The Transport Layer

Together with the network layer, the transport layer is the heart of the protocol hierarchy. The network layer provides end-to-end packet delivery using datagrams or virtual circuits. The transport layer builds on the network layer to provide data transport from a process on a source machine to a process on a destination machine with a desired level of reliability that is independent of the physical networks currently in use.

Services Provided to the Upper Layers

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services provided by the network layer. The software and/or hardware within the transport layer that does the work is called the transport entity. The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card. The first two options are most common on the Internet. The (logical) relationship of the network, transport, and application layers is illustrated in Fig

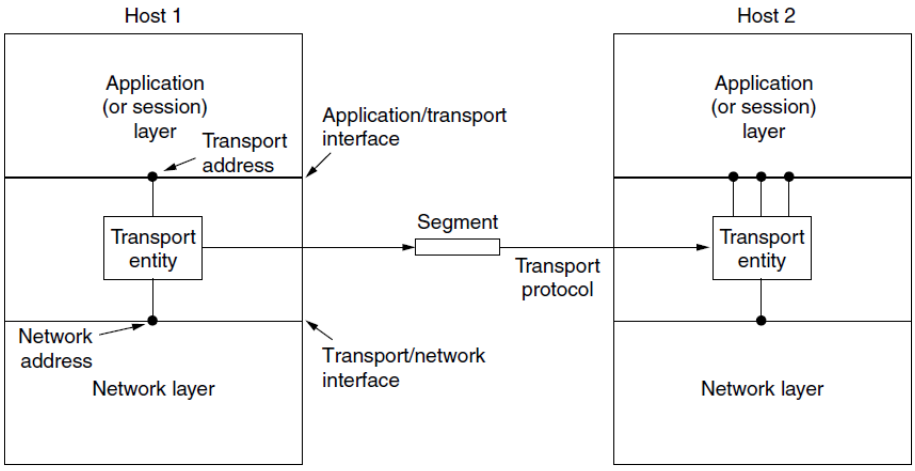


Figure 6-1. The network, transport, and application layers.

There are also two types of transport service. The connection-oriented transport service is similar to the connection-oriented network service in many ways. In both cases, connections have three phases: establishment, data transfer, and release. Addressing and flow control are also similar in both layers.

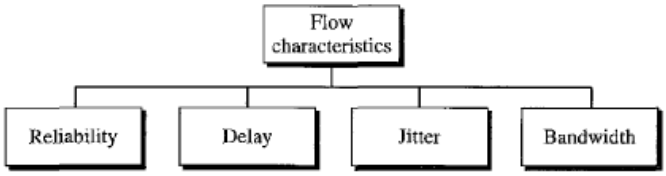
Furthermore, the connectionless transport service is also very similar to the connectionless network service. However, note that it can be difficult to provide a connectionless transport service on top of a connection-oriented network service, since it is inefficient to set up a connection to send a single packet and then tear it down immediately afterwards. The obvious question is this: if the transport layer service is so similar to the network layer service, why are there two distinct layers? Why is one layer not Problems occur, that's what? The users have no real control over the network layer, so they cannot solve the problem of poor service by using better routers or putting more error handling in the data link layer because they don't own the routers. The only possibility is to put on top of the network layer another layer that improves the quality of the service. If, in a connectionless network, packets are lost or mangled, the transport entity can detect the problem and compensate for it by using retransmissions. If, in a connection-oriented network, a transport entity is informed halfway through a long transmission that its network connection has been abruptly terminated, with no indication of what has happened to the data currently in transit, it can set up a new network connection to the remote transport entity. Using this new network connection, it can send a query to its peer asking which data arrived and which did not, and knowing where it was, pick up from where it left off.

QUALITY OF SERVICE

Quality of service (QoS) is an internetworking issue.

Flow Characteristics:

Traditionally, four types of characteristics are attributed to a flow: reliability, delay, jitter, and bandwidth, as shown in Figure.



Reliability

Reliability is a characteristic that a flow needs. Lack of reliability means losing a packet or acknowledgment, which entails retransmission. However, the sensitivity of application programs to reliability is not the same. For example, it is more important that electronic mail, file transfer, and Internet access have reliable transmissions than telephony or audio conferencing.

Delay

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

Jitter

Jitter is the variation in delay for packets belonging to the same flow. For 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. On the other hand, if the above four packets arrive at 21, 23, 21, and 28, they will have different delays: 21,22, 19, and 24.

Bandwidth

Different applications need different bandwidths. In video conferencing we need 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:

Four common methods: 1. Scheduling, 2. Traffic shaping, 3. Admission control, and 4. Resource reservation.

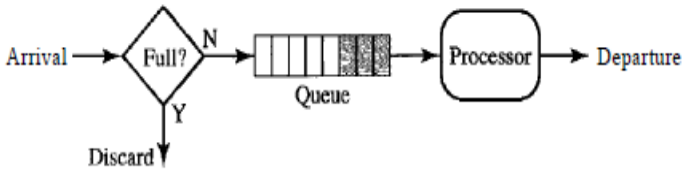
1. Scheduling

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

a) FIFO queuing,
b) Priority Queuing,
c) Weighted Fair Queuing.

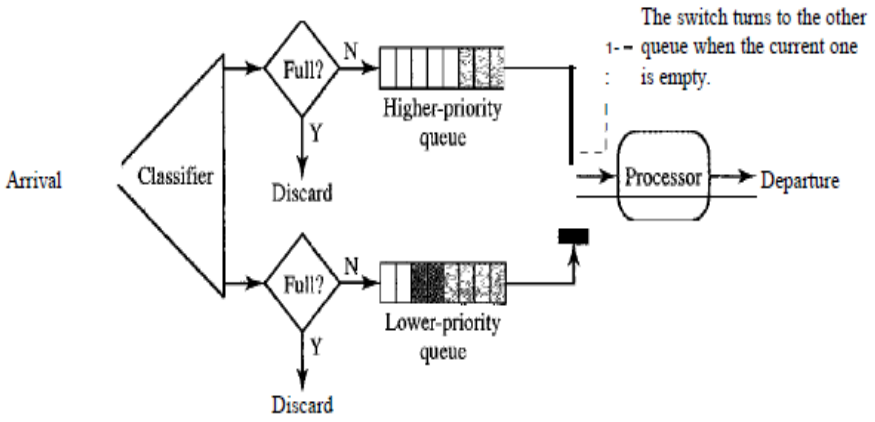
a) FIFO Queuing

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



b) Priority Queuing

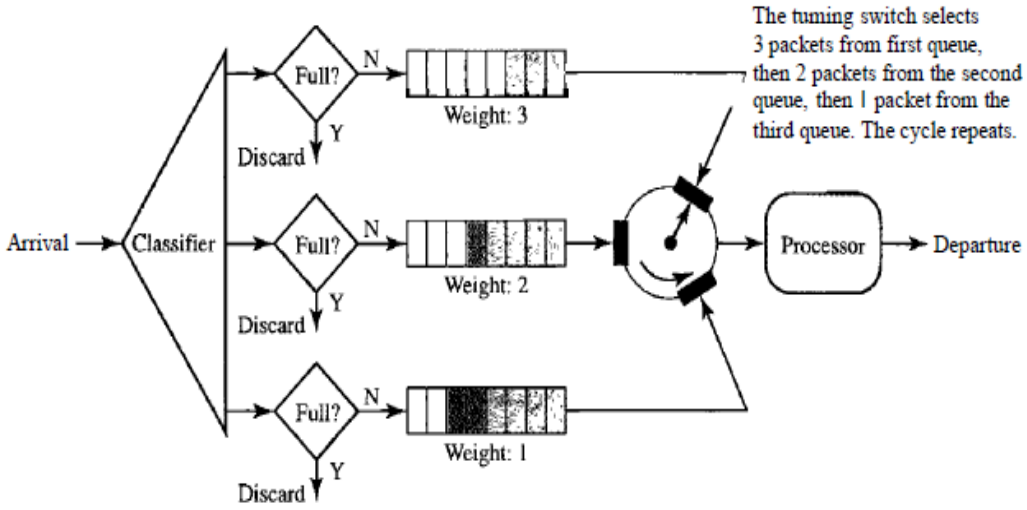
In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue. The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last. Note that the system does not stop serving a queue until it is empty. Figure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



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

c) 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. If the system does not impose priority on the classes, all weights can be equal. In this way, we have fair queuing with priority. Figure shows the technique with three classes.



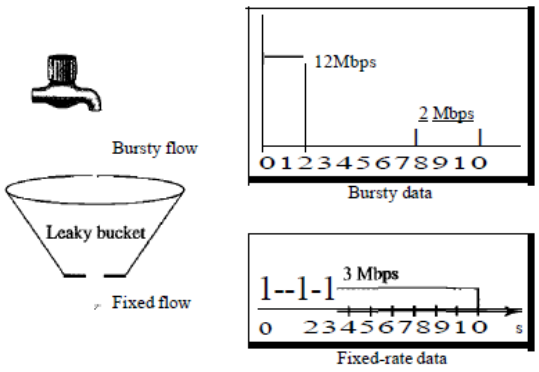
2. Traffic Shaping:

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

a) leaky bucket
b) Token bucket.

a) Leaky Bucket:

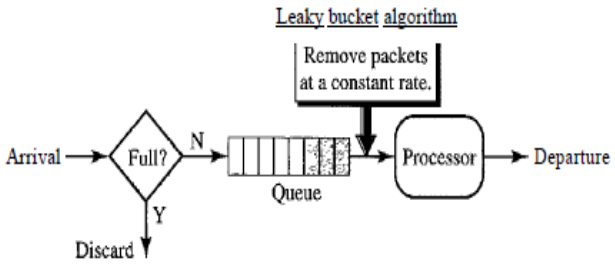
If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the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



In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure 24.19 the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in l0 s. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 s. Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion. As an analogy, consider the freeway during rush hour (bursty traffic). If, instead, commuters could stagger their working hours, congestion o'n our freeways could be avoided.

A simple leaky bucket implementation is shown in Figure 24.20. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), 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 or bits.

Figure 24.20 *Leaky bucket implementation*



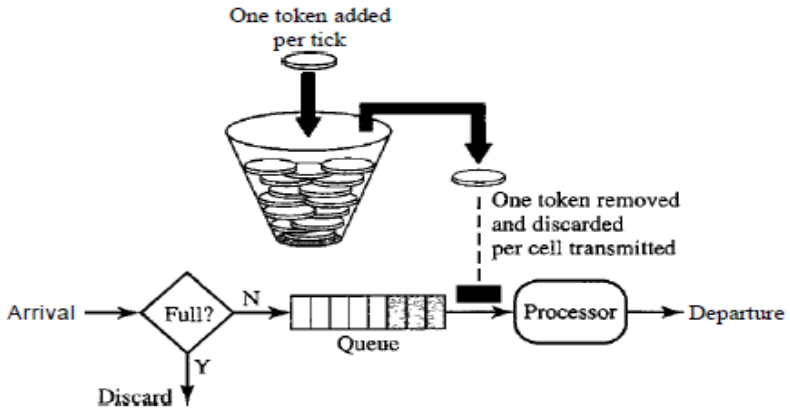
The following is an algorithm for variable-length packets:

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 step 1.

b) Token Bucket:

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

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



Combining Token Bucket and Leaky Bucket:

The two techniques can be combined to credit an idle host and at the same time regulate the traffic. The leaky bucket is applied after the token bucket; the rate of the leaky bucket needs to be higher than the rate of tokens dropped in the bucket.

3. Resource Reservation:

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

4. Admission Control:

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

Elements of Transport Protocols

To establish a reliable service between two machines on a network, transport protocols are implemented, which somehow resembles the data link protocols implemented at layer 2. The major difference lies in the fact that the data link layer uses a physical channel between two routers while the transport layer uses a subnet.

Following are the issues for implementing transport protocols −

Types of Service

The transport layer also determines the type of service provided to the users from the session layer. An error-free point-to-point communication to deliver messages in the order in which they were transmitted is one of the key functions of the transport layer.

Error Control

Error detection and error recovery are an integral part of reliable service, and therefore they are necessary to perform error control mechanisms on an end-to-end basis. To control errors from lost or duplicate segments, the transport layer enables unique segment sequence numbers to the different packets of the message, creating virtual circuits, allowing only one virtual circuit per session.

Flow Control

The underlying rule of flow control is to maintain a synergy between a fast process and a slow process. The transport layer enables a fast process to keep pace with a slow one. Acknowledgements are sent back to manage end-to-end flow control. Go back N algorithms are used to request retransmission of packets starting with packet number N. Selective Repeat is used to request specific packets to be retransmitted.

Connection Establishment/Release

The transport layer creates and releases the connection across the network. This includes a naming mechanism so that a process on one machine can indicate with whom it wishes to communicate. The transport layer enables us to establish and delete connections across the network to multiplex several message streams onto one communication channel.

Multiplexing/De multiplexing

The transport layer establishes a separate network connection for each transport connection required by the session layer. To improve throughput, the transport layer establishes multiple network connections. When the issue of throughput is not important, it multiplexes several transport connections onto the same network connection, thus reducing the cost of establishing and maintaining the network connections.

When several connections are multiplexed, they call for demultiplexing at the receiving end. In the case of the transport layer, the communication takes place only between two processes and not between two machines. Hence, communication at the transport layer is also known as peer-to-peer or process-to-process communication.

Fragmentation and re-assembly

When the transport layer receives a large message from the session layer, it breaks the message into smaller units depending upon the requirement. This process is called fragmentation. Thereafter, it is passed to the network layer. Conversely, when the transport layer acts as the receiving process, it reorders the pieces of a message before reassembling them into a message.

Addressing

Transport Layer deals with addressing or labelling a frame. It also differentiates between a connection and a transaction. Connection identifiers are ports or sockets that label each frame, so the receiving device knows which process it has been sent from. This helps in keeping track of multiple-message conversations. Ports or sockets address multiple conservations in the same location.

Connection Establishment

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, delay, corrupt, and duplicate packets. This behaviour causes serious complications. Imagine a network that is so congested that acknowledgements hardly ever get back in time and each packet times out and is retransmitted two or three times. Suppose that the network uses datagrams inside and that every packet follows a different route. Some of the packets might get stuck in a traffic jam inside the network and take a long time to arrive. That is, they may be delayed in the network and pop out much later, when the sender thought that they had been lost. The worst possible nightmare is as follows.

A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person. Unfortunately, the packets decide to take the scenic route to the destination and go off exploring a remote corner of the network. The sender then times out and sends them all again. This time the packets take the shortest route and are delivered quickly so the sender releases the connection.

Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.
2. Putting a hop counter in each packet.
3. Time stamping each packet.

The first technique includes any method that prevents packets from looping, combined with some way of bounding delay including congestion over the (now known) longest possible path. It is difficult, given that internets may range from a single city to international in scope.

The second method consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded. The network protocol simply discards any packet whose hop counter becomes zero.

The third method requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time. This latter method requires the router clocks to be synchronized, which itself is a nontrivial task, and in practice a hop counter is a close enough approximation to age.

TCP uses this three-way handshake to establish connections. Within a connection, a timestamp is used to extend the 32-bit sequence number so that it will not wrap within the maximum packet lifetime, even for gigabit-per-second connections. This mechanism is a fix to TCP that was needed as it was used on faster and faster links. It is described in RFC 1323 and called **PAWS** (**Protection Against Wrapped Sequence numbers**). Across connections, for the initial sequence numbers and before PAWS can come into play, TCP originally use the clock-based scheme just described. However, this turned out to have security vulnerability. The clock made it easy for an attacker to predict the next initial sequence number and send packets that tricked the three-way handshake and established a forged connection. To close this hole, pseudorandom initial sequence numbers are used for connections in practice.

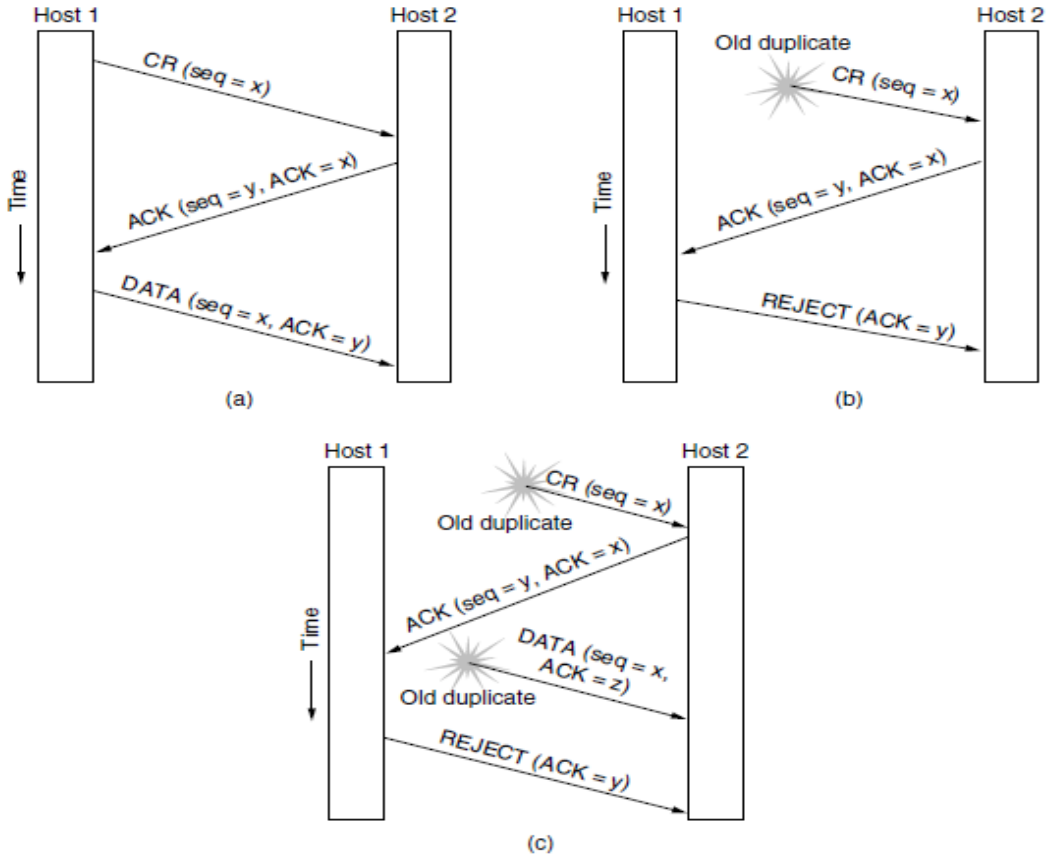


Figure 6-11. Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST. (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK.

Initial sequence numbers not repeat for an interval even though they appear random to an observer. Otherwise, delayed duplicates can wreak havoc.

Connection Release

Releasing a connection is easier than establishing one. Nevertheless, there are more pitfalls than one might expect here. As we mentioned earlier, there are two styles of terminating a connection: asymmetric release and symmetric release Asymmetric release is the way the telephone system works: when one party hangs up, the connection is broken. Symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately.

Asymmetric release is abrupt and may result in data loss. Consider the scenario of Fig. After the connection is established, host 1 sends a segment that arrives properly at host 2. Then host 1 sends another segment. Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.

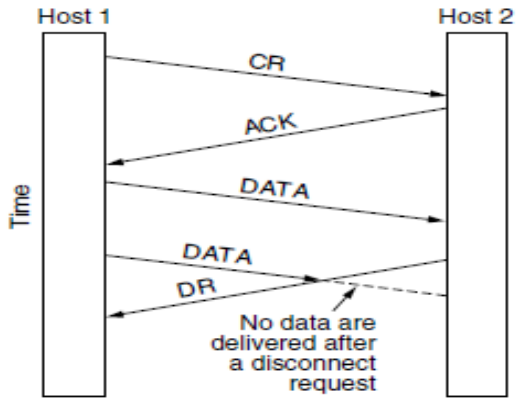


Figure 6-12. Abrupt disconnection with loss of data.

UDP Protocol

UDP provides connectionless, unreliable, datagram service. Connectionless service means that there is no logical connection between the two ends exchanging messages. Each message is an independent entity encapsulated in a datagram.

UDP does not see any relation (connection) between consequent datagram coming from the same source and going to the same destination.

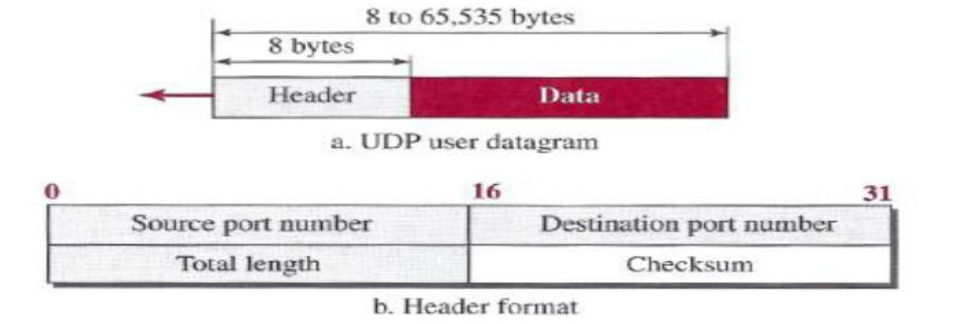
UDP has an advantage: it is message-oriented. It gives boundaries to the messages exchanged. An application program may be designed to use UDP if it is sending small messages and the simplicity and speed is more important for the application than reliability.

User Datagram

UDP packets, called *user datagram,* have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).

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.

User datagram packet format



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

As mentioned previously, UDP provides a connection less service. This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user data grams even if they are coming from the same source process and going to the same destination program.

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.

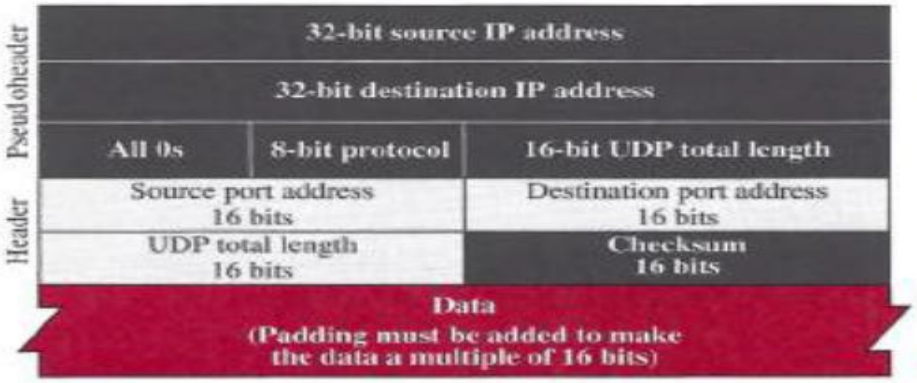
Error Control

There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated.

Checksum

UDP checksum calculation includes three sections: a pseudo header, the UDP header, and the data coming from the application layer. The pseudo header 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.

Pseudoheader for checksum calculation



UDP Features

Connectionless Service
As we mentioned previously,

- UDP is a connectionless protocol. Each UDP packet is independent from other packets sent by the same application program. This feature can be considered as an advantage or disadvantage depending on the application requirements.
- UDP does not provide error control; it provides an unreliable service. Most applications expect reliable service from a transport-layer protocol. Although a reliable service is desirable.

Typical Applications
The following shows some typical applications that can benefit more from the services of UDP

- 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. For example, the Trivial File Transfer Protocol (TFTP)
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software
- 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

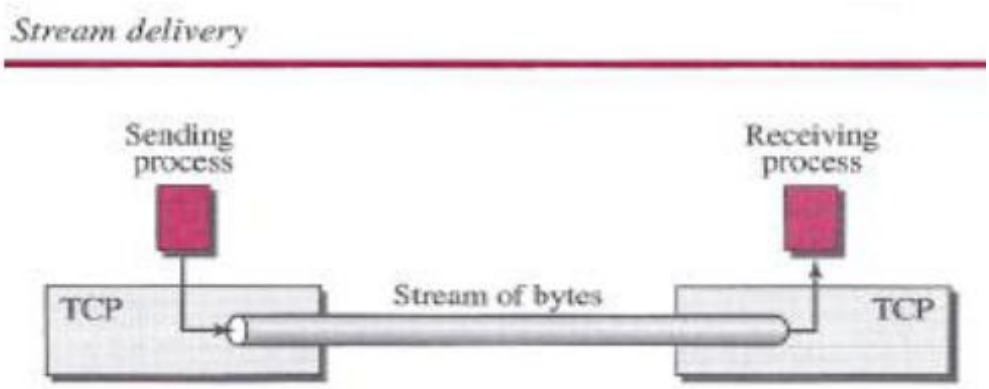
Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol. TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.

TCP Services

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

Stream Delivery Service
In UDP, a process sends messages with predefined boundaries to UDP for delivery. UDP adds its own header to each of these messages and delivers it to IP for transmission.

TCP, on the other hand, 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.

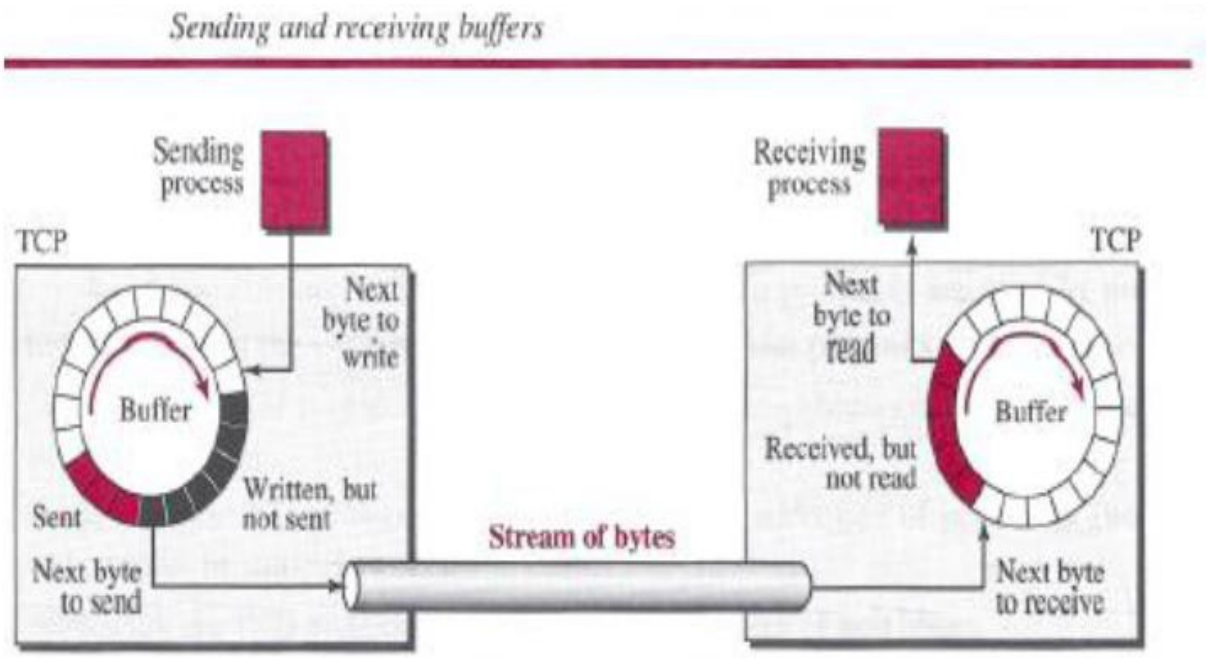


Sending and Receiving Buffers

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. The shaded area contains bytes to be sent by the sending TCP.
- The operation of the buffer at the receiver is simpler. The circular buffer is divided into two areas (shown as white and colored).
- 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. When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

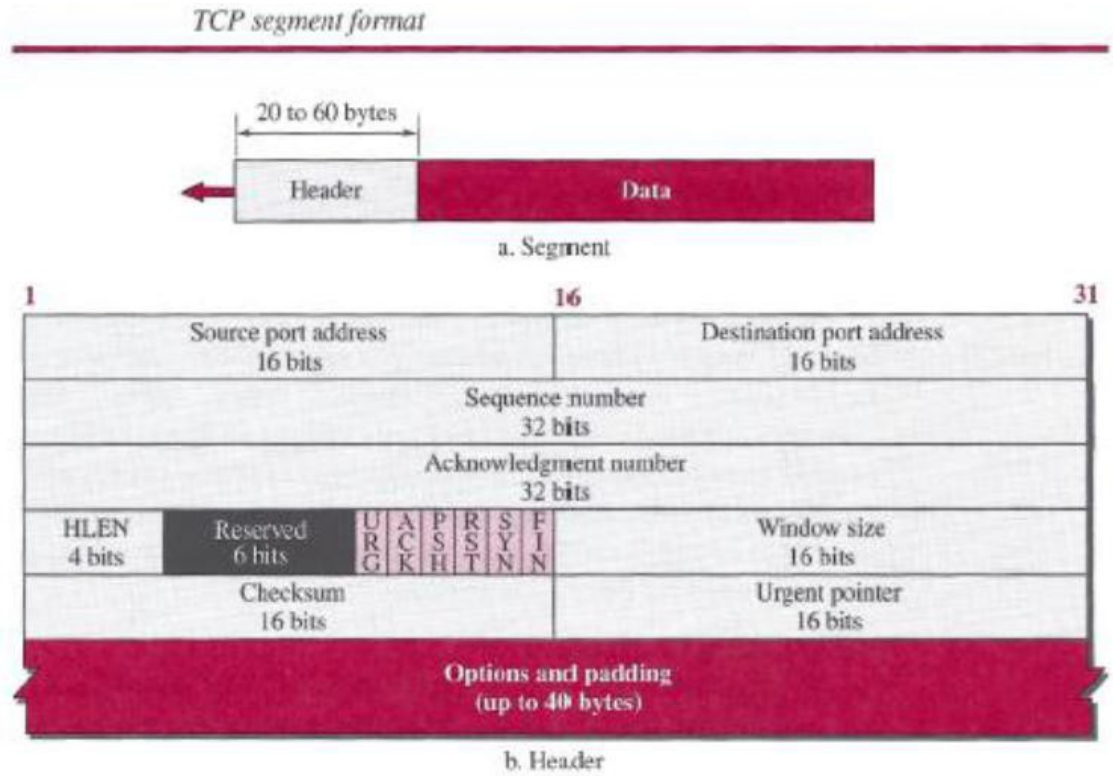


Segments

- Although buffering handles the disparity between the speed of the producing and consuming Processes, we need one more step before we can send data.
- The network layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a *segment.*
- The segments are encapsulated in an IP datagram and transmitted. This entire operation is transparent to the receiving process.

Format

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.

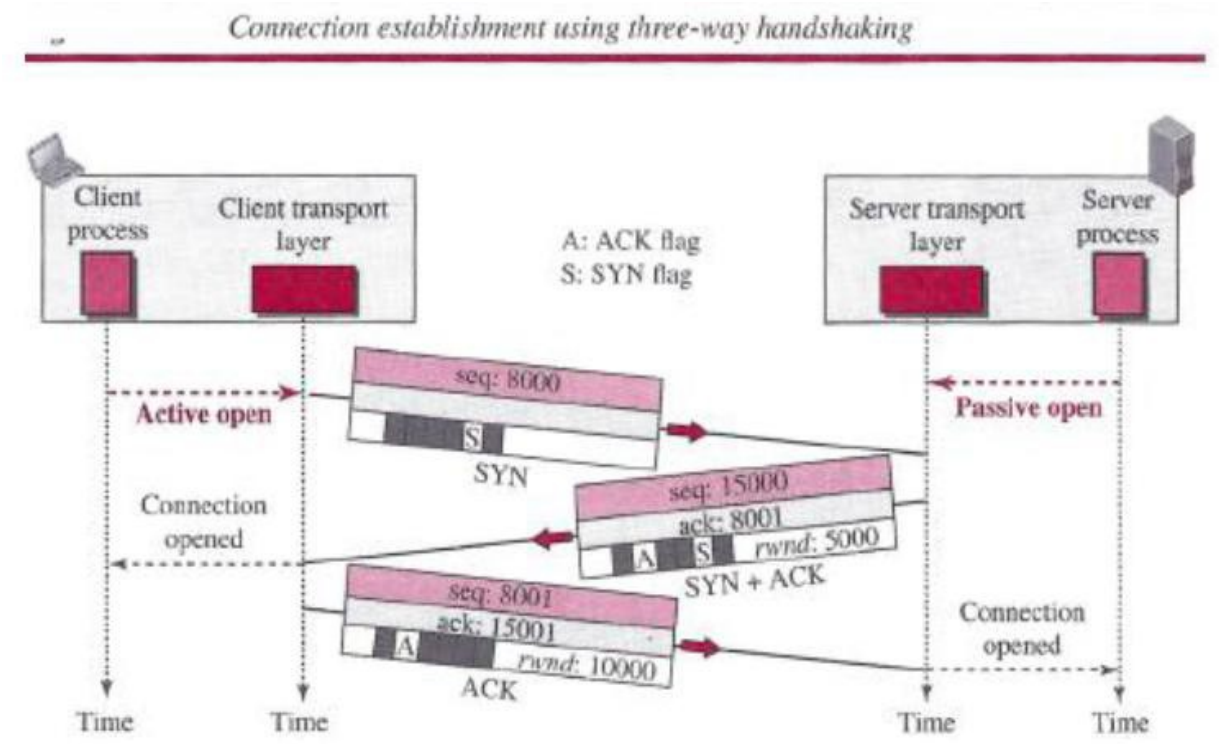
Sequence number This 32-bit field defines the number assigned to the first byte of data contained in this segment.

Acknowledgment number This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party.

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.

A 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.
- 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.
- 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.

Three- Way Handshaking

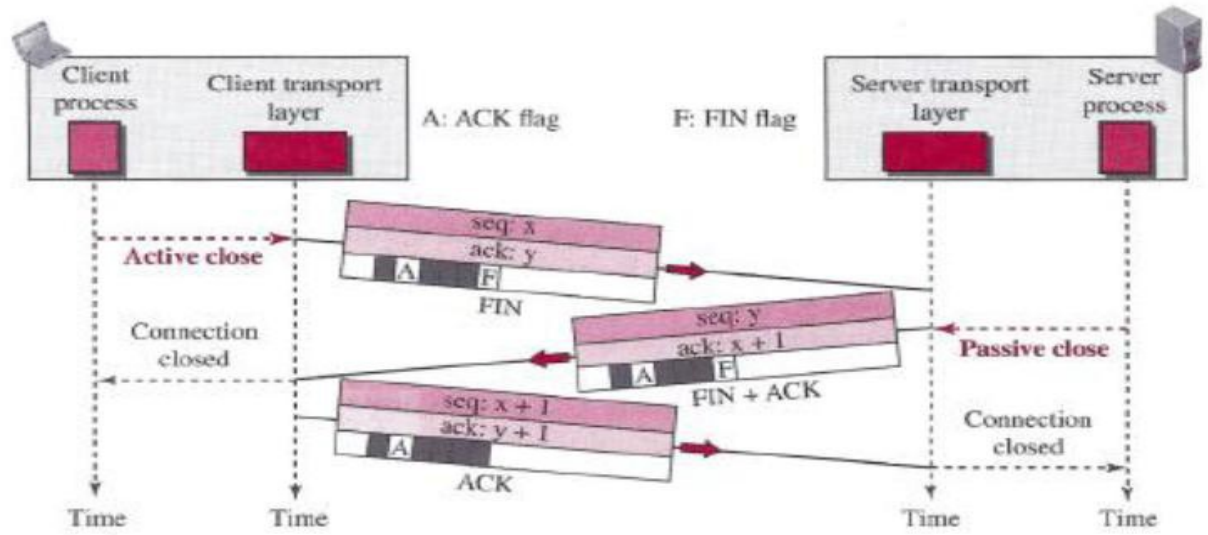
The connection establishment in TCP is called *three-way handshaking.* an application program, called the *client,* wants to make a connection with another application program, called the *server,* using TCP as the transport-layer protocol The process starts with the server. The server program tells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

Although the server TCP is ready to accept a connection from any machine in the world, it cannot make the connection itself.

The client program issues a request for an *active open.* A client that wishes to connect to an open server tells its TCP to connect to a particular server.

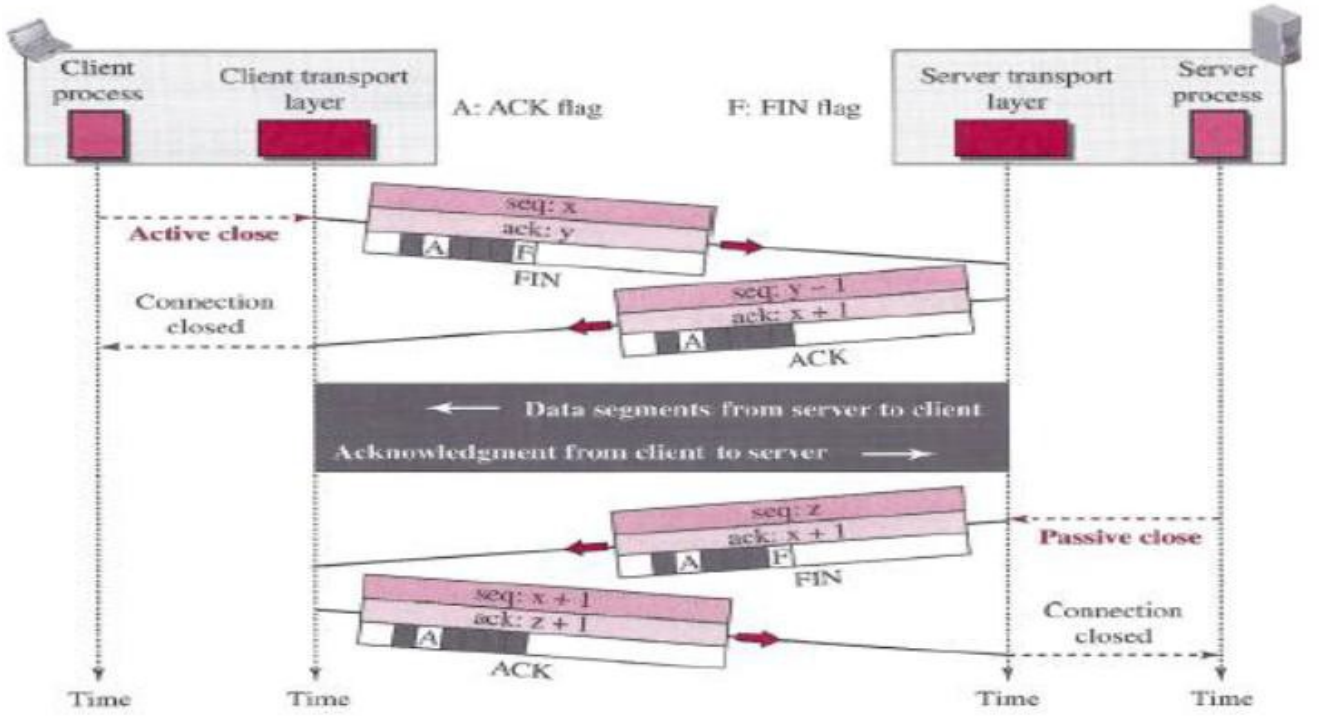
- A SYN segment cannot carry data, but it consumes one sequence number.
- A SYN + ACK segment cannot carry data, but it does consume one sequence number.
- An ACK segment, if carrying no data, consumes no sequence number.

Connection termination using three-way handshaking



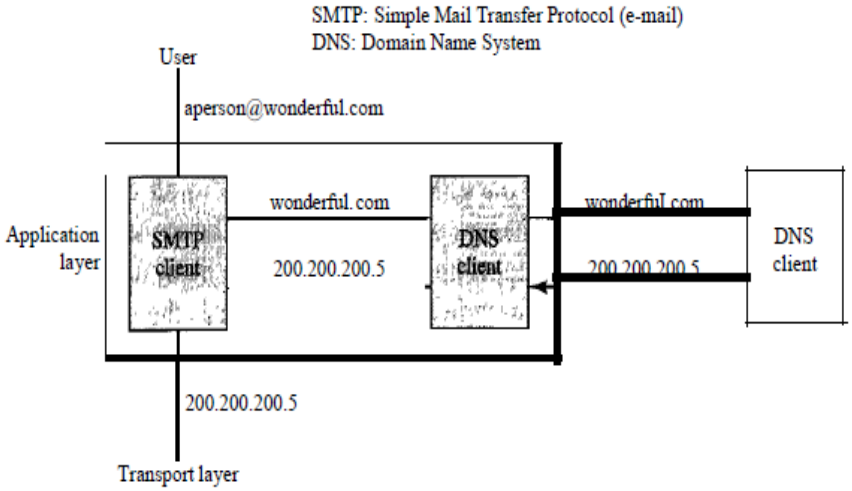
The FIN + ACK segment consumes only one sequence number if it does not carry data.

Half-close



Domain Name System (DNS)

- The DNS is a distributed database that resides on multiple machines on the internet. It provide e-mail routing information.
- The DNS protocol runs over UDP and uses port 53.
- Figure (below) shows how TCP/IP uses a DNS client and a DNS server to map a name to an address.



- A user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server name, such as afilesource.com.
- The TCP/IP suite needs the IP address of the file transfer server to make the connection.
- The following six steps map the host name to an IP address:
  1. The user passes the host name to the file transfer client.
  2. The file transfer client passes the host name to the DNS client.
  3. Each computer, after being booted, knows the address of one DNS server. The DNS client sends a message to a DNS server with a query that gives the file transfer server name using the known IP address of the DNS server.
  4. The DNS server responds with the IP address of the desired file transfer server.
  5. The DNS server passes the IP address to the file transfer client.
  6. The file transfer client now uses the received IP address to access the file transfer server.

Domain Name Space

- To have a hierarchical name space, a **domain name space** was designed.
- In this design the names are defined in an inverted-tree structure with the root at the top.
- The tree has 128 levels: level 0 (root) to level 127 (see Figure).

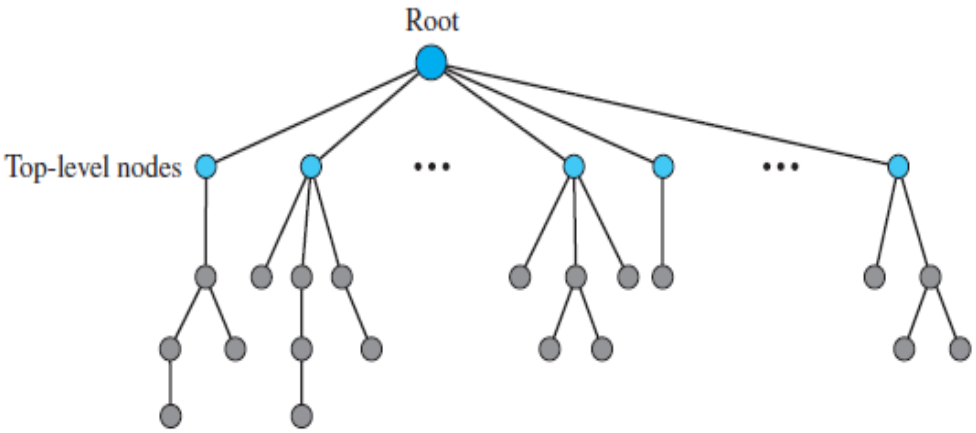


Fig: DNS tree.

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).

Domain Name

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

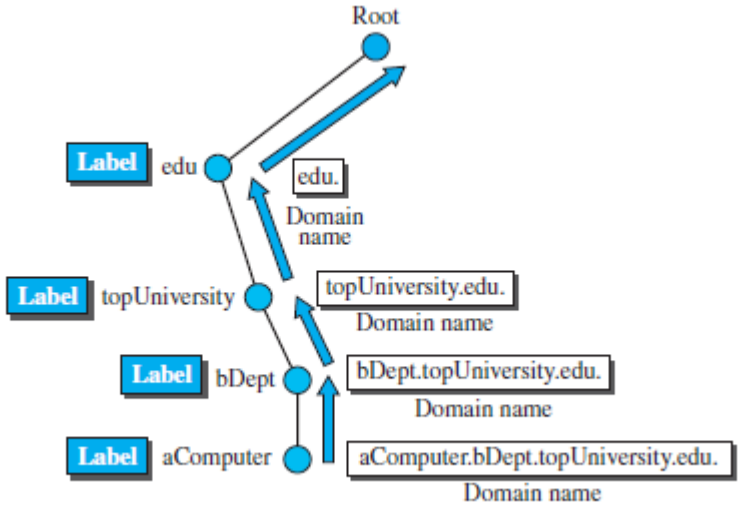


Fig: Domain names.

- If a label is terminated by a null string, it is called a **fully qualified domain name (FQDN)**. The name must end with a null label, but because null means nothing, the label ends with a dot.

- If a label is not terminated by a null string, it is called a **partially qualified domain name (PQDN).** A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client.
- Here the resolver can supply the missing part, called the suffix, to create an FQDN.

Domain

- A **domain** is a subtree of the domain name space. The name of the domain is the name of the node at the top of the subtree. Figure (below) shows some domains. A domain is divided into domains.

Distribution of Name Space

- The information contained in the domain name space must be stored.
- It is inefficient and not reliable to have just one computer store such a huge amount of information. It is inefficient because responding to requests from all over the world places a heavy load on the system.

Hierarchy of Name Servers

- To distribute the information among many computers called **DNS servers.** One way to do this is to divide the whole space into many domains based on the first level.
- DNS allows domains to be divided further into smaller domains (subdomains). Each server can be responsible (authoritative) for either a large or small domain.

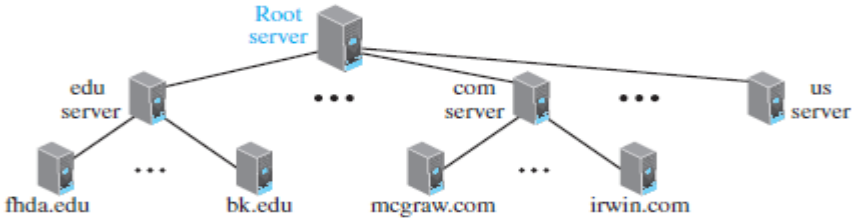


Fig: Domain name hierarchy

Zone

- If a server accepts responsibility for a domain and does not divide the domain into smallerdomains, the "domain" and the "zone" refer to the same thing. The server makes a databasecalled a zone file and keeps all the information for every node under that domain.

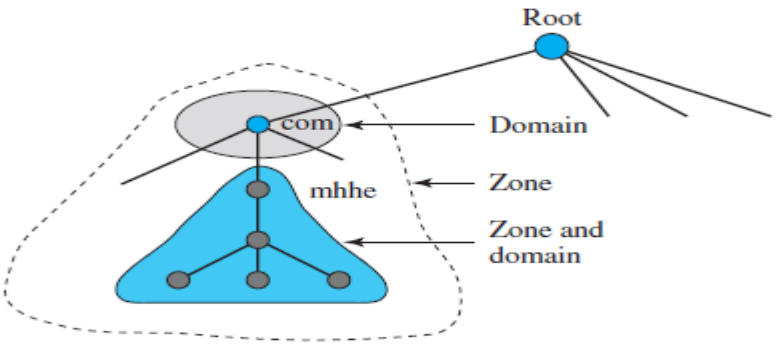


Fig: Zone

- The information about the nodes in the subdomains is stored in the servers at the lower levels, with the original server keeping some sort of reference to these lower-level servers.

Root Server

- If zone consists of the full tree then that zone server is called root server.
- A root server does not store any information about domains .
- 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.
- A secondary server is a server that loads all information from the primary server. Secondary server cannot perform any operation on zone file.

DNS in the Internet

- In the Internet, the domain name space (tree) was divided into three sections: generic domains, country domains, and the inverse domains.

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.

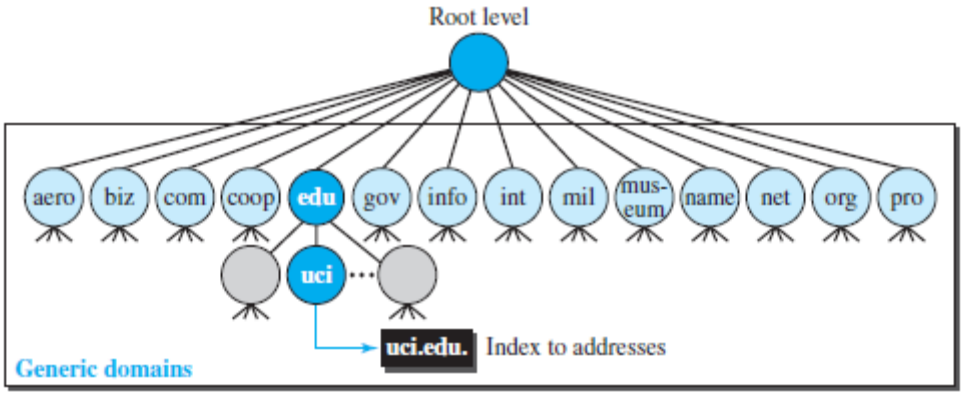


Fig: Generic domain

- Looking at the tree, the first level in the **generic domains** section allows 14 possible labels.

Country Domains

- The **country domains** section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific national designations.
- **Example**for the country domains section.
- The address **uci.ca.us.** can be translated to University of California, Irvine, in the state of California in the United States.

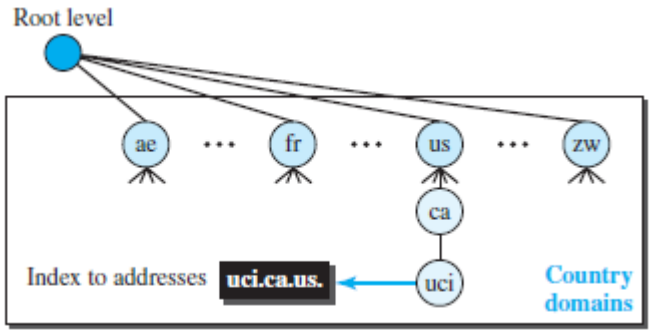


Fig: Country domain

Inverse domain

- Inverse domain is used to find the name of a host when given the IP address.

Resolution

- Mapping a name to an address is called name-address resolution.
- DNS is designed as a client-server application. A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver.
- The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the information.

Recursive Resolution

- A client request complete translation.
- If the server is authority for the domain name, it checks its database and responds.
- If the server is not authority, it sends the request to another server and waits for the response.
- When the query is resolved ,the response travel back until it finally reaches the requesting client. This is called recursive resolution.

DNS Messages

- To get information about hosts, DNS uses two types of messages: query and response. Both messages have the same format as shown in Figure.

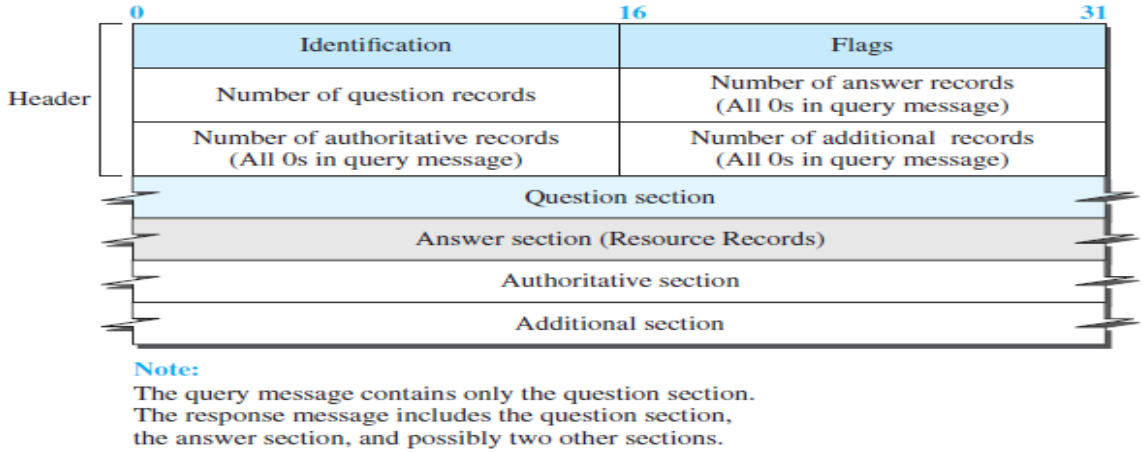


Fig: DNS messages

- The **identification field** is used by the client to match the response with the query.
- The **flag field** defines whether the message is a query or response. It also includes status of error.
- The **next four fields** in the header define the number of each record type in the message.
- The **question section** consists of one or more question records. It is present in both query and response messages.
- The **answer section** consists of one or more resource records. It is present only in response messages.
- The **authoritative section** gives information (domain name) about one or more authoritative servers for the query.
- The **additional information section** provides additional information that may help the resolver.

Encapsulation

- DNS can use either UDP or TCP. The port used by the server is port 53.
- UDP is used when the size of the response message is less than 512 bytes because most UDP packages have a 512-byte packet size limit.
- If the size of the response message is more than 512 bytes, a TCP connection is used.

Security of DNS

- Applications such as Web access or e-mail are dependent on the proper operation of DNS. DNS can be attacked in several ways :
    - The attacker may read the response of a DNS server to find the nature or names of sites the user mostly accesses. This type of information can be used to find the user's profile.
    - To prevent this attack, DNS messages need to be confidential.

TELNET

- The main task of the internet is to provide services to users. For example, users want to run different application programs at the remote site and transfers a result to the local site. This requires a client-server program such as FTP, SMTP. But this would not allow us to create a specific program for each demand.

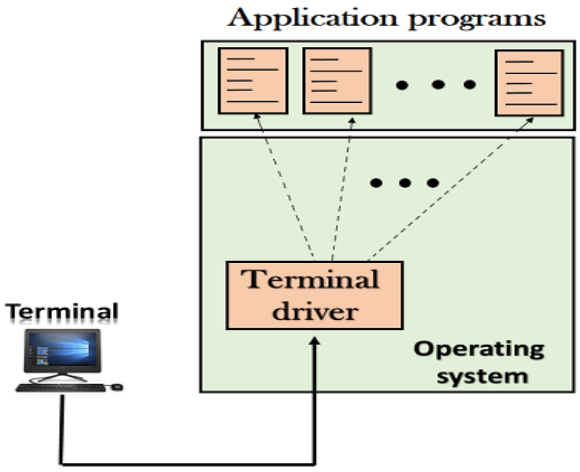
- The better solution is to provide a general client-server program that lets the user access any application program on a remote computer. Therefore, a program that allows a user to log on to a remote computer. A popular client-server program Telnet is used to meet such demands. Telnet is an abbreviation for **Terminal Network**.

- Telnet provides a connection to the remote computer in such a way that a local terminal appears to be at the remote side.

There are two types of login:

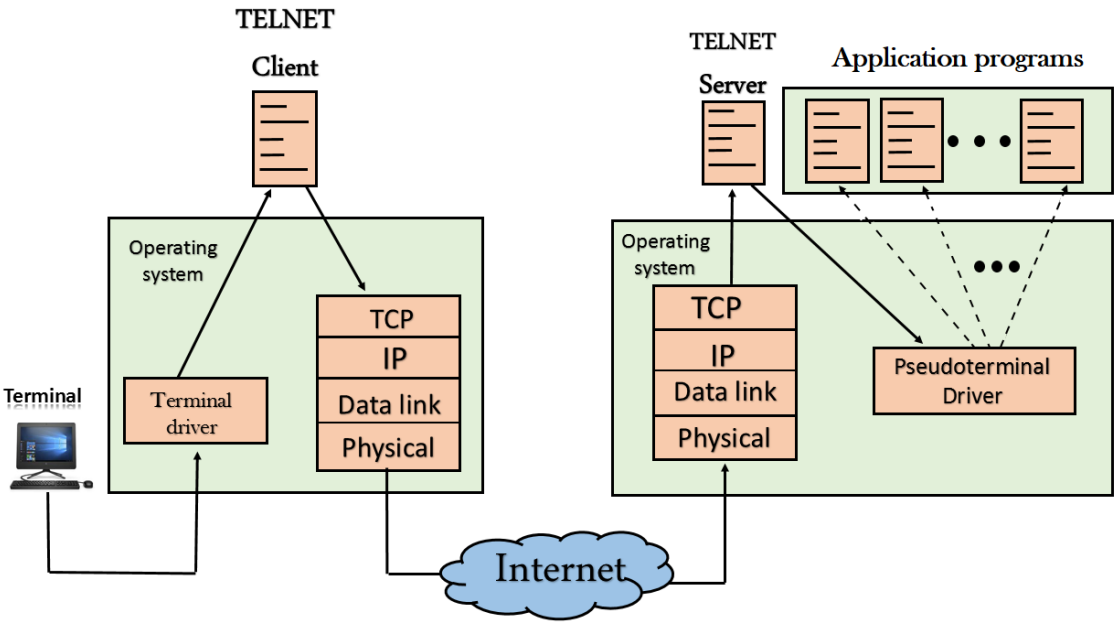
1. Local Login
2. Remote login

Local Login:



- When a user logs into a local computer, then it is known as local login.
- When the workstation running terminal emulator, the keystrokes entered by the user are accepted by the terminal driver. The terminal driver then passes these characters to the operating system which in turn, invokes the desired application program.
- However, the operating system has special meaning to special characters. For example, in UNIX some combination of characters have special meanings such as control character with "z" means suspend. Such situations do not create any problem as the terminal driver knows the meaning of such characters. But, it can cause the problems in remote login.

Remote login:



- When the user wants to access an application program on a remote computer, then the user must perform remote login.

How remote login occurs:

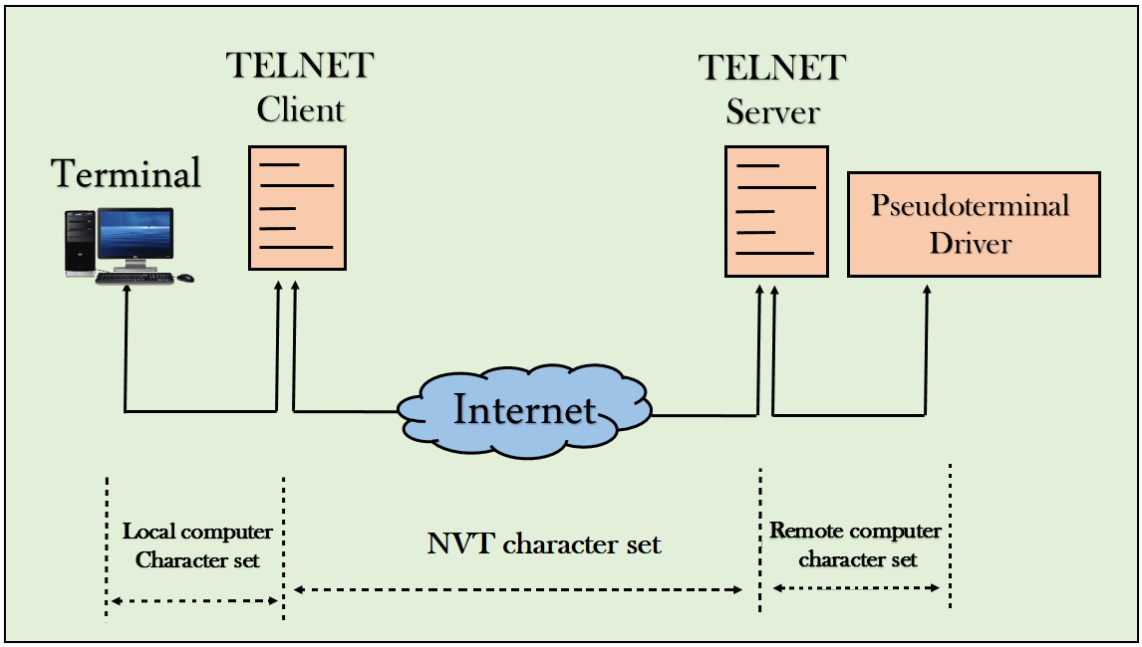
At the local site

The user sends the keystrokes to the terminal driver, the characters are then sent to the TELNET client. The TELNET client which in turn, transforms the characters to a universal character set known as network virtual terminal characters and delivers them to the local TCP/IP stack.

At the remote site

The commands in NVT forms are transmitted to the TCP/IP at the remote machine. Here, the characters are delivered to the operating system and then pass to the TELNET server. The TELNET server transforms the characters which can be understandable by a remote computer. However, the characters cannot be directly passed to the operating system as a remote operating system does not receive the characters from the TELNET server. Therefore it requires some piece of software that can accept the characters from the TELNET server. The operating system then passes these characters to the appropriate application program.

Network Virtual Terminal (NVT)



- The network virtual terminal is an interface that defines how data and commands are sent across the network.
- In today's world, systems are heterogeneous. For example, the operating system accepts a special combination of characters such as end-of-file token running a DOS operating system *ctrl+z* while the token running a UNIX operating system is *ctrl+d*.
- TELNET solves this issue by defining a universal interface known as network virtual interface.
- The TELNET client translates the characters that come from the local terminal into NVT form and then delivers them to the network. The Telnet server then translates the data from NVT form into a form which can be understandable by a remote computer.

ELECTRONIC MAIL

- Electronic mail (or e-mail) allows users to exchange messages. Electronic mail is used for sending a single message like text, voice, video or grapics to one or more recipients.
- Email is fast, easy to distribute and inexpensive.
- The sender and receiver use three different *agents*: a **user agent (UA),** a **message transfer agent (MTA),** and a **message access agent (MAA).**
- The mail server uses a queue (spool) to store messages waiting to be sent. The message, needs to be sent through the Internet from client to server site using an MTA.
- Here two message transfer agents are needed: one client and one server.

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.
- A command-driven user agent normally accepts a one character command from the keyboard to perform its task.

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 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 (or a timer). 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.

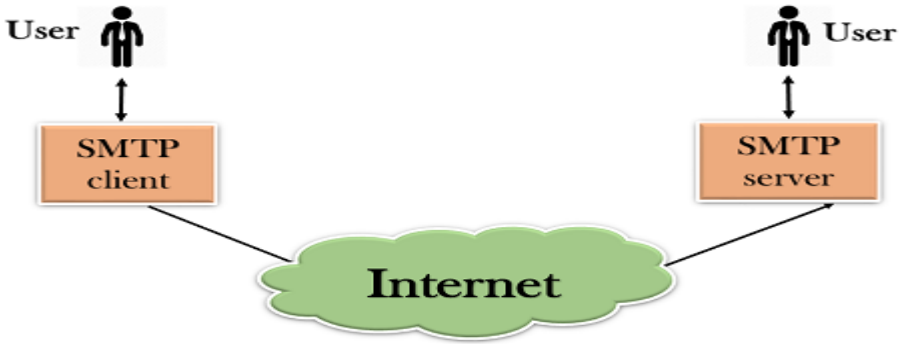
Addresses

- 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.

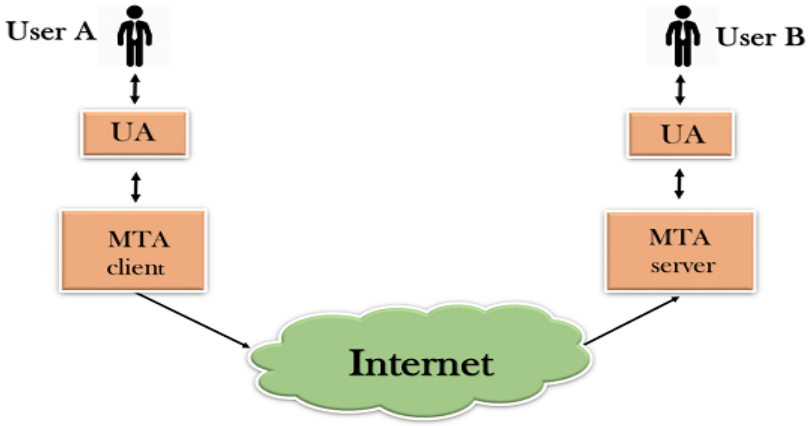
SMTP

- SMTP stands for Simple Mail Transfer Protocol.
- SMTP is a set of communication guidelines that allow software to transmit an electronic mail over the internet is called **Simple Mail Transfer Protocol**.
- It is a program used for sending messages to other computer users based on e-mail addresses.
- It provides a mail exchange between users on the same or different computers, and it also supports:
    - It can send a single message to one or more recipients.
    - Sending message can include text, voice, video or graphics.
    - It can also send the messages on networks outside the internet.
- The main purpose of SMTP is used to set up communication rules between servers. The servers have a way of identifying themselves and announcing what kind of communication they are trying to perform. They also have a way of handling the errors such as incorrect email address. For example, if the recipient address is wrong, then receiving server reply with an error message of some kind.

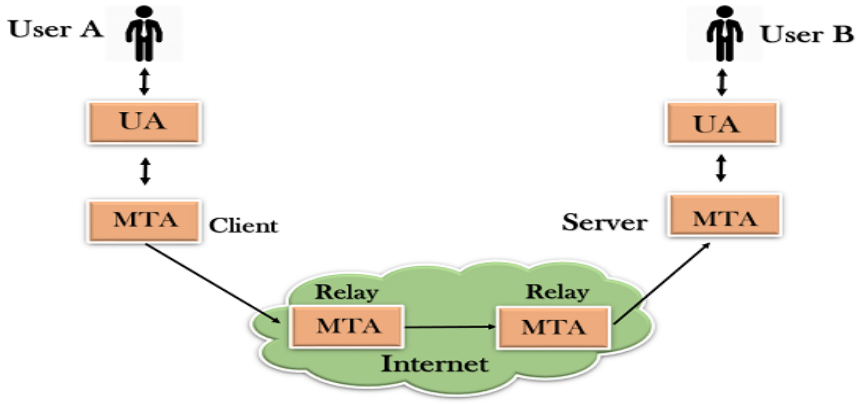
Components of SMTP



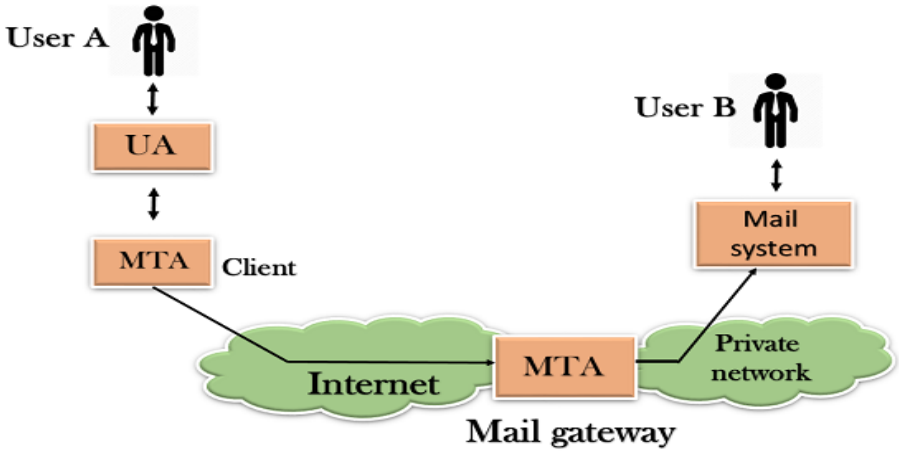
- First, we will break the SMTP client and SMTP server into two components such as user agent (UA) and mail transfer agent (MTA). The user agent (UA) prepares the message, creates the envelope and then puts the message in the envelope. The mail transfer agent (MTA) transfers this mail across the internet.



- SMTP allows a more complex system by adding a relaying system. Instead of just having one MTA at sending side and one at receiving side, more MTAs can be added, acting either as a client or server to relay the email.



- The relaying system without TCP/IP protocol can also be used to send the emails to users, and this is achieved by the use of the mail gateway. The mail gateway is a relay MTA that can be used to receive an email.



Commands and Responses

- SMTP uses commands and responses to transfer messages between an MTA client and an MTA server. The command is from an MTA client to an MTA server; the response is from an MTA server to the MTA client.

Commands

- Commands are sent from the client to the server.
- **Keyword:** argument(s)
- SMTP defines 14 commands.

S.No.	Keyword	Command form	Description	Usage
1.	HELO	HELO<SP><domain><CRLF>	It provides the identification of the sender i.e. the host name.	Mandatory
2.	MAIL	MAIL<SP>FROM : <reverse-path><CRLF>	It specifies the originator of the mail.	Mandatory
3.	RCPT	RCPT<SP>TO : <forward-path><CRLF>	It specifies the recipient of mail.	Mandatory
4.	DATA	DATA<CRLF>	It specifies the beginning of the mail.	Mandatory
5.	QUIT	QUIT<CRLF>	It closes the TCP connection.	Mandatory
6.	RSET	RSET<CRLF>	It aborts the current mail transaction but the TCP connection remains open.	Highly recommended
7.	VRFY	VRFY<SP><string><CRLF>	It is use to confirm or verify the user name.	Highly recommended
8.	NOOP	NOOP<CRLF>	No operation	Highly recommended
9.	TURN	TURN<CRLF>	It reverses the role of sender and receiver.	Seldom used
10.	EXPN	EXPN<SP><string><CRLF>	It specifies the mailing list to be expanded.	Seldom used
11.	HELP	HELP<SP><string><CRLF>	It send some specific documentation to the system.	Seldom used
12.	SEND	SEND<SP>FROM : <reverse-path><CRLF>	It send mail to the terminal.	Seldom used
13.	SOML	SOML<SP>FROM : <reverse-path><CRLF>	It send mail to the terminal if possible; otherwise to mailbox.	Seldom used
14.	SAML	SAML<SP>FROM : <reverse-path><CRLF>	It send mail to the terminal and mailbox.	Seldom used

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.

Code	*Description*
Positive Completion Reply	
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
Positive Intermediate Reply	
354	Start mail input
Transient Negative Completion Reply	
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage

Working of SMTP:

1. **Composition of Mail:** A user sends an e-mail by composing an electronic mail message using a Mail User Agent (MUA). Mail User Agent is a program which is used to send and receive mail. The message contains two parts: body and header. The body is the main part of the message while the header includes information such as the sender and recipient address. The header also includes descriptive information such as the subject of the message. In this case, the message body is like a letter and header is like an envelope that contains the recipient's address.
2. **Submission of Mail:** After composing an email, the mail client then submits the completed e-mail to the SMTP server by using SMTP on TCP port 25.
3. **Delivery of Mail:** E-mail addresses contain two parts: username of the recipient and domain name. For example, vivek@gmail.com, where "vivek" is the username of the recipient and "gmail.com" is the domain name.

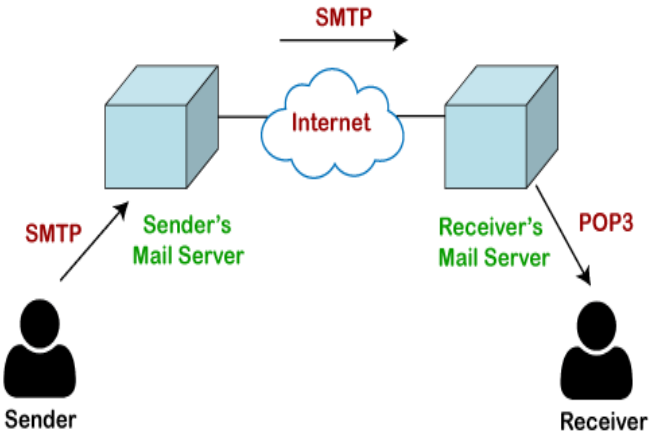
   If the domain name of the recipient's email address is different from the sender's domain name, then MSA will send the mail to the Mail Transfer Agent (MTA). To relay the email, the MTA will find the target domain. It checks the MX record from Domain Name System to obtain the target domain. The MX record contains the domain name and IP address of the recipient's domain. Once the record is located, MTA connects to the exchange server to relay the message.

4. **Receipt and Processing of Mail:** Once the incoming message is received, the exchange server delivers it to the incoming server (Mail Delivery Agent) which stores the e-mail where it waits for the user to retrieve it.
5. **Access and Retrieval of Mail:** The stored email in MDA can be retrieved by using MUA (Mail User Agent). MUA can be accessed by using login and password.

POP Protocol

The POP protocol stands for Post Office Protocol. As we know that SMTP is used as a message transfer agent. When the message is sent, then SMTP is used to deliver the message from the client to the server and then to the recipient server. But the message is sent from the recipient server to the actual server with the help of the Message Access Agent. The Message Access Agent contains two types of protocols, i.e., POP3 and IMAP.

How is mail transmitted?



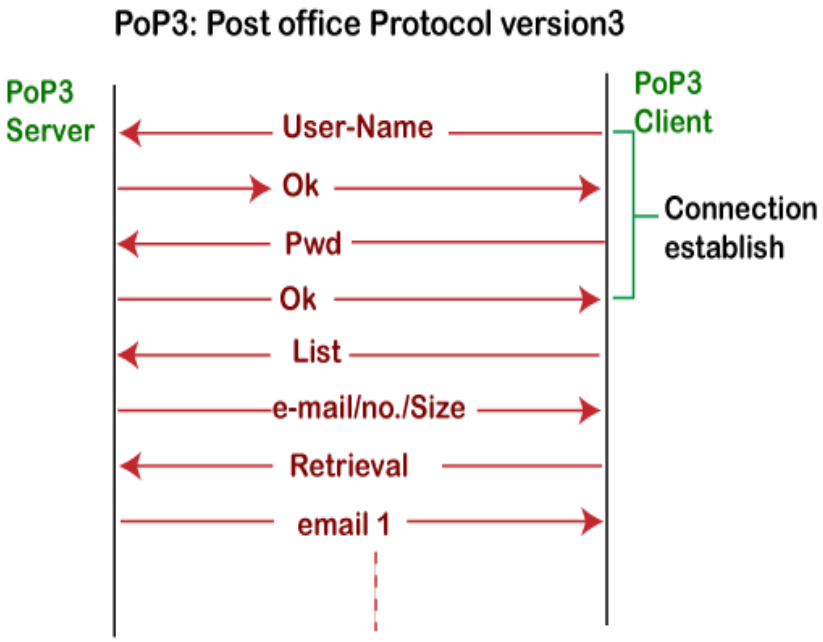
Suppose sender wants to send the mail to receiver. First mail is transmitted to the sender's mail server. Then, the mail is transmitted from the sender's mail server to the receiver's mail server over the internet. On receiving the mail at the receiver's mail server, the mail is then sent to the user. The whole process is done with the help of Email protocols. The transmission of mail from the sender to the sender's mail server and then to the receiver's mail server is done with the help of the SMTP protocol. At the receiver's mail server, the POP or IMAP protocol takes the data and transmits to the actual user.

Since SMTP is a push protocol so it pushes the message from the client to the server. As we can observe in the above figure that SMTP pushes the message from the client to the recipient's mail server. The third stage of email communication requires a pull protocol, and POP is a pull protocol. When the mail is transmitted from the recipient mail server to the client which means that the client is pulling the mail from the server.

What is POP3?

The POP3 is a simple protocol and having very limited functionalities. In the case of the POP3 protocol, the POP3 client is installed on the recipient system while the POP3 server is installed on the recipient's mail server.

Working of the POP3 protocol:



To establish the connection between the POP3 server and the POP3 client, the POP3 server asks for the user name to the POP3 client. If the username is found in the POP3 server, then it sends the ok message. It then asks for the password from the POP3 client; then the POP3 client sends the password to the POP3 server. If the password is matched, then the POP3 server sends the OK message, and the connection gets established. After the establishment of a connection, the client can see the list of mails on the POP3 mail server. In the list of mails, the user will get the email numbers and sizes from the server. Out of this list, the user can start the retrieval of mail.

Once the client retrieves all the emails from the server, all the emails from the server are deleted. Therefore, we can say that the emails are restricted to a particular machine, so it would not be possible to access the same mails on another machine. This situation can be overcome by configuring the email settings to leave a copy of mail on the mail server.

Advantages of POP3 protocol:

- It allows the users to read the email offline. It requires an internet connection only at the time of downloading emails from the server. Once the mails are downloaded from the server, then all the downloaded mails reside on our PC or hard disk of our computer, which can be accessed without the internet. Therefore, we can say that the POP3 protocol does not require permanent internet connectivity.
- It provides easy and fast access to the emails as they are already stored on our PC.
- There is no limit on the size of the email which we receive or send.
- It requires less server storage space as all the mails are stored on the local machine.
- There is maximum size on the mailbox, but it is limited by the size of the hard disk.
- It is a simple protocol so it is one of the most popular protocols used today.
- It is easy to configure and use.

Disadvantages of POP3 protocol:

- If the emails are downloaded from the server, then all the mails are deleted from the server by default. So, mails cannot be accessed from other machines unless they are configured to leave a copy of the mail on the server.
- Transferring the mail folder from the local machine to another machine can be difficult.
- Since all the attachments are stored on your local machine, there is a high risk of a virus attack if the virus scanner does not scan them. The virus attack can harm the computer.
- The email folder which is downloaded from the mail server can also become corrupted.
- The mails are stored on the local machine, so anyone who sits on your machine can access the email folder.

Hyper Text Transfer Protocol (HTTP)

- The standard web transfer protocol is HTTP.
- The **HyperText Transfer Protocol (HTTP)** is used to define how the client-server programs can be written to get 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, which is a connection-oriented and reliable protocol.

Non persistent versus Persistent Connections

Non persistent Connections
In a **nonpersistent connection**, one TCP connection is made for each request/response.
The steps 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; it then closes the connection.

- In this strategy, if a file contains links to N different pictures in different files (all located on the same server), 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.

- Figure shows an example of a nonpersistent connection.

- The client needs to access a file that contains one link to an image. The text file and image are located on the same server. Here we need two connections. For each connection, TCP requires at least three handshake messages to establish the connection, but the request can be sent with the third one.

- After the connection is established, the object can be transferred. After receiving an object, another three handshake messages are needed to terminate the connection.

- The client and server are involved in two connection establishments and two connection terminations.

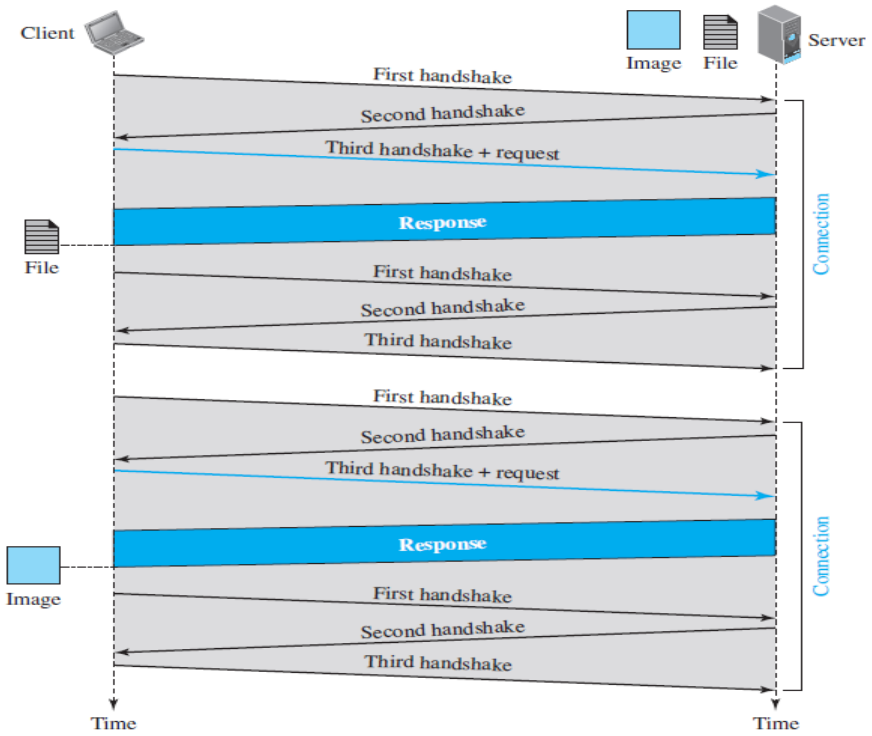


Fig: Nonpersistent connection

Persistent Connections

- HTTP 1.1 make persistent connections.
- In a persistent connection, the server keeps 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.
- There are some occasions, when the sender does not know the length of the data. This happens when a document is created dynamically or actively.
- 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.

What is RTT?

- RTT is the time it takes for a packet to travel from client to server and then back to the client.
- Figure (below) shows the example for persistent connections.

- Here Only one connection establishment and connection termination is used, but the request for the image is sent separately.

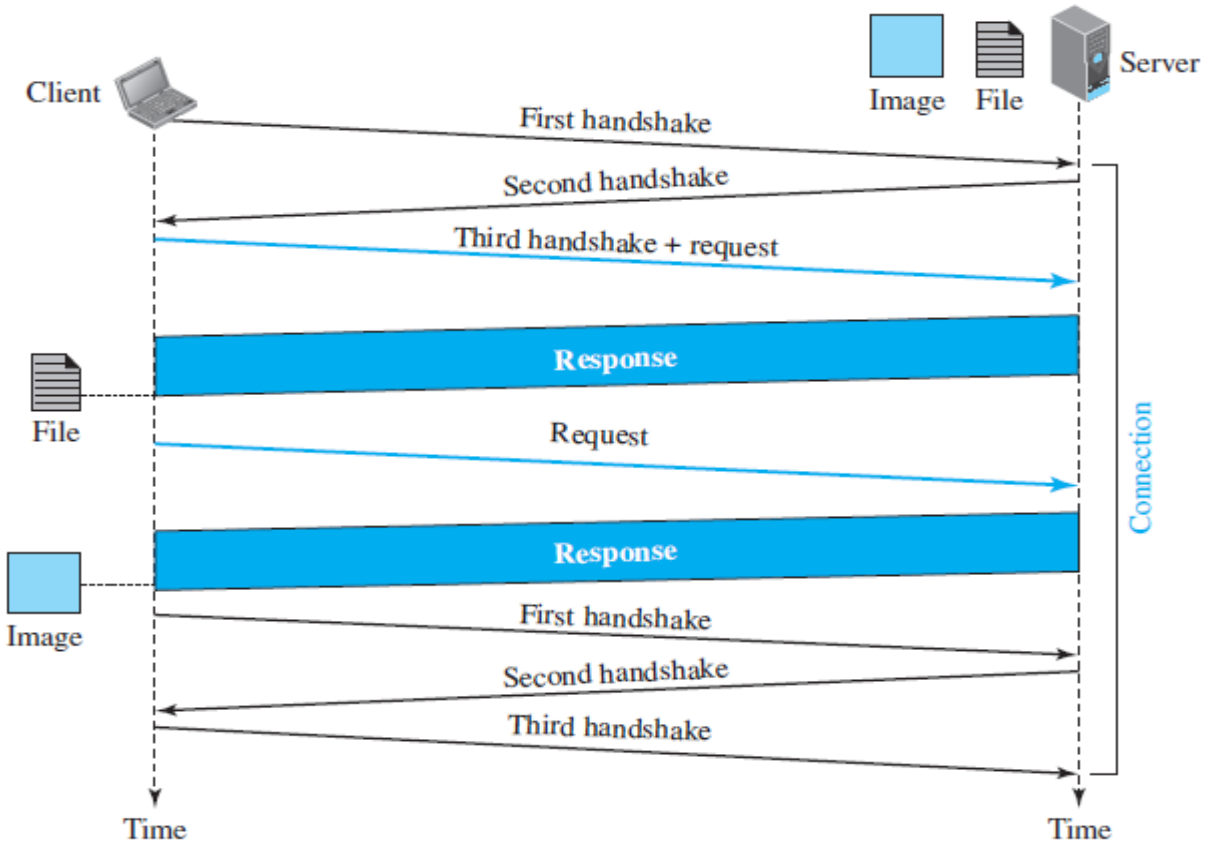


Fig: Persistent connection

HTTP Messages
Two types: request , response.

Message Formats
- The HTTP protocol defines the format of the request and response messages, as shown in Figure.
- 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.

Request Message
- The first line in a request message is called a request line.
- There are three fields in this line separated by one space and terminated by two characters (carriage return and line feed).
- The fields are called method, URL, and version.

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options

- The method field defines the request types.
- The client uses the GET method to send a request. In this case, the body of the message is empty. The HEAD method is used when the client needs only some information about the web page from the server, such as the last time it was modified. It is used to test the validity of a URL.
- The response message in this case has only the header section; the body section is empty. The PUT method is the inverse of the GET method; it allows the client to post a new web page on the server (if permitted).
- The POST method is like PUT method, but it is used to send some information to the server to be added to the web page or to modify the web page.
- The TRACE method is used for debugging; the client asks the server to echo back the request to check whether the server is getting the requests.
- The DELETE method allows the client to delete a web page on the server if the client has permission to do so.
- The CONNECT method was made as a reserve method; it may be used by proxy servers.
- The OPTIONS method allows the client to ask about the properties of a web page from the server.
- The second field, URL defines the address and name of the corresponding web page. The third field, gives the version of the protocol; the current version of HTTP is 1.1.
- After the request line, we have request header lines.
- Each header line sends additional information from the client to the server.
- **For example**, the client can request that the document be sent in a special format. Each header line has a header name, a colon, a space, and a header value
- The value field defines the values associated with each header name.
- The body can be present in a request message. It contains the comment to be sent or the file to be published on the website when the method is PUT or POST.

Response Message

- **A response message** consists of a status line, header lines, a blank line, and sometimes a body.
- The first line in a response message is called the status line. There are three fields in this line separated by spaces and terminated by a carriage return and line feed.
- The first field defines the version of HTTP protocol, currently 1.1.
- The status code field defines the status of the request. It consists of three digits.

- The codes in the 100 range are only informational, the codes in the 200 range indicate a successful request. The codes in the 300 range redirect the client to another URL, and the codes in the 400 range indicate an error at the client site.
- The codes in the 500 range indicate an error at the server site. The status phrase explains the status code in text form.
- After the status line, we can have zero or more response header lines.
- Each header line sends additional information from the server to the client.
- For example, the sender can send extra information about the document. Each header line has a header name, a colon, a space, and a header value. The body contains the document to be sent from the server to the client.

FTP

- FTP stands for File transfer protocol.
- FTP is a standard internet protocol provided by TCP/IP used for transmitting the files from one host to another.
- It is mainly used for transferring the web page files from their creator to the computer that acts as a server for other computers on the internet.
- It is also used for downloading the files to computer from other servers.

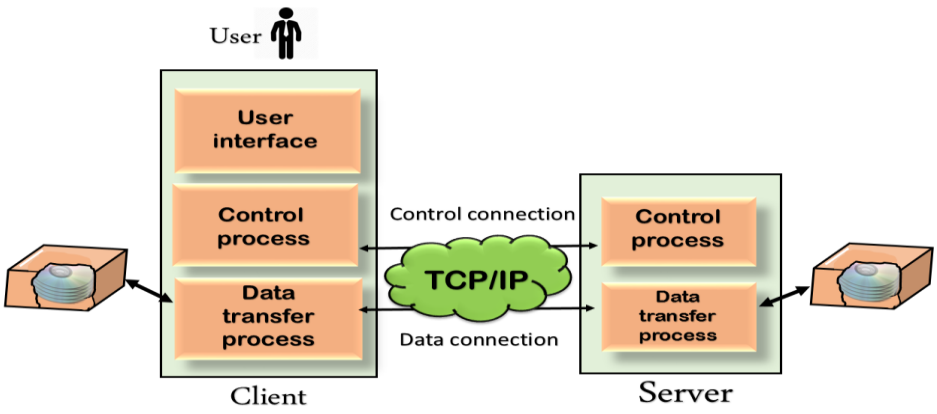
Objectives of FTP

- It provides the sharing of files.
- It is used to encourage the use of remote computers.
- It transfers the data more reliably and efficiently.

Why FTP?

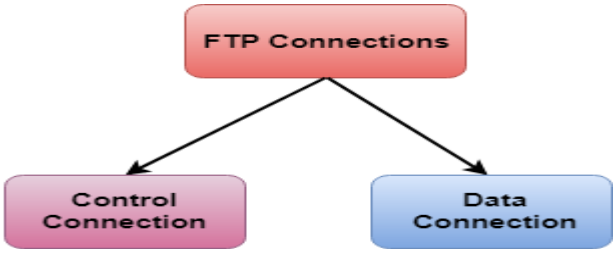
Although transferring files from one system to another is very simple and straightforward, but sometimes it can cause problems. For example, two systems may have different file conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. FTP protocol overcomes these problems by establishing two connections between hosts. One connection is used for data transfer, and another connection is used for the control connection.

Mechanism of FTP



The above figure shows the basic model of the FTP. The FTP client has three components: the user interface, control process, and data transfer process. The server has two components: the server control process and the server data transfer process.

There are two types of connections in FTP:



- **Control Connection:** The control connection uses very simple rules for communication. Through control connection, we can transfer a line of command or line of response at a time. The control connection is made between the control processes. The control connection remains connected during the entire interactive FTP session.

- **Data Connection:** The Data Connection uses very complex rules as data types may vary. The data connection is made between data transfer processes. The data connection opens when a command comes for transferring the files and closes when the file is transferred.

FTP Clients

- FTP client is a program that implements a file transfer protocol which allows you to transfer files between two hosts on the internet.
- It allows a user to connect to a remote host and upload or download the files.
- It has a set of commands that we can use to connect to a host, transfer the files between you and your host and close the connection.
- The FTP program is also available as a built-in component in a Web browser. This GUI based FTP client makes the file transfer very easy and also does not require to remember the FTP commands.

Advantages of FTP:

- **Speed:** One of the biggest advantages of FTP is speed. The FTP is one of the fastest way to transfer the files from one computer to another computer.
- **Efficient:** It is more efficient as we do not need to complete all the operations to get the entire file.
- **Security:** To access the FTP server, we need to login with the username and password. Therefore, we can say that FTP is more secure.
- **Back & forth movement:** FTP allows us to transfer the files back and forth. Suppose you are a manager of the company, you send some information to all the employees, and they all send information back on the same server.

Disadvantages of FTP:

- The standard requirement of the industry is that all the FTP transmissions should be encrypted. However, not all the FTP providers are equal and not all the providers offer encryption. So, we will have to look out for the FTP providers that provide encryption.
- FTP serves two operations, i.e., to send and receive large files on a network. However, the size limit of the file is 2GB that can be sent. It also doesn't allow you to run simultaneous transfers to multiple receivers.
- Passwords and file contents are sent in clear text that allows unwanted eavesdropping. So, it is quite possible that attackers can carry out the brute force attack by trying to guess the FTP password.
- It is not compatible with every system.