model of Proposed TCP Extensions

Packets on the control connection are sent at some regular interval, but at rates sufficiently low so as to not cause too much overhead in the network because of these control packets. The TCP source will also be sending packets on the connection with the mobile host as the application hands down packets to it. By comparing the fraction of the packets that are acknowledged within the timeout on the two connections the source will determine if there is congestion in the network which is causing packets to be lost, or is it the channel noise. If the acknowledged fraction is *significantly* different in the two cases, then the conclusion is that the channel noise is causing the packet loss. In this situation, the source will be prevented from reducing the congestion window size, and it will continue its linear increment at the same rate. If the two fractions are the same, then the normal congestion control mechanisms are allowed to proceed as it would in ordinary TCP.

At the wireless host side, to pre-empt the source to re-send packets that are lost due to the channel noise (or even otherwise), we employ the technique of fast re-transmissions [9]. The source on seeing a duplicate acknowledgement decides that its previous packet sent have been lost or corrupted. Hence, it re-transmits the subsequent packet even before the timer expires and re-starts the timer. On sending a packet the source at time t_s, it waits upto time ($t_s + sRTT + \frac{variance}{2}$) before recognizing another duplicate acknowledgement for a subsequent re-transmission. The need to await for at least the sRTT time is to ensure that these new re-transmissions had a chance to reach the destination before another re-transmission is done.

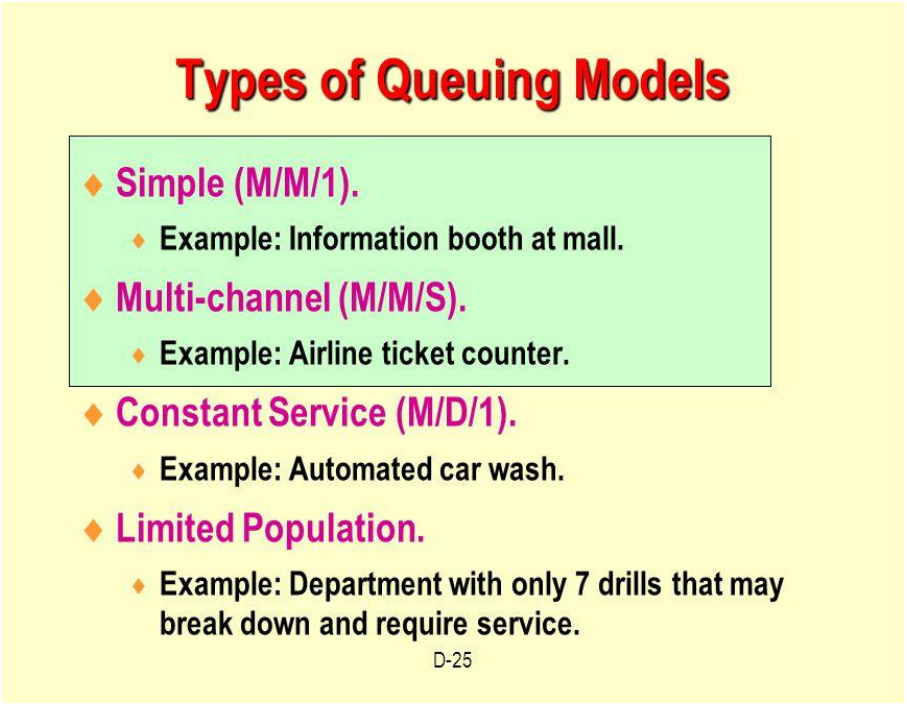
Also, to ensure that the duplicate acknowledgements manage to get through the wireless channel even under poor channel quality, the wireless host sends multiple duplicate acknowledgements on receiving a new packet at time t_r. This period extends from the time a new packet is received upto time ($t_r + timeout - sRTT$) more. Any acknowledgements sent after this time will reach around the timeout period, and the source will automatically re-transmit after that, and so further acknowledgements are wasted.

If no further packets are received upto ($t_r + timeout$), further automatic acknowledgements are made upto time ($t_r + 2 \times timeout - sRTT$). Hence, it absence of further packets being received, automatic acknowledgements are sent in regular intervals in the periods [($t_r + k \times timeout$), ($t_r + k \times timeout - sRTT$)], where t_r is the time when the last packet was received by the wireless host. The value t_r is reset each time a new packet is received.



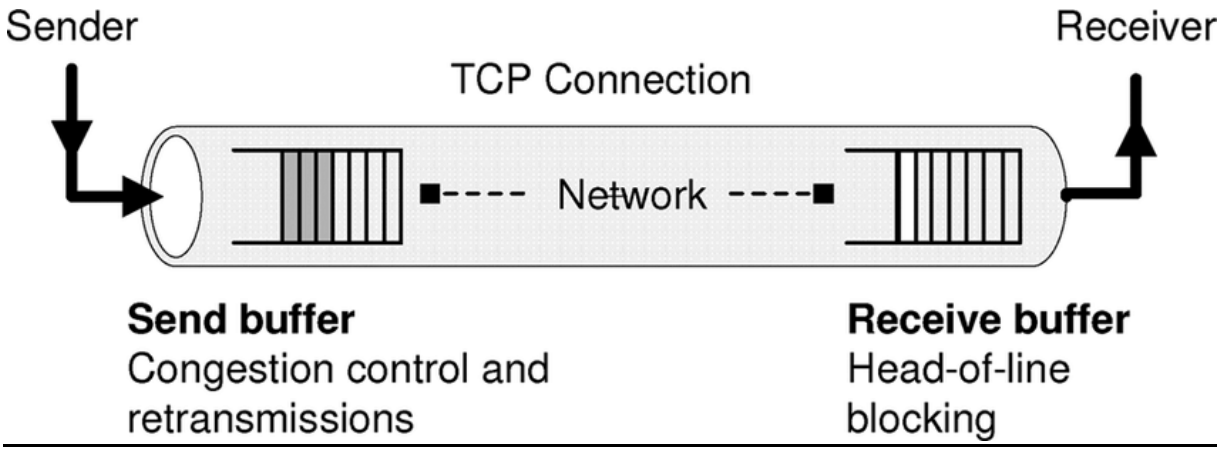
QUEUING THEORY

A set of simple queuing theory models which can model the average response of a network of computers to a given traffic load has been implemented using a spreadsheet. traffic patterns and intensities, channel capacities, and message protocols can be assessed using them because of the lack of fine detail in the network traffic rates, traffic patterns, and the hardware used to implement the networks. network of local area networks (LANs) connected via gateways and high-speed backbone communication channels.



Queuing Systems

Queuing theory establishes a powerful tool in modeling and performance analysis of many complex systems, such as computer networks, telecommunication systems, call centers, manufacturing systems and service systems. Many real systems can be modeled as networks of queues, such as the waiting line in a bank, a bus stop, waiting line in airport, material flow in factory, and signal flow in network.



Real-life applications of queuing theory cover a wide range of businesses. Its findings may be used to provide faster customer service, increase traffic flow, improve order shipments from a warehouse, or design data networks and call centers.

As a branch of operations research, queuing theory can help inform business decisions on how to build more efficient and cost-effective workflow systems.

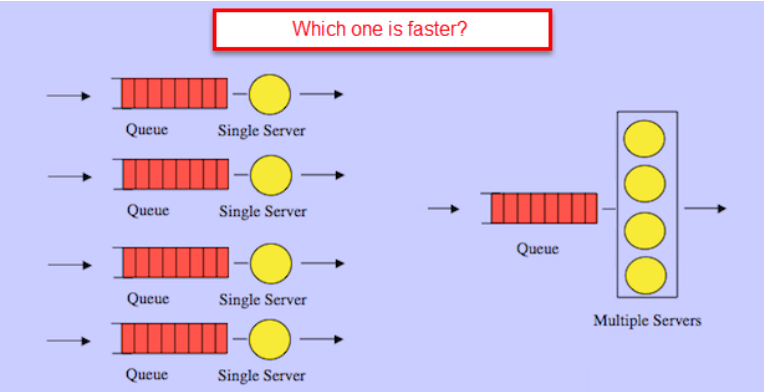
- Queuing theory is the study of the movement of people, objects, or information through a line.
- Studying congestion and its causes in a process is used to help create more efficient and cost-effective services and systems.
- Often used as an operations management tool, queuing theory can address staffing, scheduling, and customer service shortfalls.
- Some queuing is acceptable in business. If there's never a queue, it's a sign of overcapacity.
- Queuing theory aims to achieve a balance that is efficient and affordable.

How Queuing Theory Works

Queues can occur whenever resources are limited. Some queuing is tolerable in any business since a total absence of a queue would suggest a costly overcapacity.

Queuing theory aims to design balanced systems that serve customers quickly and efficiently but do not cost too much to be sustainable.

At its most basic level, queuing theory involves an analysis of arrivals at a facility, such as a bank or a fast-food restaurant, and an analysis of the processes currently in place to serve them. The end result is a set of conclusions that aim to identify any flaws in the system and suggest how they can be ameliorated.



complexities involved with the analysis of traffic flow at signalized and unsignalized intersections. Chapter 7 will address the additional complexities relating to the analysis of traffic flow at signalized intersections. For details on the analysis of traffic flow at unsignalized intersections, refer to other sources [Transportation Research Board 1975, 2000]. It should be noted that environmental conditions (day vs. night, sunny vs. rainy, etc.) can also affect the flow of traffic, but this issue is beyond the scope of this book.

5.2.1 Traffic Flow, Speed, and Density

Traffic flow, speed, and density are variables that form the underpinnings of traffic analysis. To begin the study of these variables, the basic definitions of traffic flow, speed, and density must be presented. Traffic flow is defined as

$$q = \frac{n}{t} \tag{5.1}$$

where

q = traffic flow in vehicles per unit time,
n = number of vehicles passing some designated roadway point during time t, and
t = duration of time interval.

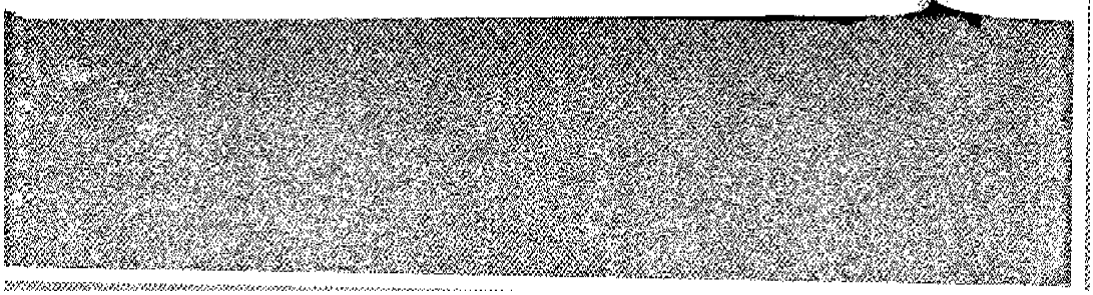
Flow is often measured over the course of an hour, in which case the resulting value is typically referred to as volume. Thus, when the term "volume" is used, it is generally understood that the corresponding value is in units of vehicles per hour (veh/h). The definition of flow is more generalized to account for the measurement of vehicles over any period of time. In practice, the analysis flow rate is usually based on the peak 15-minute flow within the hour of interest. This aspect will be described in more detail in Chapter 6.

Aside from knowing the total number of vehicles passing a point in some time interval, the amount of time between the passing of successive vehicles (or time between the arrival of successive vehicles) is also of interest. The time between the passage of the front bumpers of successive vehicles, at some designated highway point, is known as the time headway. The time headway is related to t, as defined in Eq. 5.1, by

$$t = \sum_{i=1}^{n} h_i \tag{5.2}$$

where

t = duration of time interval,
h_i = time headway of the ith vehicle (the elapsed time between the arrivals of vehicles i and $i - 1$), and
n = number of measured vehicle time headways at some designated roadway point.



Substituting Eq. 5.2 into Eq. 5.1 gives

$$q = \frac{n}{\sum_{i=1}^{n} h_i} \tag{5.3}$$

or

$$q = \frac{1}{\bar{h}} \tag{5.4}$$

where $\bar{h}$ = average time headway $(\sum h_i/n)$ in unit time per vehicle. The importance of time headways in traffic analysis will be given additional attention in forthcoming sections of this chapter.

The average traffic speed is defined in two ways. The first is the arithmetic mean of the vehicle speeds observed at some designated point along the roadway. This is referred to as the time-mean speed and is expressed as

$$\bar{u}_t = \frac{\sum_{i=1}^{n} u_i}{n} \tag{5.5}$$

where

$\bar{u}_t$ = time-mean speed in unit distance per unit time,
u_i = spot speed (the speed of the vehicle at the designated point on the highway, as might be obtained using a radar gun) of the ith vehicle, and
n = number of measured vehicle spot speeds.

The second definition of speed is more useful in the context of traffic analysis and is determined on the basis of the time necessary for a vehicle to travel some known length of roadway. This measure of average traffic speed is referred to as the space-mean speed and is expressed as (assuming that the travel time for all vehicles is measured over the same length of roadway)

$$\bar{u}_s = \frac{l}{\bar{t}} \tag{5.6}$$

where

$\bar{u}_s$ = space-mean speed in unit distance per unit time,
l = length of roadway used for travel time measurement of vehicles, and
$\bar{t}$ = average vehicle travel time, defined as

$$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_i \tag{5.7}$$

where

t_i = time necessary for vehicle i to travel a roadway section of length l, and

n = number of measured vehicle travel times.

Substituting Eq. 5.7 into Eq. 5.6 yields

$$\bar{u}_s = \frac{l}{\frac{1}{n}\sum_{i=1}^{n} t_i} \tag{5.8}$$

or

$$\bar{u}_s = \frac{1}{\frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{(l/t_i)}\right]} \tag{5.9}$$

which is the harmonic mean of speed (space-mean speed). Space-mean speed is the speed variable used in traffic models.

EXAMPLE 5.1

The speeds of five vehicles were measured (with radar) at the midpoint of a 0.5-mile (0.8-km) section of roadway. The speeds for vehicles 1, 2, 3, 4, and 5 were 44, 42, 51, 49, and 46 mi/h (70.8, 67.6, 82.1, 78.8, and 74 km/h), respectively. Assuming all vehicles were traveling at constant speed over this roadway section, calculate the time-mean and space-mean speeds.

SOLUTION

For the time-mean speed, Eq. 5.5 is applied, giving

$$\bar{u}_t = \frac{\sum_{i=1}^{n} u_i}{n}$$

$$= \frac{44 + 42 + 51 + 49 + 46}{5}$$

$$= \underline{46.4 \text{ mi/h}}$$

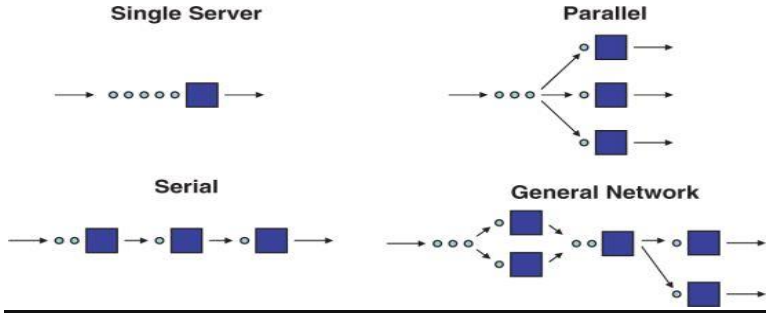
For the space-mean speed, Eq. 5.9 will be applied. This equation is based on travel time; however, because it is known that the vehicles were traveling at constant speed, we can rearrange this equation to utilize the measured speed, knowing that distance, l, divided by travel time, t, is equal to speed ($l/t_i = u$).

SINGLE AND MULTIPLE SERVER QUEUING MODELS

What is single server queue system?

The simplest queue is a line of customers, in which the customer at the head of the line **receives service from a single server** and then departs, and arriving customers join the tail of the line. ... At points where several wires meet, incoming packets are queued up, inspected, and sent out over the appropriate wire.

Queues or waiting lines form in systems when service times and arrival rates are variable. Simple queueing models provide insight into how variability subtly causes congestion. Understanding this is vital to the design and management of a wide range of production and service systems.



$$P_o = \text{Prob}\begin{bmatrix}\text{system is}\\ \text{empty (idle)}\end{bmatrix} = 1 - \frac{\lambda}{\mu}$$

$$L_q = \begin{matrix}\text{average number}\\ \text{in the queue}\end{matrix} = \frac{\lambda^2}{\mu(\mu - \lambda)}$$

$$L = \begin{matrix}\text{average number}\\ \text{in the system}\end{matrix} = \frac{\lambda}{\mu - \lambda}$$

$$W_q = \begin{matrix}\text{average time}\\ \text{in the queue}\end{matrix} = \frac{\lambda}{\mu(\mu - \lambda)}$$

$$W = \begin{matrix}\text{average time}\\ \text{in the system}\end{matrix} = \frac{1}{\mu - \lambda}$$

Note:

λ is the arrival rate.
μ is the service rate.

Multi-Server Systems

Multi-server systems include more that one server, and these provide service to the customers arriving into the customer queue(s). The models of multi-server systems can be designed with several similar servers or with different types of servers. A large set of practical problems involves the study of systems with multiple servers. Many real-world problems

can be modeled with this scheme. The simplest multi-server models include a single customer queue. Other models include multiple queues, of which the simplest are those with a separate queue for every server.

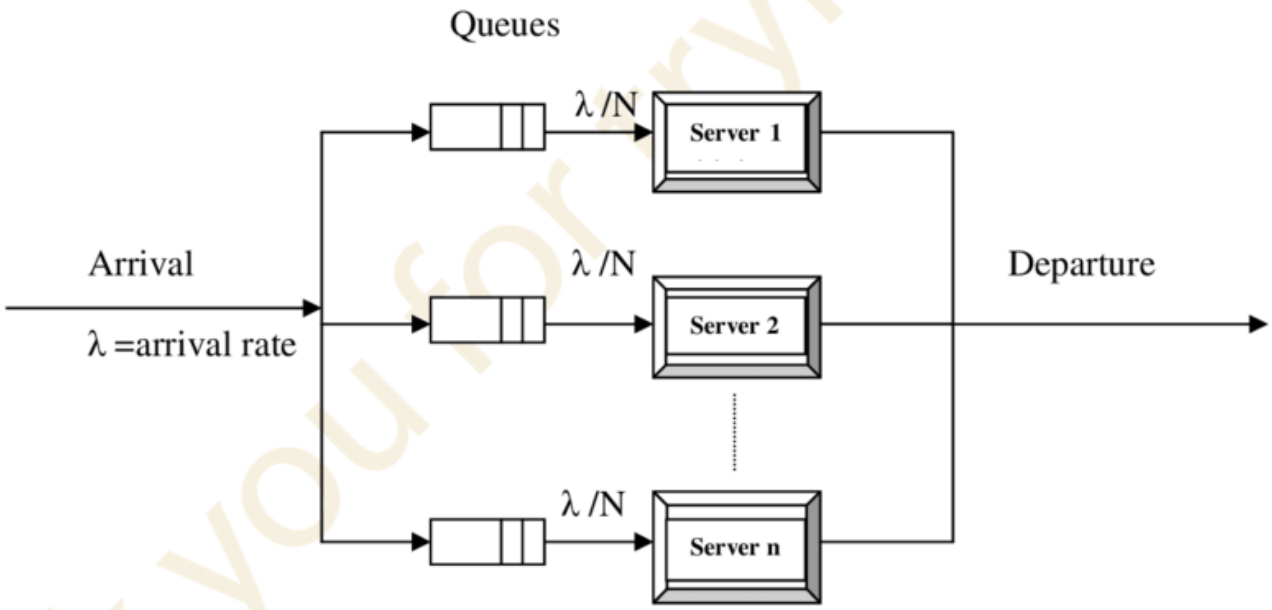
Multi-Server Queuing Models In a multi-server model, any of the servers can provide the service demanded by a customers waiting in the customer queue. the general structure of a multi-server model for the Carwash system. The servers that are not busy, are usually stored in server queue. An arriving customer will remove the server at the head of the server queue and reactivate the server. The system has several servers providing service to customers. All servers are alike, e.g., there is only one type of server. In a similar manner, all customers are alike, there is only one type of customer. If the demand for service is high enough, it may be necessary to add more servers. If the customer wait time is too high and/or the throughput (the number of serviced customers) is too low, more servers are included to improve the performance of the model. If the arrival rate of a system is not high enough, it may not be convenient to increase the number of servers because the server utilization will be lower. If the demand for service is much greater than the service capacity of the system, then adding more servers can be justified.

a: arrival rate distribution b: service rate distribution c: mean waiting time d: mean idle time

Fig(7) : case 5 results for single server model.

Multiple server Queuing Model

The generalization form of the multiple server model was shown in Figure (8). All sharii common queue. If an item arrives and at least one server is available, then the iten immediately dispatched to that server. It is assumed that all servers are identical; thus, if n than one server is available, it makes no difference which server is chosen for the item. If servers are busy, a queue begins to form. As soon as one server becomes free, an iten dispatched from the queue using the dispatching discipline in force. If we have *N* ident servers, then □□is the utilization of each server, Figure (9), shows its transition diagram.



LITTLE'S FORMULA

Little's Law

Little's Law is a theorem that determines the average number of items in a stationary queuing system, based on the average waiting time of an item within a system and the average number of items arriving at the system per unit of time.

The law provides a simple and intuitive approach for the assessment of the efficiency of queuing systems. The concept is hugely significant for business operations because it states that the number of items in the queuing system primarily depends on two key variables and is not affected by other factors, such as the distribution of the service or service order.Almost any queuing system and even any sub-system (think about a single teller in a supermarket) can be assessed using the law. In addition, the theorem can be applied in different fields, from running a small coffee shop to the maintenance of the operations of a military airbase.

Origin of Little's Law

Massachusetts Institute of Technology (MIT) professor, John Little, developed Little's Law in 1954. The initial publication of the law did not contain any proof of the theorem. However, in 1961, Little published proof that there is no queuing situation where the described relationship does not hold. Little later received recognition for his work in operations research.

Formula for Little's Law

Mathematically, Little's Law is expressed through the following equation:

Where:

L – the average number of items in a queuing system

λ – the average number of items arriving at the system per unit of time

W – the average waiting time an item spends in a queuing system

Example of Little's Law

John owns a small coffee shop. He wants to know the average number of customers queuing in his coffee shop, to decide whether he needs to add more space to accommodate more customers. Currently, his queuing area can accommodate no more than eight people.

John measured that, on average, 40 customers arrive at his coffee shop every hour. He also determined that, on average, a customer spends around 6 minutes in his store (or 0.1 hours). Given these inputs, John can find the average number of customers queuing in his coffee shop by applying Little's Law:

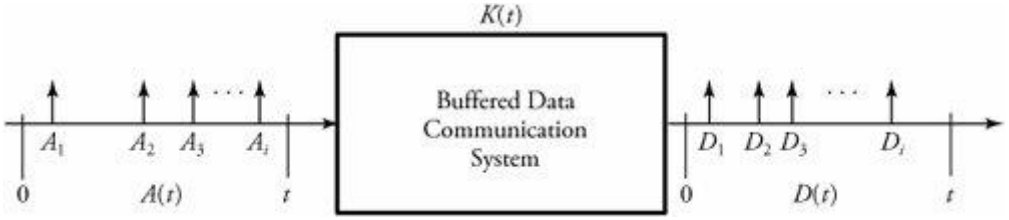
L = 40 x 0.1 = 4 customers

Little's Law shows that, on average, there are only four customers queuing in John's coffee shop. Therefore, he does not need to create more space in the store to accommodate more queuing customers.

Little's Theorem

The fundamental concept of packet queueing is based on Little's theorem , or Little's formula . This formula states that for networks reaching steady state, the average number of packets in a system is equal to the product of the average arrival rate, » , and the average time spent in the queueing system. For example, consider the communication system shown in Figure 11.1. Assume that the system starts with empty state at time t = 0. Let A_i and D_i be the i th arriving packet and i th departing packet, respectively, and let A(t) and D(t) be the total number of arriving packets and the total number of departing packets, respectively, up to a given time t .

Figure 11.1. Overview of a buffered communication system with arriving and departing packets

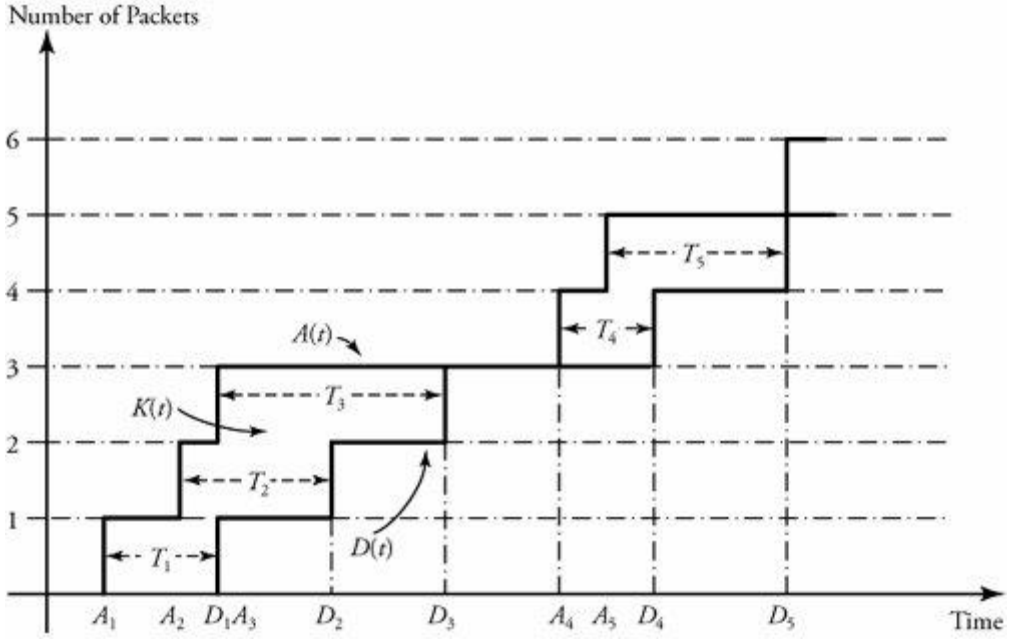


Thus, the total time a packet i spends in the system is expressed by $T_i = D_i - A_i$, and the total number of packets in the system at time t is described by $K(t) = A(t) - D(t)$. The cumulative timing chart of packets in a first-come, first-served service discipline is shown in Figure 11.2. The total time that all packets spend in the system can be expressed by $\sum_{i=1}^{A(t)} T_i$. Thus, the average number of packets in the system, K_a, can be obtained by

Equation 11.1

$$K_a = \frac{1}{t}\sum_{i=1}^{A(t)} T_i.$$

Figure 11.2. Accumulation of 5 packets in a queueing system



Similarly, the average time a packet spends in the system, T_a, can be represented by

Equation 11.2

$$T_a = \frac{1}{A(t)}\sum_{i=1}^{A(t)} T_i.$$

Also, the average number of arriving packets per second is given by » as

Equation 11.3

$$\lambda = \frac{A(t)}{t}.$$

By combining Equations (11.1), (11.2), and (11.3), we obtain

Equation 11.4

$$K_a = \lambda T_a.$$

This important result leads to a final form: Little's formula. This formula makes the assumption that t is large enough and that K_a and T_a therefore converge to their expected values of corresponding random processes E [K(t)] and E [T], respectively. Thus:

Equation 11.5

$$E[K(t)] = \lambda E[T].$$

Little's formula holds for most service disciplines with an arbitrary number of servers. Therefore, the assumption of first-come, first-served service discipline is not necessary.

Example. Consider a data-communication system with three transmission lines; packets arrive at three different nodes with arrival rates λ_1 = 200 packets/sec, λ_2 = 300 packets/sec, and λ_3 = 10 packets/sec, respectively. Assume that an average of 50,000 same- size packets float in this system. Find the average delay per packet in the system.

Solution. This delay is derived by

$$T_a = \frac{K_a}{\lambda_1 + \lambda_2 + \lambda_3} = \frac{50,000}{200 + 300 + 10} = 98.4 \text{ sec.}$$

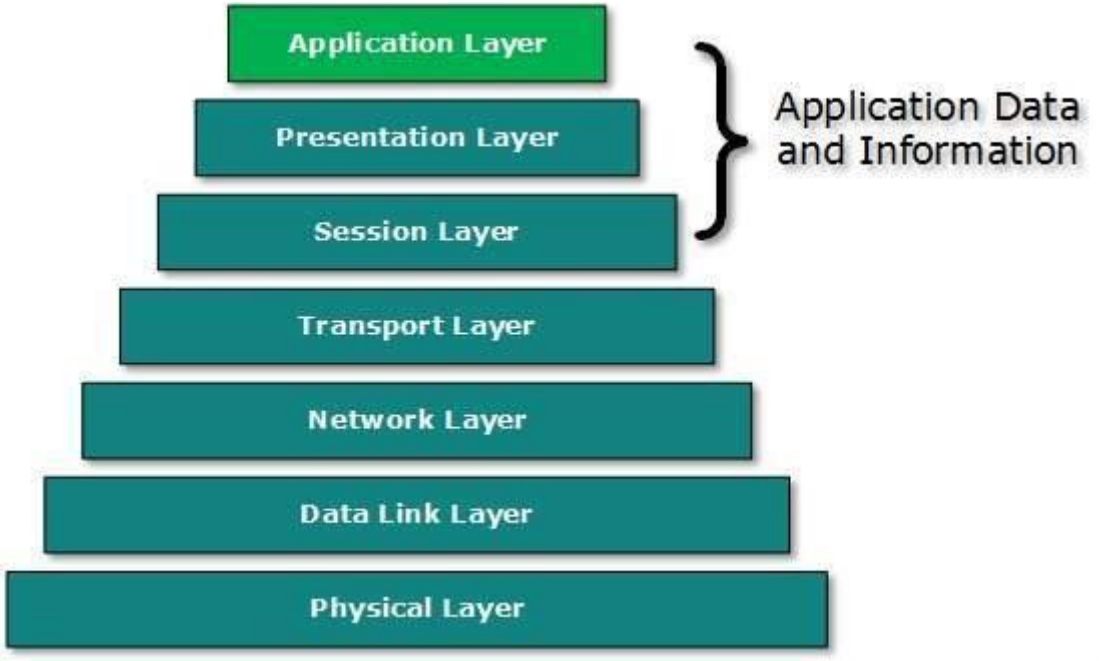
This is a simple application of Little's formula in a communication system.

APPLICATION LAYER.

Application layer is the top most layer in OSI and TCP/IP layered model. This layer exists in both layered Models because of its significance, of interacting with user and user applications. This layer is for applications which are involved in communication system.

A user may or may not directly interacts with the applications. Application layer is where the actual communication is initiated and reflects. Because this layer is on the top of the layer stack, it does not serve any other layers. Application layer takes the help of Transport and all layers below it to communicate or transfer its data to the remote host.

When an application layer protocol wants to communicate with its peer application layer protocol on remote host, it hands over the data or information to the Transport layer. The transport layer does the rest with the help of all the layers below it.



There'is an ambiguity in understanding Application Layer and its protocol. Not every user application can be put into Application Layer. except those applications which interact with the communication system. For example, designing software or text-editor cannot be considered as application layer programs.

On the other hand, when we use a Web Browser, which is actually using Hyper Text Transfer Protocol (HTTP) to interact with the network. HTTP is Application Layer protocol.

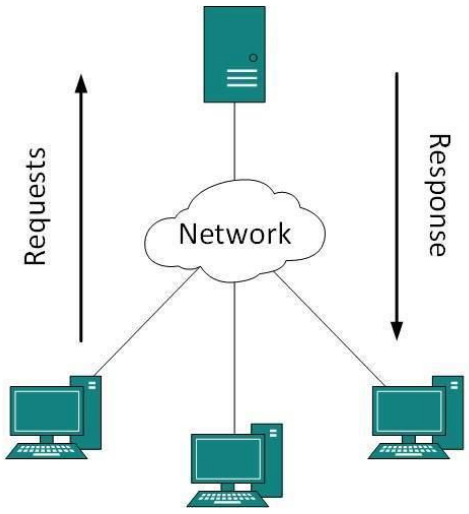
Another example is File Transfer Protocol, which helps a user to transfer text based or binary files across the network. A user can use this protocol in either GUI based software like FileZilla or CuteFTP and the same user can use FTP in Command Line mode.

Hence, irrespective of which software you use, it is the protocol which is considered at Application Layer used by that software. DNS is a protocol which helps user application protocols such as HTTP to accomplish its work.

Two remote application processes can communicate mainly in two different fashions:

- **Peer-to-peer:** Both remote processes are executing at same level and they exchange data using some shared resource.
- **Client-Server:** One remote process acts as a Client and requests some resource from another application process acting as Server.

In client-server model, any process can act as Server or Client. It is not the type of machine, size of the machine, or its computing power which makes it server; it is the ability of serving request that makes a machine a server.



A system can act as Server and Client simultaneously. That is, one process is acting as Server and another is acting as a client. This may also happen that both client and server processes reside on the same machine.

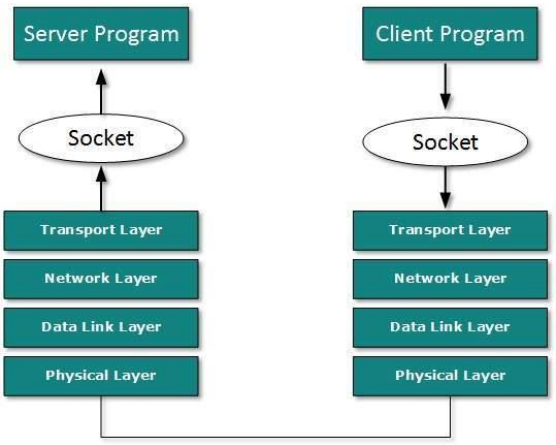
Communication

Two processes in client-server model can interact in various ways:

- Sockets
- Remote Procedure Calls (RPC)

Sockets

In this paradigm, the process acting as Server opens a socket using a well-known (or known by client) port and waits until some client request comes. The second process acting as a Client also opens a socket but instead of waiting for an incoming request, the client processes 'requests first'.



When the request is reached to server, it is served. It can either be an information sharing or resource request.

Remote Procedure Call

This is a mechanism where one process interacts with another by means of procedure calls. One process (client) calls the procedure lying on remote host. The process on remote host is said to be Server. Both processes are allocated stubs. This communication happens in the following way:

- The client process calls the client stub. It passes all the parameters pertaining to program local to it.
- All parameters are then packed (marshalled) and a system call is made to send them to other side of the network.
- Kernel sends the data over the network and the other end receives it.
- The remote host passes data to the server stub where it is unmarshalled.
- The parameters are passed to the procedure and the procedure is then executed.
- The result is sent back to the client in the same manner.

NETWORK APPLICATION SERVICES AND PROTOCOLS INCLUDING E-MAIL

Network service

In computer networking, a **network service** is an application running at the network application layer and above, that provides data storage, manipulation, presentation, communication or other capability which is often implemented using a client–server or peer-to-peer architecture based on application layer network protocols.[1]

Each service is usually provided by a server component running on one or more computers (often a dedicated server computer offering multiple services) and accessed via a network by client components running on other devices

Example: Domain Name System(DNS)

E-MAIL

E-mail Protocols are set of rules that help the client to properly transmit the information to or from the mail server. Here in this tutorial, we will discuss various protocols such as **SMTP, POP,** and **IMAP.**

SMPTP

SMTP stands for **Simple Mail Transfer Protocol**. It was first proposed in 1982. It is a standard protocol used for sending e-mail efficiently and reliably over the internet.

There are three common protocols used to deliver email over the Internet: the **Simple Mail Transfer Protocol (SMTP)**, the Post Office Protocol (POP), and the Internet Message Access Protocol (IMAP). All three use TCP, and the last two are used for accessing electronic mailboxes.

These are:

1.Transmission Control Protocol (TCP)
2.Internet Protocol (IP)
3.User Datagram Protocol (UDP)
4.Post office Protocol (POP)
5.Simple mail transport Protocol (SMTP)
6.File Transfer Protocol (FTP)
7.Hyper Text Transfer Protocol (HTTP)
8.Hyper Text Transfer Protocol Secure (HTTPS)

Key Points:

- SMTP is application level protocol.
- SMTP is connection oriented protocol.
- SMTP is text based protocol.
- It handles exchange of messages between e-mail servers over TCP/IP network.
- Apart from transferring e-mail, SMPT also provides notification regarding incoming mail.
- When you send e-mail, your e-mail client sends it to your e-mail server which further contacts the recipient mail server using SMTP client.
- These SMTP commands specify the sender's and receiver's e-mail address, along with the message to be send.
- The exchange of commands between servers is carried out without intervention of any user.
- In case, message cannot be delivered, an error report is sent to the sender which makes SMTP a reliable protocol.

SMTP Commands

The following table describes some of the SMTP commands:

S.N.	Command Description
1	**HELLO** This command initiates the SMTP conversation.
2	**EHELLO** This is an alternative command to initiate the conversation. ESMTP indicates that the sender server wants to use extended SMTP protocol.

3	**MAIL FROM** This indicates the sender's address.
4	**RCPT TO** It identifies the recipient of the mail. In order to deliver similar message to multiple users this command can be repeated multiple times.
5	**SIZE** This command let the server know the size of attached message in bytes.
6	**DATA** The **DATA** command signifies that a stream of data will follow. Here stream of data refers to the body of the message.
7	**QUIT** This commands is used to terminate the SMTP connection.
8	**VERFY** This command is used by the receiving server in order to verify whether the given username is valid or not.
9	**EXPN** It is same as VRFY, except it will list all the users name when it used with a distribution list.

IMAP

IMAP stands for **Internet Message Access Protocol.** It was first proposed in 1986. There exist five versions of IMAP as follows:

1. Original IMAP
2. IMAP2
3. IMAP3
4. IMAP2bis
5. IMAP4

Key Points:

- IMAP allows the client program to manipulate the e-mail message on the server without downloading them on the local computer.
- The e-mail is hold and maintained by the remote server.
- It enables us to take any action such as downloading, delete the mail without reading the mail.It enables us to create, manipulate and delete remote message folders called mail boxes.
- IMAP enables the users to search the e-mails.
- It allows concurrent access to multiple mailboxes on multiple mail servers.

IMAP Commands

The following table describes some of the IMAP commands:

S.N.	Command Description
1	**IMAP_LOGIN** This command opens the connection.
2	**CAPABILITY** This command requests for listing the capabilities that the server supports.
3	**NOOP** This command is used as a periodic poll for new messages or message status updates during a period of inactivity.
4	**SELECT** This command helps to select a mailbox to access the messages.
5	**EXAMINE** It is same as SELECT command except no change to the mailbox is permitted.
6	**CREATE** It is used to create mailbox with a specified name.
7	**DELETE** It is used to permanently delete a mailbox with a given name.
8	**RENAME** It is used to change the name of a mailbox.
9	**LOGOUT** This command informs the server that client is done with the session. The server must send BYE untagged response before the OK response and then close the network connection.

POP

POP stands for Post Office Protocol. It is generally used to support a single client. There are several versions of POP but the POP 3 is the current standard.

Key Points

- POP is an application layer internet standard protocol.
- Since POP supports offline access to the messages, thus requires less internet usage time.

- POP does not allow search facility.
- In order to access the messaged, it is necessary to download them.
- It allows only one mailbox to be created on server.
- It is not suitable for accessing non mail data.
- POP commands are generally abbreviated into codes of three or four letters. Eg. STAT.

POP Commands

The following table describes some of the POP commands:

S.N.	Command Description
1	**LOGIN** This command opens the connection.
2	**STAT** It is used to display number of messages currently in the mailbox.
3	**LIST** It is used to get the summary of messages where each message summary is shown.
4	**RETR** This command helps to select a mailbox to access the messages.
5	**DELE** It is used to delete a message.
6	**RSET** It is used to reset the session to its initial state.
7	**QUIT** It is used to log off the session.

Comparison between POP and IMAP

S.N.	POP	IMAP
1	Generally used to support single client.	Designed to handle multiple clients.
2	Messages are accessed offline.	Messages are accessed online although it also supports offline mode.

3	POP does not allow search facility.	It offers ability to search emails.
4	All the messages have to be downloaded.	It allows selective transfer of messages to the client.
5	Only one mailbox can be created on the server.	Multiple mailboxes can be created on the server.
6	Not suitable for accessing non-mail data.	Suitable for accessing non-mail data i.e. attachment.
7	POP commands are generally abbreviated into codes of three or four letters. Eg. STAT.	IMAP commands are not abbreviated, they are full. Eg. STATUS.
8	It requires minimum use of server resources.	Clients are totally dependent on server.
9	Mails once downloaded cannot be accessed from some other location.	Allows mails to be accessed from multiple locations.
10	The e-mails are not downloaded automatically.	Users can view the headings and sender of e-mails and then decide to download.
10	POP requires less internet usage time.	IMAP requires more internet usage time.

E-MAIL SYSTEM

E-mail system comprises of the following three components:

- Mailer
- Mail Server
- Mailbox

Mailer

It is also called **mail program, mail application** or **mail client.** It allows us to manage, read and compose e-mail.

Mail Server

The function of mail server is to receive, store and deliver the email. It is must for mail servers to be Running all the time because if it crashes or is down, email can be lost.

Mailboxes

Mailbox is generally a folder that contains emails and information about them.

Working of E-mail

Email working follows the client server approach. In this client is the mailer i.e. the mail application or mail program and server is a device that manages emails.

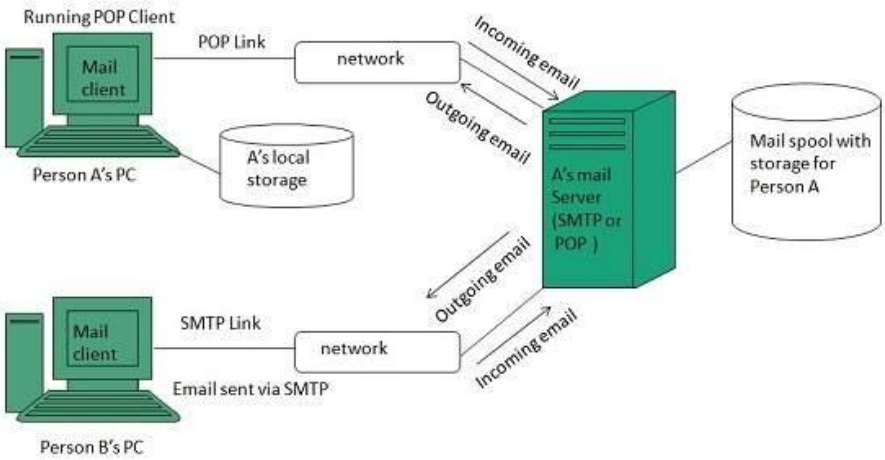
Following example will take you through the basic steps involved in sending and receiving emails and will give you a better understanding of working of email system:

- Suppose person A wants to send an email message to person B.
- Person A composes the messages using a mailer program i.e. mail client and then select Send option.
- The message is routed to **Simple Mail Transfer Protocol** to person B's mail server.
- The mail server stores the email message on disk in an area designated for person B.

The disk space area on mail server is called mail spool.

- Now, suppose person B is running a POP client and knows how to communicate with B's mail server.
- It will periodically poll the POP server to check if any new email has arrived for B.As in this case, person B has sent an email for person B, so email is forwarded over the network to B's PC. This is message is now stored on person B's PC.

The following diagram gives pictorial representation of the steps discussed above:



World Wide Web (WWW)

The **World Wide Web** abbreviated as WWW and commonly known as the web. The WWW was initiated by CERN (European library for Nuclear Research) in 1989.

SystemArchitecture:

From the user's point of view, the web consists of a vast, worldwide connection of documents or web pages. Each page may contain links to other pages anywhere in the world. The pages can be retrieved and viewed by using browsers of which internet explorer, Netscape Navigator, Google, Chrome, etc are the popular ones. The browser fetches the page requested interprets the text and formatting commands on it, and displays the page, properly formatted, on the screen.

Working of WWW:

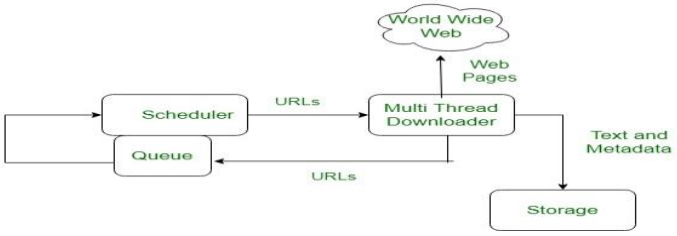
The World Wide Web is based on several different technologies: Web browsers, Hypertext Markup Language (HTML) and Hypertext Transfer Protocol (HTTP).
A Web browser is used to access webpages. Web browsers can be defined as programs which display text, data, pictures, animation and video on the Internet. Hyperlinked resources on the World Wide Web can be accessed using software interface provided by Web browsers. Initially Web browsers were used only for surfing the Web but now they have become more universal. Web browsers can be used for several tasks including conducting searches, mailing, transferring files, and much more. Some of the commonly used browsers are Internet Explorer, Opera Mini, Google Chrome.

Features of WWW:

- HyperText Information System
- Cross-Platform
- Distributed
- Open Standards and Open Source
- Uses Web Browsers to provide a single interface for many services
- Dynamic, Interactive and Evolving.
- "Web 2.0"

Components of Web: There are 3 components of web:

1. **Uniform Resource Locator (URL):** serves as system for resources on web.
2. **HyperText Transfer Protocol (HTTP):** specifies communication of browser and server.
3. **Hyper Text Markup Language (HTML):** defines structure, organisation and content of webpage.



Domain Name System(DNS)

- Domain Name System is an Internet service that translates domain names into IP addresses.
- The DNS has a distributed database that resides on multiple machines on the Internet.
- DNS has some protocols that allow the client and servers to communicate with each other.
- When the Internet was small, mapping was done by using hosts.txt file.
- The host file was located at host's disk and updated periodically from a master host file.
- When any program or any user wanted to map domain name to an address, the host consulted the host file and found the mapping.
- Now Internet is not small, it is impossible to have only one host file to relate every address with a name and vice versa.
- The solution used today is to divide the host file into smaller parts and store each part on a different computer.
- In this method, the host that needs mapping can call the closest computer holding the needed information.
- This method is used in Domain Name System (DNS).

Name space

- The names assigned to the machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses.
- There are two types of name spaces: Flat name spaces and Hierarchical names.

Flat name spaces

- In a flat name space, a name is a sequence of characters without structure.

- A name in this space is assigned to an address.
- The names were convenient and short.
- A flat name space cannot be used in a large system such as the internet because it must be centrally controlled to avoid ambiguity and duplication.

Hierarchical Name Space

- In hierarchical name space, each name consists of several parts.
- First part defines the nature of the organization, second part defines the name of an organization, third part defines department of the organization, and so on.
- In hierarchical name space, the authority to assign and control the name spaces can be decentralized.
- Authority for names in each partition is passed to each designated agent.

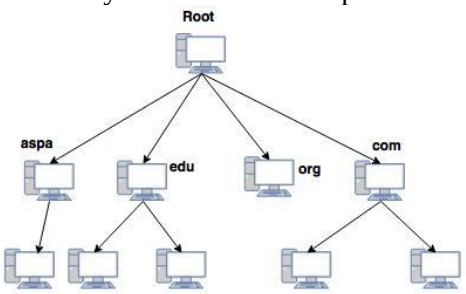


Fig: Hierarchy of DNS

DNS in the Internet

- DNS is a protocol that can be used in different platform.
- Domain Name Space is divided into different sections in the Internet: Generic domain, country domain and inverse domain.

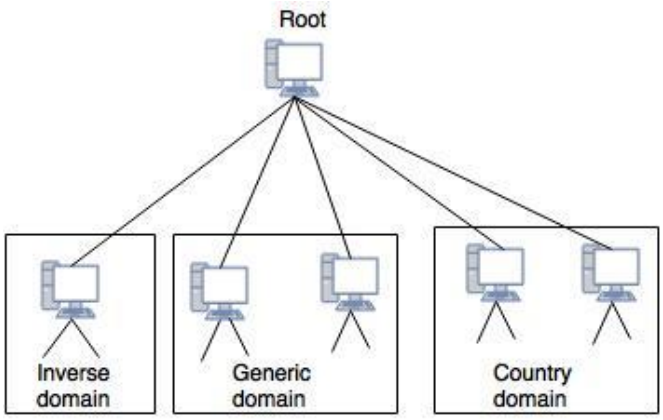


Fig. : DNS in the Internet

Generic Domains

The generic domains define registered hosts according to their generic behavior.

Generic domain labels are as stated below:

1. Country Domains

- Country domain uses two character country abbreviations.
- Second labels can be more specific, national designation.
- **For example,** for Australia the country domain is "au", Inida is .in, UK is .uk etc.

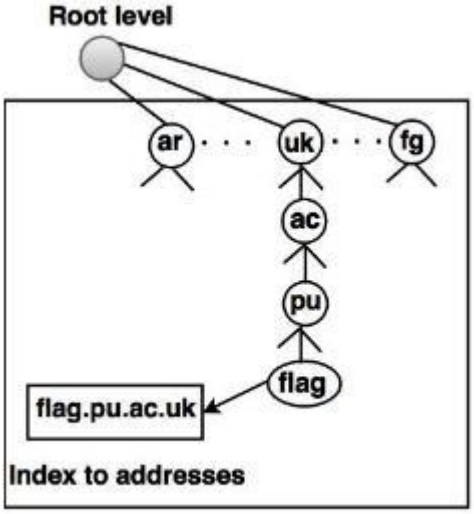


Fig: Country domains

2. Inverse Domains

- Inverse domain is used to map an address to a name.
- **For example,** a client send a request to the server for performing a particular task, server finds a list of authorized client. The list contains only IP addresses of the client.
- The server sends a query to the DNS server to map an address to a name to determine if the client is on the authorized list.
- This query is called an inverse query.
- This query is handled by first level node called arpa.

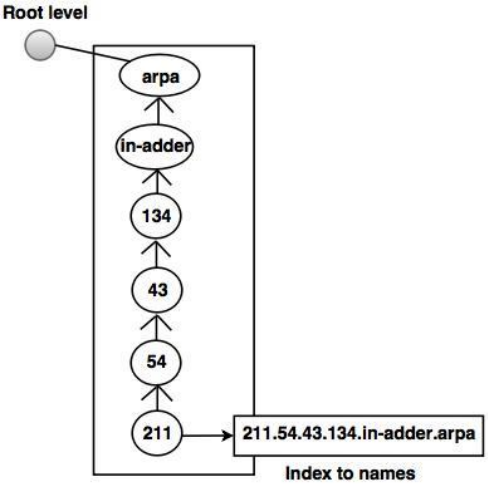


Fig. Inverse domain

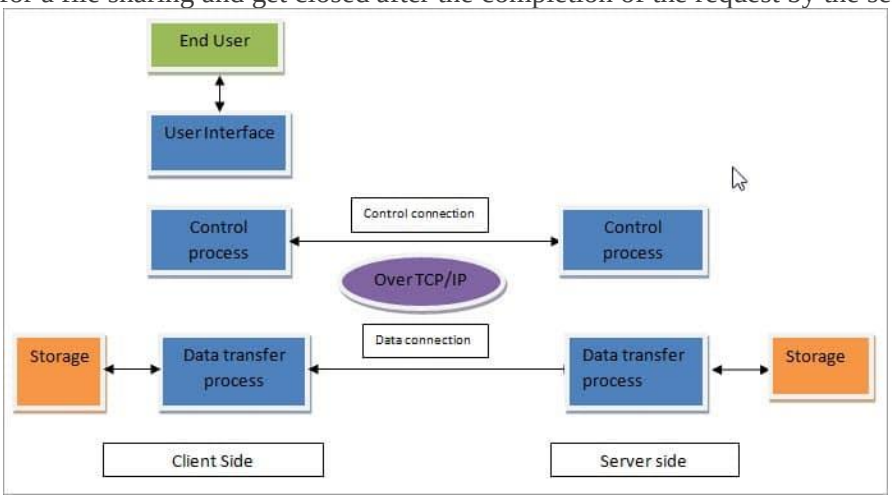
FTP (FILE TRANSFER PROTOCOL)

- **FTP** is File Transfer Protocol.
- It used to exchange files on the internet.
- To enable the data transfer FTP uses TCP/IP, FTP is most commonly used to upload and download files from the internet.
- FTP can be invoked from the command prompt or some graphical user interface.
- FTP also allows to update (delete, rename, move, and copy) files at a server.
- It uses a reserved port no 21.

- It is one of the widely used application layer protocol of the TCP/IP protocol suite. FTP is basically used to exchange data between two host devices over the Internet or Intranet securely.
- It is referred to as one of the safest modes of file sharing among systems, and thus it is deployed by large industries, universities, and offices.
- It works in the client-server model and thus the user needs an FTP client program to run FTP on its system. The common types of FTP client program include Filezilla and Dreamweaver etc.
- The data transfer takes place only in one direction at a time. The FTP protocol carry out many duties apart from file transfer like creation and deletion of data files, listing, renaming, etc.

The FTP Model

- In this model, one host behaves as the client and another host as a server. The one who requests for file-sharing or data is the client host and one which in response completes the request is the server host.
- Firstly the FTP connection is established between the client and server computer and data exchange take place after that. Two channels come into the picture of FTP connection i.e. control channel and data channel.
- The control channel establishes the connection between the client and server and remains open for the overall session. The control channel port number is 21 in TCP/IP. While the data channel opens when the client request for a file sharing and get closed after the completion of the request by the server.



- Two processes naming data transfer process (DTP) and protocol interpreter (PI) are used in managing the communication between the client and the server. The DTP establishes and manages the connection for the data channel, while PI manages the DTP by applying commands given by the control channel.
- The server host end PI is accountable for analyzing the commands received from the client host end via the control channel, connection establishment, and in running the DTP. The client PI is accountable for forwarding the FTP commands, receiving the response from the server and establishment of the connection with the FTP server.
- After the establishment of a connection between the FTP client and the FTP server, the client builds up the connection and sends the FTP commands to the server. The server analyzes them and in response completes the request.
- Now the server end PI sends the port detail on which the files will be forwarded to the client DTP. The client DTP then waits for the data to arrive at the decided port from the server.

The FTP Response

- To make out a secure and reliable file transfer between the client and server, it is important that the server and client should remain in synchronization with each other.

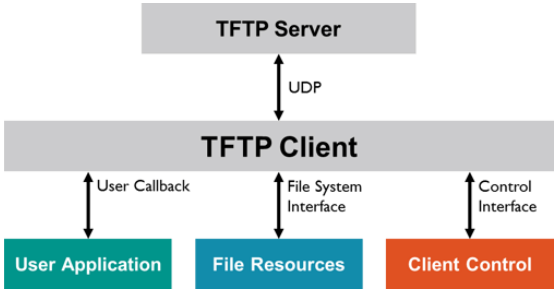
- Thus for each command executed by the client, a user is acknowledged by the response and the action is performed by the server host in order. The response consists of a 3 digit code plus a text (a character string is separated from digit by a space) denoting the processing of the commands.

TFTP (TRIVIAL FILE TRANSFER PROTOCOL)

- TFTP is used when a file transfer does not require an acknowledgment packet during file transfer. TFTP is used often in the router configuration. TFTP is similar in operation to FTP. TFTP is also a command-line-based utility.
- One of the two primary differences between TFTP and FTP is speed and authentication. Because TFTP is used without acknowledgment packets, it is usually faster than FTP. TFTP does not provide user authentication like FTP and therefore the user must be logged on to the client and the files on the remote computer must be writable. TFTP supports only unidirectional data transfer (unlike FTP, which supports bi-directional transfer). TFTP is operated over port 69.

Trivial File Transfer Protocol is very simple in design and has limited features as compared to File Transfer Protocol (FTP). TFTP provides no authentication and security while transferring files. As a result, it is usually used for transferring boot files or configuration files between machines in a local setup. Because of its simple design, it is rarely used interactively by users in a computer network. Its lack of security also makes it dangerousforuseovertheInternet.

TFTP is very useful for boot computers and devices that do not have hard disk drives or storage devices because it can easily be implemented using a small amount of memory. This characteristic of TFTP makes it one of the core Data transfer through TFTP is usually initiated through port 69.



S.NO	FTP	TFTP
1.	FTP stands for File Transfer Protocol.	TFTP stands for Trivial File Transfer Protocol.

2.	The software of FTP is larger than TFTP.	While software of TFTP is smaller than FTP.
3.	FTP works on two ports: 20 and 21.	While TFTP works on 69 Port number.
4.	FTP services are provided by TCP.	While TFTP services are provided by UDP.
5.	The complexity of FTP is higher than TFTP.	While the complexity of TFTP is less than FTP complexity.
6.	There are many commands or messages in FTP.	There are only 5 messages in TFTP.
7.	FTP need authentication for communication.	While TFTP does not need authentication for communication.
8.	FTP is generally suited for uploading and downloading of files by remote users.	While TFTP is mainly used for transmission of configurations to and from network devices.

TELNET

TELNET (**TEL**ecommunication **NET**work) is a network protocol used on the Internet or local area network (LAN) connections. It was developed in 1969 beginning with RFC 15 and standardized as IETF STD 8, one of the first Internet standards.

It is a network protocol used on the Internet or local area networks to provide a bidirectional interactive communications facility. Typically, telnet provides access to a command-line interface on a remote host via a virtual terminal connection which consists of an 8-bit byte oriented data connection over the Transmission Control Protocol (TCP). User data is interspersed in-band with TELNET control information. The user's computer, which initiates the connection,isreferredtoasthelocal computer.

The network terminal protocol (TELNET) allows a user to log in on any other computer on the network. We can start a remote session by specifying a computer to connect to. From that time until we finish the session, anything we type issenttotheothercomputer.

The Telnet program runs on the computer and connects your PC to a server on the network. We can then enter commands through the Telnet program and they will be executed as if we were entering them directly on the server console. This enables we to control the server and communicate with other servers on the network. To start a Telnet session, we must log in to a server by entering a valid username and password. Telnet is a common way to remotely controlWebservers.

The term **telnet** also refers to software which implements the client part of the protocol. TELNET clients have been available on most Unix systems for many years and are available virtually for all platforms. Most network equipment and OSs with a TCP/IP stack support some kind of TELNET service server for their remote configuration including ones based on Windows NT. TELNET is a client server protocol, based on a reliable connection oriented transport.

Typically this protocol used to establish a connection to TCP port 23, where a getty-equivalent program (telnetd) is listening, although TELNET predates.

Port* 21 ~ *File Transfer Protocol
Port* 22 – *SSH Remote Login Protocol
Port* 23 – *Telnet Server
Port* 25 – *Simple Mail Transfer Protocol (SMTP)
Port* 53 – *Domain Name Server (DNS)
Port* 69 – *Trivial File Transfer Protocol (TFTP)
Port 70 – Gopher
Port 80 – Hyper Text Transfer Protocol (HTTP)
Port 110 – Post Office Protocol* 3 *(POP3)

A personal computer can connect via Modem to a large computer and run a terminal emulation program. The most common terminal emulation is the VT100. The computer works like a dumb terminal, except it is connected via a phone line instead of a direct connection. Often, you will not be able to use graphics on the Internet, such as the WWW (World Wide Web), this kind of access, although you will be able to browse the text-only portion of the Web.

This kind of Internet account is sometimes called "Shell" account. This shell account is available with VSNL for students in India. Many terminal emulation programs can emulate DEC terminals, including the VT52 and VT200 series terminals. For example, tty pathname of your terminal's device file.

The syntax for this command is

tty [option]

The options are:

1. -1 Prints the synchronous line number.

2. -s Causes tty not to print any output but sets the exit status to 0 if the standard input file is a terminal, and to 1 if it is not.

TELNET is generally used with the following applications :

(1) Enterprise networks to access host applications, e.g. on IBM Mainframes.

(2) Administration of network elements, e.g., in commissioning, integration and maintenance of core network elements in mobile communication networks.

(3) MUD games played over the Internet, as well as talkers, MUSHes, MUCKs, MOOes, and the resurgent BBS community.

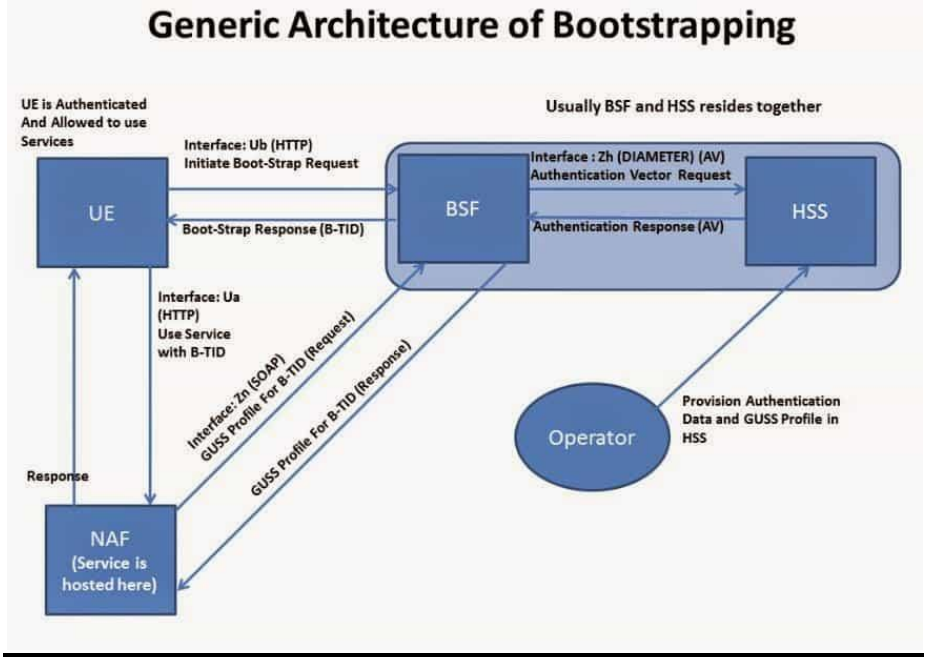
(4) embedded systems.

BOOTP

The **Bootstrap Protocol** (**BOOTP**) is a computer networking protocol used in Internet Protocol networks to automatically assign an IP address to network devices from a configuration server. The BOOTP was originally defined in RFC 951.

When a network-connected computer boots-up, its IP-stack broadcasts BOOTP network-messages requesting an IP-address assignment. A BOOTP configuration-server replies to the request by assigning an IP address from a pool of addresses, which is preconfigured by an administrator.

BOOTP is implemented using the User Datagram Protocol (UDP) for transport protocol, port number 67 is used by the (DHCP) server for receiving client-requests and port number 68 is used by the client for receiving (DHCP) server responses. BOOTP operates only on IPv4 networks.



If BOOTP is to be used in your network, the server and client are usually on the same physical LAN segment. BOOTP can only be used across bridged segments when source-routing bridges are being used, or across subnets, if you have a router capable of BOOTP forwarding.

BOOTP is a draft standard protocol. Its status is recommended. There are also updates to BOOTP, some relating to interoperability with DHCP .BOOTP are draft standards with a status of elective and recommended, respectively. The BOOTP protocol was originally developed as a mechanism to enable diskless hosts to be remotely booted over a network as workstations, routers, terminal concentrators, and so on.

It allows a minimum IP protocol stack with no configuration information to obtain enough information to begin the process of downloading the necessary boot code. BOOTP does not define how the downloading is done, but this process typically uses TFTP “Trivial File Transfer Protocol (TFTP)”. Although still widely used for this purpose by diskless hosts, BOOTP is also commonly used solely as a mechanism to deliver configuration information to a client that has not been manually configured.

The BOOTP process involves the following steps :

(1) The client determines its own hardware address; this is normally in a ROM on the hardware.

(2) A BOOTP client sends its hardware address in a UDP datagram to the server.

BOOTP stands for **Bootstrap Protocol.** and **DHCP** stands for Dynamic host configuration protocol. These protocols square measure used for getting the information science address of the host along side the bootstrap info. The operating of each protocols is totally different in some manner.Dynamic host configuration protocol is also the extended version of the Bootstrap Protocol.

Let’s see that the difference between that BOOTP and DHCP:

S.NO	BOOTP	DHCP
1.	BOOTP stands for Bootstrap Protocol.	While DHCP stands for Dynamic host configuration protocol.
2.	BOOTP does not provide temporary IP addressing.	While DHCP provides temporary IP addressing for only limited amount of time.

3.	BOOTP does not support DHCP clients.	While it support BOOTP clients.
4.	In BOOTP, manual-configuration takes place.	While in DHCP, auto-configuration takes place.
5.	BOOTP does not support mobile machines.	Whereas DHCP supports mobile machines.
6.	BOOTP can have errors due to manual-configuration.	Whereas in DHCP errors do not occure mostly due to auto-configuration.

HTTP

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (i.e. internet) . HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers.

Basically, HTTP is a TCP/IP based communication protocol, that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web. The default port is TCP 80, but other ports can be used as well. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients' request data will be constructed and sent to the server, and how the servers respond to these requests.

Basic Features

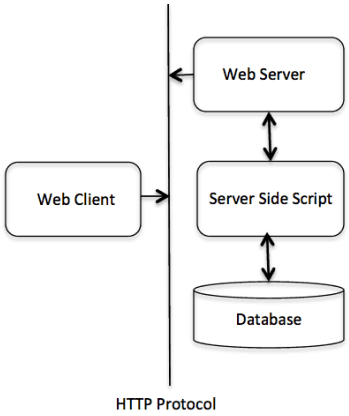
There are three basic features that make HTTP a simple but powerful protocol:

- **HTTP is connectionless:** The HTTP client, i.e., a browser initiates an HTTP request and after a request is made, the client waits for the response. The server processes the request and sends a response back after which client disconnect the connection. So client and server knows about each other during current request and response only. Further requests are made on new connection like client and server are new to each other.
- **HTTP is media independent:** It means, any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content. It is required for the client as well as the server to specify the content type using appropriate MIME-type.
- **HTTP is stateless:** As mentioned above, HTTP is connectionless and it is a direct result of HTTP being a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across the web pages.

HTTP/1.0 uses a new connection for each request/response exchange, where as HTTP/1.1 connection may be used for one or more request/response exchanges.

Basic Architecture

The following diagram shows a very basic architecture of a web application and depicts where HTTP sits:



The HTTP protocol is a request/response protocol based on the client/server based architecture where web browsers, robots and search engines, etc. act like HTTP clients, and the Web server acts as a server.

Client

The HTTP client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a TCP/IP connection.

Server

The HTTP server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta information, and possible entity-body content.

HTTP Version

HTTP uses a **<major>.<minor>** numbering scheme to indicate versions of the protocol. The version of an HTTP message is indicated by an HTTP-Version field in the first line. Here is the general syntax of specifying HTTP version number:

```
HTTP-Version   = "HTTP" "/" 1*DIGIT "." 1*DIGIT
```

Example

```
HTTP/1.0

or

HTTP/1.1
```

IP security (IPSec)

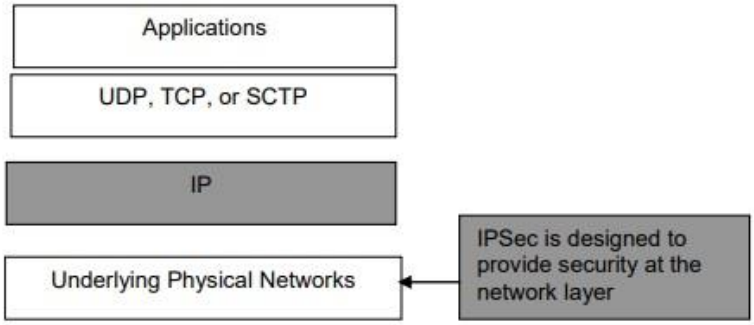
IP Security (IPSec) is a collection of protocols which is designed by Internet Engineering Task Force (IETF) to provide security for a packet at the network level. It helps to create confidential and authenticated and packets for the IP layer as shown in below diagram –

The last three topics cover the three main IPsec protocols:

IPsec Authentication Header (AH),

IPsec Encapsulating Security Payload (ESP),

IPsec Internet Key Exchange (IKE).



IPSec protocol aim is to provide security services for IP packets like encrypting sensitive data/packets, authentication, and protection against replay and data confidentiality. It can be configured to operate in two different modes −

- Tunnel Mode
- Transport mode.

The original packet is generated as follows −

IP Header	UDP Header	Data

Let us discuss each mode in detail.

Tunnel mode

IPSec tunnel mode is the default mode. IPSec Tunnel mode is most widely used to create site-to-site IPSec VPN.

Let see the packet format of IPSec tunnel mode with ESP header −

|□-----Original Packet----------□|

NewIP Header	ESP Header	IP Header	TCP/UDP Header	Data	ESP Trailer	EXP Auth.trailer

|□-------Encrypted--------------------------□|

|----------------------Authenticated----------------------□|

From the above format we can conclude the following −

- The encrypted part of the packet contains the following −

IP Header	UDP Header	Data	ESP Trailer

- The authenticated part of the packet contains the following −

ESP Header	IP Header	UDP Header	Data	ESP Trailer

Transport Mode

IPSec Transport mode is used for end-to-end communications. In this only, the Data Payload of the IP datagram is secured by IPSec.

IP Header	ESP Header	TCP/UDP Header	Data	ESP Trailer	EXP Auth.trailer

|□-------Encrypted-------------□|

|----------Authenticated---------□|

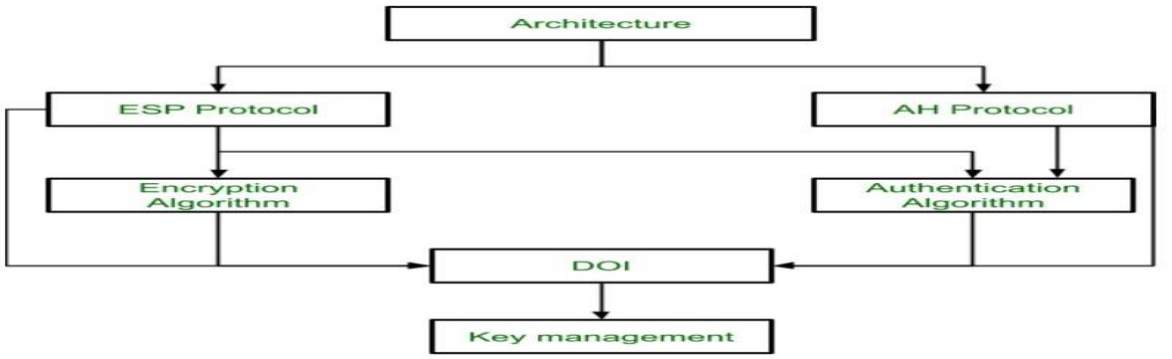
From the above format we conclude the following −

- The encrypted part of the packet contains the following −

UDP Header	Data	ESP Trailer

- The authenticated part of the packet contains the following −

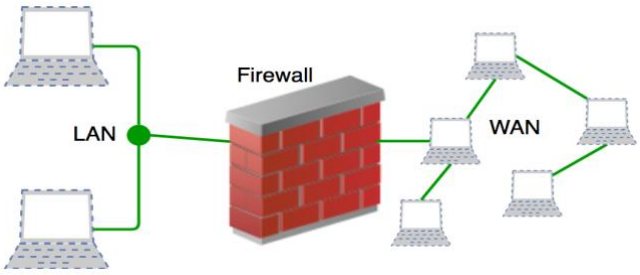
ESP Header	UDP Header	Data	ESP Trailer



FIREWALLS

A firewall is a network security device, either hardware or software-based, which monitors all incoming and outgoing traffic and based on a defined set of security rules it accepts, rejects or drops that specific traffic.

Accept : allow the traffic
Reject : block the traffic but reply with an "unreachable error"
Drop : block the traffic with no reply
A firewall establishes a barrier between secured internal networks and outside untrusted network, such as the Internet.



TYPES OF FIREWALL

There are several types of Firewalls which can be installed depending on the requirements of the users. We shall explore what are the different types of Firewalls available.

1.Proxy Firewall: This is one of the early types of Firewalls. One of the characteristics of this Firewall is that it prevents direct connection to a computer for outside networks**.** This Firewall functions for specific applications as a gateway from one network to another.
2.Packet Filtering Firewalls: This is about the most basic type and this Firewall application is for creating a barrier at the router. The working of the Firewall is only for checking the source and destination IPs without going into packet details. This is a Firewall example of one of the oldest architectures.

3.Circuit level Gateways: This simple Firewall configuration works by checking the transmission control protocol, TCP, handshake to ensure that the packet is from a legitimate source.

4.Stateful Inspection Firewalls: This is a different type of Firewall which combines the above two methods to create a configuration with a higher level of security.

5.Next-Generation Firewalls(NGFW): As the threats increase, most of the bigger companies are going in for the next-generation Firewalls which provide far greater benefits of a Firewall. These NGFW types come with many high-end capabilities to provide top-end Firewall advantages to large organizations with a higher risk profile.

6.Software Firewalls: A software Firewall definition is that it is a software installed on a device without any additional hardware. This answers the question of what is a Firewall software as well. Is Windows Firewall enough might be a legitimate doubt regarding stand-alone or individual computers? While windows give some basic protection, the need for firewalls in the form of an anti-virus plus application is always there.

7.Hardware Firewalls: Hardware Firewalls comprise devices that act as the first line of defense to filter data packets before they arrive at the servers. The major disadvantages of Firewalls of the hardware type are that they are easy for insider attacks to bypass them and the hardware capabilities vary with each supplier.

USES OF A FIREWALL

The uses of a Firewall are numerous. Questions like what is the use of a forewall or what does a Firewall do or what is a Firewall and what is its function are only natural when some big investment is required.A Firewall and its uses are given below for a better understanding.

1.The useful thing about a Firewall is that it prevents unauthorized remote access and remote control of your computer by a hacker for evil purposes.

2.Data security is ensured based on IP address and protocol.

3.It ensures continuity of operations and availability of information.

4.It blocks destructive or unsuitable or pornographic content.

5.It prevents ransomware, malware, and phishing attacks.

6.It shields older PCs with earlier versions of OS.

7.It is safer for online gamers.

The above points explain a Firewall adequately.

FEATURES OF A FIREWALL

A list given below answers the common query What are the features of a Firewall.

1.Bandwidth control and monitoring.

2.Web filtering.

3.Internet aggregation.

4.Sandboxing.

5.VPN, Virtual Private Networks.

6.Deep Packet Inspection, DPI.

LIST OF FIREWALLS

A small list of Firewalls available is provided.

1.CISCO

2.Checkpoint

3.Fortinet

4.Watchguard