EC3401 NETWORKS AND SECURITY

UNIT III TRANSPORT AND APPLICATION LAYERS

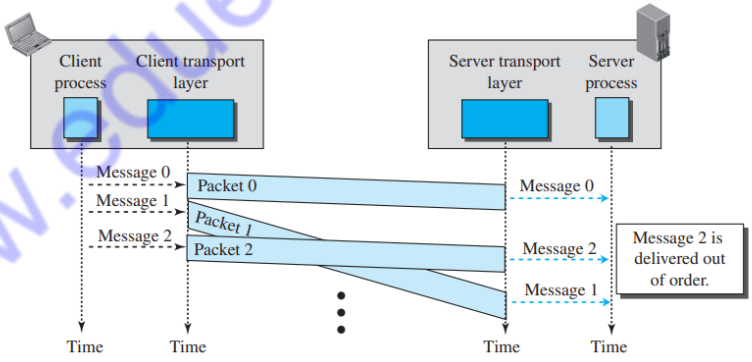
Transport Layer Protocols – UDP and TCP Connection and State Transition Diagram - Congestion Control and Avoidance (DEC bit, RED)- QoS - Application Layer Paradigms – Client – Server Programming – Domain Name System – World Wide Web, HTTP, Electronic Mail.

The transport layer is located between the application layer and the network layer. It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host.

A transport-layer protocol, like a network-layer protocol, can provide two types of services: connectionless and connection-oriented

Connectionless Service:

- In a connectionless service, the source process (application program) needs to divide its message into chunks of data of the size acceptable by the transport layer and deliver them to the transport layer one by one.
- The transport layer treats each chunk as a single unit without any relation between the chunks.
- When a chunk arrives from the application layer, the transport layer encapsulates it in a packet and sends it.
- Since there is no dependency between the packets at the transport layer, the packets may arrive out of order at the destination and will be delivered out of order to the server process



- The situation would be worse if one of the packets were lost. Since there is no numbering on the packets, the receiving transport layer has no idea that one of the messages has been lost. It just delivers two chunks of data to the server process.

Connection-Oriented Service:

- In a connection-oriented service, the client and the server first need to establish a logical connection between them. The data exchange can only happen after the connection establishment. After data exchange, the connection needs to be torn down.

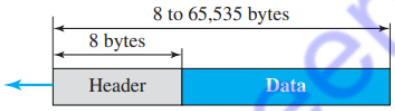
USER DATAGRAM PROTOCOL

- The User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol.

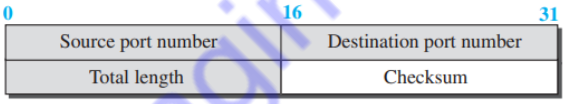
- It does not add anything to the services of IP except for providing process-to-process communication instead of host-to-host communication.
- UDP is a very simple protocol using a minimum of overhead.
- If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a small message using UDP takes much less interaction between the sender and receiver

User Datagram

- UDP packets, called user datagrams, have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).
- The first two fields define the source and destination port numbers.
- The third field defines the total length of the user datagram, header plus data.
- The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be less because a UDP user datagram is stored in an IP datagram with the total length of 65,535 bytes
- The last field can carry the optional checksum



a. UDP user datagram



b. Header format

Example:

The following is the content of a UDP header in hexadecimal format. **CB84000D001C001C**

a. What is the source port number?

b. What is the destination port number?

c. What is the total length of the user datagram?

d. What is the length of the data?

Solution

a. The source port number is the first four hexadecimal digits (CB84)16, which means that the source port number is 52100.

b. The destination port number is the second four hexadecimal digits (000D)16, which means that the destination port number is 13.

c. The third four hexadecimal digits (001C)16 define the length of the whole UDP packet as 28 bytes.

d. The length of the data is the length of the whole packet minus the length of the header, or 28 − 8 = 20 bytes.

UDP Services:

Process-to-Process Communication:

UDP provides process-to-process communication using socket addresses, a combination of IP addresses and port numbers

Connectionless Services

- UDP provides a connectionless service.
- There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
- The user datagrams are not numbered

Flow Control

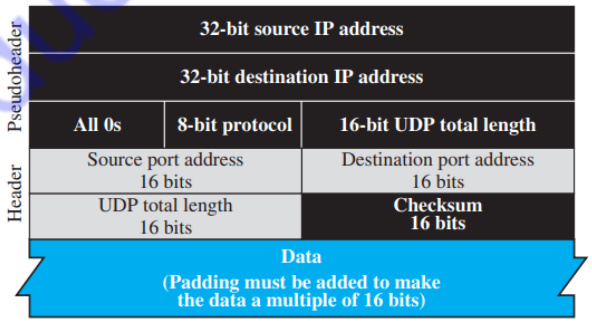
- UDP is a very simple protocol. There is no flow control, and hence no window mechanism. The receiver may overflow with incoming messages.

Error Control

- There is no error control mechanism in UDP except for the checksum.
- 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.

Checksum

- UDP checksum calculation includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer.
- The pseudoheader is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s
- If the checksum does not include the pseudo header, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host.
- If the checksum does not include the pseudo header, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host.



Applications of UDP:

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

- UDP is normally used for interactive real-time applications that cannot tolerate uneven delay between sections of a received message

TRANSMISSION CONTROL PROTOCOL:

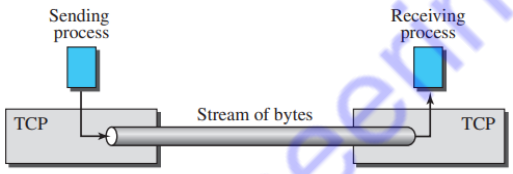
Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol. TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.

TCP Services:

Process-to-Process Communication: As with UDP, TCP provides process-to-process communication using port numbers.

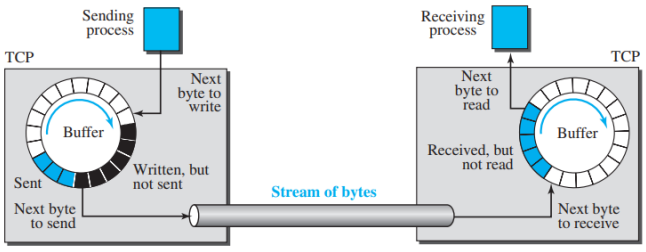
Stream Delivery Service:

- 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. TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their bytes across the Internet



Sending and Receiving Buffers:

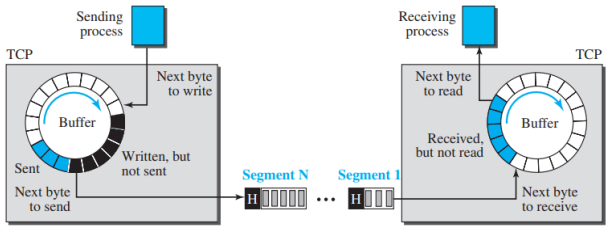
- Because the sending and the receiving processes may not necessarily write or read data at the same rate, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction
- At the sender, the buffer has three types of chambers. The white section contains empty chambers that can be filled by the sending process (producer). The colored area holds bytes that have been sent but not yet acknowledged. The TCP sender keeps these bytes in the buffer until it receives an acknowledgment
- TCP may be able to send only part of this shaded section. This could be due to the slowness of the receiving process or to congestion in the network
- The operation of the buffer at the receiver is simpler. The circular buffer is divided into two areas. The white area contains empty chambers to be filled by bytes received from the network. The colored sections contain received bytes that can be read by the receiving process.



Segments:

- At the transport layer, TCP groups a number of bytes together into a packet called a segment

- TCP adds a header to each segment and delivers the segment to the network layer for transmission
- The segments are encapsulated in an IP datagram and transmitted. This entire operation is transparent to the receiving process.



Full-Duplex Communication: TCP offers full-duplex service, where data can flow in both directions at the same time

Multiplexing and Demultiplexing: TCP performs multiplexing at the sender and demultiplexing at the receiver

Connection-Oriented Service TCP: When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:

1. The two TCP's establish a logical connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

Numbering System: Although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields, called the sequence number and the acknowledgment number. These two fields refer to a byte number and not a segment number.

Byte Number: TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, TCP stores them in the sending buffer and numbers them. The numbering does not necessarily start from 0. Instead, TCP chooses an arbitrary number between 0 and 232 − 1 for the number of the first byte.

Sequence Number: After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number, in each direction, is defined as follows:

1. The sequence number of the first segment is the ISN (initial sequence number), which is a random number.

2. The sequence number of any other segment is the sequence number of the previous segment plus the number of bytes (real or imaginary) carried by the previous segment.

TCP Segment Format:

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

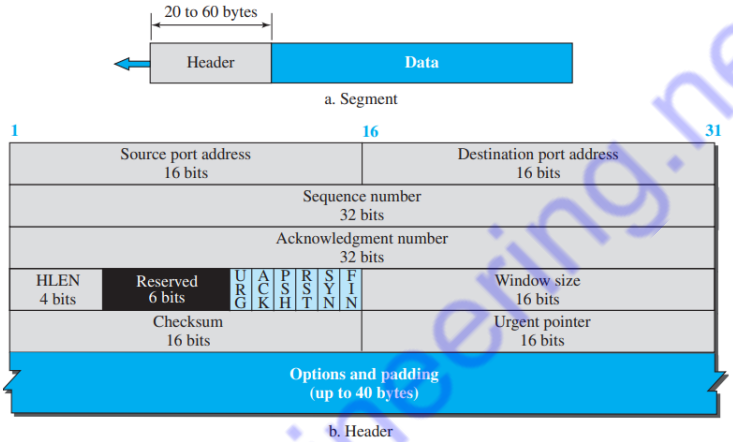
Source port address. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.

Destination port address. This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

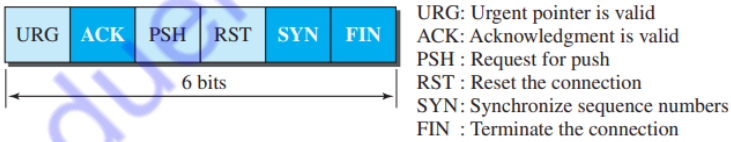
Sequence number. This 32-bit field defines the number assigned to the first byte of data contained in this segment. The sequence number tells the destination which byte in this sequence is the first byte in the segment.

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

Header length. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field is always between 5 (5 × 4 = 20) and 15 (15 × 4 = 60).



Control. This field defines 6 different control bits or flags. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.



Window size. This field defines the window size of the sending TCP in bytes. The length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes

Checksum. This 16-bit field contains the checksum

Urgent pointer. This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data

TCP Connection:

TCP is connection-oriented. A connection-oriented transport protocol establishes a logical path between the source and destination. All of the segments belonging to a message are then sent over this logical path.

Using a single logical pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames

In TCP, connection-oriented transmission requires three phases: **connection establishment, data transfer, and connection termination**

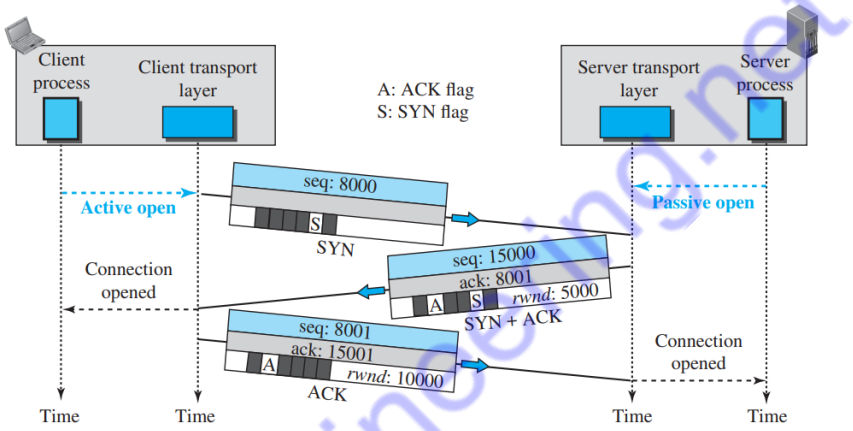
Connection Establishment:

- TCP transmits data in full-duplex mode.
- When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.

- This implies that each party must initialize communication and get approval from the other party before any data are transferred.

Three-Way Handshaking:

- The connection establishment in TCP is called three-way handshaking. An application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport-layer protocol.
- The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This request is called a passive open
- The client program issues a request for an active open. A client that wishes to connect to an open server tells its TCP to connect to a particular server

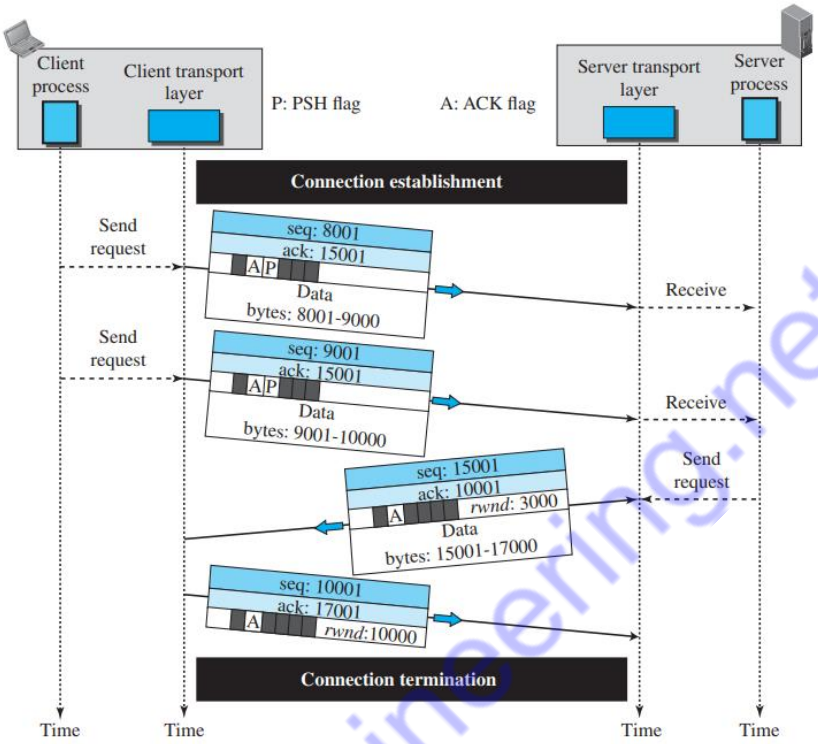


- The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers. This sequence number is called the initial sequence number (ISN). This segment does not contain an acknowledgment number
- The SYN segment is a control segment and carries no data. However, it consumes one sequence number because it needs to be acknowledged.
- The server sends the second segment, a SYN + ACK segment with two flag bits set as: SYN and ACK. This segment has a dual purpose. First, it is a SYN segment for communication in the other direction. The server uses this segment to initialize a sequence number for numbering the bytes sent from the server to the client.
- The server also acknowledges the receipt of the SYN segment from the client by setting the ACK flag and displaying the next sequence number it expects to receive from the client. Because the segment contains an acknowledgment, it also needs to define the receive window size, rwnd (to be used by the client)
- The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field

Data Transfer

- After connection is established, bidirectional data transfer can take place. The client and server can send data and acknowledgments in both directions
- In the diagram shown in figure the client sends 2,000 bytes of data in two segments. The server then sends 2,000 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 is 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

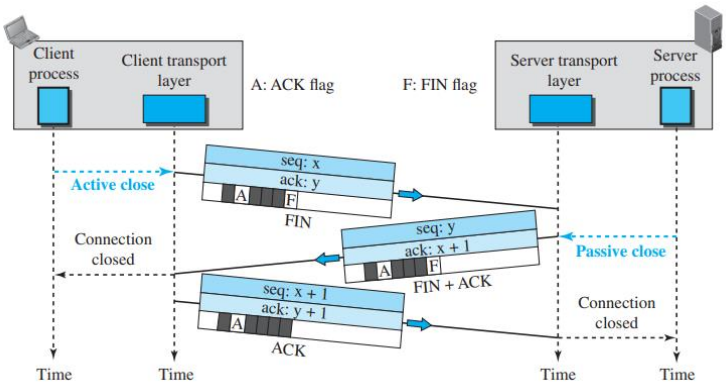


Connection Termination:

- Either of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client
- Most implementations today allow two options for connection termination: three-way handshaking and four-way handshaking with a half-close option.

Three-Way Handshaking:

- In this situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set.
- The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN + ACK segment, to confirm the receipt of the FIN segment
- The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is one plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

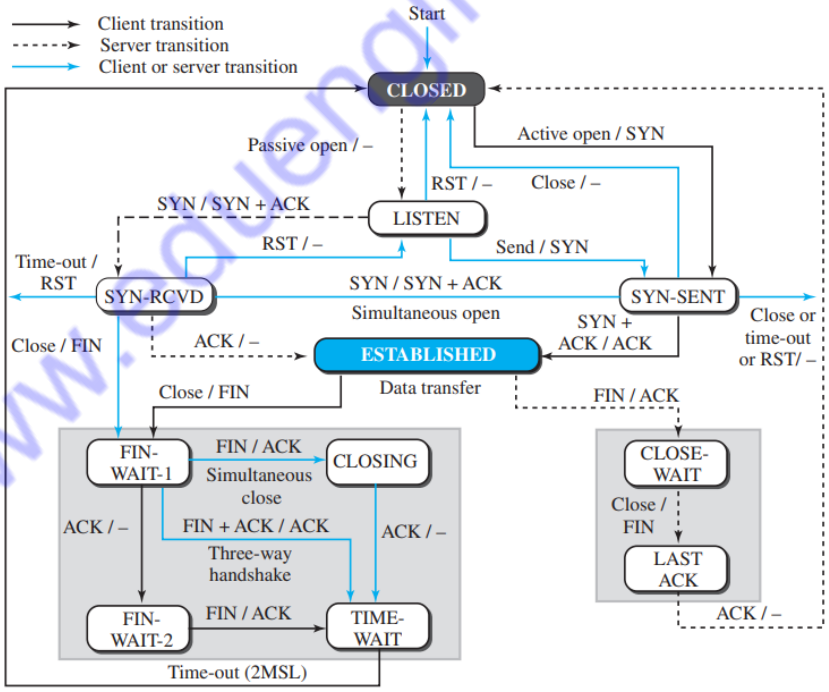


Connection Reset

- TCP at one end may deny a connection request, may abort an existing connection, or may terminate an idle connection

State Transition Diagram

- To keep track of all the different events happening during connection establishment, connection termination, and data transfer, TCP is specified as the finite state machine (FSM)
- The two FSMs used by the TCP client and server combined in one diagram. The rounded-corner rectangles represent the states. The transition from one state to another is shown using directed lines. Each line has two strings separated by a slash. The first string is the input, what TCP receives. The second is the output, what TCP sends.



- The dotted black lines in the figure represent the transition that a server normally goes through; the solid black lines show the transitions that a client normally goes through. However, in some situations, a server transitions through a solid line or a client transitions through a dotted line. The colored lines show special situations

State	Description
CLOSED	No connection exists
LISTEN	Passive open received; waiting for SYN
SYN-SENT	SYN sent; waiting for ACK
SYN-RCVD	SYN + ACK sent; waiting for ACK
ESTABLISHED	Connection established; data transfer in progress
FIN-WAIT-1	First FIN sent; waiting for ACK
FIN-WAIT-2	ACK to first FIN received; waiting for second FIN
CLOSE-WAIT	First FIN received, ACK sent; waiting for application to close
TIME-WAIT	Second FIN received, ACK sent; waiting for 2MSL time-out
LAST-ACK	Second FIN sent; waiting for ACK
CLOSING	Both sides decided to close simultaneously

TCP Congestion Control:

- In flow control the size of the send window is controlled by the receiver using the value of rwnd, which is advertised in each segment traveling in the opposite direction.
- The use of this strategy guarantees that the receive window is never overflowed with the received bytes
- However, does not mean that the intermediate buffers, buffers in the routers, do not become congested. A router may receive data from more than one sender. No matter how large the buffers of a router may be, it may be overwhelmed with data, which results in dropping some segments sent by a specific TCP sender
- TCP needs to worry about congestion in the middle because many segments lost may seriously affect the error control.
- To control the number of segments to transmit, TCP uses another variable called a congestion window, cwnd, whose size is controlled by the congestion situation in the network
- The cwnd variable and the rwnd variable together define the size of the send window in TCP. The first is related to the congestion in the middle .The second is related to the congestion at the end. The actual size of the window is the minimum of these two.

Actual window size = minimum (rwnd, cwnd)

Congestion Detection:

- The TCP sender uses the occurrence of two events as signs of congestion in the network: time-out and receiving three duplicate ACKs.
- The first is the time-out. If a TCP sender does not receive an ACK for a segment or a group of segments before the time-out occurs, it assumes that the corresponding segment or segments are lost and the loss is due to congestion
- When a TCP receiver sends a duplicate ACK, it is the sign that a segment has been delayed, but sending three duplicate ACKs is the sign of a missing segment, which can be due to congestion in the network.
- When a receiver sends three duplicate ACKs, it means that one segment is missing, but three segments have been received. The network is either slightly congested or has recovered from the congestion

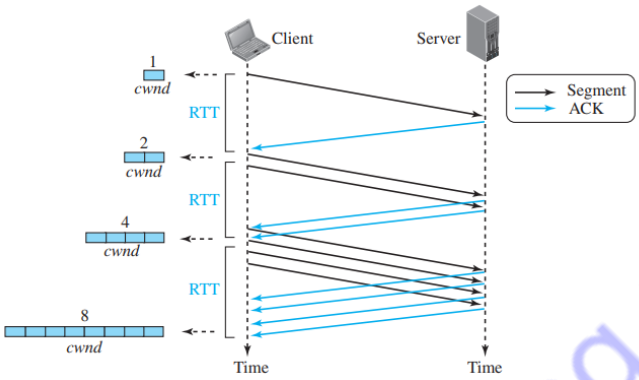
Congestion Policies:

- TCP's general policy for handling congestion is based on three algorithms: slow start, congestion avoidance, and fast recovery

Slow Start (Exponential Increase):

- The slow-start algorithm is based on the idea that the size of the congestion window (cwnd) starts with one maximum segment size (MSS), but it increases one MSS each time an acknowledgment arrives.

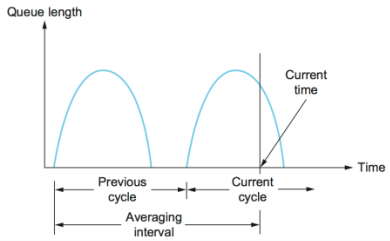
- The algorithm starts slowly, but grows exponentially. The rwnd is much larger than cwnd, so that the sender window size always equals cwnd.
- The sender starts with cwnd = 1. This means that the sender can send only one segment. After the first ACK arrives, the acknowledged segment is purged from the window, which means there is now one empty segment slot in the window.



- The size of the congestion window is also increased by 1 because the arrival of the acknowledgement is a good sign that there is no congestion in the network.
- The size of the window is now 2. After sending two segments and receiving two individual acknowledgments for them, the size of the congestion window now becomes 4, and so on.
- The size of the congestion window in this algorithm is a function of the number of ACKs arrived and can be determined as follows.
- If an ACK arrives, cwnd = cwnd + 1.
- A slow start cannot continue indefinitely. There must be a threshold to stop this phase. The sender keeps track of a variable named ssthresh (slow-start threshold). When the size of the window in bytes reaches this threshold, slow start stops and the next phase starts

DEC bit

- It is first mechanism was developed for use on the Digital Network Architecture (DNA), a connectionless network with a connection-oriented transport protocol.
- The idea is to more evenly split the responsibility for congestion control between the routers and the end nodes.
- Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur. This notification is implemented by setting a binary congestion bit in the packets that flow through the router, hence the name DECbit.
- The destination host then copies this congestion bit into the ACK it sends back to the source. Finally, the source adjusts its sending rate so as to avoid congestion.
- A single congestion bit is added to the packet header. A router sets this bit in a packet if its average queue length is greater than or equal to 1 at the time the packet arrives.
- This average queue length is measured over a time interval that spans the last busy+idle cycle, plus the current busy cycle.
- Figure shows the queue length at a router as a function of time. Essentially, the router calculates the area under the curve and divides this value by the time interval to compute the average queue length.
- Using a queue length of 1 as the trigger for setting the congestion bit is a trade-off between significant queuing (and hence higher throughput) and increased idle time (and hence lower delay). In other words, a queue length of 1 seems to optimize the power function.



Random Early Detection(RED)

- A second mechanism, called random early detection (RED), is similar to the DECbit scheme in that each router is programmed to monitor its own queue length and, when it detects that congestion is imminent, to notify the source to adjust its congestion window.
- The first is that rather than explicitly sending a congestion notification message to the source, RED is most commonly implemented such that it implicitly notifies the source of congestion by dropping one of its packets.
- The source is, therefore, effectively notified by the subsequent timeout or duplicate ACK.
- In case you haven't already guessed, RED is designed to be used in conjunction with TCP, which currently detects congestion by means of timeouts.
- As the "early" part of the RED acronym suggests, the gateway drops the packet earlier than it would have to, so as to notify the source that it should decrease its congestion window sooner than it would normally have.
- In other words, the router drops a few packets before it has exhausted its buffer space completely, so as to cause the source to slow down, with the hope that this will mean it does not have to drop lots of packets later on.

Quality of service (QoS):

- It is an internetworking issue that refers to a set of techniques and mechanisms that guarantee the performance of the network to deliver predictable service to an application program

QoS Characteristics:

Reliability :

Reliability is a characteristic that a flow needs in order to deliver the packets safe and sound to the destination. Lack of reliability means losing a packet or acknowledgment, which entails retransmission

Delay:

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

Jitter:

Jitter is the variation in delay for packets belonging to the same flow. For example, 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

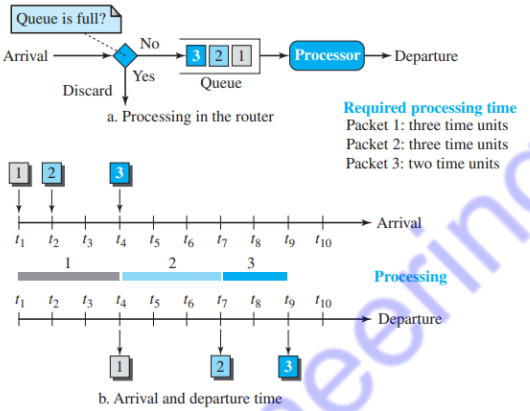
Flow Control to Improve QoS:

Scheduling:

- Treating packets in the Internet based on their required level of service can mostly happen at the routers. It is at a router that a packet may be delayed, suffer from jitters, be lost, or be assigned the required bandwidth.
- A good scheduling technique treats the different flows in a fair and appropriate manner. Several scheduling techniques are designed to improve the quality of service- FIFO queuing, priority queuing, and weighted fair queuing.

FIFO Queuing

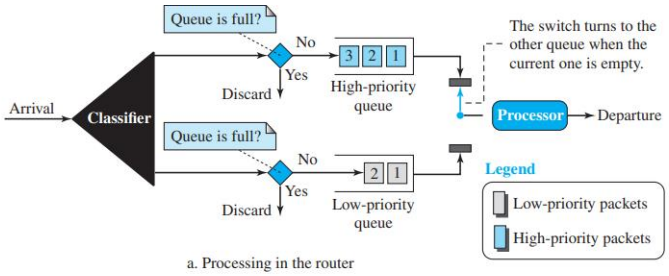
- In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router) is ready to process them.
- If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded.
- A larger packet definitely may need a longer processing time. In the figure, packets 1 and 2 need three time units of processing, but packet 3, which is smaller, needs two time units.
- This means that packets may arrive with some delays but depart with different delays. If the packets belong to the same application, this produces jitters. If the packets belong to different applications, this also produces jitters for each application.



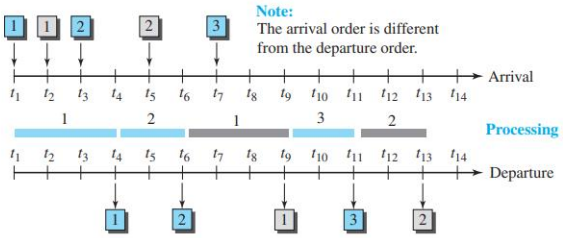
- FIFO queuing is the default scheduling in the Internet.
- With FIFO queuing, all packets are treated the same in a packet-switched network.
- The bandwidth allocated for each application depends on how many packets arrive at the router in a period of time.

Priority Queuing:

- Queuing delay in FIFO queuing often degrades quality of service in the network. A frame carrying real-time packets may have to wait a long time behind a frame carrying a small file.
- 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.
- A priority queue can provide better QoS than the FIFO queue because higher-priority traffic, such as multimedia, can reach the destination with less delay
- If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called **starvation**. Severe starvation may result in dropping of some packets of lower priority.



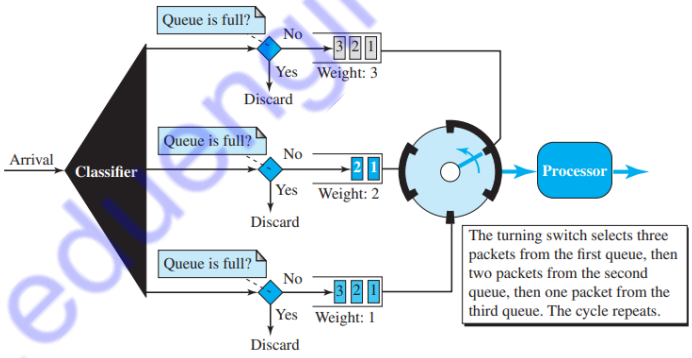
a. Processing in the router



b. Arrival and departure time

Weighted Fair Queuing :

- A better scheduling method is weighted fair queuing.
- In this technique, the packets are still assigned to different classes and admitted to different queues.
- The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight.



- The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight.
- For 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.
- In weighted fair queuing, each class may receive a small amount of time in each time period. In other words, a fraction of time is devoted to serve each class of packets, but the fraction depends on the priority of the class

Client Server Programming:

In a client-server paradigm, communication at the application layer is between two running application programs called processes: a client and a server.

A client is a running program that initializes the communication by sending a request; a server is another application program that waits for a request from a client.

The server handles the request received from a client, prepares a result, and sends the result back to the client.

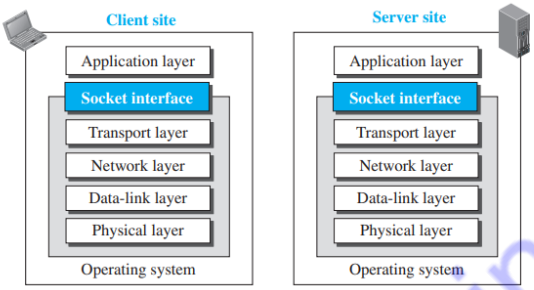
The server program needs to be started before start running the client program. The lifetime of server is infinite and client is finite

Application Programming Interface

A computer manufacturer needs to build the first four layers of the suite in the operating system and include an API.

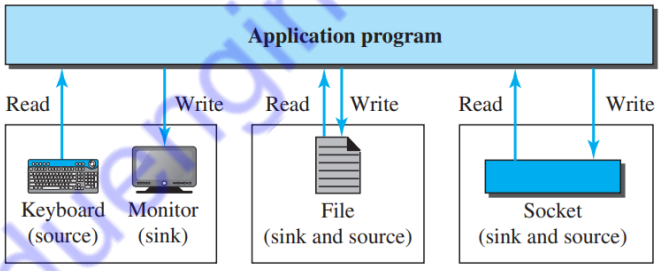
In this way, the processes running at the application layer are able to communicate with the operating system when sending and receiving messages through the Internet.

Several APIs have been designed for communication. Three among them are common: **socket interface, Transport Layer Interface (TLI), and STREAM.**

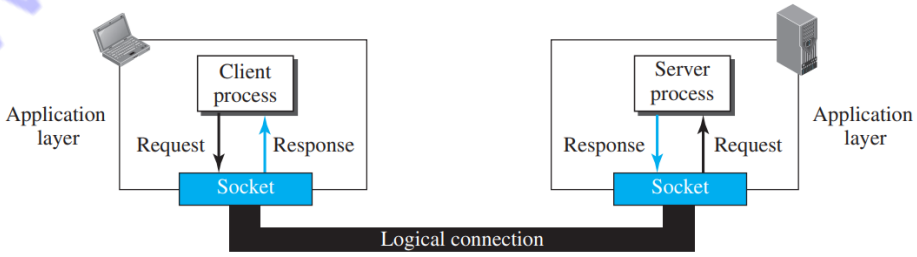


Socket Interface:

- Socket is supposed to behave like a terminal or a file. It is an abstraction.
- It is an object that is created and used by the application program.



- The client thinks that the socket is the entity that receives the request and gives the response; the server thinks that the socket is the one that has a request and needs the response.
- If we create two sockets, one at each end, and define the source and destination addresses correctly, we can use the available instructions to send and receive data



- The interaction between a client and a server is two-way communication.
- In a two-way communication, there are pair of addresses needed: local (sender) and remote (receiver).

- The local address in one direction is the remote address in the other direction and vice versa. Since communication in the client-server paradigm is between two sockets, we need a pair of socket addresses for communication: a local socket address and a remote socket address.

World Wide Web

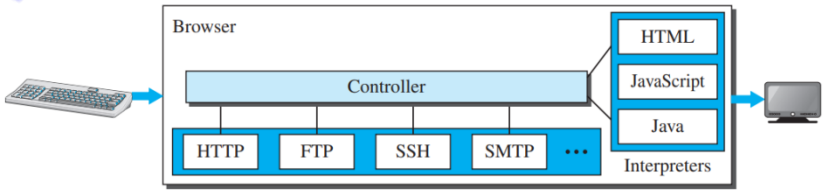
- The idea of the Web was first proposed by Tim Berners-Lee in 1989 at CERN in Europe. The commercial Web started in the early 1990s.
- The Web today is a repository of information in which the documents, called web pages, are distributed all over the world and related documents are linked together.
- The idea was to use a machine that automatically retrieved another document stored in the system when a link to it appeared in the document.
- The Web implemented this idea electronically to allow the linked document to be retrieved when the link was clicked by the user.
- Today, the term hypertext, coined to mean linked text documents, has been changed to hypermedia, to show that a web page can be a text document, an image, an audio file, or a video file.

Architecture:

- The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server.
- However, the service provided is distributed over many locations called sites. Each site holds one or more web pages.
- Each web page, however, can contain some links to other web pages in the same or other sites.
- A web page can be simple or composite. A simple web page has no links to other web pages; a composite web page has one or more links to other web pages. Each web page is a file with a name and address.

Web Client (Browser):

- A variety of vendors offer commercial browsers that interpret and display a web page, and all of them use nearly the same architecture.
- Each browser usually consists of three parts: a controller, client protocols, 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. Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox.

Web Server

- The web page is stored at the server. Each time a 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 can also become more efficient through multithreading or multiprocessing.
- In this case, a server can answer more than one request at a time. Some popular web servers include **Apache and Microsoft Internet Information Server**.

Uniform Resource Locator (URL)

- A web page, as a file, needs to have a unique identifier to distinguish it from other web pages. To define a web page, we need three identifiers: host, port, and path.
- However, before defining the web page, it needs to tell the browser what clientserver application or protocol required for usage

  Protocol. The first identifier is the abbreviation for the client-server program that we need in order to access the web page.

  Host. The host identifier can be the IP address of the server or the unique name given to the server.

  Port. The port, a 16-bit integer, is normally predefined for the client-server application.

  Path. The path identifies the location and the name of the file in the underlying operating system.

  To combine these four pieces together, the uniform resource locator (URL) has been designed

protocol://host/path	Used most of the time
protocol://host:port/path	Used when port number is needed

Web Documents

- The documents in the WWW can be grouped into three broad 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 a copy of the document only.
- When a client accesses the document, a copy of the document is sent. The user can then use a browser to see the document.
- Static documents are prepared using one of several languages: HyperText Markup Language (HTML), Extensible Markup Language (XML), Extensible Style Language (XSL), and Extensible Hypertext Markup Language (XHTML).

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 or a script that creates the dynamic document.
- The server returns the result of the program or script as a response to the browser that requested the document.
- Because a fresh document is created for each request, the contents of a dynamic document may vary from one request to another.
- A very simple example of a dynamic document is the retrieval of the time and date from a server.

- The Common Gateway Interface (CGI) was used to retrieve a dynamic document in the past, some other options included are Java Server Pages (JSP), which uses the Java language for scripting, or Active Server Pages (ASP), a Microsoft product that uses Visual Basic language for scripting, or ColdFusion, which embeds queries in a Structured Query Language (SQL) database in the HTML document.

Active Documents:

- For many applications, we 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. One way to create an active document is to use Java applets, a program written in Java on the server.

Hyper Text Transfer Protocol (HTTP)

- The Hyper Text Transfer Protocol (HTTP) is used to define how the client-server programs can be written to retrieve web pages from the Web.
- An HTTP client sends a request; an HTTP server returns a response.
- The server uses the port number 80; the client uses a temporary port number.
- HTTP uses the services of TCP, a connection-oriented and reliable protocol.
- The client and server do not needs to worry about errors in messages exchanged or loss of any message, because the TCP is reliable and will take care of this matter

The hypertext concept embedded in web page documents may require several requests and responses.

- If the web pages, objects to be retrieved, are located on different servers, we do not have any other choice than to create a new TCP connection for retrieving each object.
- However, if some of the objects are located on the same server, we have two choices: to retrieve each object using a new TCP connection or to make a TCP connection and retrieve them all. The first method is referred to as a nonpersistent connection, the second as a persistent connection.

  Non persistent Connections:

  In a nonpersistent connection, one TCP connection is made for each request/response.

  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; it then closes the connection.

     - If a file contains links to N different pictures in different files the connection must be opened and closed N + 1 times.
     - The nonpersistent strategy imposes high overhead on the server because the server needs N + 1 different buffers each time a connection is opened.

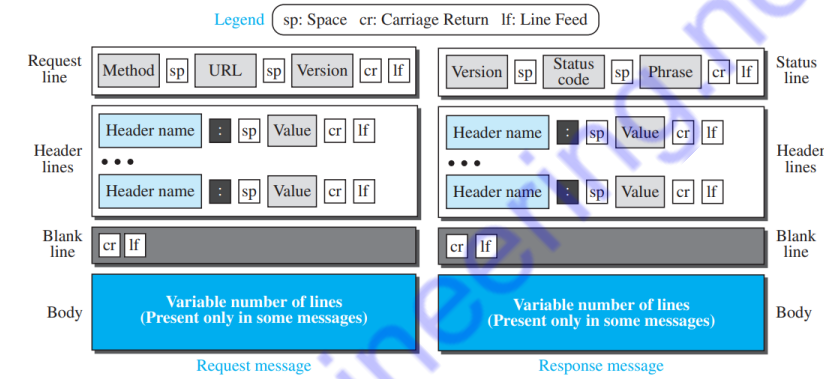
  Persistent Connections

     - In a persistent connection, the server leaves the connection open for more requests after sending a response.
     - The server can close the connection at the request of a client or if a time-out has been reached.
     - The sender usually sends the length of the data with each response. However, there are some occasions when the sender does not know the length of the data.

- In these cases, 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.
- Time and resources are saved using persistent connections.
- Only one set of buffers and variables needs to be set for the connection at each site. The round trip time for connection establishment and connection termination is saved.

Message Formats

- The HTTP protocol defines the format of the request and response messages.
- Each message is made of four sections. The first section in the request message is called the request line; the first section in the response message is called the status line.
- The other three sections have the same names in the request and response messages.

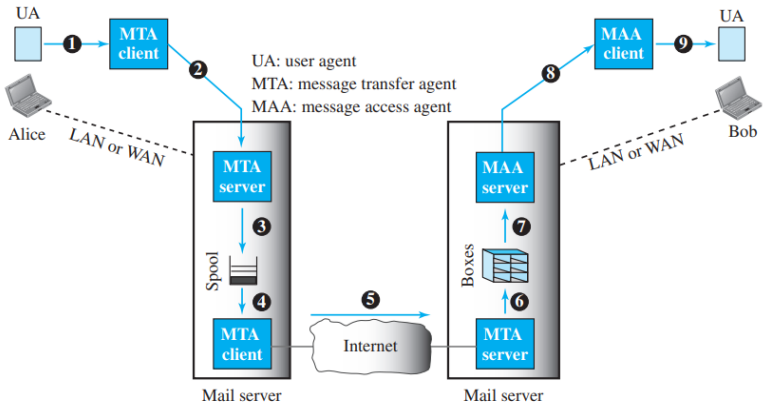


ELECTRONIC MAIL

- Electronic mail (or e-mail) allows users to exchange messages.
- In an application such as HTTP or FTP, the server program is running all the time, waiting for a request from a client.
- When the request arrives, the server provides the service. There is a request and there is a response.
- First, e-mail is considered a one-way transaction. When Alice sends an email to Bob, she may expect a response, but this is not a mandate.
- Bob may or may not respond. If he does respond, it is another one-way transaction. Second, it is neither feasible nor logical for Bob to run a server program and wait until someone sends an e-mail to him. Bob may turn off his computer when he is not using it.

Architecture:

- The sender and the receiver of the e-mail, Alice and Bob respectively, are connected via a LAN or a WAN to two mail servers.
- The administrator has created one mailbox for each user where the received messages are stored.
- A mailbox is part of a server hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. The administrator has also created a queue (spool) to store messages waiting to be sent.
- A simple e-mail from Alice to Bob takes nine different steps. Alice and Bob use three different agents: a user agent (UA), a message transfer agent (MTA), and a message access agent (MAA).



- When Alice needs to send a message to Bob, she runs a UA program to prepare the message and send it to her mail server.
- The mail server at her site uses a queue (spool) to store messages waiting to be sent. The message, however, needs to be sent through the Internet from Alice's site to Bob's site using an MTA.
- Here two message transfer agents are needed: one client and one server. Like 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, on the other hand, can be triggered by the system when there is a message in the queue to be sent.
- The user agent at the Bob site allows Bob to read the received message. Bob later uses an MAA client to retrieve the message from an MAA server running on the second server
- Bob cannot bypass the mail server and use the MTA server directly. To use the MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive
- Second, note that Bob needs another pair of client-server programs: message access programs. This is because an MTA client-server program is a push program: the client pushes the message to the server. Bob needs a pull program

User Agent

- The first component of an electronic mail system is the user agent (UA). It provides service to the user to make the process of sending and receiving a message easier.
- A user agent is a software package (program) that composes, reads, replies to, and forwards messages. It also handles local mailboxes on the user computers. There are two types of user agents: command-driven and GUI-based. Commanddriven user agents belong to the early days of electronic mail.
- They are still present as the underlying user agents. A command-driven user agent normally accepts a onecharacter command from the keyboard to perform its task
- Modern user agents are GUI-based. They contain graphical user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse

Sending Mail

- To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an envelope and a message. The envelope usually contains the sender address, the receiver address, and other information.

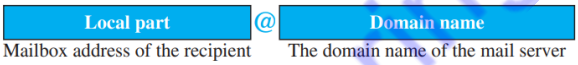
- The message contains the header and the body. The header of the message defines the sender, the receiver, the subject of the message, and some other information.
- The body of the message contains the actual information to be read by the recipient

Receiving Mail

- The user agent is triggered by the user .
- If a user has mail, the UA informs the user with a notice. If the user is ready to read the mail, a list is displayed in which each line contains a summary of the information about a particular message in the mailbox.
- The summary usually includes the sender mail address, the subject, and the time the mail was sent or received.
- The user can select any of the messages and display its contents on the screen.

Addresses

- To deliver mail, a mail handling system must use an addressing system with unique addresses.
- In the Internet, the address consists of two parts: a local part and a domain name, separated by an @ sign



- The local part defines the name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the message access agent.
- The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; they are sometimes called mail servers or exchangers.
- The domain name assigned to each mail exchanger either comes from the DNS database or is a logical name

EDU ENGINEERING

PIONEER OF ENGINEERING NOTES

TAMIL NADU'S BEST EDTECH PLATFORM FOR ENGINEERING

CONNECT WITH US

WEBSITE: www.eduengineering.net

TELEGRAM: @eduengineering

INSTAGRAM: @eduengineering

- **Regular Updates for all Semesters**
- **All Department** Notes AVAILABLE
- **Handwritten** Notes AVAILABLE
- **Past Year** Question Papers AVAILABLE
- **Subject wise** Question Banks AVAILABLE
- **Important Questions** for Semesters AVAILABLE
- Various **Author Books** AVAILABLE