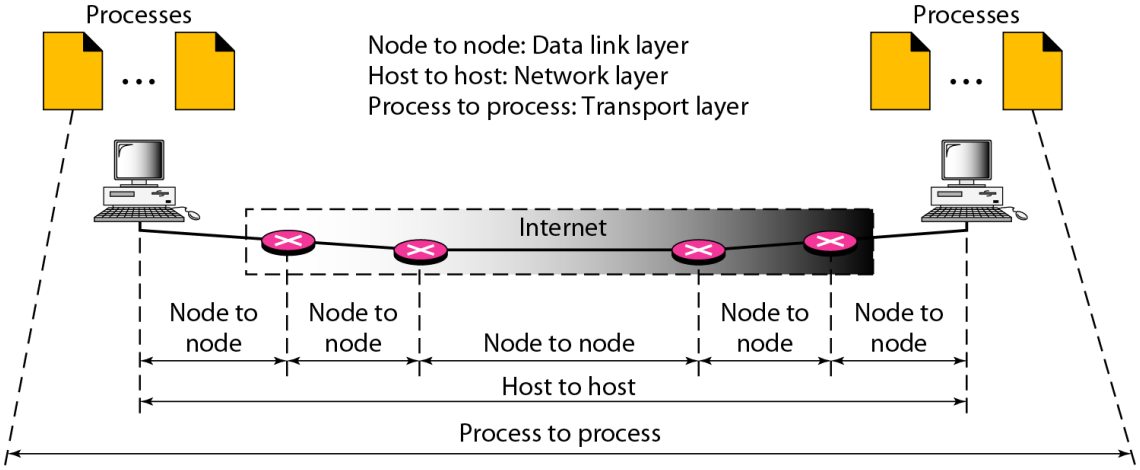
TRANSPORT LAYER

PROCESS TO PROCESS DELIVERY

A Process is an Application program that runs on the host. At any moment several processes may be running on the source host and destination host.

Transport Layer is responsible for delivery of a packet from one process on source host to another process on destination host.

Two processes communicate in a client/server relationship.



Client/Server Paradigm

A Client is a process on a local host, whereas Server is a process on remote host. A Client needs services from Server. Both Client and Server processes have the same name.
Example: To get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.

For communication to be done between two processes we must have to define following:

1. Local host
2. Local process
3. Remote host
4. Remote process

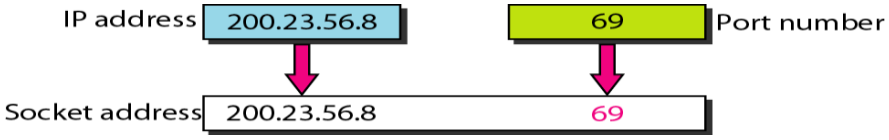
Addressing

- Transport layer needs an address called a **Port Number** or **Port Address** to choose among multiple processes running on the destination host.
- The port numbers are 16-bit integers between 0 and 65,535.
- Destination Port number is needed for Delivery. Source Port number is needed for the Reply.
- At the client side the port numbers are defined randomly whereas at the server side it uses Well-Known Port numbers.

Socket address

- The combination of an IP address and a port number is called a **Socket address**. UDP or TCP header contains the Port numbers.

- A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address.



Internet Assigned Number Authority (IANA) Ranges

Port Name	Range	Description
Well-known	0-1023	These are assigned and controlled by IANA
Registered	1024-49151	Not Assigned or controlled by IANA but these are registered with IANA to prevent duplication.
Dynamic or Private	49152-65535	These are neither controlled nor registered. They can be used by any process

Multiplexing and Demultiplexing

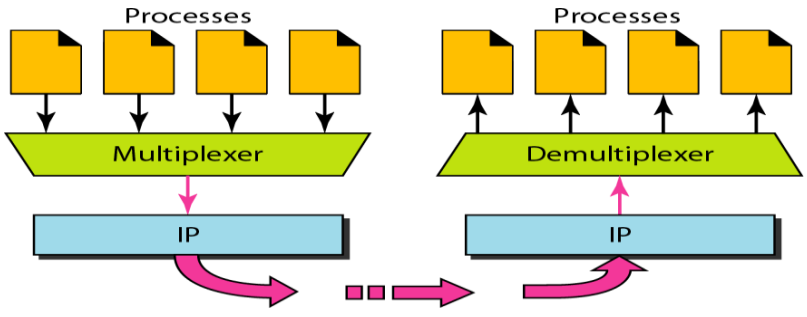
The addressing mechanism allows multiplexing and de-multiplexing by the transport layer.

Multiplexing

- At the sender site, there may be several processes that need to send packets. But there is only one transport layer protocol at anytime.
- The relationship in multiplexing is a many-to-one.
- The protocol accepts messages from different processes, differentiated by their assigned port numbers.
- After adding the header, the transport layer passes the packet to the network layer.

Demultiplexing

- At the receiver site, the relationship is one-to-many and requires demultiplexing.
- The transport layer receives datagrams from the network layer.
- After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.



Connectionless Service versus Connection-Oriented Service

Connection-Oriented Service (CO)	Connectionless Service (CL)
• In a CO service, to transfer the data 3 steps are followed by sender &receiver: 1. A connection is established 2. Data are transferred.	• In a CL service to transfer the data there is no need for connection establishment or connection release.

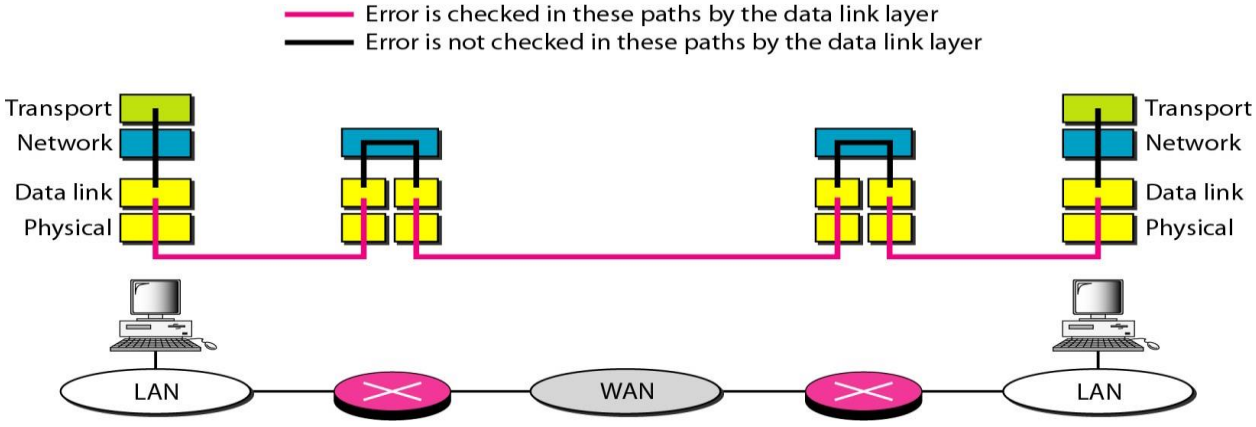
3. At the end, the connection is released. • The packets are numbered. • The packets arrived sequentially at the destination. Example: TCP, SCTP.	• The packets are not numbered. • The packets may arrive out of sequence at the destination. Example: UDP

Reliable Versus Unreliable service

Reliable service	**Unreliable service**
• Reliable service implements flow and error control at the transport layer. • It is a slower and more complex service. • Example:TCP	• Unreliable service does not implements flow and error control at the transport layer. • It is a faster service. • Example:UDP

Note: Reliability **(**Flow and Error control) mechanisms need to be implemented in both Data-link layer and transport layer.

- Reliability at the data link layer is between two nodes. It is node-to-node communication.
- The network layer in the Internet is unreliable (best-effort delivery). Hence we need to implement the reliability between two ends at the transport layer.
- Error control at the data link layer does not guarantee error control at the transport layer.



TRANSPORT LAYER PROTOCOL

There are 3 transport layer protocols are implemented:

1. UDP
2. TCP
3. SCTP

USER DATAGRAM PROTOCOL (UDP)

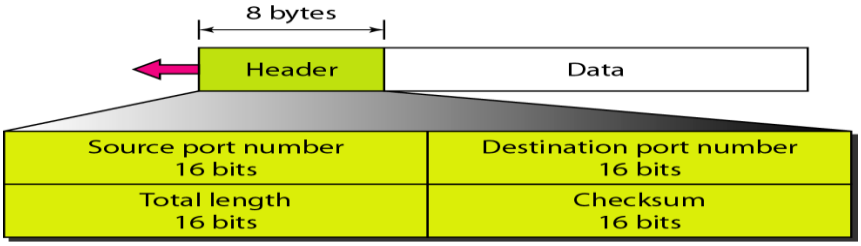
- UDP is called a connectionless, unreliable transport protocol.
- UDP is a very simple protocol using a minimum of overhead. UDP performs very limited error checking.
- UDP is used when a process wants to send a small message and does not need reliability.
- Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

Well-Known Ports for UDP

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
161	SNMP	Simple Network Management Protocol

User Datagram

UDP packets are called User Datagrams. Datagrams have fixed size header of length 8 bytes.



Fields of user datagram are:

Source port number (16 bits)

- This is the port number used by the process running on the source host.
- If the source host is the client (a client sending a request) the port number is an ephemeral port number requested by the process and chosen by the UDP software running on the source host.
- If the source host is the server (a server sending a response) the port number is a well-known port number.

Destination port number

- This is the port number used by the process running on the destination host.
- If the destination host is the server (a client sending a request) the port number is a well-known port number.
- If the destination host is the client (a server sending a response) the port number is an ephemeral port number, the server copies the ephemeral port number it has received in the request packet.

Note: Source and Destination port number can range from 0 to 65,535.

Total Length (16 bits)

The total length includes UDP header plus Data, total length ranges from 0-65535.

Checksum (16 bits)

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

UDP Operation

UDP uses the following four concepts:

1. Connectionless Service
2. No Flow and Error control
3. Encapsulation and Decapsulation
4. Queuing

Connectionless Services

- UDP provides a connectionless service. There is no connection establishment and no connection termination. Each user datagram sent by UDP is an independent datagram.
- The user datagrams are not numbered. Each user datagram can travel on a different path.
- There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
- UDP processes are capable of sending short messages only.

No Flow and Error Control

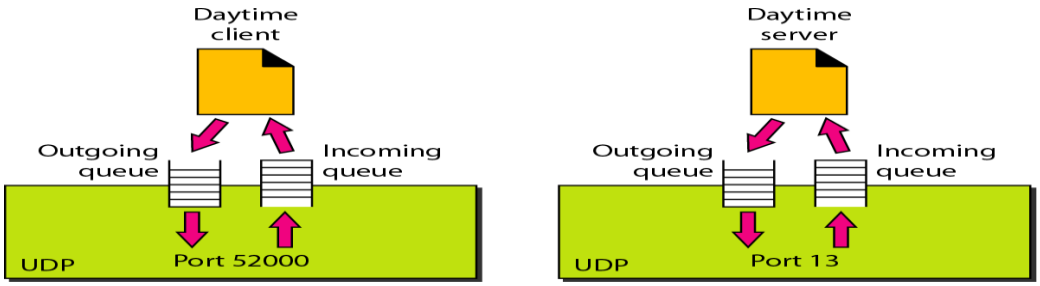
- There is no error control mechanism in UDP except for the checksum.
- There is no flow control in UDP; the receiver may overflow with incoming messages. The sender does not know if a message has been lost or duplicated.
- When the receiver detects an error through the checksum, the user datagram is silently discarded.

Encapsulation and Decapsulation

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

Queuing

- Queues are associated with ports in UDP. Queues are associated with each process.
- Queues are created when a process is started. There are two types of queues are created: Incoming queue and Outgoing queue.
- A process wants to communicate with multiple processes; it obtains only one port number and one outgoing queue and one incoming queue.
- The queues function as long as the process is running. When the process terminates, the queues aredestroyed.



Queues at Client site

- The queues opened by the client are identified by ephemeral port numbers.
- The client process can send messages to the outgoing queue by using the source port number specified in the request.
- After adding the UDP header, UDP removes the messages one by one and delivers them to IP.
- If there is an overflow in an outgoing queue then the operating system can ask the client process to wait before sending any more messages.
- When a message arrives for a client, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram.

- If there is such a queue, UDP sends the received user datagram to the end of the queue.
- If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a **Port unreachable** message to the server.
- All the incoming messages for one particular client program, whether coming from the same or a different server, are sent to the same queue.
- If there is an overflow in an incoming queue then UDP drops the user datagram and asks for a port unreachable message to be sent to the server.

Queues at Server site

- At the server site a server asks for incoming and outgoing queues using its well-known port, when it starts running. The queues remain open as long as the server isrunning.
- When a message arrives for a server, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram.
- If there is such a queue, UDP sends the received user datagram to the end of the queue.
- If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the client.
- All the incoming messages for one particular server, whether coming from the same or a different client, are sent to the same queue.
- If there is an overflow in an incoming queue then UDP drops the user datagram and asks for a port unreachable message to be sent to the client.
- When a server wants to respond to a client, it sends messages to the outgoing queue, using the source port number specified in the request.
- After adding the UDP header, it removes the messages one by one and delivers them to IP.
- If there is an overflow in an outgoing queue then the operating system asks the server to wait before sending any more messages.

Uses of UDP

The following lists some uses of the UDP protocol:

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
- UDP is suitable for a process with internal flow and error control mechanisms.
- UDP is a suitable transport protocol for multicasting.
- UDP is used for management processes such as SNMP.
- UDP is used for Route Updating Protocols such as Routing Information Protocol(RIP).

TRANSMISSION CONTROL PROTOCOL (TCP)

TCP is called a connection-oriented, reliable, process-to-process transport protocol. It adds connection-oriented and reliability features to the services of IP.

TCP Services

The following five services offered by TCP to the processes at the application layer

1. Process-to-Process Communication
2. Full-Duplex Communication
3. Connection-Oriented Service

4. Reliable Service
5. Stream Delivery Service

Process-to-Process Communication

TCP provides process-to-process communication using Well-known port numbers.

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, connection	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Service
67	BOOTP	Bootstrap protocol
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Full-Duplex Communication

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

Connection-Oriented Service

- When a process at site A wants to send and receive data from another process at site B, the following steps will occur:
  1. The two TCPs establish a virtual connection between them.
  2. Data are exchanged in both directions.
  3. The connection is terminated.
- The TCP segment is encapsulated in an IP datagram and can be sent out of order or lost or corrupted must be resent.
- TCP creates a stream-oriented environment in which it accepts the responsibility of delivering the bytes in order to the other site.

Reliable Service

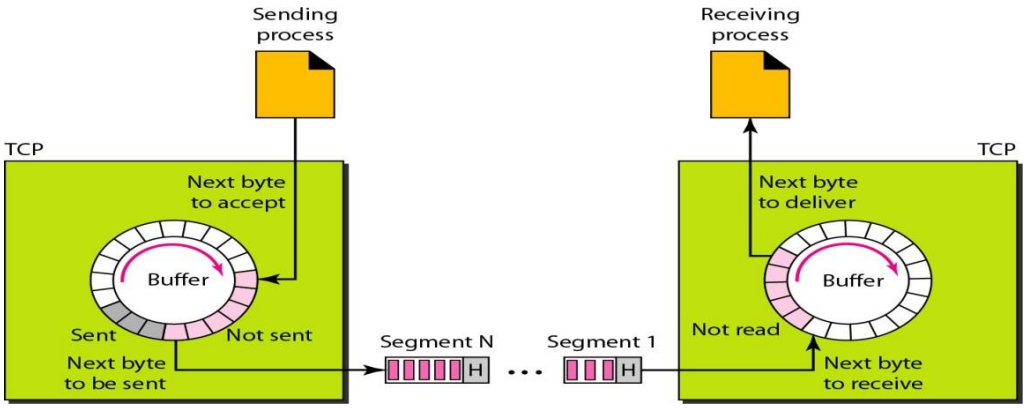
TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

Stream Delivery Service

- TCP allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.
- The sending process produces the stream of bytes, and the receiving process consumes them.

Sending and Receiving Buffers

- TCP needs buffers for storage, because the sending and the receiving processes may not write or read data at the same speed.
- The buffers are implemented by using Circular Array where each location carries1-byte.
- There are two types of buffers are implemented: **Sending buffer** and **Receiving buffer**.



- At the sending site TCP keeps bytes in the buffer that have been sent but not yet acknowledged until it receives an acknowledgment.
- After the bytes in the buffer locations are acknowledged, the locations are recycled and they are available for use by the sending process.
- At the receiver site the circular buffer is divided into two areas:
- One area contains empty chambers to be filled by bytes received from the network.
- Other are contain received bytes that can be read by the receiving process.
- When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

Segments

- TCP groups a number of bytes together into a packet called a segment.
- TCP adds a header to each segment and delivers them to the IP layer for transmission.
- The segments are encapsulated in IP datagrams and transmitted.
- The segments are not necessarily the same size.

TCP Features

TCP has following features:

1. Numbering system
2. Flow control
3. Error control
4. Congestion control

Numbering System

TCP uses 3-types of numbering: Byte, Sequence and Acknowledgement numbering.

Byte Number

- All data bytes that are transmitted in a connection will be given a number called Byte number.
- When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them. Byte numbering is independent in each direction.
- The numbering does not necessarily start from0.
- The first byte number will be any a random number between **0** and $\mathbf{2^{32}-1.}$

Example: If the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056.

Sequence Number

- After the bytes have been numbered, TCP assigns a sequence number to each **Segment** that is being sent.
- The Sequence number for each segment is the **First Byte Number** of carried in that segment.

Example: Suppose a TCP connection is transferring a file of 5000 bytes. Each segment carries 1000 bytes. The first byte number in the first segment is numbered as 10,001. The list of sequence numbers for each segment is:

Segment 1 → Sequence Number: 10,001 (range: 10,001 to 11,000)
Segment 2 → Sequence Number: 11,001 (range: 11,001 to 12,000)
Segment 3 → Sequence Number: 12,001 (range: 12,001 to 13,000)
Segment 4 → Sequence Number: 13,001 (range: 13,001 to 14,000)
Segment 5 → Sequence Number: 14,001 (range: 14,001 to 15,000)

Acknowledgment Number

- The acknowledgment number defines the number of the next byte that the party expects to receive.
- An acknowledgment number used to confirm the bytes that have received.
- The acknowledgment number is cumulative, that the party takes the number of the last byte that it has received is safe and sound and adds 1 to the number and announces this sum as the acknowledgment number.

Example: If a party uses 5643 as an acknowledgment number, it has received all bytes from the beginning number up to 5642.

Flow Control

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

Error Control

- TCP implements an error control mechanism to provide reliable service.
- Error control is byte-oriented. Error control considers a segment as the unit of data for error detection.

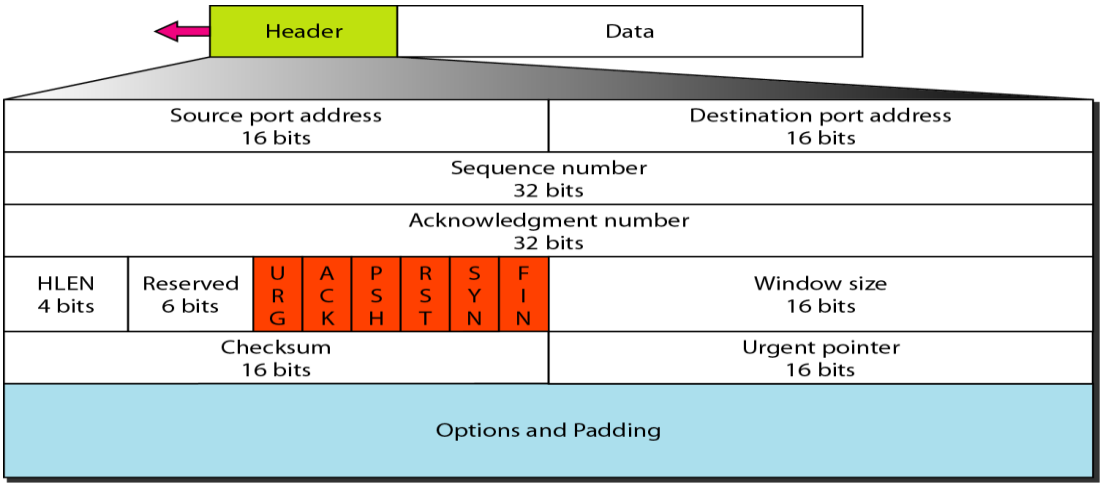
Congestion Control

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

TCP Segment

- A packet in TCP is called a segment.
- A segment consists of Segment Header and Data.
- The segment Header consists of 20-60 Byte. Header is 20 bytes if there are no options. If there are any options the length of the header varied upto 60bytes.

The segment format can be given as:



- **Source port address (16bit)**
  It defines the port number of the application program in the host that is sending the segment.
- **Destination port address(16-bit)**
  It defines the port number of the application program in the host that is receiving the segment.
- **Sequence number (32bit)**
  This field defines the number assigned to the first byte of data contained in this segment. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment.
- **Acknowledgment number (32 bit)**
  This field defines the byte number that the receiver of the segment is expecting to receive from the other party. Acknowledgment and data can be piggybacked together.
- **Header length (4bit)**
  This field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes.
  The value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).
- **Reserved (6 bit)**
  This field is reserved for future use.
- **Window size (16bit)**
  This field defines the size of the window in bytes that the other party must maintain. The maximum size of the window is 65,535 bytes.
  This value is normally referred to as the receiving window (rwnd) and is determined by the receiver.
- **Checksum (16bit)**
  The checksum field in TCP is mandatory. For the TCP pseudo header, the value for the protocol field is 6.
- **Urgent pointer(l6-bit)**
  This field is valid only if the urgent flag is set. It is used when the segment contains urgent data.
  It defines the number that must be added to the sequence number to obtain the number

of the last urgent byte in the data section of the segment.

- **Options (up to 40 bytes)** It defines the optional information in the TCP header.
- **Control flags (6bit)**

  This field defines 6 different control bits or flags. One or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.

Flag	Description
URG	The value of the urgent pointer field is valid
ACK	The value of the acknowledgement field is valid.
PSH	Push the data
RST	Reset the connection
SYN	Synchronize sequence numbers during connection
FIN	Terminate the connection

TCP Connection

- TCP is connection-oriented transport protocol establishes a virtual path between the source and destination.
- All the segments belonging to a message are then sent over this virtual path.
- TCP operates at a higher level.TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself.
- If a segment is lost or corrupted then the segment is retransmitted.
- If a segment arrives out of order then TCP holds it until the missing segments arrives. IP is unaware of this reordering.

Connection-oriented TCP transmission requires three phases:

1. Connection establishment
2. Data transfer
3. Connection termination

Connection Establishment Phase

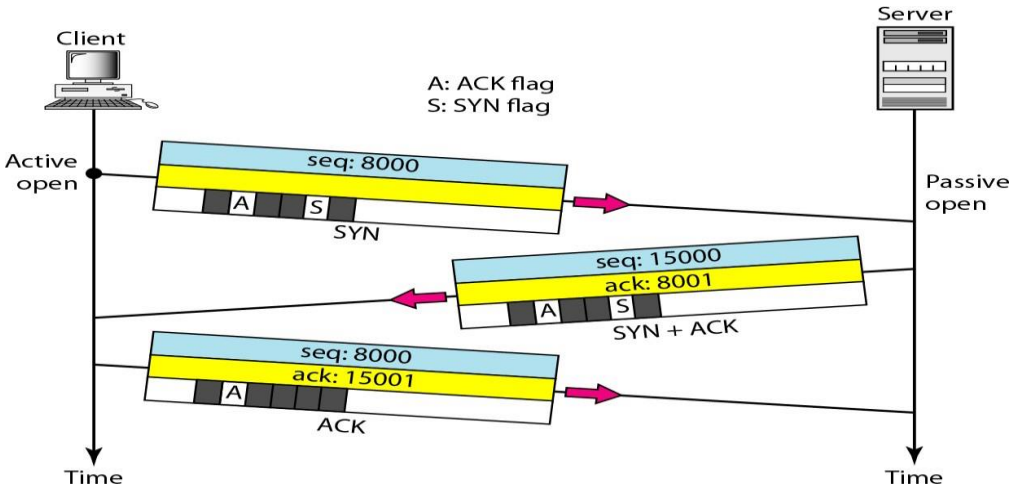
TCP uses Three way handshaking to establish a connection. The three way handshaking uses the sequence number, the acknowledgment number, the control flags and the window size.

- The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a **Passive open**.
- A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. The client program issues a request for an **Active open.**
- Now TCP starts the Three way handshaking protocol process.

The three steps in this phase are as follows:

1. The client sends the first segment a SYN segment. The SYN flag in control field is set. This segment is for synchronization of sequence numbers. The SYN segment does not carry real data. SYN consumes one sequence number. When the data transfer starts the sequence number is incremented by1.
2. The server sends the second segment a SYN + ACK segment with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

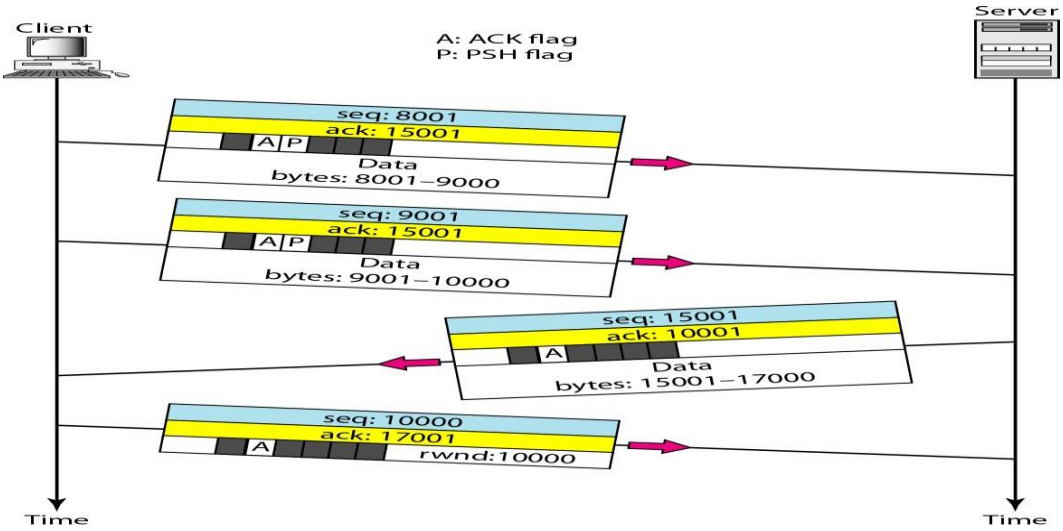
3. The client sends the third segment an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. The sequence number in this segment is the same as the one in the SYN segment.



Data Transfer Phase

- After connection is established, bidirectional data transfer can take place.
- The client and server can both send data and acknowledgments. The acknowledgment is piggybacked with the data.

Consider the below figure that shows the data transfer after connection is established.

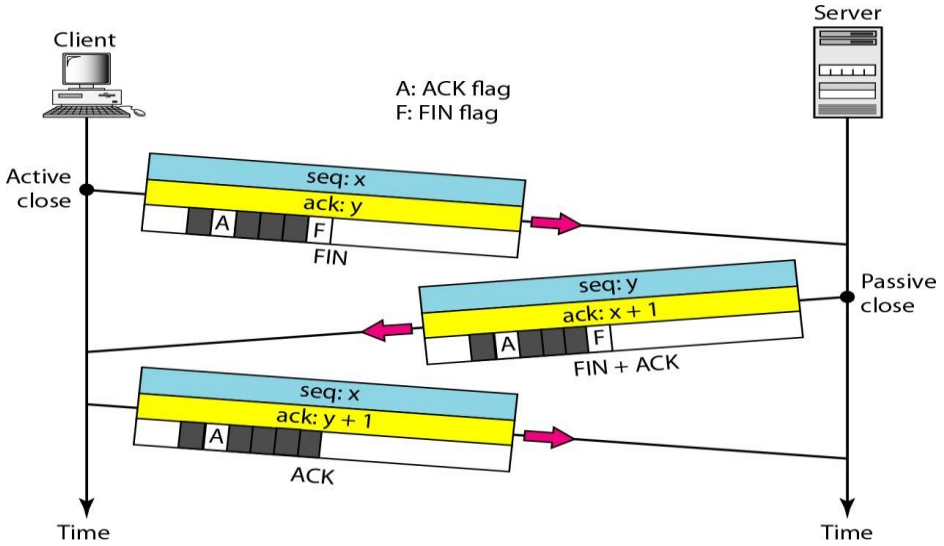


- The client sends 2000 bytes of data in two segments.
- The server then sends 2000 bytes in one segment.
- The client sends one more segment.
- The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.
- The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.
- The segment from the server does not set the push flag.

Connection Termination

Client and Server involved in exchanging data can close the connection is called Connection Termination. It is usually initiated by the client.

Three way Handshaking is also used to terminate the connection by following steps:



1. The client TCP after receiving a close command from the client process sends the first segment a **FIN** segment in which the FIN flag is set.

   A FIN segment can include the last chunk of data sent by the client or it can be just a control segment. If it is only a control segment, it consumes only one sequence number.
2. The server TCP after receiving the FIN segment informs server process of the situation and sends the second segment a **FIN + ACK** segment to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction.

   This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.
3. The TCP client sends the last segment an ACK segment to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is **(sequence number + 1)** received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

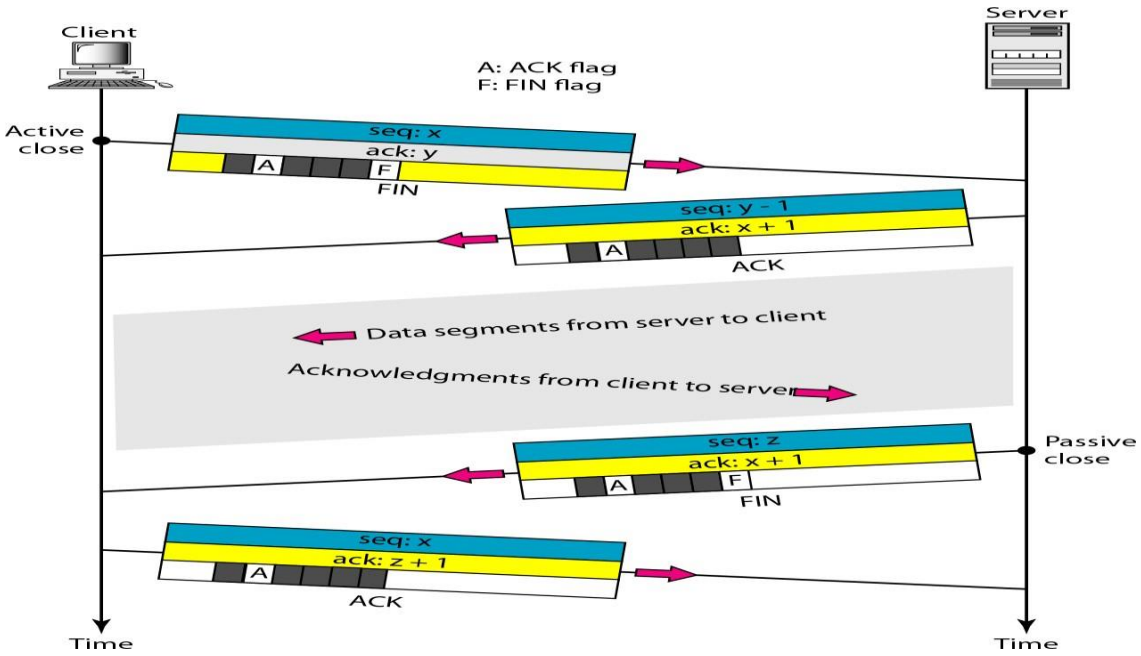
Note: In some rare situations Connection Termination can be done in Four way handshaking by using Half-close option.

Four way Handshaking using Half-Close

- In TCP one end can stop sending data while still receiving data is called a **Half-Close**. It is normally initiated by the client.
- It can occur when the server needs all the data before processing can begin.

Example: Sorting.

- When the client sends data to the server to be sorted, the server needs to receive all the data before sorting can start.
- This means the client after sending all the data can close the connection in the outbound direction.
- The inbound direction must remain open to receive the sorted data.
- The server after receiving the data still needs time for sorting. Its outbound direction must remain open.



- The client half-closes the connection by sending a FIN segment. The server accepts the half-close by sending the ACK segment.
- The data transfer from the client to the server stops. The server can still send data.
- When the server has sent all the processed data, it sends a FIN segment which is acknowledged by an ACK from the client.
- After half-closing of the connection, the data can travel from the server to the client and acknowledgments can travel from the client to the server but the client cannot send any more data segments to the server.
- The second segment (ACK) consumes no sequence number. Although client has received sequence number *y* - 1 and is expecting *y*, the server sequence number is still *y* -1.
- When the connection finally closes, the sequence number of the last ACK segment is still *x*, because no sequence numbers are consumed during data transfer in that direction.

Rare Scenarios of TCP communication

There are some rare scenarios happened at the time of communicating through TCP, they are:

- Simultaneous Open
- SYN Flooding Attack
- Pushing Data
- Urgent Data

Simultaneous Open

- It is a rare situation that may occur when both processes issue an active open.
- Both TCPs transmit a SYN + ACK segment to each other and only one single connection is established between them.

SYN Flooding Attack or Denial-of-Service Attack

This will occur at the time connection establishment phase.

- SYN Flooding Attack happens when a malicious attacker sends a large number of SYN segments to a server pretending that each of them is coming from a different client by faking the source IP addresses in the datagrams.

- The server assuming that the clients are issuing an active open request and it allocates the necessary resources such as creating communication tables and setting timers.
- The TCP server then sends the **SYN +ACK** segments to the fake clients, which are lost.
- During this time a lot of resources are occupied without being used.
- If during this short time the number of SYN segments is large, the server eventually runs out of resources and may crash.
- This SYN flooding attack belongs to security attack known as a **Denial-of-Service** attack, in which an attacker monopolizes a system by sending so many service requests that the system collapses and denies service to every request.

Pushing Data

Pushing Data concept is implemented where Delayed transmission and delayed delivery of data may not be acceptable by the application program that communicates interactively with another application program on the other end.

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

Urgent Data

There are some rare occasions an application program needs to send **Urgent bytes**. This means that the sending application program wants a piece of data to be read out of order by the receiving application program.

- The Urgent data can send a segment with the URG bit set.
- The sending application program tells the sending TCP that the piece of data is urgent.
- The sending TCP creates a segment and inserts the urgent data at the beginning of the segment.
- The rest of the segment can contain normal data from the buffer.
- The urgent pointer field in the header defines the end of the urgent data and the start of normal data.
- When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment using the value of the urgent pointer and delivers them to the receiving application program in out of order.
- The receiving TCP does not discard these urgent data packets even though they are received out of order.

CONGESTION

Congestion in a network may occur if the **load** on the network is greater than the *capacity* of the network.

Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

Note:

1. **Load** refers to the number of packets sent to the network.

2. **Capacity** refers to the number of packets a network can handle.

Congestion control involves two factors that measure the performance of a network: Delay and Throughput.

Delay versus Load

- The load is much less than the capacity of the network the delay is at a minimum. This minimum delay is composed of propagation delay and processing delay.
- When the load reaches the network capacity, the delay increases sharply due to additional waiting time in the queues will be added to the total delay.
- The delay becomes infinite when the load is greater than the capacity.
- When a packet is delayed, the source that is not receiving the acknowledgment will retransmit the packet. This makes the delay and the congestion even worse.

Throughput versus Load

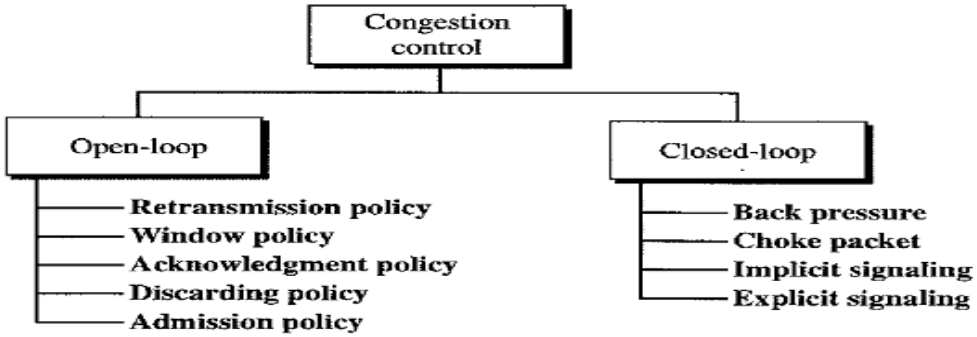
- Throughput in a network is defined as the number of packets passing through the network in a unit of time.
- The load is below the capacity of the network, the throughput increases proportionally with the load.
- When the load exceeds the capacity, the queues become full and the routers have to discard some packets.
- Discarding packets does not reduce the number of packets in the network because the sources retransmit the packets using time-out mechanisms.

CONGESTION CONTROL

Congestion control refers to techniques and mechanisms that can either prevent congestion before it happens or remove congestion after it has happened.

Congestion control mechanisms can be divided into two categories:

1. Open-loop congestion control(prevention)
2. Closed-loop congestion control(removal)



Open-Loop Congestion Control

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

Retransmission Policy

- The packet needs to be retransmitted by sender, when a packet is lost or corrupted.
- Retransmission is sometimes unavoidable. It may increase congestion in the network.
- The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion.

Example: Retransmission policy used by TCP is designed to prevent or alleviate congestion.

Window Policy

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

Acknowledgment Policy

- The acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing less load on the network.
- If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.
- A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires.

Discarding Policy

- A good discarding policy by routers may prevent congestion.
- Example: In audio transmission if the policy is to discard less sensitive packets when congestion happens, the quality of sound is still preserved and congestion is prevented.

Admission Policy

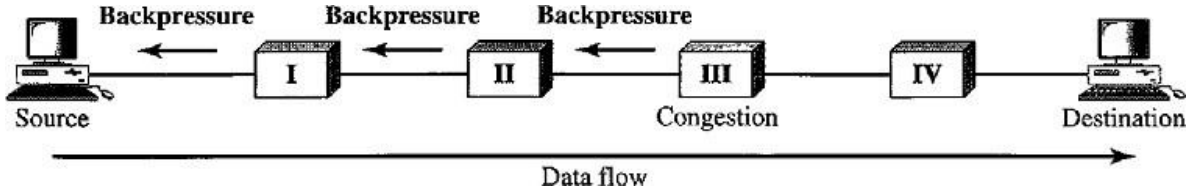
- An admission policy can prevent congestion in virtual-circuit networks.
- Switches first check the resource requirement of a data flow before admitting it to the network.
- A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility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

Closed-Loop Congestion Control

Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols are: Back pressure, Choke packet, Implicit signaling, Explicit signaling.

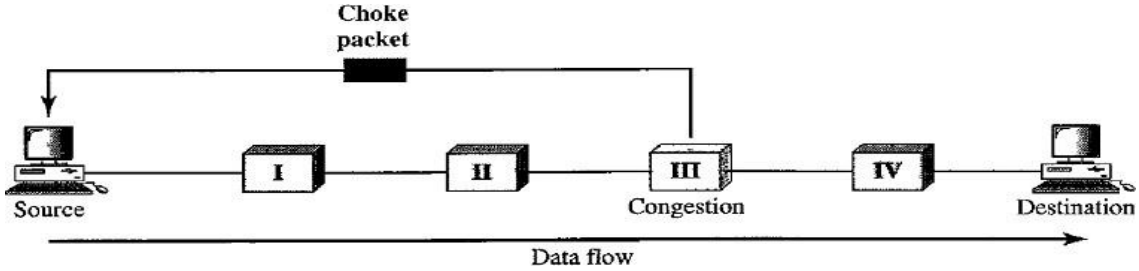
Backpressure

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



Choke Packet

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



Implicit Signaling

- In implicit signaling, there is no communication between the congested nodes and the source.
- Source guesses that there is congestion somewhere in the network from other symptoms.

Example: when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested. The delay in receiving an acknowledgment is interpreted as congestion in the network and the source should slow down sending speed.

Explicit Signaling

- The node that experiences congestion can explicitly send a signal to the source or destination.
- In explicit signaling method, the signal is included in the packets that carry data.
- Explicit signaling can occur in either the forward or the backward direction.

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

QUALITY OF SERVICE (QoS)

It is an internetworking issue. There are four Flow characteristics are related to Qos.
They are: Reliability, Delay, Jitter and Bandwidth.

Reliability: Lack of reliability means that losing a packet or acknowledgment, which entails retransmission. Electronic mail, file transfer, and Internet access have reliable transmissions.

Delay: Applications can tolerate delay in different degrees. Telephony, audio conferencing, video conferencing, and remote log-in need minimum delay.

Jitter: It is the variation in delay for packets belonging to the same flow. If four packets depart at times 0, 1, 2,3 and arrive at 20, 21, 22, 23 all have the same delay = 20 units of time. Audio and video applications accept the delay of packets as long as if the delay is same for all the packets.

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

TECHNIQUES TO IMPROVE QoS

There are four techniques that will improve the QoS:

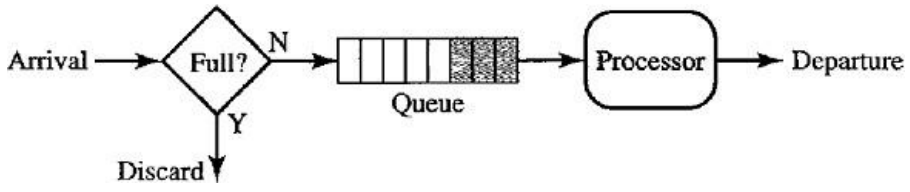
1. Scheduling
2. Traffic shaping
3. Admission control
4. Resource reservation

Scheduling

Packets from different flows arrive at a switch or router for processing.
Several scheduling techniques are designed to improve the quality of service such as: FIFO Queuing, Priority Queuing and Weighted fair queuing.

First-In-First-Out Queuing (FIFO)

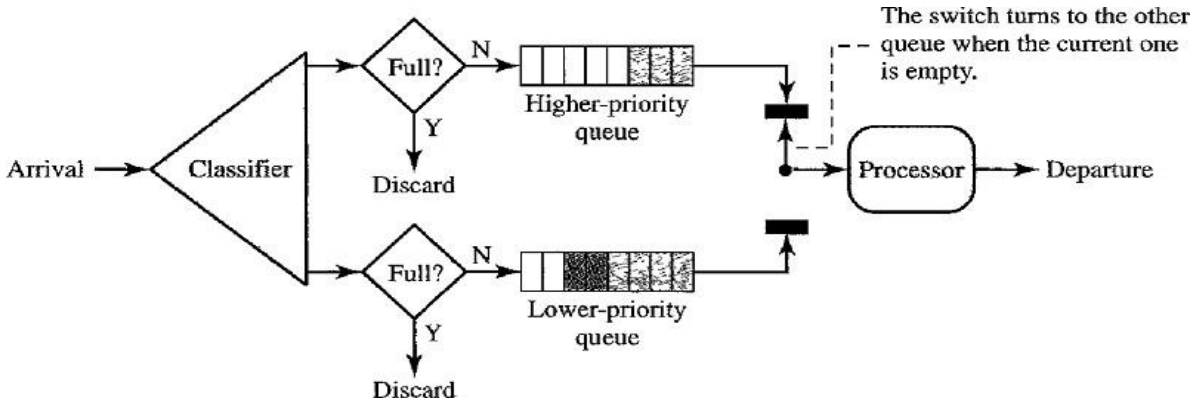
- In FIFO queuing, the packets wait in a buffer (queue) until the node (router or switch) is ready to process them.
- If the average arrival rate is higher than the average processing rate then the queue will fill up and new packets will be discarded.
- That means the speed of receiving the packets in to buffer is more than the speed of processing the packets by processor then the buffer will be filled completely and then there is no place for newly arrived packets hence these packets will be discarded.



Priority Queuing

- In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue.
- The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last.
- The system does not stop serving a queue until it is empty.

- There is drawback with priority queues called **Starvation** (i.e.) if there is a continuous data flow in a high-priority queue, the packets in the low-priority queues will never have a chance to be processed.

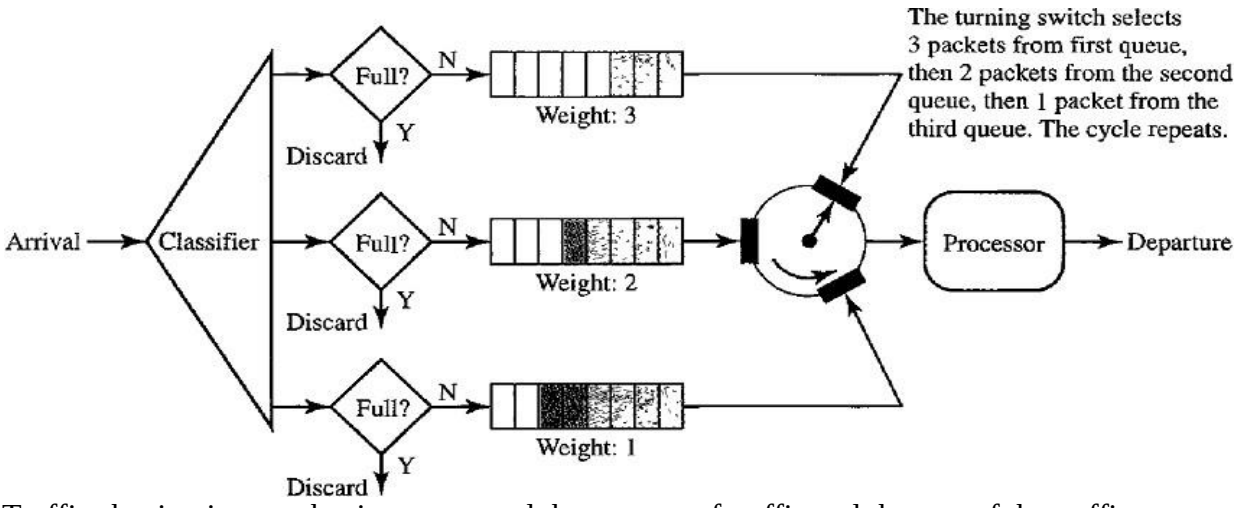


Weighted Fair Queuing

- In this technique, the packets are still assigned to different classes and admitted to different queues.
- The queues are weighted based on the priority of the queues; higher priority means a higher weight.
- The system processes packets in each queue in a **Round-Robin** fashion with the number of packets selected from each queue based on the corresponding weight.

Example: If the weights are 3, 2, and 1, three packets are processed from the first queue, two from the second queue, and one from the third queue. If the system does not impose priority on the classes, all weights can be equal. In this way, we will achieve fair queuing with priority.

Traffic Shaping

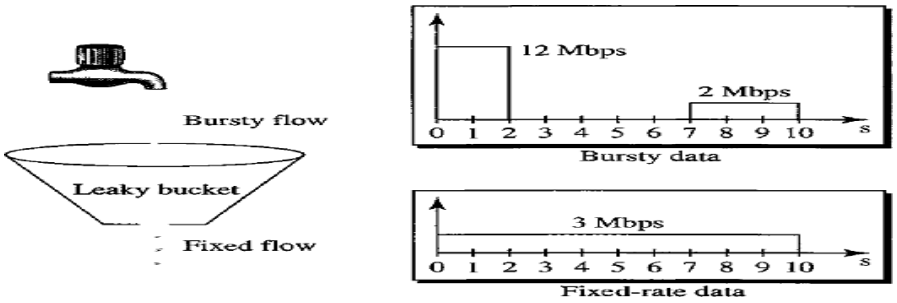


Traffic shaping is a mechanism to control the amount of traffic and the rate of the traffic sent to the network. Two techniques can shape traffic: **Leaky bucket** and **Token bucket.**

Leaky Bucket

- If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant.

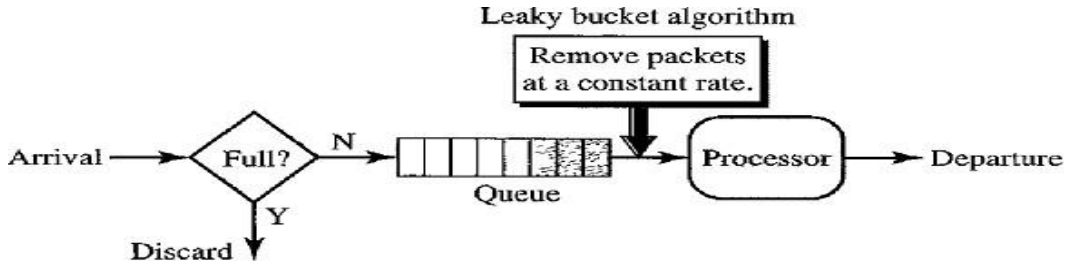
- In networking, a technique called leaky bucket can smooth out Bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.



- In the above figure, the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment.
- The host sends a burst of data at a rate of 12 Mbps for 2 sec, for a total of 24 M bits of data.
- The host is silent for 5 sec and then sends data at a rate of 2 Mbps for 3 sec, for a total of 6 M bits of data.
- In total the host has sent 30 M bits of data inl0s.
- The leaky bucket smoothen the traffic by sending out data at a rate of 3 Mbps during the same 10sec.
- Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host.

This way the leaky bucket may prevent congestion.

Consider the below figure that shows implementation of Leaky Bucket:



- A FIFO queue holds the packets. If the traffic consists of fixed-size packets the process removes a fixed number of packets from the queue at each tick of the clock.
- If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes orbits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to *n* at the tick of the clock
2. If *n* is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until *n* is smaller than the packet size.
3. Reset the counter and go to step1.

Problems with Leaky Bucket

1. The leaky bucket is very restrictive. If a host is not sending for a while, its bucket becomes empty.
2. After some time if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account.

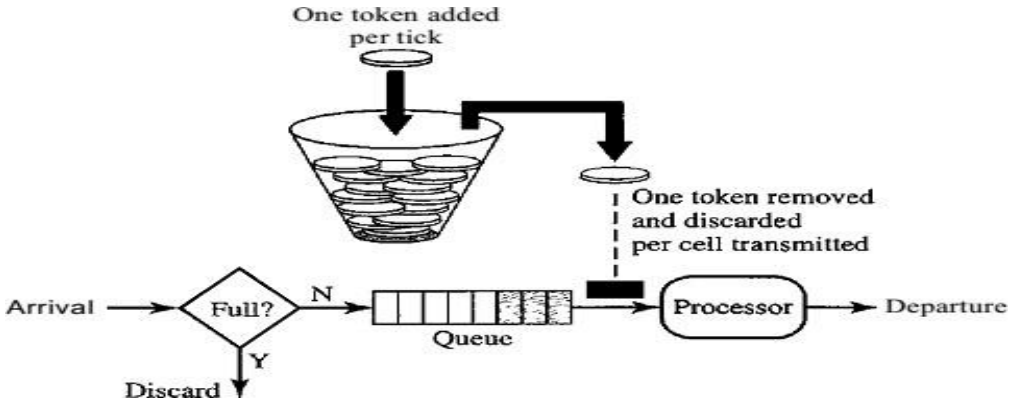
These problems can be overcome by Token bucket algorithm.

Token Bucket

- The token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens.
- For each tick of the clock, the system sends *n* tokens to the bucket.
- The system removes one token for every cell (or byte) of data sent.

Example: If *n* is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens.

- Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick.
- The host can send bursty data as long as the bucket is not empty.



The token bucket can easily be implemented with a counter.

- The token is initialized to zero.
- Each time a token is added, the counter is incremented by1.
- Each time a unit of data is sent, the counter is decremented by1.
- When the counter is zero, the host cannot send data.

Resource Reservation

- A flow of data needs resources such as a buffer, bandwidth, CPU time, and soon.
- The quality of service is improved if these resources are reserved before data transfer.

Admission Control

- Admission control refers to the mechanism used by a router or a switch to accept or reject a flow based on predefined parameters called flow specifications.
- Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity and its previous commitments to other flows can handle the new flow.

Note: Capacity is in terms of bandwidth, buffer size, CPU speed, etc.

APPLICATION LAYER

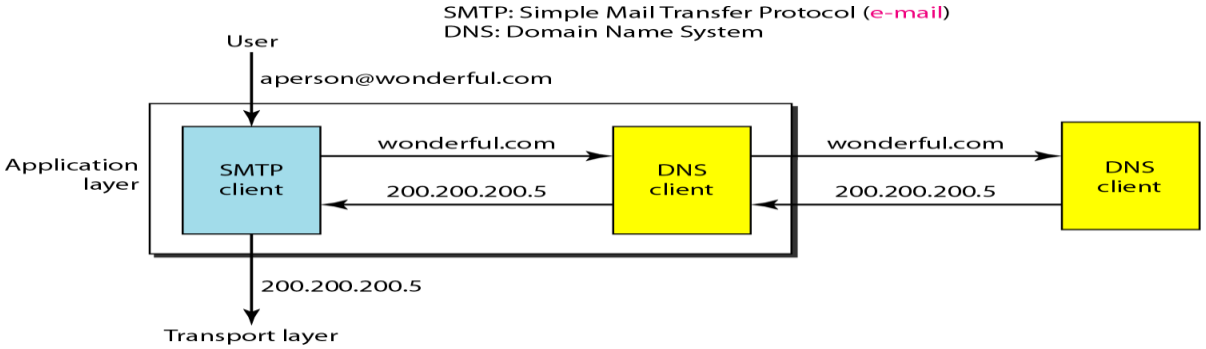
DOMAIN NAME SYSTEM

The client/server programs can be divided into two categories:

1. Programs that can be directly used by user.
2. Programs that support other application programs.

Domain Name System (DNS) is a supporting program that is used by other programs such as E-mail.

Basic concept of DNS



The above figure shows a DNS client/server program can support an E-mail program to find the IP address of an E-mail recipient.

- A user of an E-mail program knows the E-mail address of the recipient but the IP protocol needs the IP address.
- The DNS client program sends a request to a DNS server to map the E-mail address to the corresponding IP address.
- To identify an entity TCP/IP protocols uses the IP address, which uniquely identifies the connection of a host to the Internet.
- People prefer to use names instead of numeric addresses. Hence we need a system that can map a name to an address or an address to a name. DNS is designed for this purpose.

NAMESPACE

The names assigned to machines must be unique because the addresses are unique.
A name space that maps each address to a unique name can be organized in two ways:

- Flat Name Space
- Hierarchical Name Space

Flat Name Space

- In a flat name space, a name is assigned to an address.
- A name in this space is a sequence of characters without structure.

Hierarchical Name Space

- In Hierarchical name space, each name is made of several parts.
- The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization etc.

Example:

Assume three Education institutions named one of their computers **Challenger**. The three colleges have given names by the central authority such as iitm.ac.in, berkeley.edu and smart.edu.

When these organizations add the name **Challenger** the names will be :

- challenger.iitm.ac.in
- challenger.berkeley.edu
- challenger.smart.edu

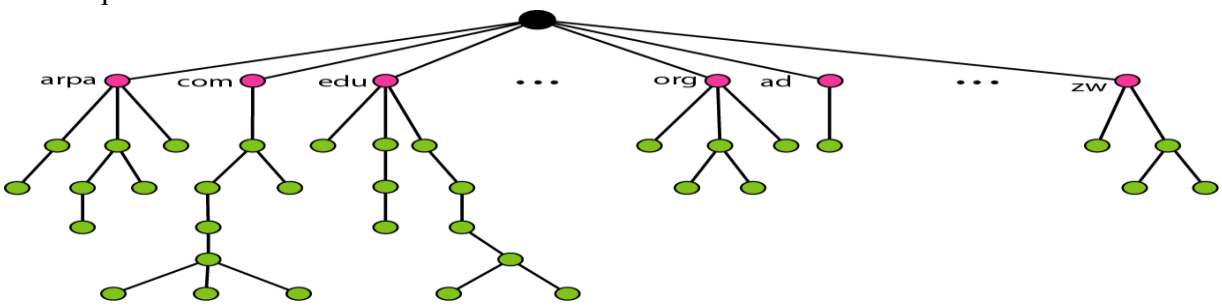
DOMAIN NAME SPACE

A domain name space was designed to have a Hierarchical Name Space.

In this design the names are defined in a tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127.

Label

- Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string).
- DNS requires that children of a node have different labels, which guarantees the uniqueness of the domain names.

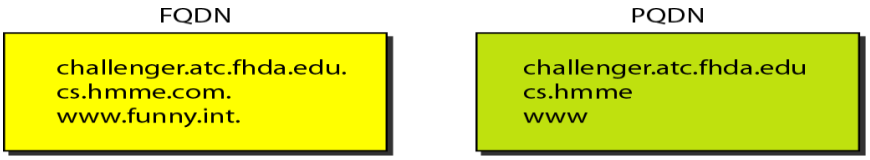


Domain Name

- Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots(.).
- Domain names are always read from the bottom to top.
- The last label is the label of the root (null). (i.e.) a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing.

Fully Qualified Domain Name (FQDN)

- If a label is terminated by a null string, it is called a fully qualified domain name(FQDN).
- Example:**challenger.atc.fhda.edu.**

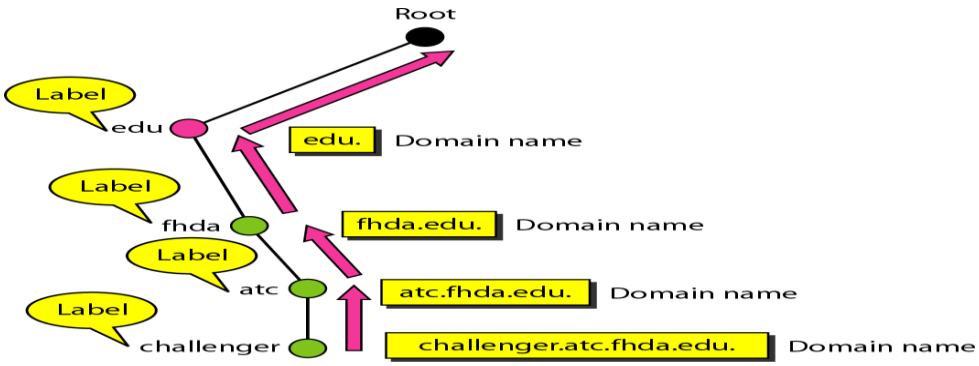


Partially Qualified Domain Name (PQDN)

- If a label is not terminated by a null string, it is called a PQDN.
- Example:**challenger**

If a user at the **"fhda.edu."** site wants to get the IP address of the challenger

computer, he or she can define the partial name "**challenger**".
The DNS client adds the suffix **"atc.fhda.edu."** before passing the address to the DNS server.



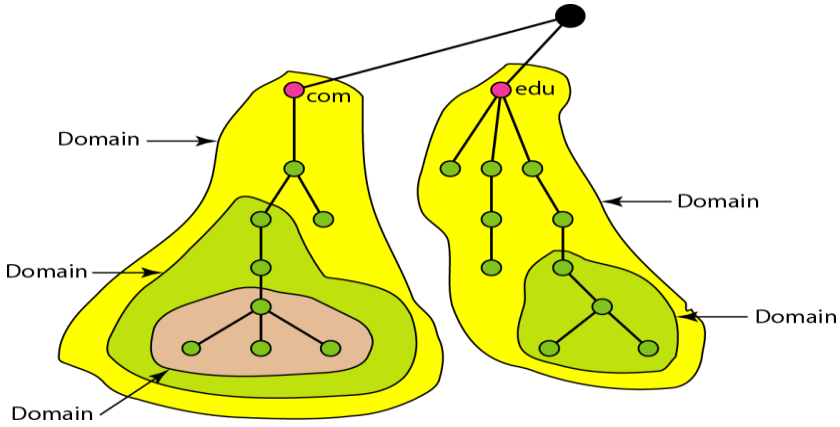
Note: The DNS client normally holds a list of suffixes such as:

- **atc.fhda.edu**
- **fhda.edu**
- **null**

These suffixes were added when the user defines an FQDN.

Domain

A **domain** is a sub-tree of the domain name space. The name of the domain is the domain name of the node at the top of the sub tree. A domain may itself be divided into sub-domains.



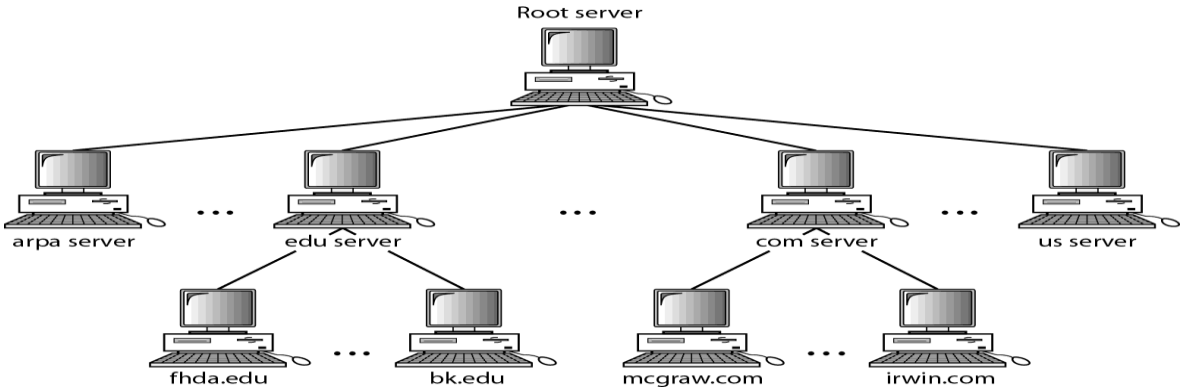
DISTRIBUTION OF NAME SPACE

- The information contained in the domain name space must be stored.
- This information is distributed among different computers and in different places.

Hierarchy of Name Servers

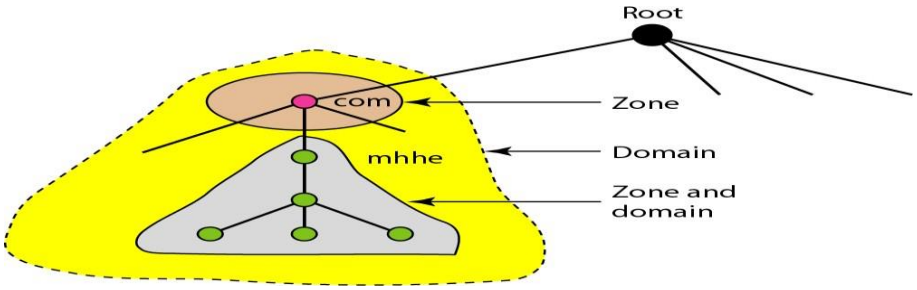
- Distribution of the information among many computers called DNS servers.
- The whole space is divided into many domains based on the first level. The root stand alone and create as many domains(sub trees).
- A domain created in this way could be very large; DNS allows domains to be divided further into smaller domains(sub domains).

- Each server can be responsible (authoritative) for either a large or a small domain.



Zone

- The complete domain name hierarchy cannot be stored on a single server, it is divided among many servers.
- A zone is a contiguous part of the entire tree and it defines what a server is responsible for or server has authority over.



Root Server

- A root server is a server whose zone consists of the whole tree.
- A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers.
- The servers are distributed all around the world.

DNS defines two types of servers: Primary and Secondary servers.

Primary Server

- A primary server is a server that stores a file about the zone for which it is an authority.
- It is responsible for creating, maintaining, and updating the zone file.
- It stores the zone file on a local disk.

Secondary Server

- A secondary server is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk.
- The secondary server neither creates nor updates the zone files.
- If updating is required, it must be done by the primary server, which sends the updated version to the secondary.

Note:

i. These secondary servers are used for crash recovery. These are redundant servers which are created when one server fails the other server can continue serving the clients for their data requests.

ii. A server can be a primary server for a specific zone and a secondary server for another zone.

Zone transfer

A primary server loads all information from the disk file. The secondary server loads all information from the primary server. When the secondary server downloads information from the primary server it is called zone transfer.

DNS IN THE INTERNET

In the Internet the domain name space tree is divided into three different sections:

- Generic Domains,
- Country Domains
- The Inverse Domain

Generic Domains

- The **generic domains** define Registered hosts according to their generic behavior.
- Each node in the tree defines a domain, which is an index to the domain name space database.

Generic domain labels are listed as:

Label	Description
com	Commercial organizations
org	Nonprofit organizations
net	Network support centers
edu	Educational institutions
gov	Government institutions

Country Domains

- The country domains section uses two-character country abbreviations. Ex: in for India, us for USA.
- Second labels can be organizational, or they can be more specific, national designations. Ex: .ac.in for nptel.ac.in, .gov.in for tspsc.gov.in etc.

Inverse Domain

The inverse domain is used to map an Address to a Name. It uses inverse query or pointer query.

RESOLUTION

Mapping a name to an address or an address to a name is called Name-Address Resolution.

Caching

- When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client.
- If the client asks for the same mapping, it can check its cache memory and returns the result.
- To inform the client that the response is coming from the cache memory and not from an authoritative source, the server marks the response as Un authoritative.

REGISTRARS

Registrar adds new domains to DNS. A registrar first verifies that the requested domain

name is unique and then enters it into the DNS database. The registrars and their names, addresses found at: http://www.intenic.net

Example: Domain name: WS.wonderful.com (ws is a server name) IP address: 200.200.200.5 (new IP address).

REMOTE LOGGING

In the Internet, users may want to run application programs at a remote site and create results that can be transferred to their local site.

Example: Students may want to connect to their university computer lab from their home to access application programs for doing homework assignments or projects.

A General purpose client/server program that allows a user to log-on to a remote computer to access any application program on that remote computer.

After logging on, a user can use the available services on the remote computer and transfer the results back to the local computer.

TELNET (TErminaL NETwork)

TELNET is a client/server application program. It is the standard TCP/IP protocol for virtual terminal service as proposed by the International Organization for Standards (ISO).

TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.

Timesharing Environment

TELNET works in Time Sharing Environment. The interaction between a user and the computer occurs through a terminal.

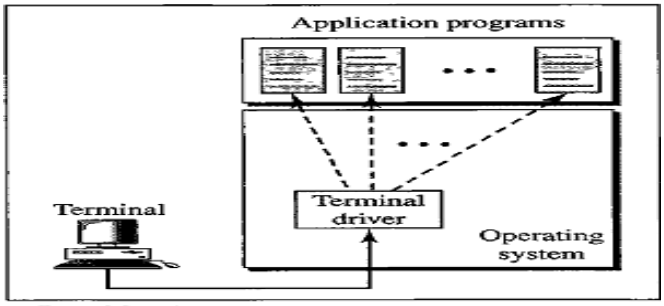
Logging

- In a timesharing environment, users are part of the system with some right to access resources. Each authorized user has Identification (User ID) and a password.
- To access the system the user logs into the system with a user id or log-in name.
- The system also includes password checking to prevent an unauthorized user from accessing the resources.

Local log-in

- When a user logs into a local timesharing system it is called local log-in.
- As a user types at a terminal the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system.

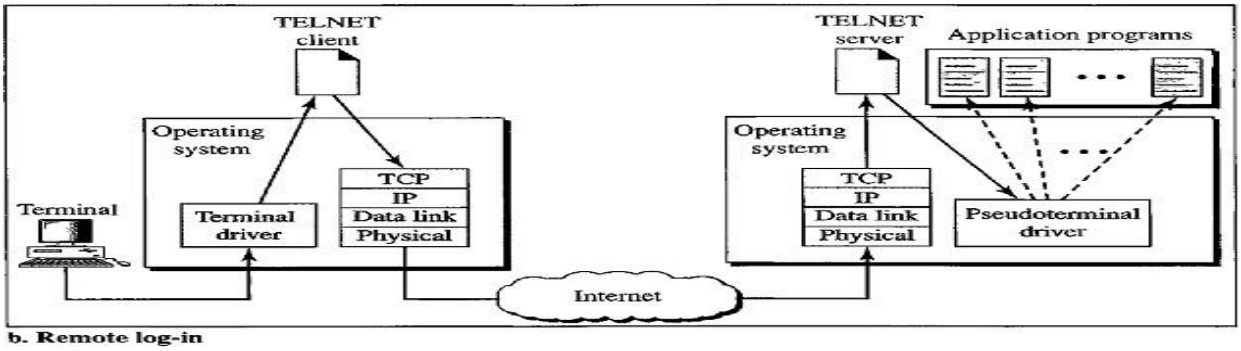
- The operating system interprets the combination of characters and invokes the desired application program or utility.



a. Local log-in

Remote Log-in

- When a user wants to access an application program or utility located on a remote machine, the user performs remote log-in. TELNET uses client and server programs.
- The user sends the keystrokes to the terminal driver, where the local operating system accepts the characters but does not interpret them.
- The characters are sent to the TELNET client, which transforms the characters to a universal character set called Network Virtual Terminal (NVT) characters and delivers them to the local TCP/IP protocol stack.
- The commands or text in NVT form travel through the Internet and arrive at the TCP/IP stack at the remote machine.



b. Remote log-in

- The characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer.
- The characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server. It is designed to receive characters from a terminal driver.
- Hence the characters can be passed to **Pseudo-terminal driver** which passes the characters to operating system.
- The operating system then passes the characters to the appropriate application program.

Network Virtual Terminal (NVT)

- TELNET defines a universal interface called the Network Virtual Terminal character set. Via this interface the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network.
- The server TELNET translates data and commands from NVT form into the form

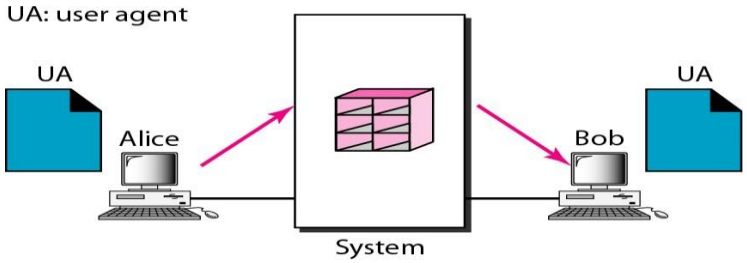
acceptable by the remote computer.

ELECTRONIC MAIL

Electronic mail (E-mail) is one of the most popular Internet services. E-mail allows a message to include text, audio, and video. There are Four Scenarios of E-mail:

First Scenario

- In the first scenario, the sender and the receiver of the E-mail are user application programs on the same system. They are directly connected to a shared system.
- The administrator has created one mailbox for each user where the received messages are stored.
- A mailbox is part of a local hard drive and it a special file with permission restrictions. Only the owner of the mailbox has access to it.

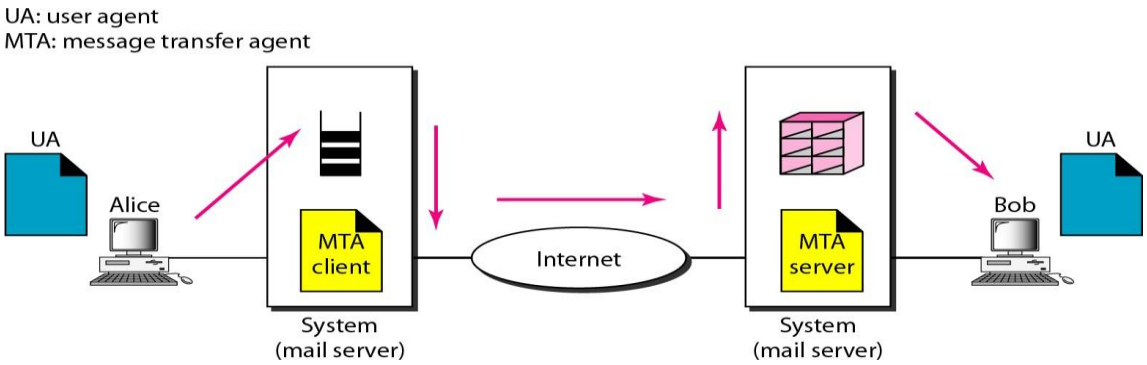


Example: Consider the above figure, Alice and Bob are two users of mail server.

- When a user Alice needs to send a message to Bob, Alice runs a User Agent (UA) program to prepare the message and store it in Bob's mailbox.
- The message has the sender and recipient mailbox addresses (names offiles).
- Bob can retrieve and read the contents of his mailbox using a User Agent.

Second Scenario

- In the second scenario, the sender and the receiver of the E-mail are user application programs on two different systems. The message needs to be sent over the Internet.
- We need User Agents (UAs) and Message Transfer Agents(MTA's).

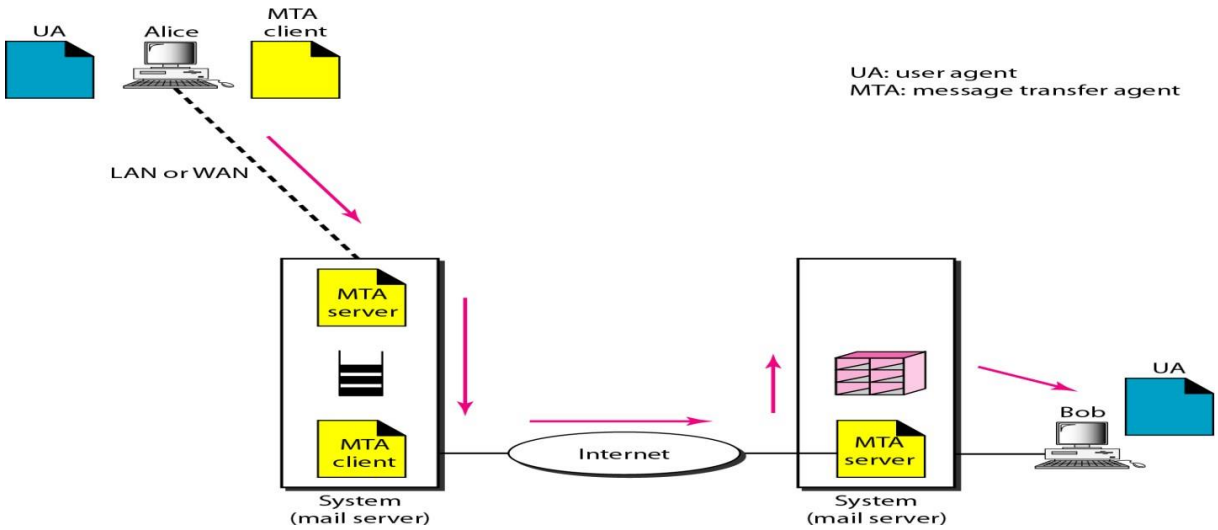


- Alice needs to use a user agent program to send her message to the mail server at her own site.
- The mail server at Alice site uses a queue to store messages waiting to be sent.
- Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site.
- The message needs to be sent through the Internet from Alice's site to Bob's site.

Here two Message Transfer Agents are needed: one for client and one for server.

- Most client/server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection.
- The client can be alerted by the system when there is a message in the queue to be sent.

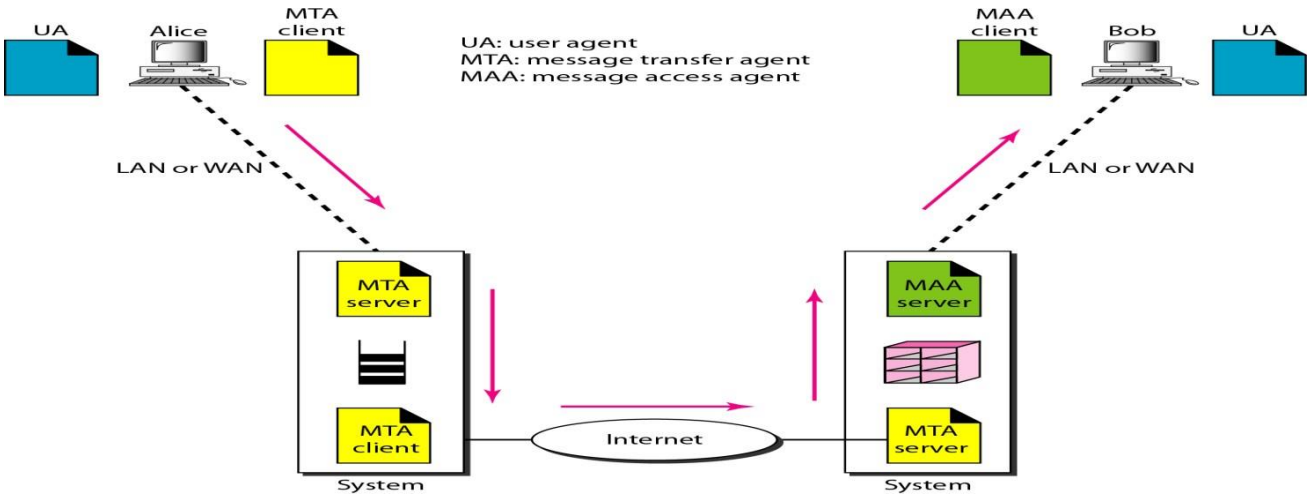
Third Scenario



- In the third scenario, Bob is directly connected to his system (i.e. Mail Server). Alice is separated from her system.
- Alice is connected to the mail server via WAN or LAN.
- In an organization that uses one mail server for handling E-mails, all users need to send their messages to this mail server.
- User agent of Alice prepares message and sends the message through the LAN or WAN.
- Whenever Alice has a message to send, Alice calls the user agent and user agent calls the MTA client.
- The MTA client establishes a connection with the MTA server on the system.
- The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site. The system receives the message and stores it in Bob's mailbox.
- Bob uses his user agent to retrieve the message and reads it.
- It needs two MTA client and two MTA server programs.

Fourth Scenario

- It is the most common scenario, Alice and Bob both are connected to their mail server by a WAN or a LAN.
- After the message has arrived at Bob's mail server, Bob needs to retrieve it. Now Bob needs another set of client/server agents called Message Access Agents (MAA). Bob uses an MAA client to retrieve his messages.
- The client sends a request to the MAA server and requests the transfer of the messages.



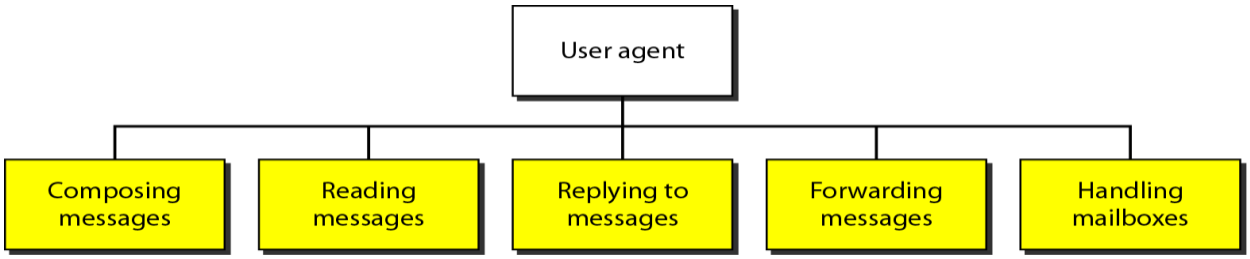
Architecture of E-mail

There are three major components in the architecture of E-mail:

1. User Agent
2. Message Transfer Agent
3. Message Access Agent

User Agent

User Agent provides services to the user to make the process of sending and receiving a message easier. Services provided by User agent are:



Composing Messages

A user agent helps the user to compose the E-mail message to be sent out. Most user agents provide a template on the screen to be filled in by the user.

Reading Messages

The user agent reads the incoming messages. When a user invokes a user agent, it first checks the mail in the incoming mailbox. Each E-mail contains the following fields:

1. A number field.
2. A flag field that shows the status of the mail such as new, already read but not replied to, or read and replied to.
3. The size of the message.
4. The sender.
5. The optional subject field.

Replying to Messages

After reading a message, a user can use the user agent to reply to a message. A user agent allows the user to reply to the original sender or to reply to all recipients of the message.

Forwarding Messages

It means sending a message to a third party. A user agent allows receiver to forward message.

Handling Mailboxes

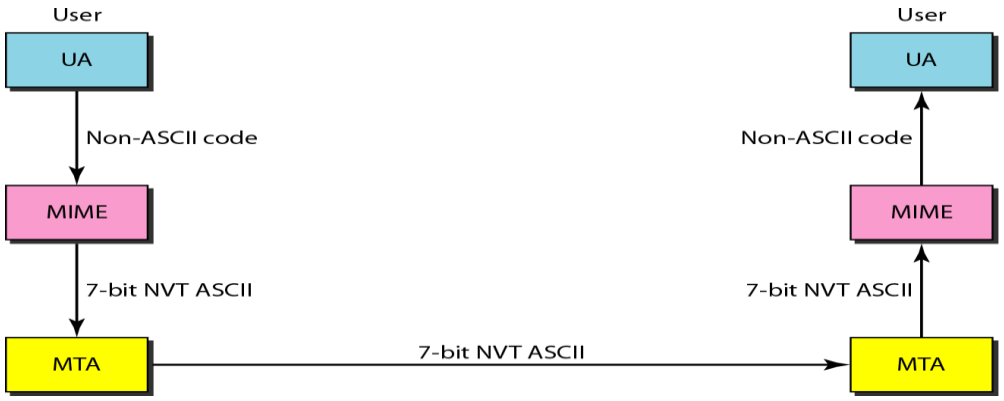
- A user agent normally creates two mailboxes: **Inbox** and **Outbox**.
- Inbox and Outbox is a file with a special format that can be handled by user agent.
- **Inbox** keeps all the received E-mails until they are deleted by the user.
- **Outbox** keeps all the sent E-mails until the user deletes them.

Mailing List

- Electronic mail allows one name (an alias) to represent several different E-mail addresses is called a mailing list.
- Every time a message is to be sent, the system checks the recipient's name against the alias database.

MIME (Multipurpose Internet Mail Extensions)

- MIME is a supplementary protocol that allows non-ASCII data to be sent through E-mail.
- French, German, Hebrew, Russian, Chinese, and Japanese are non-ASCII characters.
- MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client MTA to be sent through the Internet.
- The message at the receiving side is transformed back to the original data.



SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

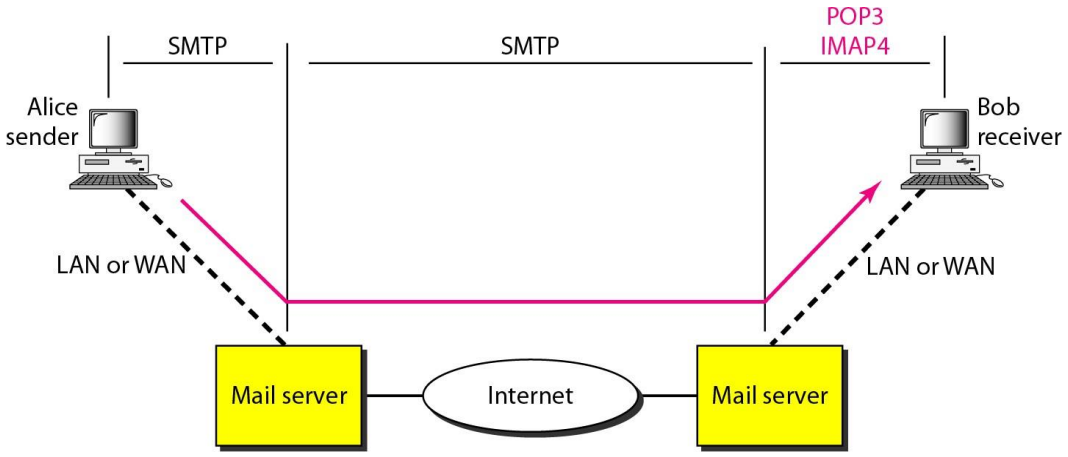
Message Transfer Agent: SMTP

- The actual mail transfer is done through message transfer agents (MTA).
- Simple Mail Transfer Protocol defines the MTA Client and MTA server in the internet.
- MTA Client is used to send mail and MTA Server is used to receive

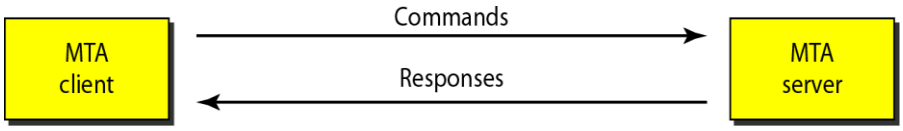
a mail. SMTP is used two times:

1. Between sender and sender mail server.

2. Between sender mail server and receiver mail server.



SMTP uses commands and responses to transfer messages between an MTA client and an MTA server. Each command or reply is terminated by a two-character end-of-line token.



Commands

Commands are sent from the client to the server. It consists of a keyword followed by zero or more arguments.

There are five mandatory commands are used. Every implementation must support these five commands: HELO (Sender's Host name), MAIL FROM, RCPT TO, DATA, QUIT.

Responses

Responses are sent from the server to the client. A response is a three digit code that may be followed by additional textual information.

Responses are divided into four categories: The leftmost digit of the code 2, 3, 4, and 5 defines the category.

- 2-Positive Completion Reply
- 3-Positive Intermediate Reply
- 4-Transient Negative Completion Reply
- 5-Permanent Negative Completion Reply

Mail Transfer Phases

The process of transferring a mail message occurs in three phases:

- Connection Establishment
- Mail Transfer
- Connection Termination

FILE TRANSFER PROTOCOL (FTP)

File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file from one host to another. FTP uses the services of TCP. FTP implemented to solve below problems.

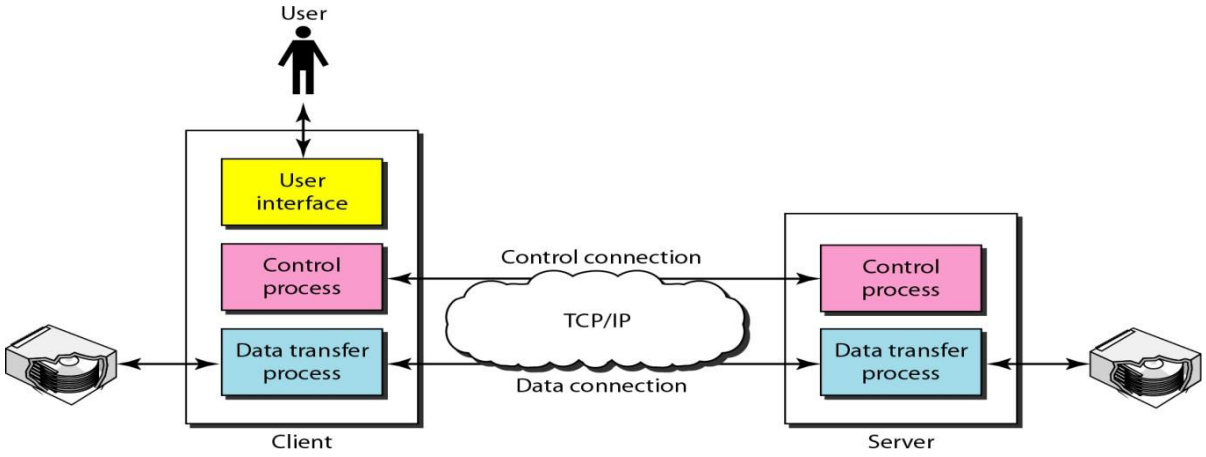
Problems with File transfer

- Two systems may use different file name conventions.
- Two systems may have different ways to represent text and data.
- Two systems may have different directory structures.

FTP differs from other client/server applications in that it establishes two connections between the hosts. FTP uses two well-known ports these connections.

1. Data transfer connection (Port 20 is used)
2. Control connection (Port 21 is used)

Basic Model of FTP



Consider the above figure that shows basic model of FTP that contains client and server components.

- Client has three components: User Interface, Client Control Process, and Client Data Transfer Process.
- Server has two components: Server Control Process and Server Data Transfer Process.
- The control connection is made between the control processes.
- The data connection is made between the data transfer processes.
- The control connection remains connected during the entire interactive FTP session.
- The data connection is opened and then closed for each file transferred.
- When a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.

WORLD WIDE WEB (WWW)

World Wide Web (WWW) is a repository of information linked together from locations all over the world. WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet.

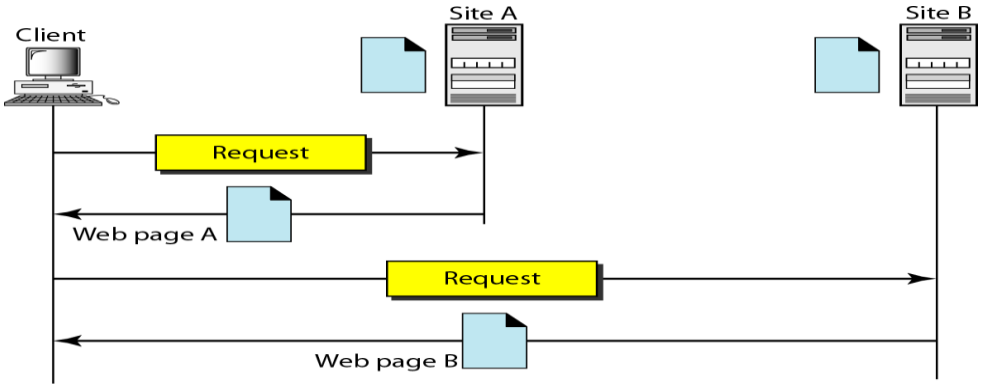
Architecture

The WWW is a distributed client/server service, in which a client using a browser can access a service using a server.

- The service provided is distributed over many locations called site.
- Each site holds one or more documents, referred to as Web pages.
- Each Web page can contain a link to other pages in the same site or at other sites is

called Hyperlink.

- The pages can be retrieved and viewed by using browsers.



Architecture of WWW contains four parts: **1. Client 2. Server 3.URL 4. Cookies Client(Browser)**

A Client is a browser that interprets and displays a Web document.

Each browser consists of three parts: **Controller**, **Client protocol**, and **Interpreters.**

- The controller receives input from the keyboard or the mouse and uses the client programs to access the document.
- After the document has been accessed, the controller uses one of the interpreters to display the document on the screen.
- The interpreter can be HTML, Java, or JavaScript depending on the type of document.
- The client protocol can be FTP or HTTP.

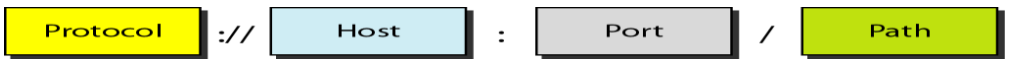
Server

- The Web page is stored at the server. Each time a client request arrives, the corresponding document is sent to the client.
- To improve efficiency, servers normally store requested files in a cache in memory.
- A server uses multi-threading or multi-processing for answering more than one request at a time to increase the efficiency.

Uniform Resource Locator (URL)

The Uniform Resource Locator (URL) is a standard for specifying any kind of information on the Internet. A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators.

URL defines four things: Protocol, Host computer, Port, and Path.



- **Protocol:** It is the client/server program used to retrieve the document. Ex:FTP or HTTP.
- **Host:** The host is the computer on which the information is located. Web pages are usually stored in computers and computers are given **alias names** that usually begin with the characters "www". This is not mandatory.

- **Port:** The URL can optionally contain the port number of the server.
- **Path:** It is the pathname of the file where the information is located.

Note: The path can itself contain slashes that separate the directories from the subdirectories and files.

Cookies

Cookies are used to devise the following functionalities:

- Some websites need to allow access to registered clients only.
- Websites are being used as electronic stores (such as Flipkart or Amazon) that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.
- Some websites are used as portals: the user selects the Web pages he wants to see.
- Some websites are just advertising.

Creation and Storage of Cookies

The creation and storage of cookies depend on the implementation:

1. When a server receives a request from a client, it stores information about the client in a file or a string.

   The information may include the domain name of the client, a timestamp, the contents of the cookie such as client name, client registration number and other information depending on the implementation.
2. The server includes the cookie in the response that it sends to the client.
3. When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the domain server name.

Using Cookies

A cookie is used for following purposes:

1. The site that restricts access to registered clients only sends a cookie to the client when the client registers for the first time. For any repeated access, only those clients that send the appropriate cookie are allowed.
2. An electronic store such as Flipkart or Amazon can use a cookie for its client shoppers. When a client selects an item and inserts it into a cart, a cookie that contains information about the item, such as its number and unit price, is sent to the browser.

   If the client selects a second item the cookie is updated with the new selection information.

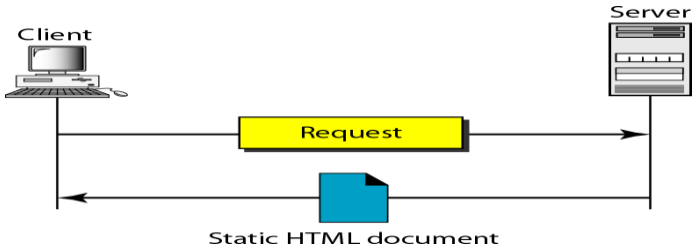
   When the client finishes shopping and wants to check out, the last cookie is retrieved and the total charge is calculated.
3. A Web portal uses the cookie, when a user selects her favorite pages, a cookie is made and sent. If the site is accessed again, the cookie is sent to the server to show what the client is looking for.

WEB DOCUMENTS

Documents in the WWW can be grouped into three categories: **Static, Dynamic** and **Active. Static Documents**

- Static documents are fixed-content documents that are created and stored in a server.

- The client can get only a copy of the document (i.e.) the contents of the file are determined when the file is created, not when it is used.
- The contents in the server can be changed but the user cannot change them.
- When a client accesses the document, a copy of the document is sent and the user can then use a browsing program to display the document.



Hypertext Markup Language (HTML)

- Hypertext Markup Language (HTML) is a language for creating Web pages.
- Data for a Web page are formatted for interpretation by a browser.
- HTML allows us to embed formatting instructions in the file itself. The instructions are included with the text.

A Web page is made up of two parts: **Head** and **Body.**

Head: The head is the first part of a Web page. The head contains the title of the page and other parameters that the browser will use.

Body: The actual contents of a page are in the body, which includes the text and the tags. The text is the actual information contained in a page. The tags define the appearance of the document.

HTML Tags

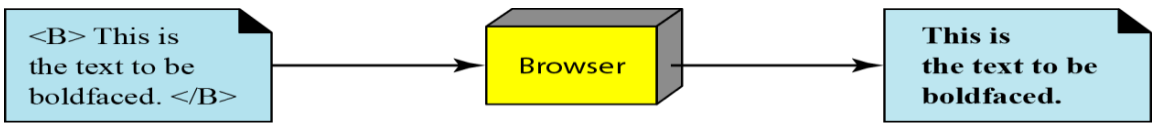
- Every HTML tag is a name followed by an optional list of attributes enclosed between less-than and greater-than symbols (< and>).
- An attribute is followed by an equals sign and the value of the attribute.
- The browser makes a decision about the structure of the text based on the tags, which are embedded into the text.

a. Beginning tag

b. Ending tag

The common tags used in HTML are : **Bold, Italic, Underline** the text.

- The two **Bold** face tags **<B>**and **</B>**are instructions for the browser.
- The two **Italic** tags **<I>**and **</I>**make the text italic
- The two **Underline** tags **<U>**and **</U>**put underline below the text.



There are two other tags used in HTML are: **Image** and **Hyperlink** tags.

Image tag

- Non-textual information such as digitized photos or graphic images is not a physical part of an HTML document.
- The image tag defines the address (URL) of the image to be retrieved. An image tag is used to point to the file of a photo or image.
- It also specifies how the image can be inserted after retrieval. Image tag contains attributes such as: SRC ,ALIGN.
- SRC (source) defines the source address
- ALIGN defines the alignment of the image

<IMG SRC=”/bin/images/image1.gif” ALIGN=MIDDLE>

Hyperlink tag

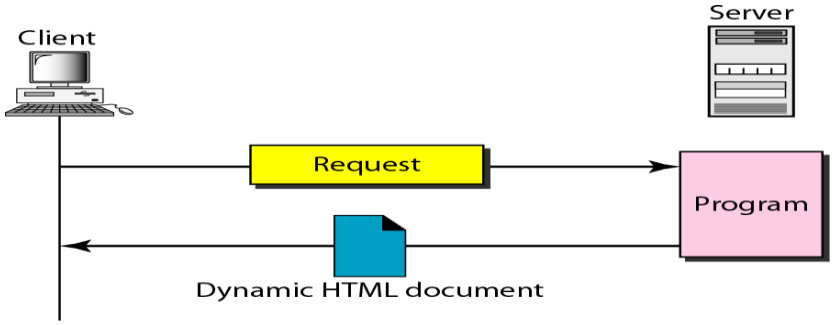
- Hyperlink tag is needed to link documents together. Any item such as word, phrase, paragraph or image can refer to another document through a mechanism called an **anchor**.
- The anchor is defined by <A ... > and </A> tags and the anchored item uses the URL to refer to another document.
- When the document is displayed, the anchored item is underlined, blinking, or boldfaced.
- The user can click on the anchored item to go to another document.
- The reference phrase is embedded between the beginning and ending tags.
- The beginning tag can have an attributes called HREF (Hyperlink Reference) defines the address (URL) of the linked document.

<A HREF= http://www/deanza.edu/forouzan>

Author </A> Dynamic Documents

- A **dynamic document** is created by a Web server whenever a browser requests the document.
- When a request arrives, the Web server runs an application program that creates the dynamic document.
- The server returns the output of the program as a response to the browser.
- Because a fresh document is created for each request, the contents of a dynamic document can vary from one request to another.

Example: the retrieval of the time and date from a server is a dynamic document.



Common Gateway Interface (CGI)

- CGI is a technology that creates and handles dynamic documents.
- CGI is a set of standards that defines how a dynamic document is written, how data are input to the program and how the output result is used
- The CGI also defines a set of rules and terms that the programmer must follow.

- CGI allows programmers to use any of several languages such as C, C++, Bourne Shell, Korn Shell, C Shell, Tcl, or Perl.
- CGI program can be used to access resources such as databases, graphical packages etc.

Input

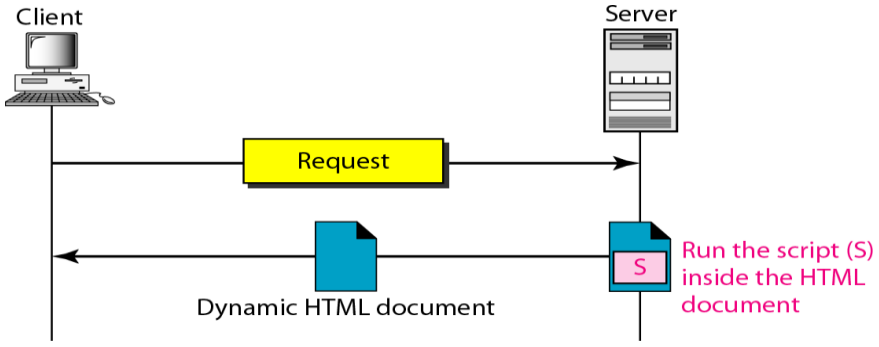
- The input from a browser to a server is sent by using a Form. If the information in a form is small (such as a word), it can be appended to the URL after a question mark.
  Example: The following URL is carrying Form information (23, a value):
  http://www.deanza/cgi-bin/prog.pl?23
- If the input from a browser is too long to fit in the query string, the browser can ask the server to send a form. The browser can then fill the form with the input data and send it to the server.
- The information in the form can be used as the input to the CGI program.

Output

- CGI executes a CGI program at the server site and send the output to the client(browser).
- The output can be a plain text, graphics or binary data, a status code, instructions to the browser to cache the result, or instructions to the server to send an existing document instead of the actual output.
- The output of the CGI program always consists of two parts: **Header** and **Body**.
- The Header is separated by a blank line from the body.

Scripting Technologies for Dynamic Documents

- The problem with CGI technology is the inefficiency that results, if part of the dynamic document that is to be created is fixed and not changing from request to request.
- The solution is to create a file containing the fixed part of the document using HTML and embed a script, a source code that can be run by the server to provide the dynamic part.
- PHP, JSP, ASP, ColdFusion are the technologies have been involved in creating dynamic documents using scripts.



Active Documents

- Applications need a program or a script to be run at the client site. These are called active documents.
- When a browser requests an active document, the server sends a copy of the document or a script. The document is then run at the client site(browser).

Example: Suppose we want to run a program that creates animated graphics on the

screen. The program definitely needs to be run at the client site where the animation takes place.

Java Applets

- By using java applets we can create an active document.
- Java is a combination of a high-level programming language, a run-time environment, and a class library that allows a programmer to write an active document (an applet) and a browser to run it.
- Java can also be a stand-alone program that doesn't use a browser.
- An applet is a program written in Java on the server. It is compiled and ready to be run.
- The document is in byte-code (binary) format.
- The client process (browser) creates an instance of this applet and runs it.

JavaScript

- Java script in dynamic documents can also be used for active documents.
- If the active part of the document is small, it can be written in a scripting language; then it can be interpreted and run by the client at the same time.
- The script is in source code (text) and not in binary form.
- JavaScript is a very high level scripting language developed for this purpose.

HYPERTEXT TRANSFER PROTOCOL (HTTP)

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP uses the services of TCP on well-known port 80.

HTTP functions as a combination of FTP and SMTP.

Similarity between HTTP and FTP

- HTTP is similar to FTP because it transfers files and uses the services of TCP.
- HTTP uses only TCP data connection to transfer the data between client and server and there is no control connection.

Similarity between HTTP and SMTP

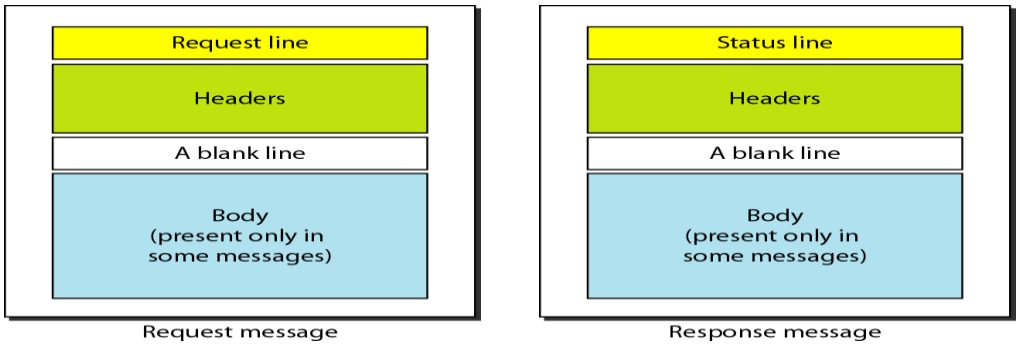
- In HTTP, the data transferred between the client and the server look like SMTP messages.
- The format of the messages is controlled by MIME-like headers.
- HTTP messages are read and interpreted by the HTTP server and HTTP client(browser).
- SMTP messages are stored & forwarded, but HTTP messages are delivered immediately.
- The commands from the client to the server are embedded in a request message.
- The contents of the requested file or information are embedded in a response message.

HTTP Transaction

HTTP is a stateless protocol even though it uses TCP services. The client initializes the transaction by sending a request message. The server replies by sending a response.

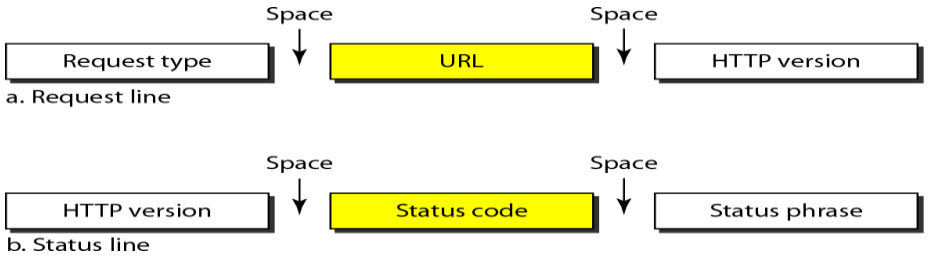
HTTP uses two types of messages: Request, Response

- A request message consists of a request line, a header, and optional body.
- A response message consists of a status line, a header, and optional body.



Request and Status Lines

- The first line in a request message is called a request line.
- The first line in the response message is called the status line.



Request type

This field is used in the request message. In version 1.1 of HTTP defines several request types. The request type is categorized into methods.

Methods	Action
GET	Requests a document from the server
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client

URL: By using URL, clients can access the webpage.

Version The most current version of HTTP is 1.1.

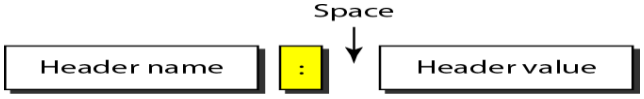
Status code is used in the response message. It consists of 3-digits. 100, 200, 300, 400, 500.

Status phrase: It explains the status code in text form, it is used in the response message.

Status Code	Status phrase	Description
100	Continue, switch	Informational
200	OK, CREATED, ACCEPTED	Successful request
300	Moved Permanently or temporarily, Not modified.	Redirect Client to another URL
400	400- Bad request, 404- Not found,	Error at client side
500	500- Internal server error 503-Service unavailable	Error at server side

Header

The header exchanges additional information between the client and the server. The header can consist of one or more header lines.



A Header line can be divided into 4 categories: 1. **General** 2.**Request** 3.**Response** 4.**Entity**.

1. A request message can contain Request header, General header, Entity header.
2. A response message can contain Response header, General header, Entity header.

General header

General header gives general information about the message such as Date, MIME version.

Request header

Request header specifies the client's configuration and the client's preferred documentformat. Example: Accept: Shows the medium format the client can accept

From: Shows the E-mail address of the user
Host: Shows the host and port number of the server Referrer: Specifies the URL of the linked document User agent: Identifies the client program.

Response header

This header specifies the server's configuration and special information about the request. Example: Age: shows the age of the document, public: shows the supported list of methods, server: shows the server name and address.

Entity header

The entity header gives information about the body of the document. some request messages such as POST or PUT methods may contain a body also use this type of header.

Examples: E tag- Gives an entity tag, Content-type- Specifies the medium type, Last-modified- Gives the date and time of the last change etc.

Body

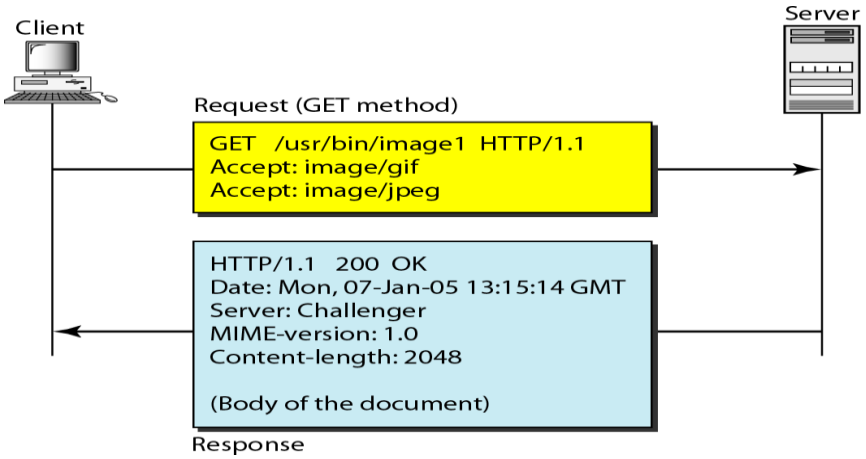
The body can be present in a request or response message. Body contains the document to be sent or received.

Example of HTTP transaction

Consider the below figure that shows how to retrieve a document.

- We use the GET method to retrieve an image with the path/usr/bin/imagel.
- The request line shows the method (GET), the URL, and the HTTP version(1.1).
- Header has two lines that show the client can accept images in the GIF or JPEG format.
- The request does not have a body.
- The response message contains the status line and four lines of header.
- The header lines define the date, server, MIME version, and length of the document.

- The body of the document follows the header.



HTTP Connections

Non-persistent connection

Versions before 1.1 use Non-persistent method as the default connection. In this connection, one TCP connection is made for each request/response.

The steps involved in this strategy:

1. The client opens a TCP connection and sends a request.
2. The server sends the response and closes the connection.
3. The client reads the data until it encounters an end-of-file marker after that the client closes the connection.

In this strategy, for N different pictures in different files, the connection must be opened and closed N times.

The Non-persistent strategy imposes high overhead on the server because the server needs N different buffers and requires a slow start procedure each time a connection is opened.

Persistent Connection

- Persistent connection is the default in HTTP version1.1.
- In this connection, the server leaves the connection open for more requests after sending a response.
- Server can close the connection at the request of a client or if a time-out has been reached.
- The sender sends the length of the data with each response. If the sender does not know the length of the data then document is created dynamically or actively.
- The server informs the client that the length is not known and closes the connection after sending the data so the client knows that the end of the data has been reached.

Proxy Server

HTTP supports Proxy Servers. A Proxy server is a computer that keeps copies of responses to recent requests.

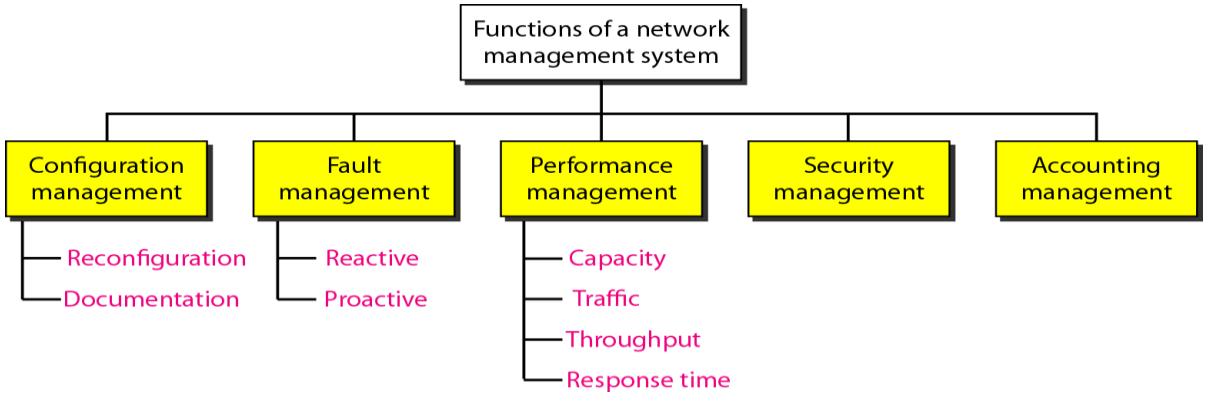
- The HTTP client sends a request to the proxy server. The proxy server checks its cache.
- If the response is not stored in the cache, the proxy server sends the request to the corresponding server.
- Incoming responses are sent to the proxy server and stored for future requests from other clients.
- Proxy server reduces the load on the original server, decreases traffic and improves

latency.

- To use the proxy server, the client must be configured to access the proxy instead of the targetserver.

NETWORK MANAGEMENT: SNMP

Network Management can be defined as monitoring, testing, configuring and troubleshooting network components to meet a set of requirements defined by an



organization.

Functions of Network Management System

Network Management System can be divided into five broad categories:

Configuration Management

- A large network is usually made up of hundreds of entities that are physically or logically connected to one another.
- These entities have an initial configuration when the network is set up, but can change with time.
- The configuration management system must know the status of each entity and its relation to other entities.

Configuration management can be divided into two subsystems: Reconfiguration and Documentation.

Reconfiguration

Reconfiguration means in a large network, the network components and features are adjusted daily. There are three types of reconfiguration:

i. **Hardware reconfiguration:** It covers all changes to the hardware. These are handled manually. Example: A sub network (Router) may be added or removed from the network.

ii. **Software reconfiguration:** It covers all changes to the software. Most of the software reconfiguration can be automated. Example: Updating Operating system.

iii. **User-account reconfiguration:** It covers adding and deleting the users on a system and it also considers user's individual privileges and Group privileges. Example: A user may have read and write permission with regard to some files, but only read permission with regard to other files.

Documentation

The network configuration and each subsequent change in hardware, software and user accounts must be documented.

Hardware documentation

- It involves two sets of documents: Maps andSpecifications.
- **Maps** track each piece of hardware and its connection to the network.
- General maps that shows the logical relationship as well as physical relationship between eachsub network.
- For each sub-network, there are one or more maps that show all pieces of equipment.
- **Specification** information such as hardware type, serial number, vendor address and phone number, time of purchase and warranty information must be included for each piece of hardware connected to the network.

Software Documentation It includes information such as the software type, the version, the time installed etc.

User documentation Operating system utilities allows the documentation of user accounts and their privileges. The information in these files are updated and secured.

Fault Management

Fault management is the area of network management that handles the issues in network components. Example: A fault may be a damaged communication medium. Fault management system has two subsystems: Reactive and Proactive.

Reactive Fault Management

The responsibilities of reactive fault management can be divided into 4 steps:

i. **Detect the fault:** Fault management system must have to detect the exact location of the fault.
ii. **Isolate the fault:** If a fault is isolated that affects only a few users. After isolation, the affected users are immediately notified and given an estimated time of correction.
iii. **Correct the fault:** This may involve replacing or repairing the faulty components.
iv. **Record the fault:** After the fault is corrected, it must be recorded (i.e. documented). The record should show the exact location of the fault, the possible cause, the action or actions taken to correct the fault, the cost and time it took for each step.

Proactive Fault Management

Proactive fault management tries to prevent faults from occurring. Some failures can be predicted and prevented.

Performance Management

Performance management tries to monitor and control the network to ensure that it is running as efficiently. It can be measured by the following concepts: Capacity, Traffic, Throughput, Response time.

- **Capacity of the Network** Every network has a limited capacity, and the performance management system must ensure that it is not used above this capacity. Example: If a LAN is designed for 100 stations at an average data rate of2 Mbps, it will not operate properly if 200 stations are connected to the network.
- **Traffic** Traffic can be measured in two ways: Internally and Externally.
  Internal traffic is measured by the number of packets (or bytes) traveling inside the network.
  External traffic is measured by the exchange of packets (or bytes) outside the network.
- **Throughput** It can be measured by an individual device such as a router or a part

of the network. Throughput makes sure that, the device is not reduced to unacceptable levels.

- **Response Time** It is measured from the time a user requests a service to the time the service is granted.

Security Management

Security management is responsible for controlling access to the network based on the predefined policy.

Accounting Management

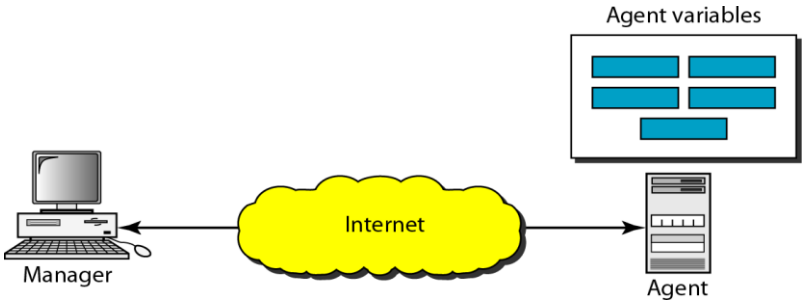
- Accounting management is the control of users' access to network resources through charges.
- Under accounting management, individual users, departments, divisions, or even projects are charged for the services they receive from the network.

SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

The Simple Network Management Protocol (SNMP) is a framework for managing devices in an internet using the TCP/IP protocol suite. It provides a set of fundamental operations for monitoring and maintaining an internet.

Concept of SNMP

- SNMP is an application level protocol that uses the concept of Manager and Agent.
- A manager controls and monitors a set ofagents.
- A manager may be a host and an Agent may be a router.
- SNMP can monitor devices made by different manufacturers and installed on different physical networks.
- SNMP can be used in LANs and WANs connected by routers.



Managers and Agents

- A Manager or a management station is a host that runs the SNMP client program.
- An Agent or a Managed station is a router or a host that runs the SNMP server program.

Management is achieved through simple interaction between a manager and an agent.

- Agent keeps performance information in a database.
- Manager has access to the values in the database.

Example: A router can store in variables such as the number of packets received and forwarded. The manager can fetch and compare the values of these two variables to see if the router is congested or not.

Management with SNMP is based on three basic ideas:

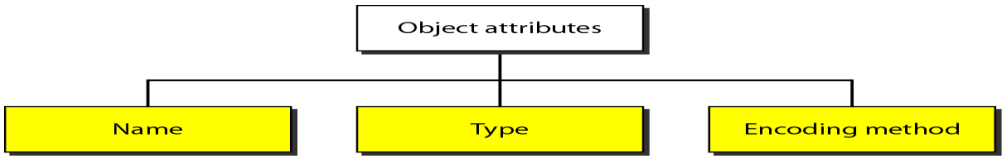
1. A manager checks an agent by requesting information that reflects the behavior of the agent.
2. A manager forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes to the management process by sending warning message to the manager of an unusual situation. The warning message is called the trap.

Management Components

- SNMP uses two other protocols to do management tasks: Structure of Management Information (SMI) and Management Information Base(MIB).

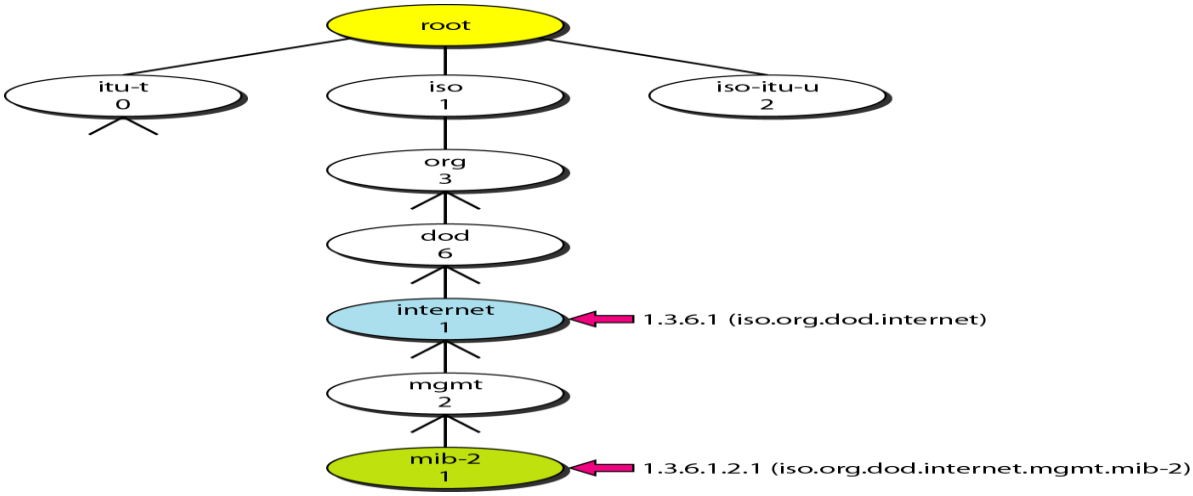
Structure of Management Information (SMI)

The Structure of Management Information version 2 (SMIv2) is a component for network management. Functions of SMI are:



Name

- SMI requires that each managed object (such as a router, a variable in a router, a value) have a unique name.
- To name objects globally SMI uses an object identifier, which is a hierarchical identifier based on a tree structure. The tree structure starts with an unnamed root.



- Each object can be defined by using a sequence of integers separated by dots.
- Tree structure can also define an object by using a sequence of textual names separated by dots.
- The integer-dot representation is used in SNMP. The name-dot notation is used by people.

  Example: The following shows the same object in two different notations:

  iso.org.dod.internet.mgmt.mib-2 □1.3.6.1.2.1

Type

- To define the data type, SMI uses fundamental Abstract Syntax Notation 1(ASN.l).

- SMI has two broad categories of data type: Simple and Structured.

Simple Type

The simple data types are atomic data types such a as Integer32, Octet String, IP address etc.

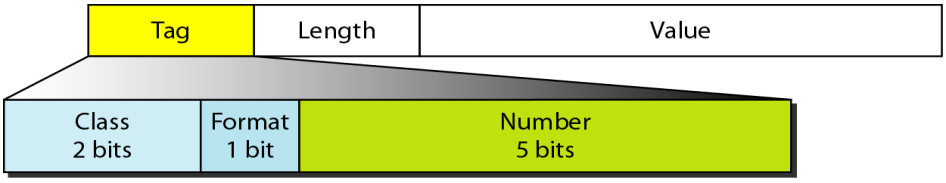
Structured Type

By combining simple and structured data types, we can make new structured data types. SMI defines two structured data types: Sequence and Sequence of.

- A **Sequence** data type is a combination of simple data types, not necessarily of the same type. It is similar to the concept of a struct used in C programming.
- A **Sequence of** data type is a combination of simple data types all of the same type. It is similar to the concept of an array used in C-programming.

Encoding Method

SMI uses another standard Basic Encoding Rules (BER) to encode data to be transmitted over the network. BER specifies data to be encoded in following format: Tag, Length and Value.



Tag

The tag is a 1-byte field that defines the type of data. It is composed of three subfields:

- **Class (2 bits):** It defines the scope of the data.
  Four classes are defined: 00- universal, 01-Application wide, 10- context specific, 11- private.
- **Format** subfield indicates whether the data are simple (0) or structured(1).
- **Number** subfield further divides simple or structured data into subgroups.
  Example: In the universal class with simple format, INTEGER has a value of 2, OCTET STRING has a value of 4.

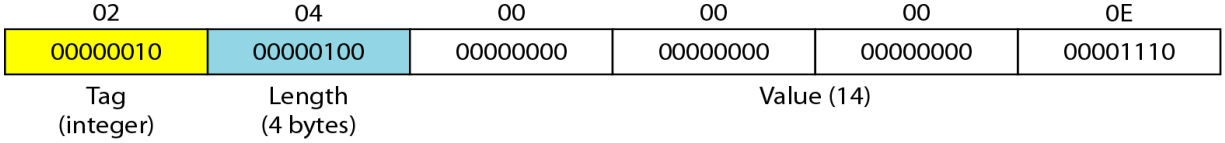
Length

- The length field is 1 or more bytes.
- If it is 1 byte, the most significant bit must be 0. Other 7 bits define the length of the data.
- If it is more than 1 byte, the most significant bit of the first byte must be 1. The other 7 bits of the first byte define the number of bytes needed to define the length.

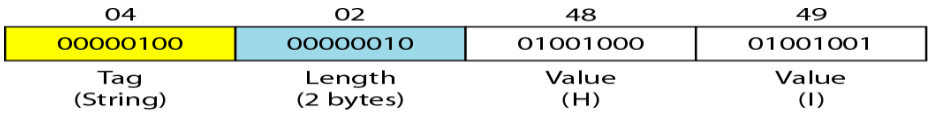
Value field codes the value of the data according to the rules defined in BER.

Example: Show the following in encoding representation:

1. Define INTEGER14.

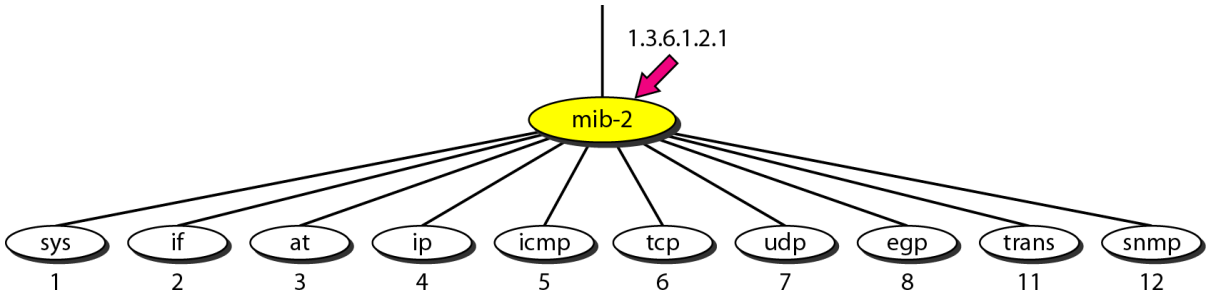


2. Define OCTET STRING“HI”



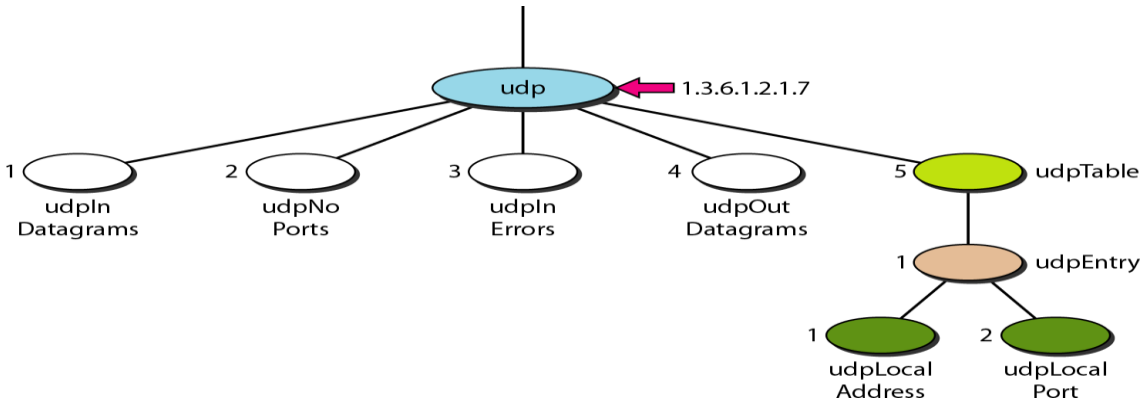
Management Information Base(MIB2)

- MIB version 2 is the second component used in network management.
- MIB creates a collection of named objects, their types and their relationships to each other in an entity to be managed.
- Each agent has its own MIB2, which is a collection of all the objects that the manager can manage.
- The objects in MIB2 are categorized under 10 different groups: system, interface, address translation, ip, icmp, tcp, udp, egp, transmission, and snmp.
- These groups are under the MIB-2 object in the object identifier tree.
- Each group has defined variables and tables.



Accessing MIB Variables

Let us take UDP group to show how to access different variables. To access any of the simple variables, we use the id of the group (1.3.6.1.2.1.7) followed by the id of the variable.



The following shows how to access each variable:

UdpIn Datagrams	□	1.3.6.1.2.1.7.1
UdpNo Ports	□	1.3.6.1.2.1.7.2
UdpIn Errors	□	1.3.6.1.2.1.7.3
UdpOut Datagrams	□	1.3.6.1.2.1.7.4

The object identifiers define the variable not instances (contents). An instance suffix "**0**" should be added to show the instance of each variable.

udpInDatagrams.0	□	1.3.6.1.2.1.7.1.0
udpNoPorts.0	□	1.3.6.1.2.1.7.2.0
udpInErrors.0	□	1.3.6.1.2.1.7.3.0
udpOutDatagrams.0	□	1.3.6.1.2.1.7.4.0

Tables

To identify a table, we first use the table id. To access the table, we have to define the table entries.

Udp Table 1.3.6.1.2.1.7.5
udp Entry 1.3.6.1.2.1.7.5.1

To access the entry we need to define each entity (field) in the entry.

Udp Local Address 1.3.6.1.2.1.7.5.1.1
Udp Local Port 1.3.6.1.2.1.7.5.1.2

- To access a specific instance (row) of the table, we add the index to the above ids. To access the instance of the local address for the first row, we use the identifier augmented with the instance index:

  udpLocalAddress.181.23.45.14.23 1.3.6.1.2.7.5.1.1.181.23.45.14.23

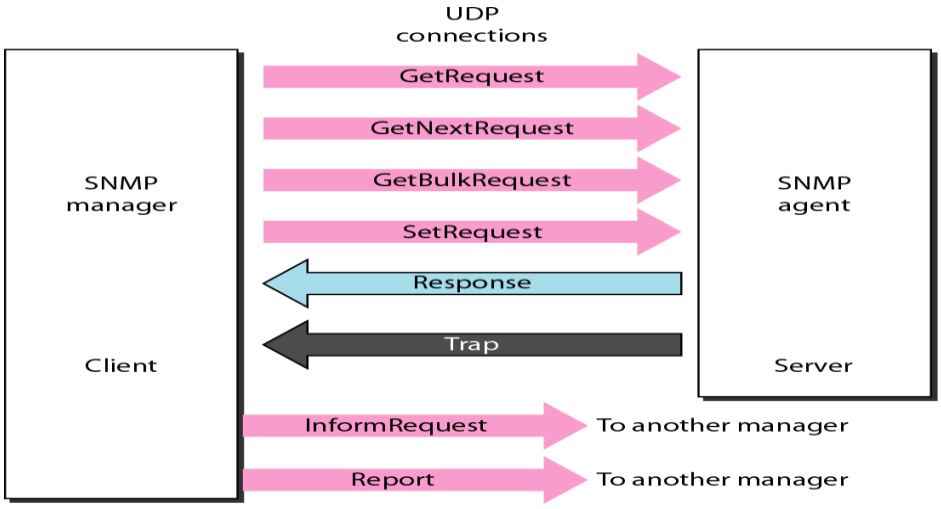
Lexicographic Ordering

- The object identifiers follow in lexicographic order.
- Tables are ordered column by column from the top to the bottom.
- The lexicographic ordering enables a manager to access a set of variables one after another by defining the first variable.

SNMP version3 (SNMPv3)

- SNMP defines the format of packets exchanged between a manager and an agent.
- SNMP interprets the result and creates statistics.
- The packets exchanged contain the object (variable) names and their status(values).
- SNMP is responsible for reading and changing these values.
- SNMP uses both SMI and MIB in Internet network

management. SNMPv3 defines eight types of packets or

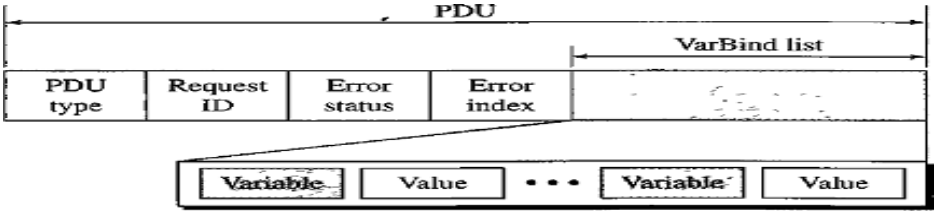


PDUs.

1. **GetRequest** PDU is sent from the manager (client) to the agent (server) to retrieve the value of a variable or a set of variables.
2. **GetNextRequest** PDU is sent from the manager to the agent to retrieve the value of a variable. It is mostly used to retrieve the values of the entries in a table.
3. **GetBulkRequest** PDU is sent from the manager to the agent to retrieve a large amount of data.
4. **SetRequest** PDU is sent from the manager to the agent to set (store) a value in a variable.

5. **Response** PDU is sent from an agent to a manager in response to GetRequest or GetNextRequest.
6. **Trap** PDU is sent from the agent to the manager to report an event. For example, if the agent is rebooted, it informs the manager and reports the time of rebooting.
7. **InformRequest** PDU is sent from one manager to another remote manager to get the value of some variables from agents under the control of the remote manager. The remote manager responds with a Response PDU.
8. **Report** PDU is designed to report some types of errors between managers. It is not yet in use.

Format of PDU



- **PDU type.** This field defines the type of the PDU.
- **Request ID** This field is a sequence number used by the manager in a Request PDU and repeated by the agent in a response. It is used to match a request to a response.
- **Error status**. This is an integer that is used only in Response PDUs to show the types of errors reported by the agent. Its value is 0 in Request PDUs.
- **Error index**. It is an offset that tells the manager which variable caused the error.
- **Non-repeaters**. This field is used only in GetBulkRequest and replaces the error status field, which is empty in Request PDUs.
- **Max-repetition**. This field is also used only in GetBulkRequest and replaces the error index field, which is empty in Request PDUs.
- **VarBind list**. This is a set of variables with the corresponding values the manager wants to retrieve or set. The values are null in GetRequest and GetNextRequest.

Messages

SNMP embeds the PDU in a message. A message in SNMPv3 is made of four elements: Version, Header, Security parameters and Data.

- **Version** defines the current version(3).
- **Header** contains values for message identification, maximum message size, message flag, and a message security model.
- **Security parameter** is used to create a message digest.
- **Data** contain the encoded PDU. The data may or may not be encrypted.

SNMP UDP Ports

- SNMP uses the services of UDP on two well-known ports, 161 and162.
- The well-known port 161 is used by the server (agent).
- The well-known port 162 is used by the client (manager).

Security

- SNMPv3 provides two types of security: General and Specific.

- SNMPv3 provides message authentication, privacy, and manager authorization.
  SNMPv3 allows a manager to remotely change the security configuration, which means that the manager does not have to be physically present at the manager station.