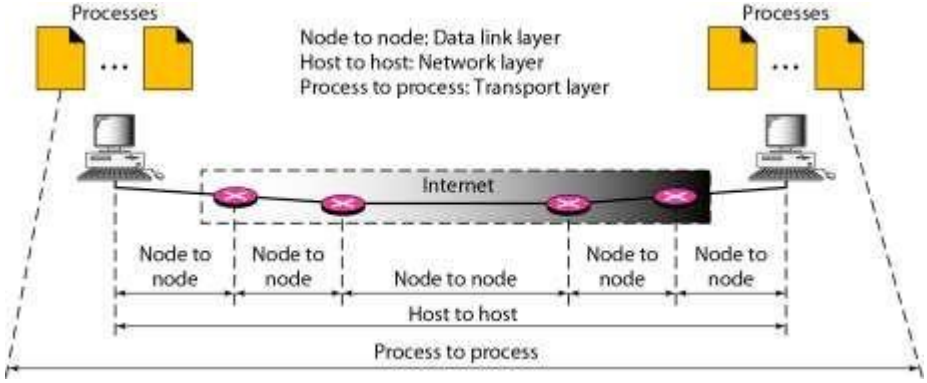
Unit -5
TRANSPORT LAYER

Process-to-process delivery

The transport layer ensures process-to-process delivery. The Internet model has three protocols at the transport layer: UDP, TCP, and SCTP. First we discuss UDP, which is the simplest of the three. We see how we can use this very simple transport layer protocol that lacks some of the features of the other two. We then discuss TCP, a complex transport layer protocol. We finally discuss SCTP, the new transport layer protocol that is designed for multi homed,multi stream applications such as multimedia.



Socket address: Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a socket address

UDP (USER DATAGRAM 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.

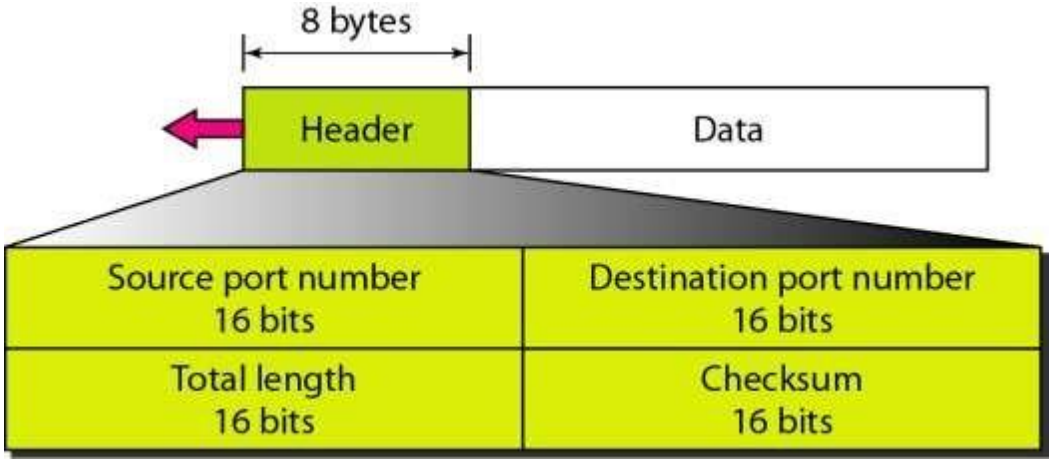
UDP does not see any relation (connection) between consequent datagram coming from the same source and going to the same destination.

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

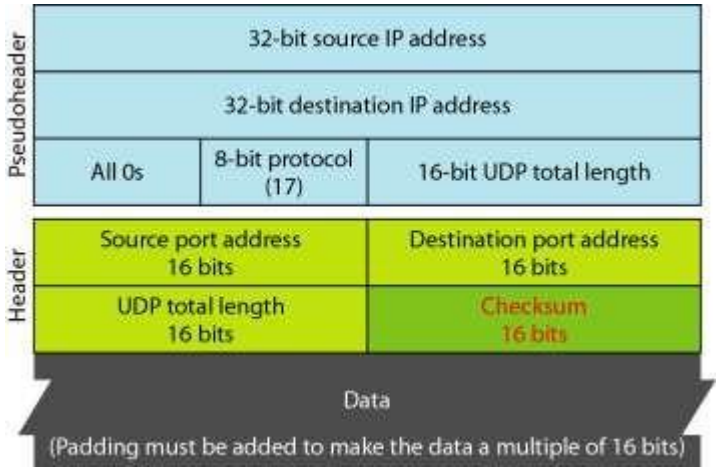
UDP packets, called user datagram, have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).

. 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



UDP Services

- Process-to-Process Communication.
- Connectionless Services
- Flow Control
  UDP is a very simple protocol. There is no flow control'
- Error Control
  There is no error control mechanism in UDP except for the checksum.
- 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

Typical Applications

The following shows some typical applications that can benefit more from the services of UDP
1. UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control
2. UDP is suitable for a process with internal flow- and error-control mechanisms. For example, the Trivial File Transfer Protocol (TFIP)
3. UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software
4. UDP is used for management processes such as SNMP
5. UDP is used for some route updating protocols such as Routing Information Protocol (RIP)
6. UDP is normally used for interactive real-time applications that cannot tolerate uneven delay between sections of a received message

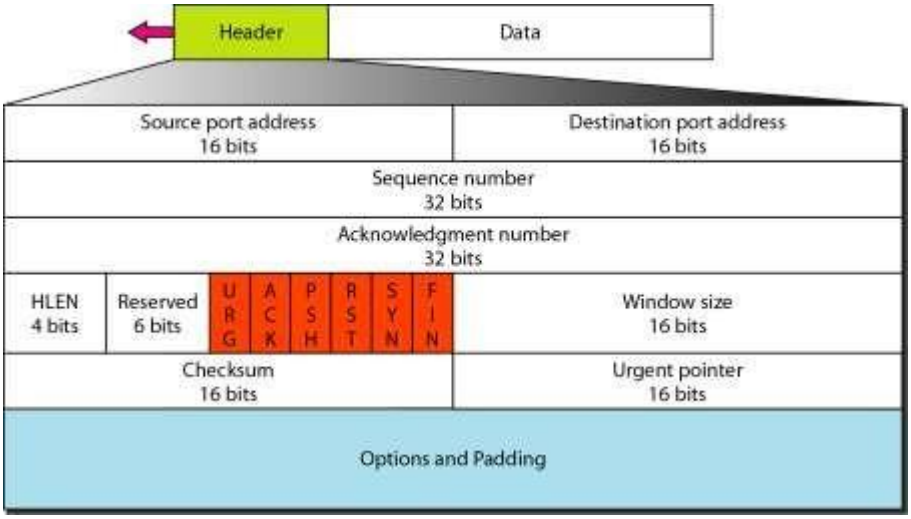
TRANSMISSION CONTROL PROTOCOL (TCP):

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
- Stream Delivery Service
- Segments
- Full-Duplex Communication
- Reliable Service

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.

Reserved. This is a 6-bit field reserved for future use.

Control. This field defines 6 different control bits or flags as shown in Figure .One or more of these bits can be set at a time.

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection

Window size. 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. The calculation of the checksum for TCP follows the same procedure as the one described for UDP.

Urgent pointer. This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.

Options. There can be up to 40 bytes of optional information in the TCP header.

TCP Features:

Numbering System

- Byte Number
- Sequence Number
- Acknowledgment Number

.

Flow Control

TCP, unlike UDP, provides flow control.

Error Control

To provide reliable service, TCP implements an error control mechanism.

Congestion Control

TCP, unlike UDP, takes into account congestion in the network.

A TCP Connection

TCP is connection-oriented. a connection-oriented transport protocol establishes a logical path between the source and destination. In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

Connection Establishment

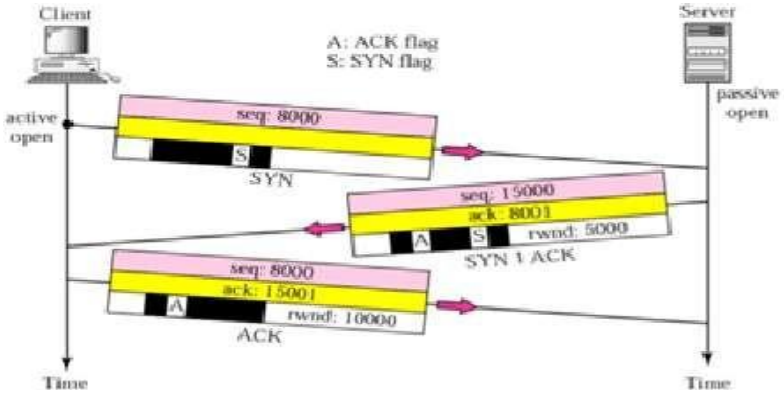
TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.

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.

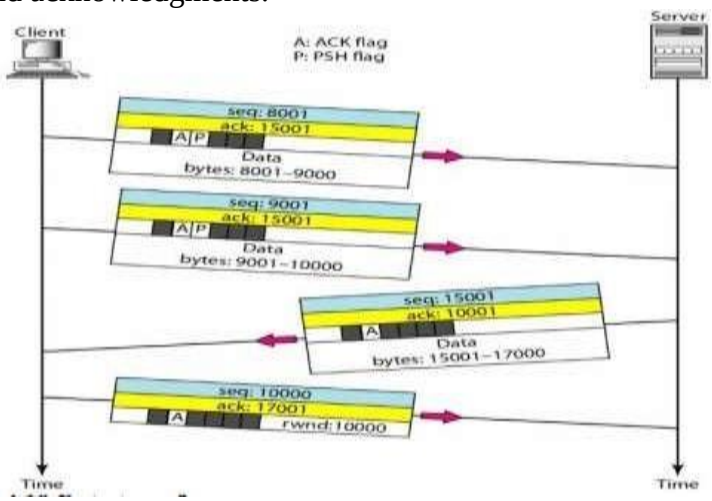
.

- 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



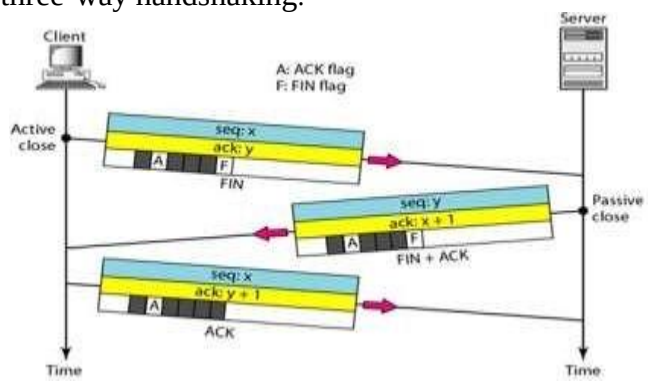
Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments.



Connection Termination

Any 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.



Congestion Control Techniques

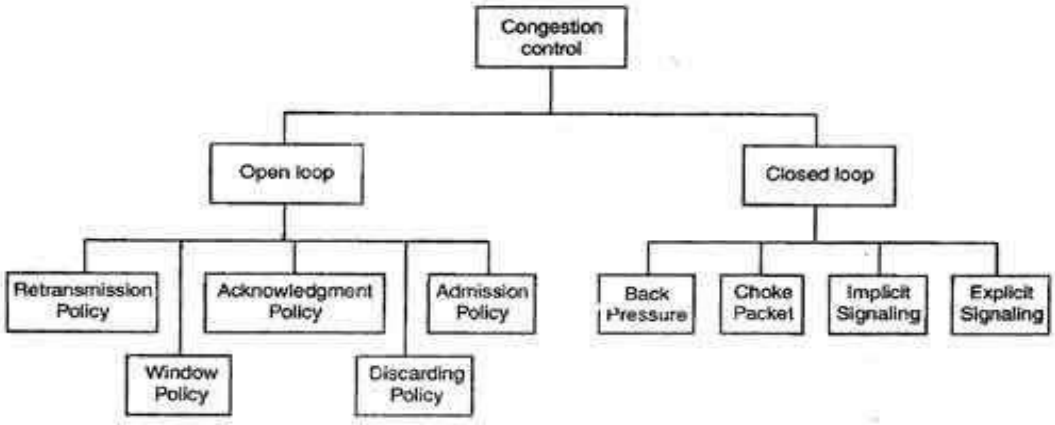
Congestion control refers to the mechanisms and techniques used to control congestion and keep the traffic below the capacity of the network. As shown in Fig. , the congestion control techniques can be broadly classified two broad categories:

• Open loop: Protocols to prevent or avoid congestion, ensuring that the system (or network under consideration) never enters a Congested State.

• Close loop: Protocols that allow system to enter congested state, detect it, and remove it.

Factors that Cause Congestion

a) Packet arrival rate exceeds the outgoing link capacity.
b) Insufficient memory to store arriving packets
c) Bursty traffic
d) Slow processor



Types of Congestion Control Methods

• The various methods used for open loop congestion control are:

Retransmission Policy

• The sender retransmits a packet, if it feels that the packet it has sent is lost or corrupted..

The retransmission policy and the retransmission timers need to be designed to optimize efficiency and at the same time prevent the congestion

Window Policy

• To implement window policy, selective reject window method is used for congestion control.

• Selective Reject method is preferred because it sends only the specific lost or damaged packets.

Acknowledgement Policy

• The acknowledgement policy imposed by the receiver may also affect congestion.

• If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.

Discarding Policy

• A router may discard less sensitive packets when congestion is likely to happen.

• Such a discarding policy may prevent congestion and at the same time may not harm the integrity of the transmission.

Admission Policy

• An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual circuit networks.

• Switches in a flow should first check the resource requirement of a network flow before admitting it to the network.

• A router can deny establishing a virtual circuit connection if there is congestion in the network or if there is a possibility of future congestion.

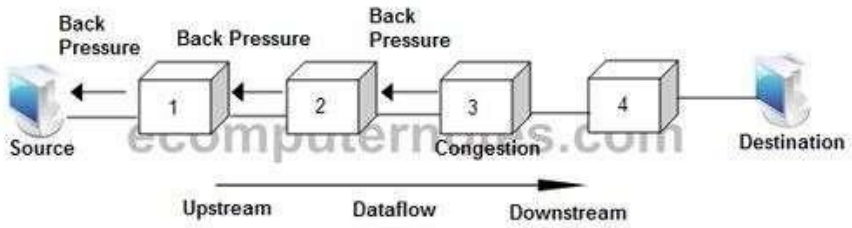
Closed Loop Congestion Control

• Closed loop congestion control mechanisms try to remove the congestion after it happens.

• The various methods used for closed loop congestion control are:

Backpressure

• Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow.



Backpressure Method

• In this method of congestion control, the congested node stops receiving data from the immediate upstream node or nodes.

This may cause the upstream node on nodes to become congested, and they, in turn, reject data from their upstream node or nodes.

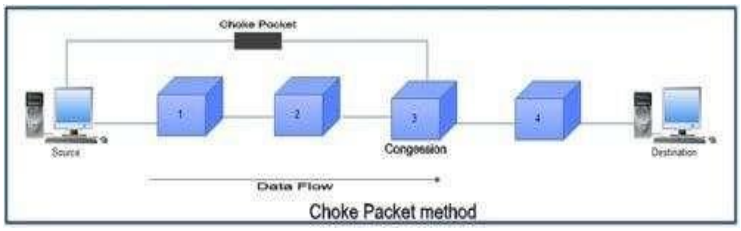
• As shown in fig node 3 is congested and it stops receiving packets and informs its upstream node 2 to slow down. Node 2 in turns may be congested and informs node 1 to slow down. Now node 1 may create congestion and informs the source node to slow down. In this way the congestion is alleviated. Thus, the pressure on node 3 is moved backward to the source to remove the congestion.

Choke Packet

• In this method of congestion control, congested router or node sends a special type of packet called choke packet to the source to inform it about the congestion.

• Here, congested node does not inform its upstream node about the congestion as in backpressure method.

• In choke packet method, congested node sends a warning directly to the source station



Choke Packet method

Implicit Signaling

• In implicit signaling, there is no communication between the congested node or nodes and the source.

• The source guesses that there is congestion somewhere in the network when it does not receive any acknowledgment. Therefore the delay in receiving an acknowledgment is interpreted as

congestion in the network.

• On sensing this congestion, the source slows down.

• This type of congestion control policy is used by TCP.

Explicit Signaling

• In this method, the congested nodes explicitly send a signal to the source or destination to inform about the congestion.

• Explicit signaling is different from the choke packet method. In choke packed method, a separate packet is used for this purpose whereas in explicit signaling method, the signal is included in the packets that carry data .

• Explicit signaling can occur in either the forward direction or the backward direction .

• In backward signaling, a bit is set in a packet moving in the direction opposite to the congestion. This bit warns the source about the congestion and informs the source to slow down.

• In forward signaling, a bit is set in a packet moving in the direction of congestion. This bit warns the destination about the congestion.

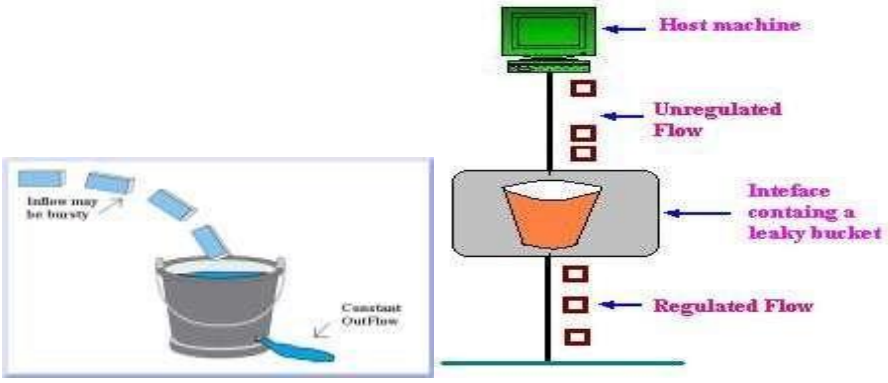
Congestion Control Algorithms:

Leaky Bucket Algorithm

Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket; the rate at which water comes out from that small hole is constant. This scenario is depicted in figure. Once the bucket is full, any additional water entering it spills over the sides and is lost The same idea of leaky bucket can be applied to packets, as shownin Fig Conceptually each network interface contains a leaky bucket. And the following steps are performed:

- When the host has to send a packet, the packet is thrown into the bucket.
- The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
- Bursty traffic is converted to a uniform traffic by the leaky bucket.
- In practice the bucket is a finite queue that outputs at a finite rate.

Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded.



Token Bucket Algorithm

The leaky bucket algorithm described above, enforces a rigid pattern at the output stream, irrespective of the pattern of the input. For many applications it is better to allow the output to speed up somewhat when a larger burst arrives than to loose the data. Token Bucket algorithm

provides such a solution. In this algorithm leaky bucket holds token, generated at regular intervals. Main steps of this algorithm can be described as follows:

- In regular intervals tokens are thrown into the bucket.
- The bucket has a maximum capacity.
- If there is a ready packet, a token is removed from the bucket, and the packet is send.
- If there is no token in the bucket, the packet cannot be send.

Figure shows the two scenarios before and after the tokens present in the bucket have been consumed. In Fig. (a) the bucket holds two tokens, and three packets are waiting to be sent out of the interface, in Fig.(b) two packets have been sent out by consuming two tokens, and 1 packet is still left.

The implementation of basic token bucket algorithm is simple; a variable is used just to count the tokens. This counter is incremented every t seconds and is decremented whenever a packet is sent. Whenever this counter reaches zero, no further packet is sent out.

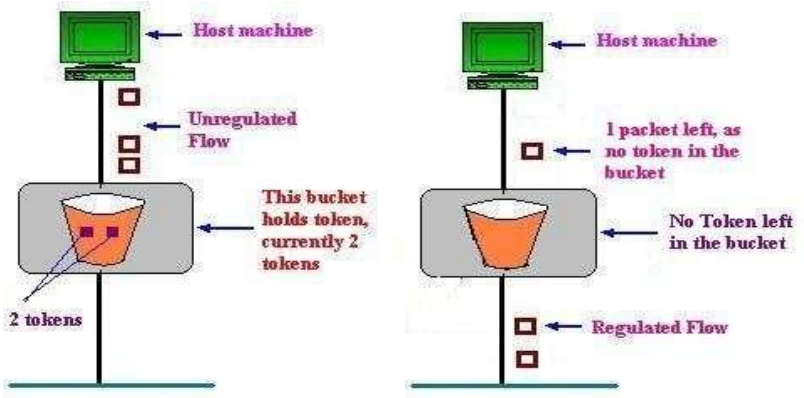


Figure 7.5.4(a) Token bucket holding two tokens, before packets are send out, (b) Token bucket after two packets are sending, one packet still remains as no token is left.

TECHNIQUES TO IMPROVE QoS

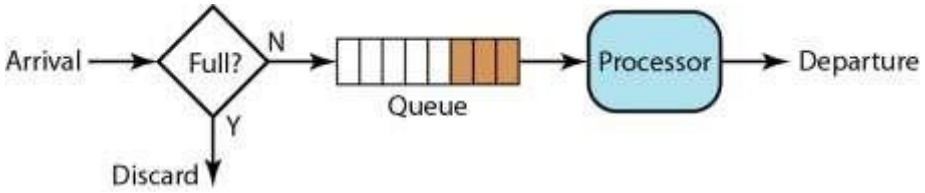
We define QoS in terms of its characteristics. In this section, we discuss some techniques that can be used to improve the quality of service. We briefly discuss four common methods: scheduling, traffic shaping, admission control, and resource reservation.

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: FIFO queuing, priority queuing, and weighted fair queuing.

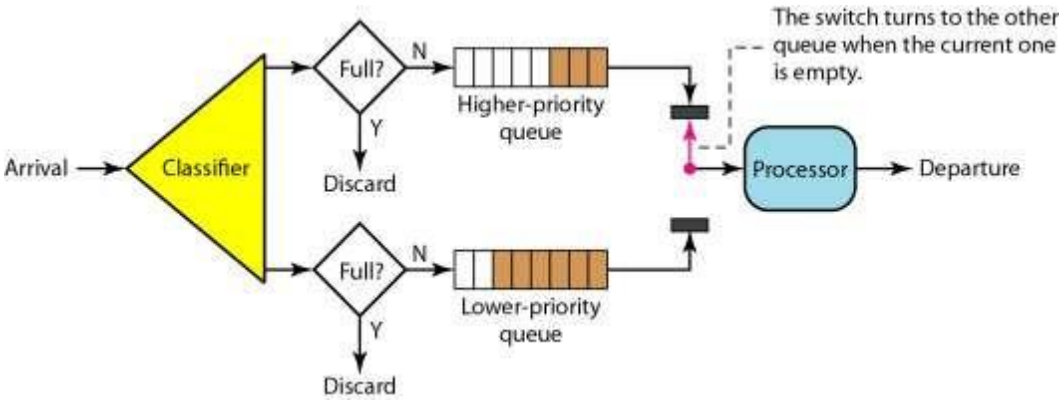
FIFO Queuing

In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router 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 24.16 shows a conceptual view of a FIFO queue.



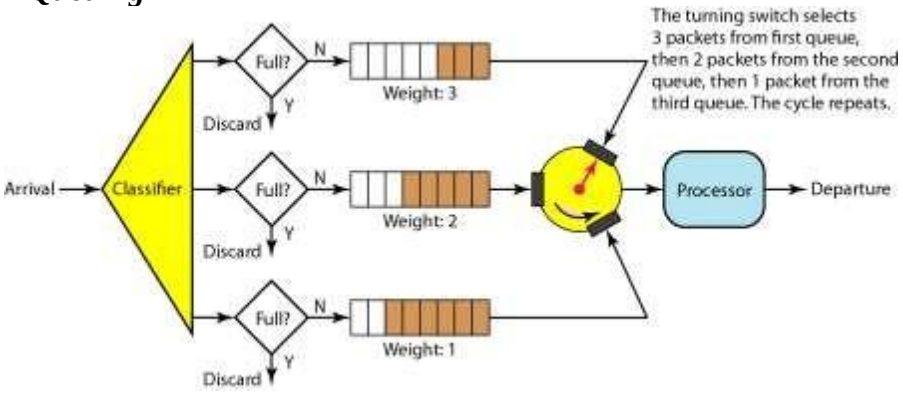
Priority Queuing

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 24.17 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.*

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 24.18 shows the technique with three classes.

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.

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.

NAME SPACE

A name space that maps each address to a unique name can be organized in two ways: flat or hierarchical.

Flat Name

In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section.The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication

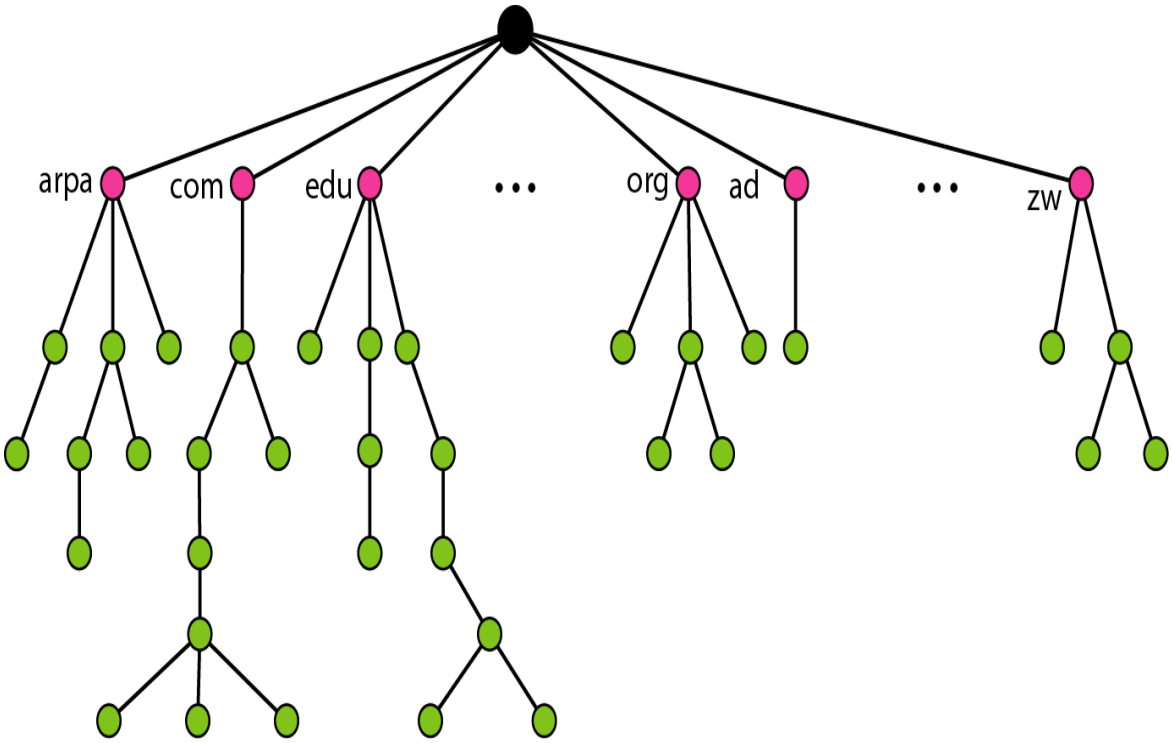
Hierarchical Name Space

In a hierarchical name space, each name is made of several parts. The **first part** can define the nature of the organization, the **second part** can define the name of an organization, the **third part** can define departments in the organization, and so on. In this case, the authority to assign and control the name spaces can be decentralized. A **central authority** can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility of the rest of the name can be given to the organization itself. The organization canadd suffixes (or prefixes) to the name to define its host or resources. The management of the organization need not worry that the prefix chosen for a host is taken by another organization because, even if part of an address is the same, the whole address is different.

The Domain Name System (DNS)

A DNS translates or resolves a hostname (eg. www.zoho.com) into a language of numbers that a computer can understand (eg. an IP address).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 can have only 128 levels: level 0 (root) to level 127

Domain name space



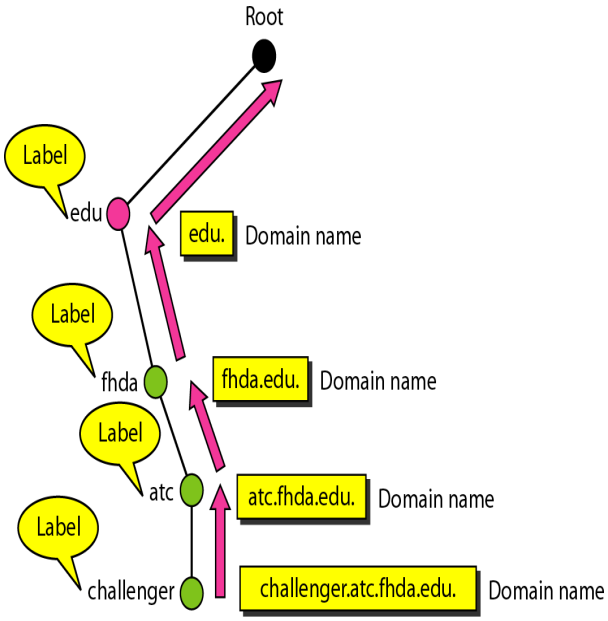
Label

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

Domain Name

Each node in the tree has a domain name. A full domain name is a sequence of labels separated by dots (.). 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.

Domain names and labels



Fully Qualified Domain Name

If a label is terminated by a null string, it is called a fully qualified domain name (FQDN). An FQDN is a domain name that contains the full name of a host. It contains all labels, from the most specific to the most general, that uniquely define the name of the host. For example, the domain name challenger.atc.fhda.edu. is the FQDN of a computer named *challenger installed at the Advanced Technology* Center (ATC) at De Anza College.

A **DNS server** can only match an FQDN to an address. Note that the name must end with a null label, but because null means nothing,the label ends with a dot (.).

Partially Qualified Domain Name

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.

For example, if a user at the *fhda.edu. site wants to get the IP address of the challenger computer, he or she* can define the partial name challenger

The DNS client adds the suffix *atc.fhda.edu.* ***before*** *passing the address to the DNS* server.

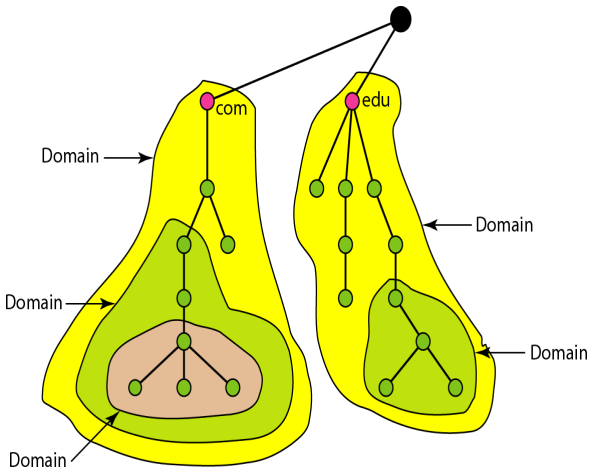
The DNS client normally holds a list of suffixes. The following can be the list of suffixes at De Anza College.

atc.fhda.edu
fhda.edu
null

The null suffix defines nothing. This suffix is added when the user defines an FQDN.

A **domain** is a subtree of the domain name space. The name of the domain is the domain name of the node at the top of the subtree. Note that a domain may itself be divided into domains (or subdomains as they are sometimes called).

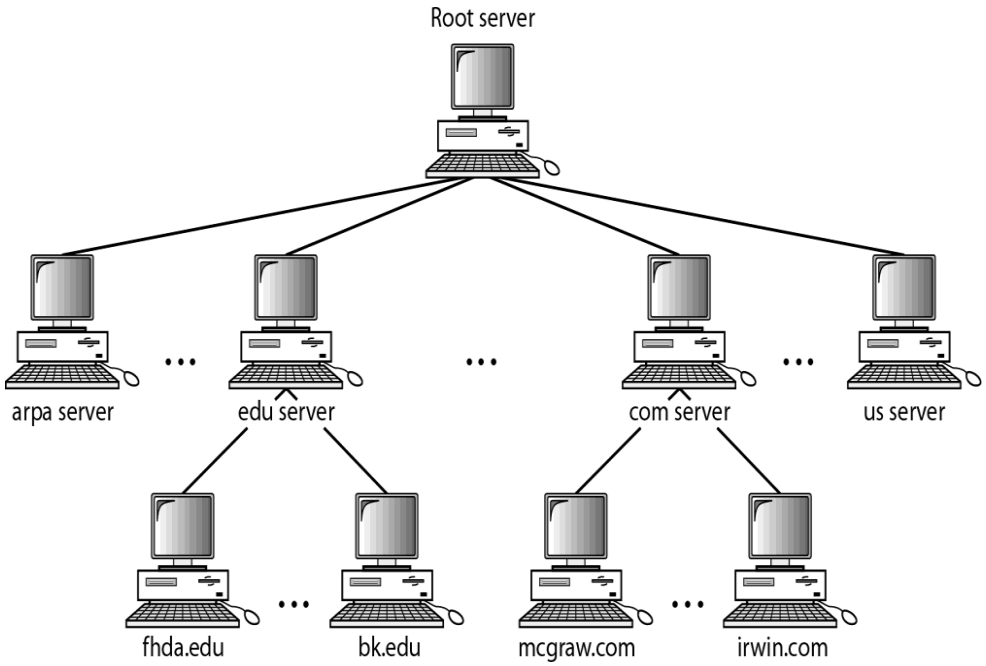
Domains: subtree of the domain name space



Hierarchy of Name Servers

The information contained in the Domain Name Space is distributed among many computers called DNS servers. Whole space is divide d the into many domains to based on the first level. Each server can be responsible (authoritative) for either a large or a small domain.

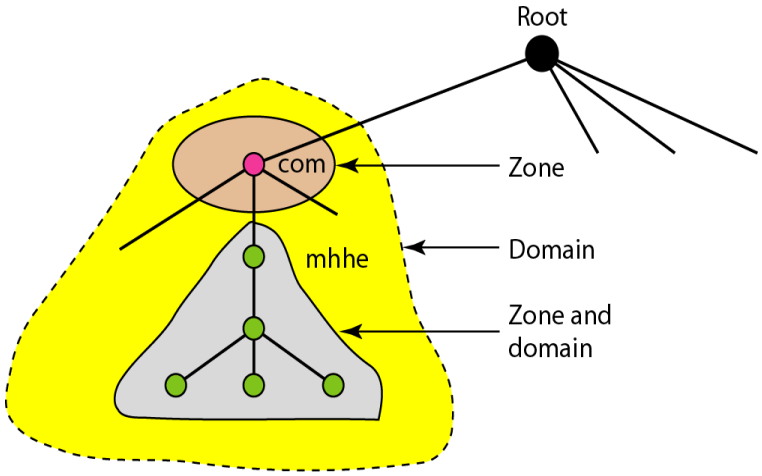
Hierarchy of name servers



Zone

Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a zone. We can define a zone as a contiguous part of the entire tree. If a server accepts responsibility for a domain and does not divide the domain into smaller domains, the domain and the zone refer to the same thing. The server makes a database called a zone file and keeps all the information forevery node under that domain. However, if a server divides its domain into subdomains and delegates part of its authority to other servers, domain and zone refer to different things. 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. Of course the original server does not free itself from responsibility totally: It still has a zone, but the detailed information is kept by the lower-level servers A server can also divide part of its domain and delegate responsibility but still keep part of the domain for itself. In this case, its zone is made of detailed information for the part of the domain that is not delegated and references to those parts that are delegated.

Zones and domains



Root Server

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

Primary and Secondary Servers

DNS defines two types of servers: primary and secondary. A **primary server** is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It **stores** the zone file on a local disk.

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

The primary and secondary servers are both authoritative for the zones they serve.

The idea is not to put the secondary server at a lower level of authority but to create redundancy for the data so that if one server fails, the other can continue serving clients. Note also that a server can be a primary server for a specific zone and a secondary server for another zone.

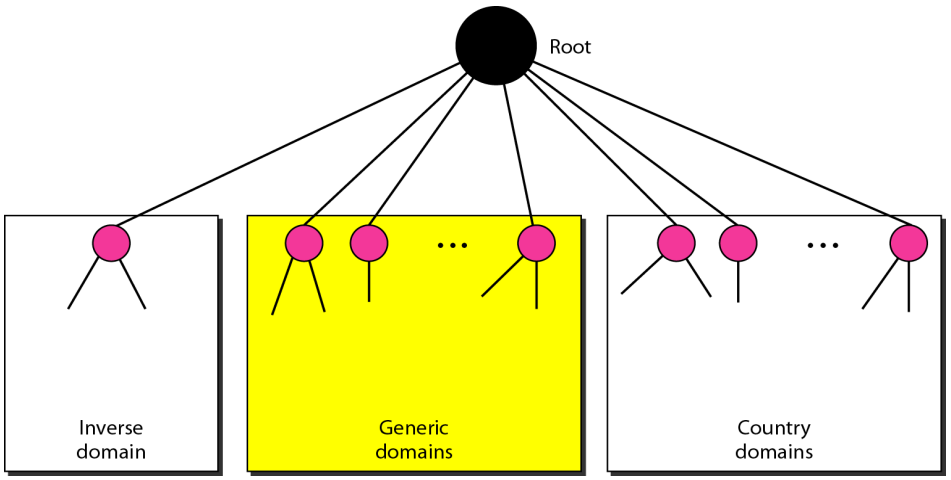
DNS IN THE INTERNET

DNS is a protocol that can be used in different platforms.

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

- Generic domains
- Country domains
- Inverse domain

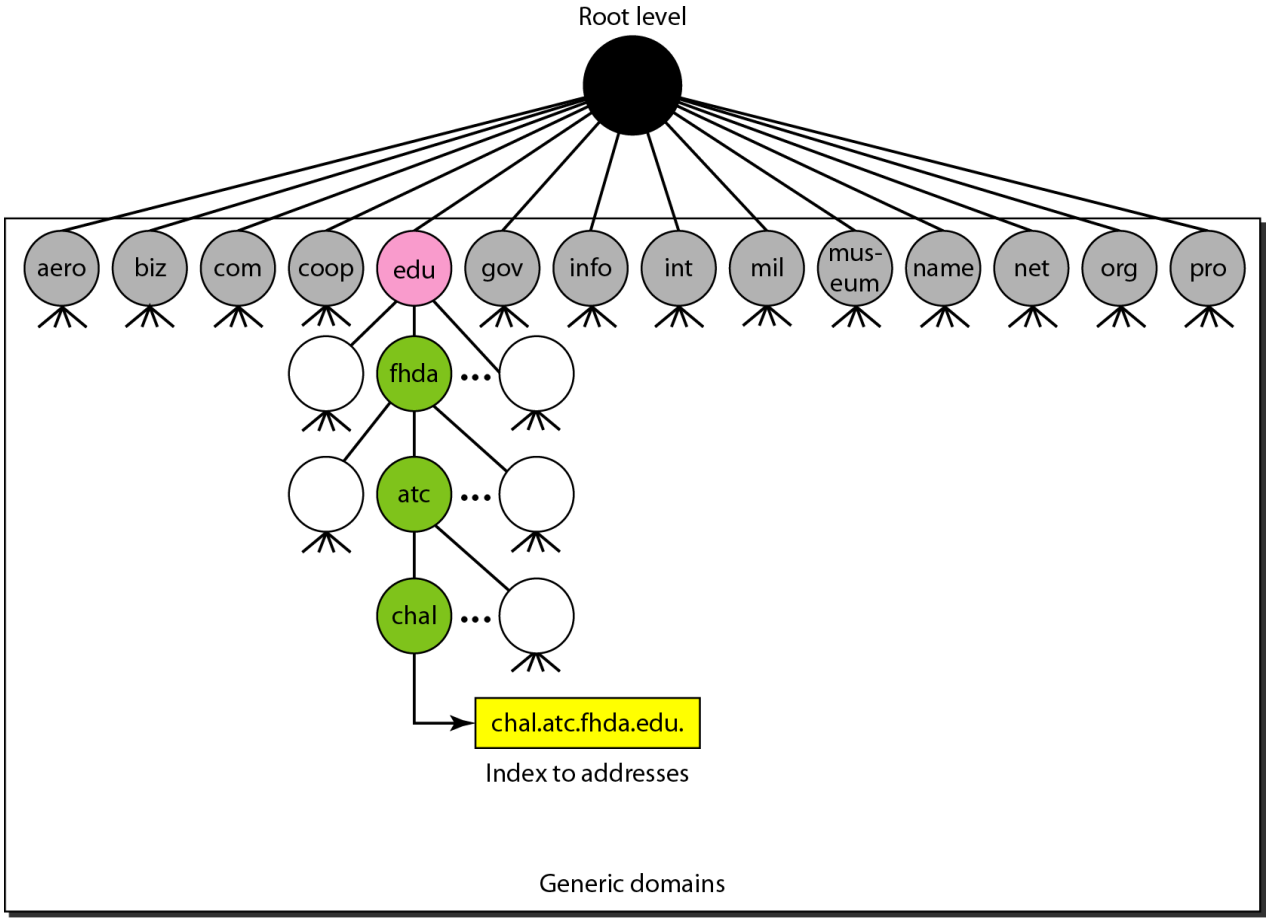
DNS IN THE INTERNET



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.Looking at the tree, we see that the first level in the generic domains section allows 14 possible labels

Generic domains



Label	*Description*
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to "com")
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military groups
museum	Museums and other nonprofit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations
pro	Professional individual organizations

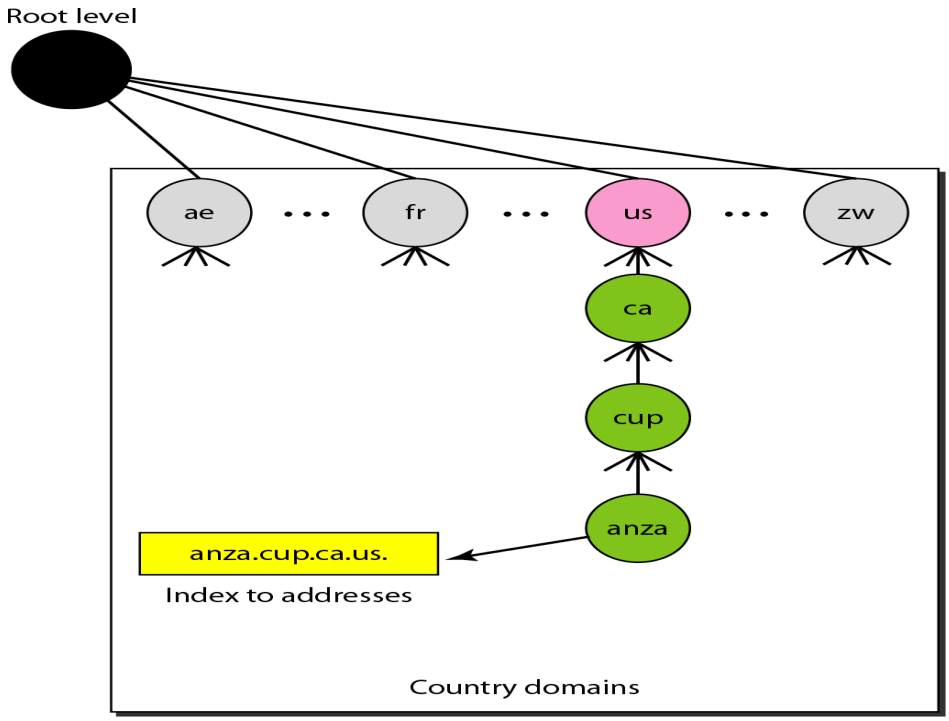
Country Domains

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.

The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).

The address ***anza.cup.ca.us*** *can* be translated to De Anza College in Cupertino, California, in the United States.

Figure shows the country domains section.



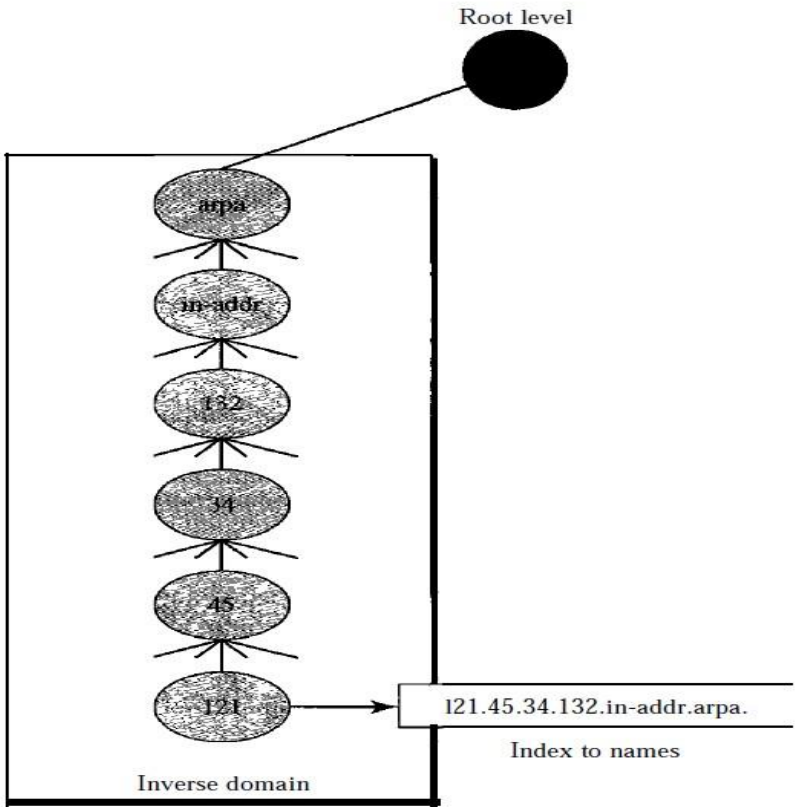
Inverse Domain

The inverse domain is used to map an address to a name. This may happen, for example, when a server has received a request from a client to do a task. Although the server has a file that contains a list of authorized clients, only the IP address of the client (extracted from the received IP packet) is listed. The server asks its resolver to send a query to the DNS server to map an address to a name to determine if the client is on the authorized list. This type of query is called an inverse or pointer (PTR) query.To handle a pointer query, the inverse domain is added to the

domain name space with the first-level node called *arpa (for historical reasons).* The second level is also one single node named in-addr (for inverse address). The rest of the domain defines IP addresses.The servers that handle the inverse domain are also hierarchical.

This means the netid part of the address should be at a higher level than the subnetid part, and the subnetid part higher than the hostid part. In this way, a server serving the whole site is at a higher level than the servers serving each subnet. This configuration makes the domain look inverted when compared to a generic or country domain. To follow the convention of reading the domain labels from the bottom to the top, an IP address such as 132.34.45.121 (a class B address with netid 132.34) is read as 121.45.34.132.in-addr. arpa.

Figure 25.11 *Inverse domain*



RESOLUTION

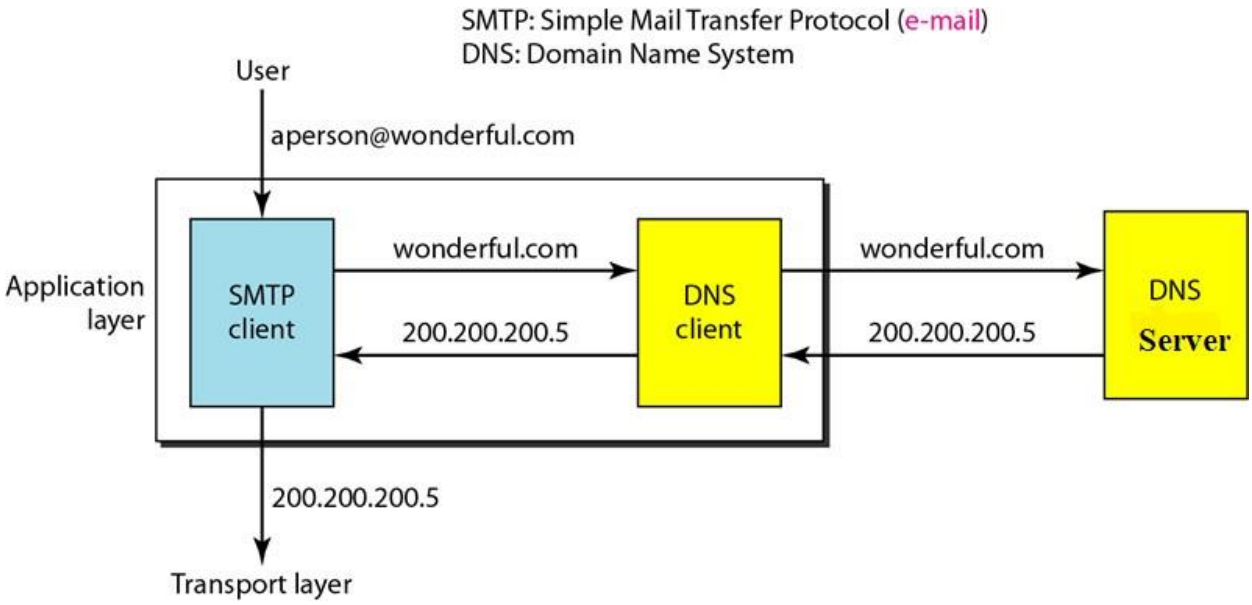
Mapping a name to an address or an address to a name is called *name-address resolution*

Resolver

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 resolve otherwise, it either refers the resolver to other servers or asks other servers to provide the information.

Example of using the DNS service



After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it.

Mapping Names to Addresses

Most of the time, the resolver gives a domain name to the server and asks for the corresponding address. In this case, the server checks the generic domains or the countrydomains to find the mapping.If the domain name is from the generic domains section, the resolver receives a domain name such as *"chal.atc.fhda.edu."*. The query is sent by the resolver to the local DNS server for resolution. If the local server cannot resolve the query, it either refers the resolver to other servers or asks other servers directly.If the domain name is from the country domains section, the resolver receives a domain name such as "ch.fhda.cu.ca.us.". The procedure is the same.

Mapping Addresses to Names

A client can send an IP address to a server to be mapped to a domain name this is called a PTR query. To answer queries of this kind, DNS uses the inverse domain. However, in the request,

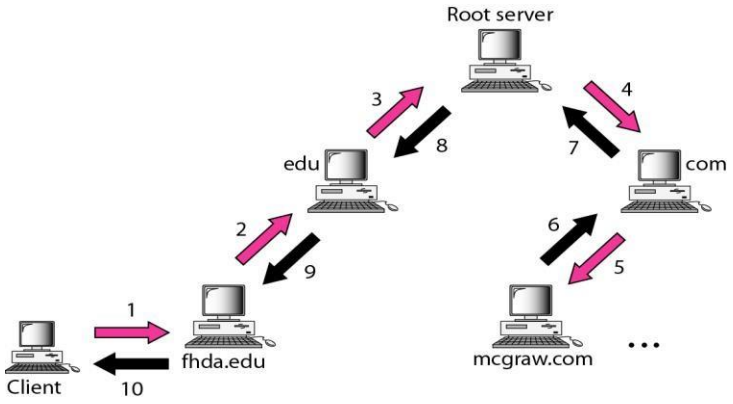
the IP address is reversed and the two labels in-addr and arpa are appended to create a domain acceptable by the

Inverse domain section.

For example, if the resolver receives the IP address 132.34.45.121, the resolver first inverts the address and then adds the two labels before sending. The domain name sent is "121.45.34.132.in-addr.arpa." which is received by the local DNS and resolved.

Recursive Resolution

The client (resolver) can ask for a recursive answer from a name server. This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response.If the parent is the authority, it responds; otherwise, it sends the query to yet another server. When the query is finally resolved, the response travels back until it finally reaches the requesting client. This is called recursive resolution

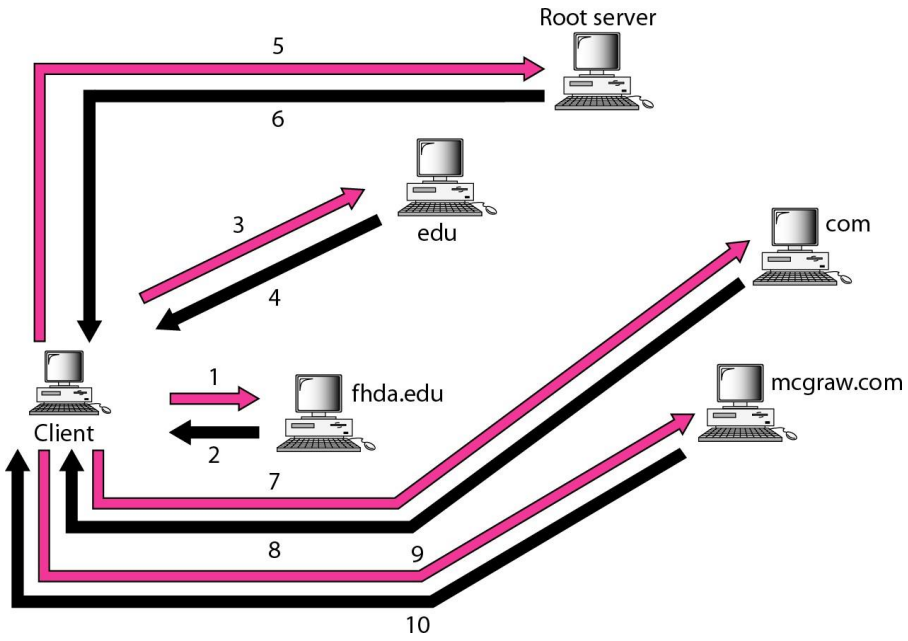


Iterative Resolution

If the client does not ask for a recursive answer, the mapping can be done iteratively. If the server is an authority for the name, it sends the answer. If it is not, it returns (to the client) the IPaddress of the server that it thinks can resolve the query. The client is responsible for repeating the query to this second server. If the newly addressed server can resolve the problem, it answersthe query with the IP address; otherwise, it returns the IP address of a new server to the client.

Now the client must repeat the query to the third server. This process is called iterative resolution because the client repeats the same query to multiple servers. In Figure the client queries four servers before it gets an answer from the mcgraw.com server.

Iterative resolution



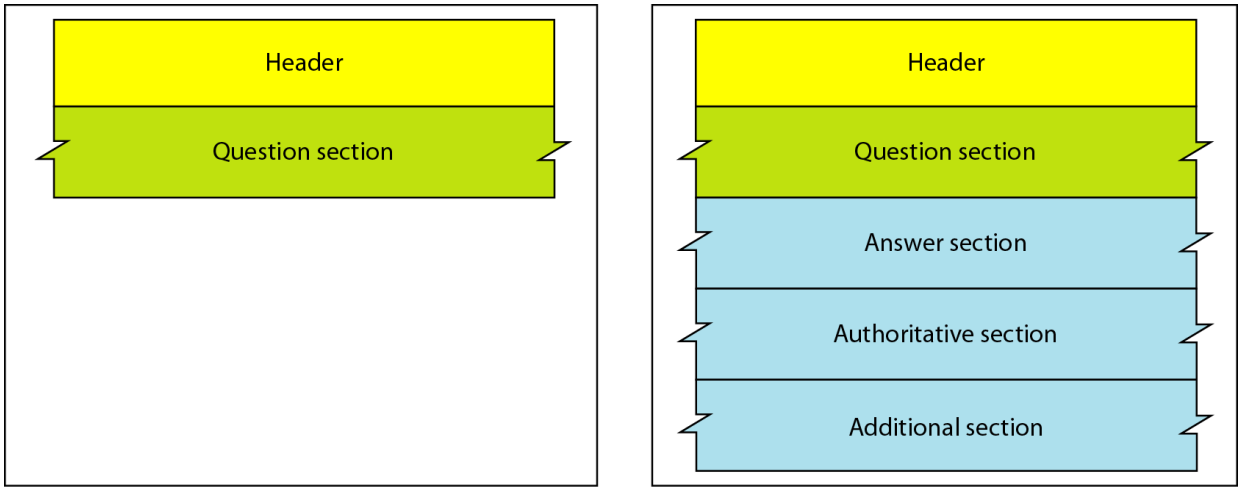
Caching

Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase efficiency. DNS handles this with a mechanism called caching. When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client. If the same or another client asks for the same mapping, it can check its cache memory and solve the problem. However, to inform the client that the response is coming from the cache memory and not from an authoritative source, the server marks the response as **unauthoritative.** Caching speeds up resolution, but it can also be problematic. If a server caches a mapping for a long time, it may send an outdated mapping to the client. To counter this, two techniques are used. First, the authoritative server always adds information to the mappingcalled *time-to-live (TTL).* It defines the time in seconds that the receiving server can cache the information. After that time, the mapping is invalid and any query must be sent again to the authoritative server. Second, DNS requires that each server keep a TTL counter for each mapping it caches. The cache memory must be searched periodically, and those mappings with an expired TTL must be purged

DNS MESSAGES

DNS has two types of messages: query and response. Both types have the same format. The query message consists of a header and question records .The response message consists of a header, question records, answer records, authoritative records, and additional records.

Figure *Query and response messages*



a. Query b. Response

Figure: Header *format*

Identification	Flags
Number of question records	Number of answer records (all 0s in query message)
Number of authoritative records (all 0s in query message)	Number of additional records (all 0s in query message)

Header

Both query and response messages have the same header format with some fields set to zero for the query messages. The header is **12 bytes**, and its format is shown in Figure .The identification subfield is used by the client to match the response with the query.The client uses a different identification number each time it sends a query. The server duplicates this number in the corresponding response. The flags subfield is a collection of subfields that define the type of the message, the type of answer requested, the type of desired resolution (recursive or iterative), and so on. The number of question records subfield contains the number of queries in the question

section of the message. The number of answer records subfield contains the number of answer records in the answer section of the response message. Its value is zero in the query message.

The number of authoritative records subfields contains the number of authoritative records in the authoritative section of a response message. Its value is zero in the query message. Finally, the number of additional records subfields contains the number additional records in the additional section of a response message. Its value is zero in the query message.

Question Section

This is a section consisting of one or more question records. It is present on both query and response messages.

Answer Section

This is a section consisting of one or more resource records. It is present only on response messages. This section includes the answer from the server to the client (resolver).

Authoritative Section

This is a section consisting of one or more resource records. It is present only on response messages. This section gives information (domain name) about one or more authoritative servers for the query.

Additional Information Section

This is a section consisting of one or more resource records. It is present only on response messages. This section provides additional information that may help the resolver. For example, a server may give the domain name of an authoritative server to the resolver in the authoritative section, and include the IP address of the same authoritative server in the additional information section.

TYPES OF RECORDS

Two types of records are used in DNS. The question records are used in the question section of the query and response messages. The resource records are used in the answer, authoritative, and additional information sections of the response message.

Question Record

A question record is used by the client to get information from a server. This contains the domain name. Resource records are also what is returned by the server to the client

Resource Record

Each domain name (each node on the tree) is associated with a record called the resource record. The server database consists of resource records.

REGISTRARS

New domains added to DNS through a Registrar, a commercial entity accredited by ICANN. (Internet Corporation For Assigned Names and Numbers).A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged. Today, there are many registrars; their names and addresses can be found a **http://www.intenic.net**.To register, the organization needs to give the name of its server and the IP address of the server.

For example, a new commercial organization named *wonderful with a* server named *ws and IP address 200.200.200.5 needs to give the following information* to one of the registrars:

Domain name: WS.wonderful.com ,IP address: 200.200.200.5

ENCAPSULATION

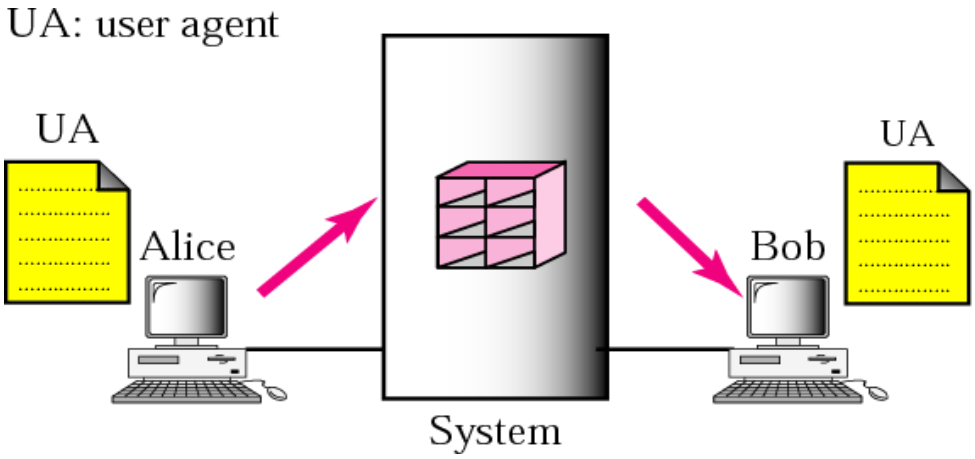
DNS can use either UDP or TCP. In both cases the well-known port used by the server is port 53. UDP is used when the size of the response message is less than 512 bytes because most UDP packages have a 512-byte packet size limit. If the size of the response message is more than 512 bytes, a TCP connection is used. In that case, one of two scenarios can occur:If the resolver has prior knowledge that the size of the response message is more than 512 bytes, it uses the TCP connection.

EMAIL

The general architecture of an e-mail system including the three main components:

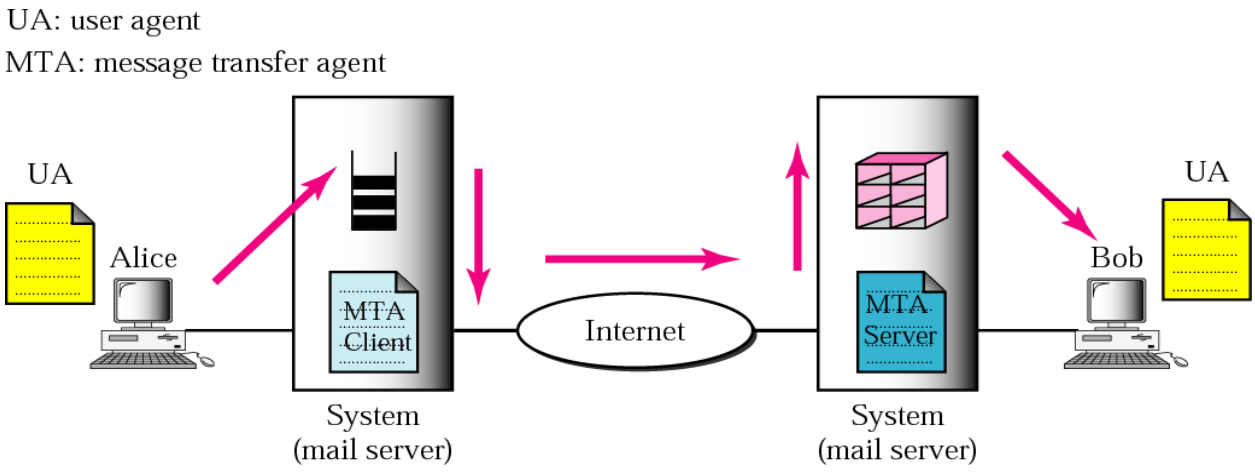
- ✓ User agent
- ✓ Message Transfer Agent
- ✓ Message Access Agent.

First Scenario



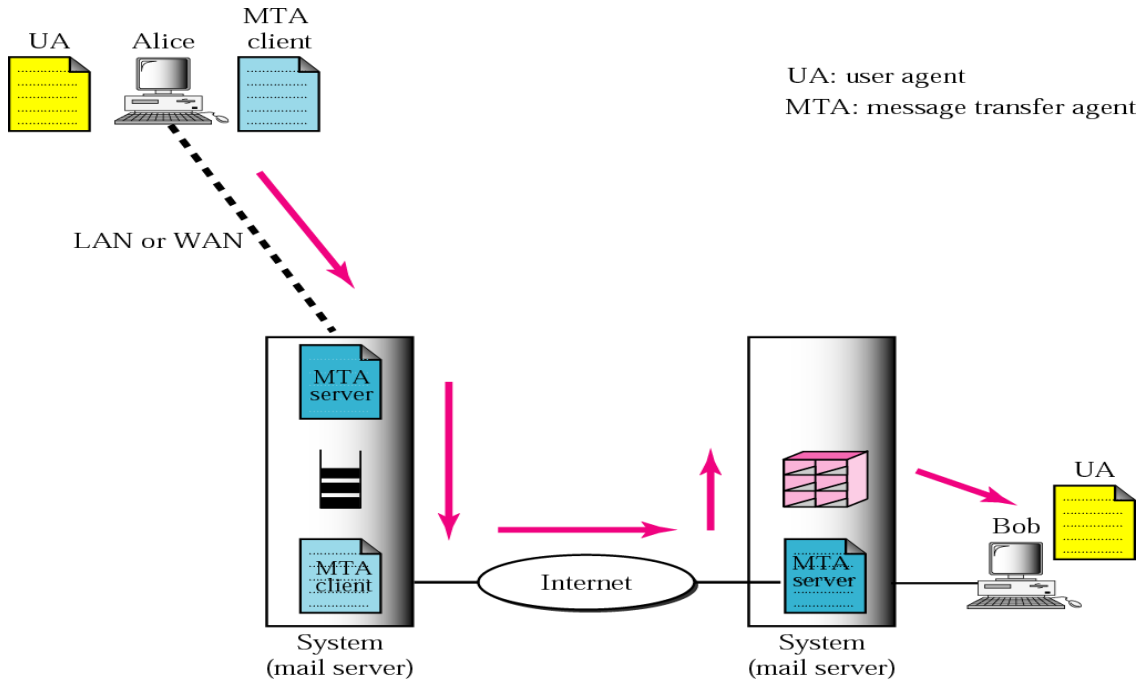
First scenario: sender and receiver on same system so need only two UAs.

Second scenario:



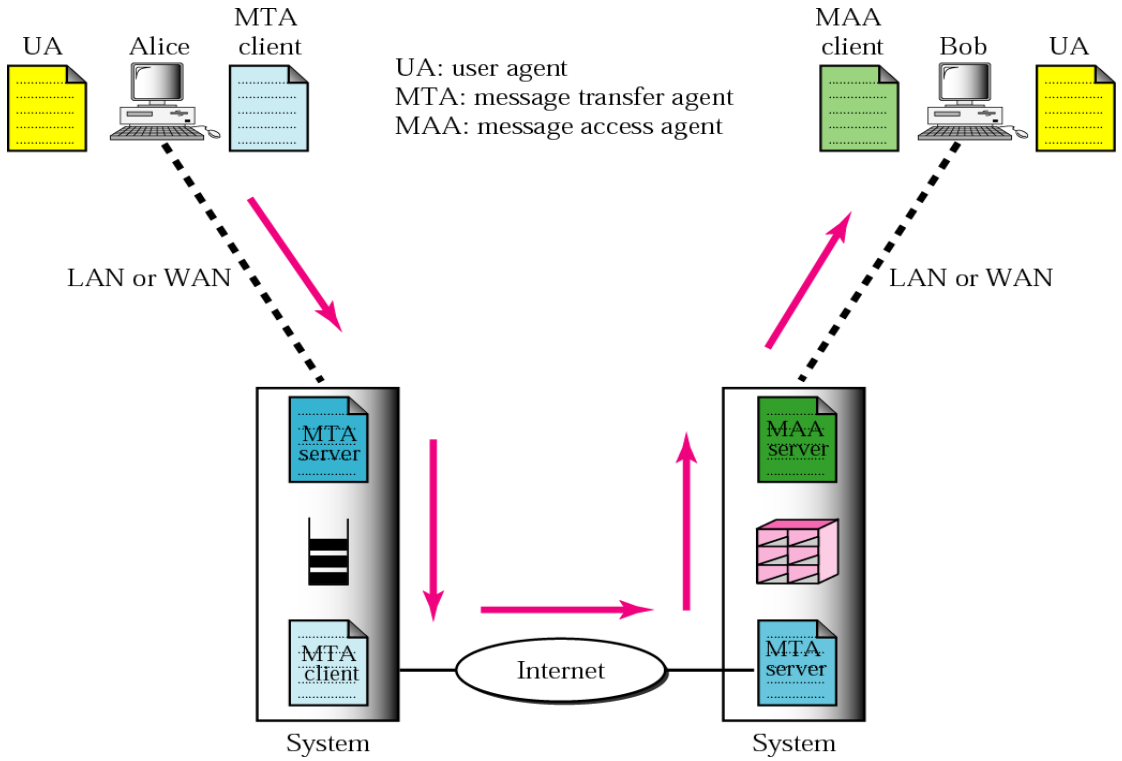
Second scenario: sender and receiver on different systems so need two UAs and pair of MTAs

Third scenario:



Third scenario: When the sender is connected to the mail server via a LAN or a WAN we need two UAs and two pairs of MTAs (client and server).

Fourth scenario:



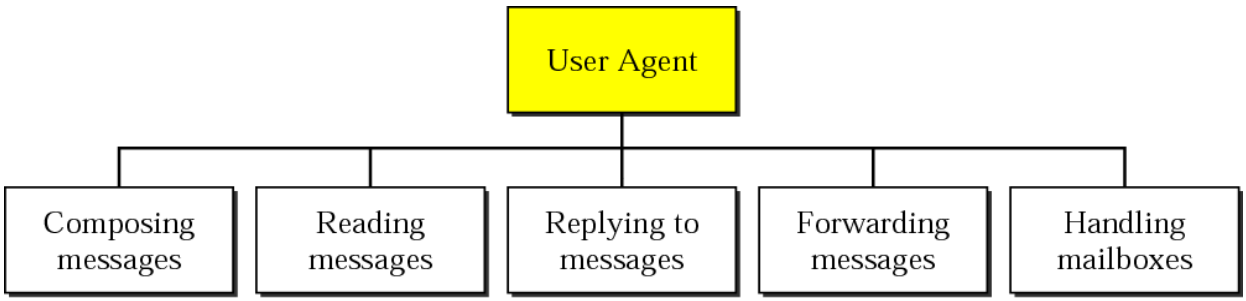
When both sender and receiver are connected to the mail server via a LAN or a WAN, we need two UAs, two pairs of MTAs (client and server),and a pair of MAAs (client and server).

1. When the sender and the receiver of an e-mail are on the same system, we need only two user agents.
2. When the sender and the receiver of an e-mail are on different systems, we need two UAs and a pair of MTAs (client and server).
3. When the sender is connected to the mail server via a LAN or a WAN, we need two *UAs and two pairs of MTAs (client and server).*
4. When both sender and receiver are connected to the mail server via a LAN or a WAN, we need two UAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server).

User Agent

A user agent is a software package (program) that composes, reads, replies to, and forwards messages.

It also handles mailboxes



Composing Messages

A user agent helps the user compose the e-mail message to be sent out. Most user agents provide a template on the screen to be filled in by the user.Some even have a built-in editor that can do spell checking, grammar checking, and other tasks expected from a sophisticated word processor. A user, of course, could alternatively use his or her favorite text editor or word processor to create the message and import it, or cut and paste it, into the user agent template

Reading Messages

The second duty of the user agent is to read the incoming messages. When a user invokes a user agent, it first checks the mail in the incoming mailbox. Most user agents show a one-line summary of each received mail.

Each e-mail contains the following fields.

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

Replying to Messages

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

The reply message may contain the original message (for quick reference) and the new message

Forwarding Messages

Replying is defined as sending a message to the sender or recipients of the copy. Forwarding is defined as sending the message to a third party. A user agent allows the receiver to forward the message, with or without extra comments, to a third party.

Handling Mailboxes

A user agent normally creates two mailboxes: an inbox and an outbox. Each box is a file with a special format that can be handled by the user agent. The inbox keeps all the received e-mails until they are deleted by the user. The outbox keeps all the sent e-mails until the user deletes them. Most user agents today are capable of creating customized mailboxes

User Agent Types

There are two types of user agents: command-driven and GUI-based

Command-Driven

A command-driven user agent normally accepts a one-character command from the keyboard to perform its task. For example, a user can type the character r, at the command prompt, to reply to the sender of the message, or type the character R to reply to the sender and all recipients.

Some examples of command-driven user agents are **mail, pine, and elm.**

GUI-Based

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. They have

graphical components such as icons, menu bars, and windows that make the services easy to access. Some examples of GUI-based user agents are **Eudora, Microsoft's Outlook, and Netscape**

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*

Envelope

The envelope usually contains the sender and the receiver addresses.

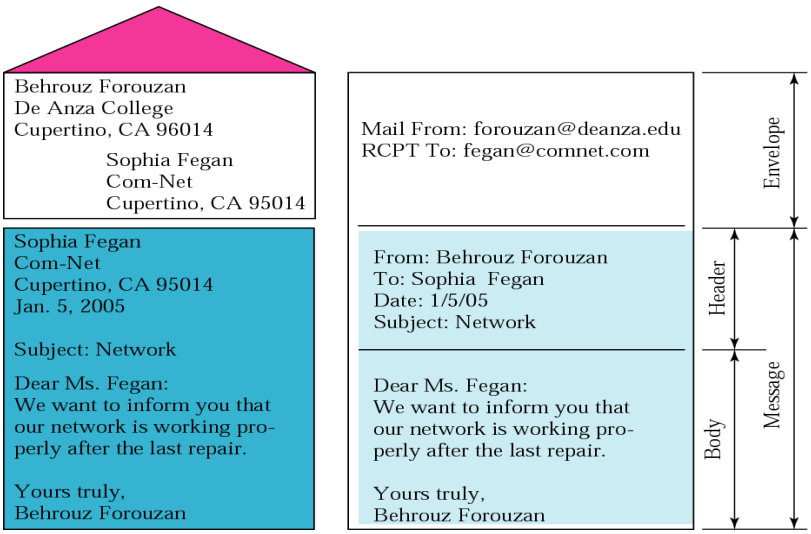
Message

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 (such as encoding type).The body of the message contains the actual information to be read by the recipient.

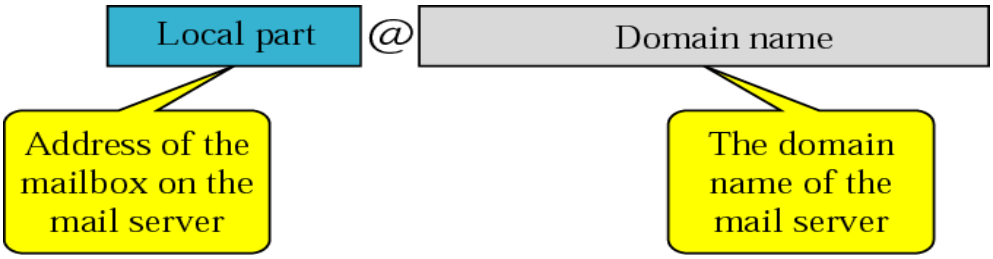
Receiving Mail

The user agent is triggered by the user (or a timer). If a user has mail, the **User Agent 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.

Format of Email



Email Address



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

Local Part

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.

Domain Name

The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; the hosts 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 (for example, the name of the organization).

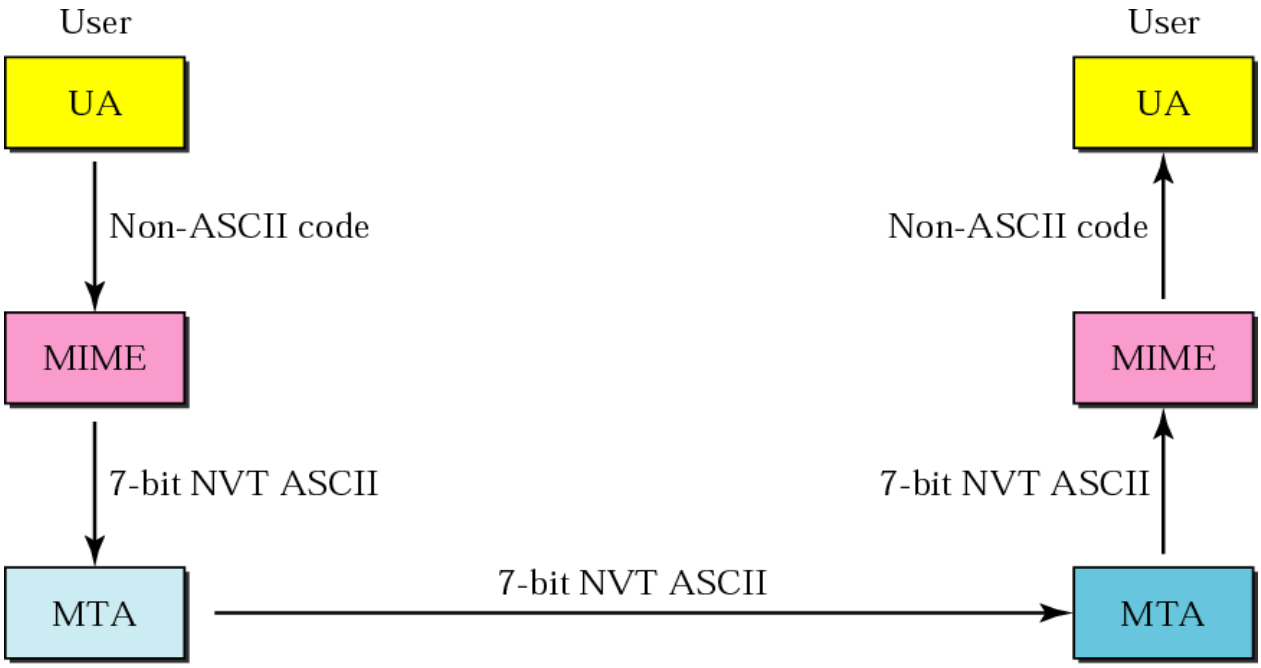
Mailing List

Electronic mail allows one name, an alias, to represent several different e-mail addresses; this is called a mailing list. Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry in the list, must be prepared and handed to the MTA. If there is no mailing list for the alias, the name itself is the receiving address and a single message is delivered to the mail transfer entity.

MIME

Electronic mail can send messages only in NVT 7-bit ASCII format. It has some limitations.

It cannot be used for languages that are not supported by 7-bit. ASCII characters (such as French, German, Hebrew, Russian, Chinese, and Japanese). Also, it cannot be used to send binary files or video or audio data. Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail.



MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client MTA to be sent through the Internet. The message at the receiving side is transformed back to the original data. MIME is a set of software functions that transforms non-ASCII data (stream of bits) to ASCII data and vice versa

MIME defines five headers that can be added to the original e-mail header section to define the transformation parameters:

1. MIME-Version

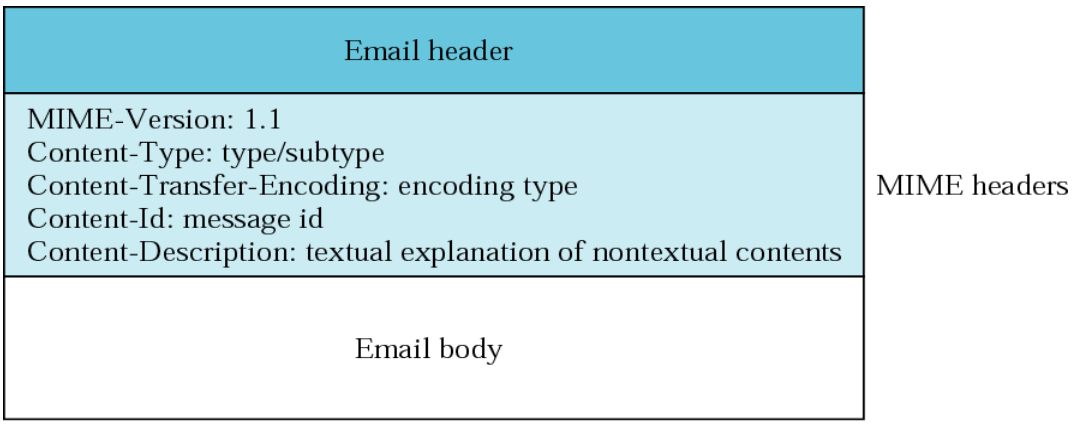
2. Content-Type

3. Content-Transfer-Encoding

4. Content-Id

5. Content-Description

MIME HEADER



Message Transfer Agent: SMTP

The actual mail transfer is done through message transfer agents. To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA. The formal protocol that defines the MTA client and server in the Internet is called the Simple Mail Transfer Protocol (SMTP).

SMTP is used two times, between the sender and the sender's mail server and between the two mail servers.SMTP simply defines how commands and responses must be sent back and forth

Commands and Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server

Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.

Commands

Commands are sent from the client to the server. It consists of a keyword followed by zero or more arguments.SMTP defines 14 commands. The first five are mandatory; every implementation must support these five commands. The next three are often used and highly recommended. The last six are seldom used.

Table 26.7 *Commands*

Keyword	*Argument(s)*
HELO	Sender's host name
MAIL FROM	Sender of the message
RCPTTO	Intended recipient of the message
DATA	Body of the mail
QUIT	
RSET	
VRFY	Name of recipient to be verified
NOOP	
TURN	
EXPN	Mailing list to be expanded
HELP	Command name

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.

Table 26.8 *Responses*

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
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Mail Transfer Phases

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

- Connection establishment
- Mail transfer
- Connection termination.

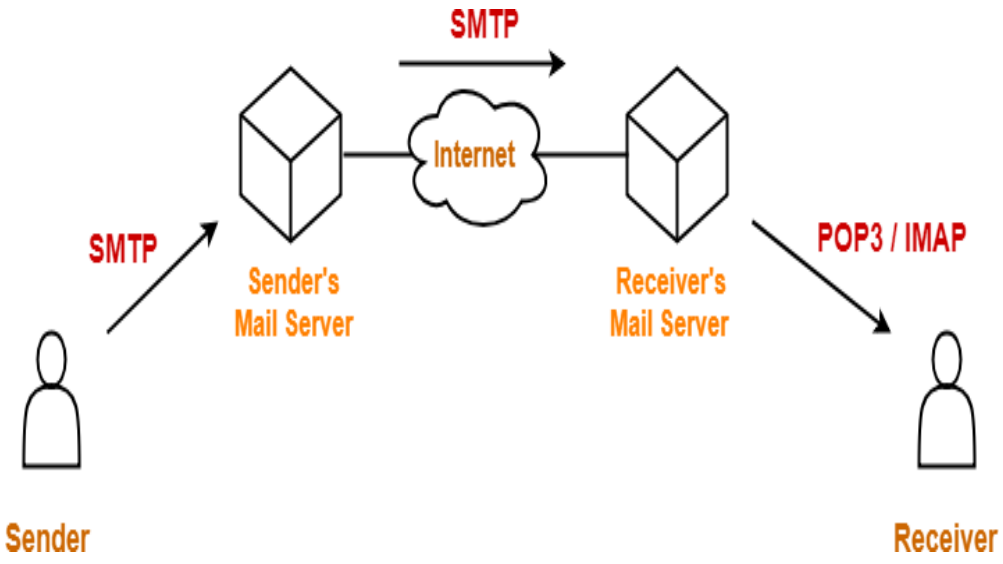
- Two message access protocols are available: Post Office Protocol, version 3 (POP3) and Internet Mail Access Protocol, version 4 (IMAP4).

Simple Mail Transfer Protocol

SMTP (**Simple Mail Transfer Protocol)** is an application layer protocol. It is used for sending the emails efficiently and reliably over the internet.

Working-

SMTP server is always on a listening mode. Client initiates a TCP connection with the SMTP server. SMTP server listens for a connection and initiates a connection on that port. The connection is established. Client informs the SMTP server that it would like to send a mail. Assuming the server is OK, client sends the mail to its mail server. Client's mail server use DNS to get the IP Address of receiver's mail server. Then, SMTP transfers the mail from sender's mail server to the receiver's mail server.



While sending the mail, SMTP is used two times-

1. Between the sender and the sender's mail server

2. Between the sender's mail server and the receiver's mail server

To receive or download the email, another protocol is needed between the receiver's mail server and the receiver.The most commonly used protocols are **Post Office Protocol 3 (**POP3) and **Internet Message Access Protocol** (IMAP)

Characteristics of SMTP-

- SMTP is a push protocol.
- SMTP uses TCP at the transport layer.
- SMTP uses port number 25.
- SMTP uses persistent TCP connections, so it can send multiple emails at once.
- SMTP is a connection-oriented protocol.
- SMTP is an in-band protocol.
- SMTP is a stateless protocol.
- SMTP is a pure **text-based** protocol.
- SMTP can only handle the messages containing **7-bit ASCII text**.
- SMTP cannot transfer other types of data like images, video, audio etc.
- SMTP cannot transfer **executable files and binary objects**.
- SMTP cannot transfer the text data of other languages like French, Japanese, Chinese etc. (since they are represented in 8-bit codes)

MIME extends the limited capabilities of email.

Multipurpose Internet Mail Extension (MIME) is an extension to the internet emailprotocol. It extends the limited capabilities of email by enabling the users to send and receive graphics, audio files, video files etc in the message. MIME was specially designed for SMTP.

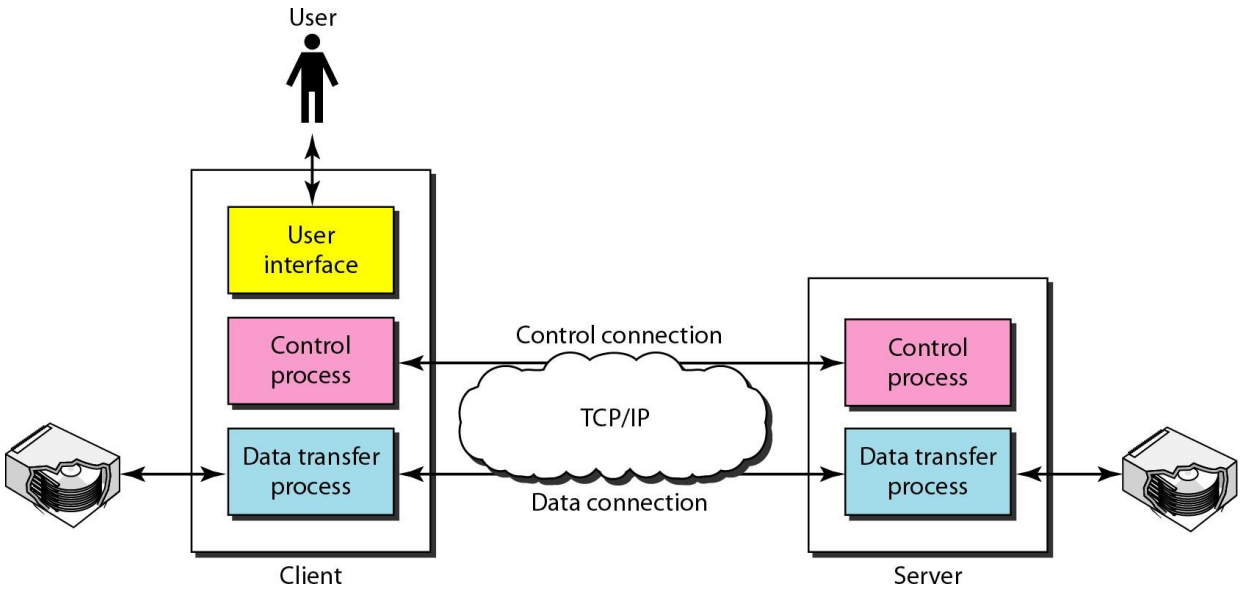
File Transfer Protocol

File Transfer Protocol Transfers files from one computer to another is the standard mechanism provided by TCP/IP for copying a file from one host to another

FTP differs from other client/server applications.FTP establishes two connections between the hosts. One connection is used for data transfer, the other for control information (commands and responses). Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication.FTP uses two well-known TCP ports: **Port**

21 is used for the control connection, and **port 20** is used for the data connection

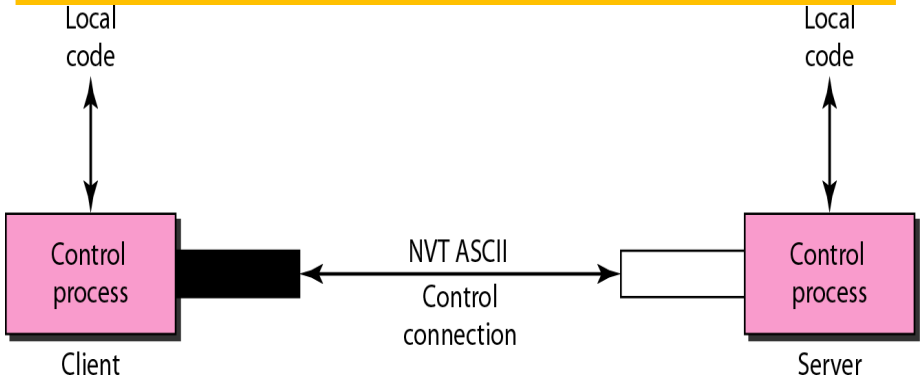
Figure : FTP



The client has three components: user interface, client control process, and the client datatransfer process. The server has two components: the server control process and the server data transfer process. The control connection is made between the control processes. The data connection is made between the data transfer processes. The control connection remains connected during the entire interactive FTP session.The data connection is opened and then closed for each file transferred. It opens each time commands that involve transferring files are used, and it closes when the file is transferred. In other words, when a user starts an FTP

session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred.

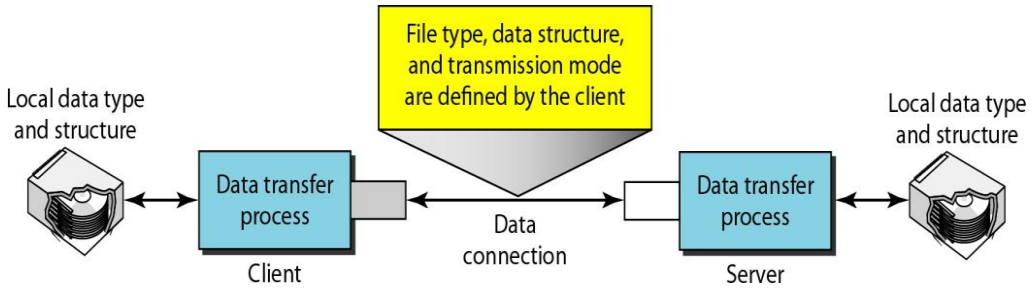
Figure **Using the control connection**



Communication over Control Connection

FTP uses the same approach as SMTP to communicate across the control connection. It uses the 7-bit ASCII character set. Communication is achieved through commands and responses. This simple method is adequate for the control connection because we send one command (or response) at a time. Each command or response is only one short line, so we need not worry about file format or file structure.Each line is terminated with a two-character end-of-line token.

Figure *Using the data connection*



Communication over Data Connection

The purpose of the data connection is different from that of the control connection.

Files are transferred through the data connection. File transfer occurs over the data connection under the control of the commands sent over the control connection.

File transfer in FTP means one of three things:

1.A file is to be copied from the server to the client. This is called retrieving a file.

It is done under the supervision of the RETR command

2. A file is to be copied from the client to the server. This is called **storing** *a file.*

It is done under the supervision of the STOR command

3. A list of directory or file names is to be sent from the server to the client.

This is done under the supervision of the LIST command.

File Type

FTP can transfer one of the following file types across the data connection: an ASCII file,

EBCDIC file, or image file

Data Structure

FTP can transfer a file across the data connection by using one of the following interpretations about the structure of the data: file structure, record structure, and page structure. In the file structure format, the file is a continuous stream of bytes.

In the record structure, the file is divided into records. This can be used only with text files. In the page structure, the file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly or sequentially

Transmission Mode

FTP can transfer a file across the data connection by using one of the following three transmission modes:

- Stream mode
- Block mode
- Compressed mode.

The **stream mode** is the default mode. Data are delivered from FTP to TCP as a continuous stream of bytes. TCP is responsible for chopping data into segments of appropriate size. If the data are simply a stream of bytes (file structure), no end-of-file is needed. End-of-file in this case is the closing of the data connection by the sender. If the data are divided into records (record structure), each record will have a 1-byte end of- record (EOR) character and the end of the file will have a 1-byte end-of-file (EOF) character.

In **block mode**, data can be delivered from FTP to TCP in blocks. In this case, each blockis preceded by a 3-byte header. The first byte is called the block descriptor; the next 2 bytes define the size of the block in bytes

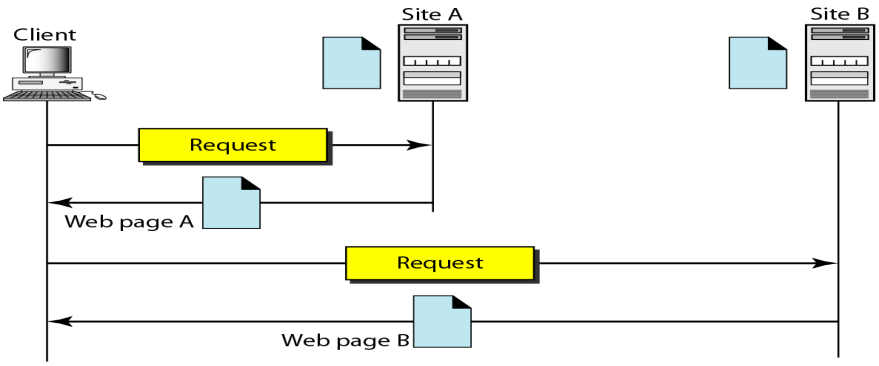
In **the compressed mode**, if the file is big, the data can be compressed. The compression method normally used is run-length encoding. In this method, consecutive appearances ofa data unit are replaced by one occurrence and the number of repetitions.

In a text file, this is usually spaces (blanks). In a binary file, null characters are usually compressed

WWW (World Wide Web)

The WWW 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 documents, referred to as Web pages. Each Web page can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers.

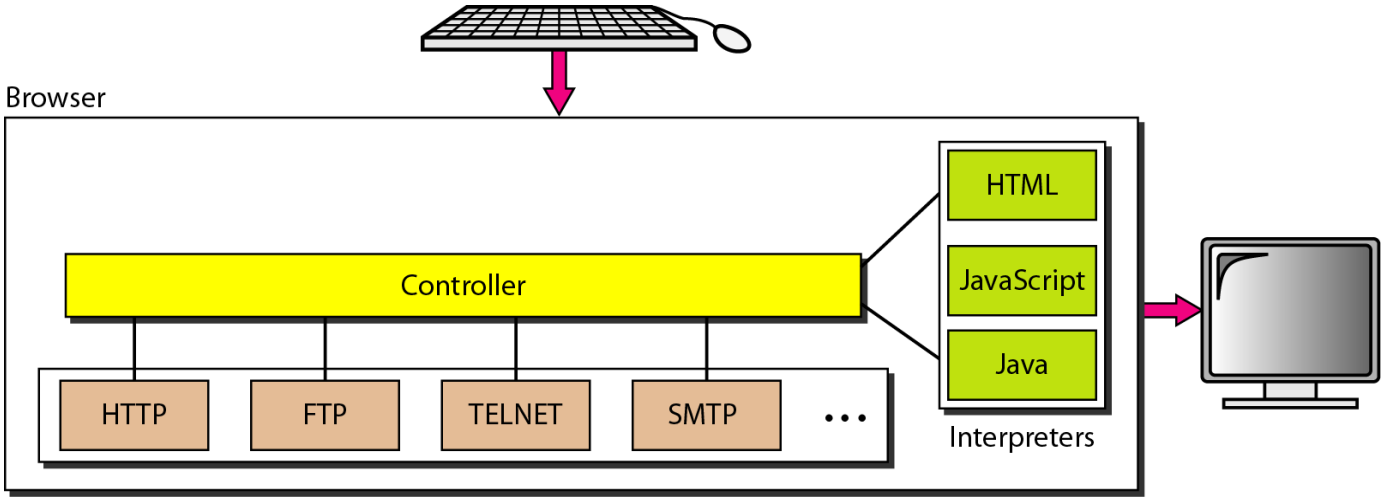
Figure Architecture of WWW



Client (Browser)

A variety of vendors offer commercial browsers that interpret and display a Web document, and all use nearly the same architecture. Each browser usually consists of three parts: a controller, client protocol, and interpreters. The controller receives input from the keyboard or the mouse and uses the client programs to access the document. After the document has been accessed, the controller uses one of the interpreters to display the document on the screen. The client protocol can be one of the protocols described previously such as FTP or HTTP .The interpreter can be HTML, Java, or JavaScript, depending on the type of document.

Figure *Browser*



Server

The Web page is stored at the server. Each time a client request arrives the server, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than disk. 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.

Cookies

The World Wide Web was originally designed as a stateless entity.

A client sends a request; a server responds. Their relationship is over. The original design of WWW, retrieving publicly available documents, exactly fits this purpose.

Today the Web has other functions; some are listed here.

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

For these purposes, the cookie mechanism was devised.

Creation and Storage of Cookies

The creation and storage of cookies depend on the implementation; however, the principle is the same

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

The information may include the domain name of the client, the contents of the cookie (information

the server has gathered about the client such as name, registration number, and so on), a timestamp, and other information depending on the implementation.

2. The server includes the cookie in the response that it sends to the client.

3. When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the domain server name

WEB DOCUMENTS

The documents in the WWW can be grouped into three broad categories: static, dynamic, and active. The category is based on the time at which the contents of the document are determined.

Static Documents

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

The client can get only a copy of the document. In other words, the contents of the file are determined when the file is created, not when it is used. Of course, the contents in the server can be changed, but the user cannot change them. When a client accesses the document, a copy of the document is sent. The user can then use a browsing program to display the document

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 output 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 can vary from one request to another.

Active Documents

For many applications, we need a program or a script to be run at the client site. These are called active documents.

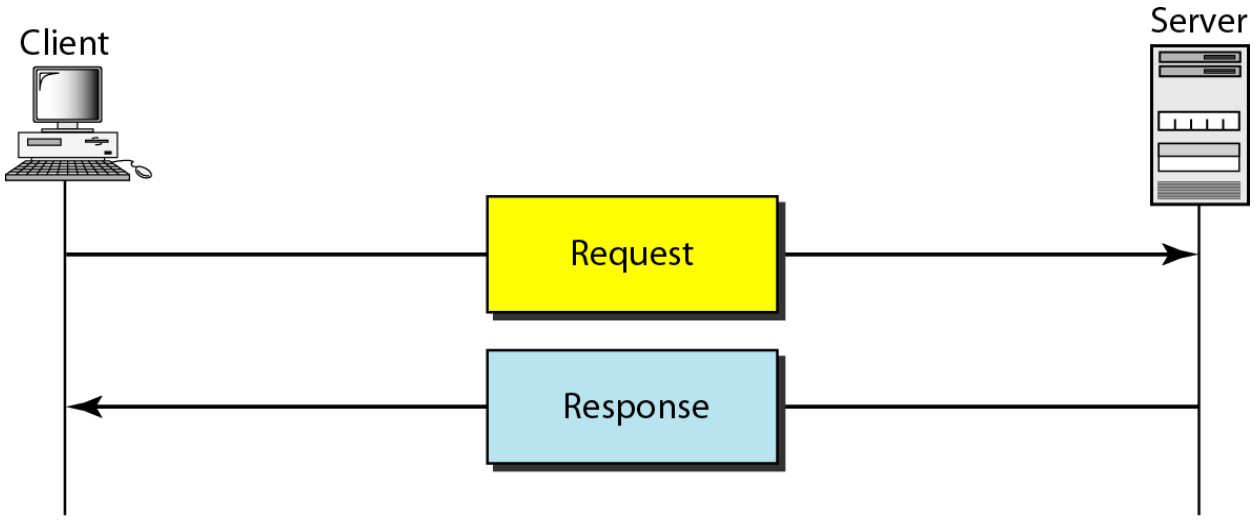
HTTP (Hypertext Transfer Protocol)

The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. HTTP functions as a combination of FTP and SMTP. It is similar to FTP because it transfers files and uses the services of TCP. However, it is much simpler than FTP because it uses only one TCP connection. There is no separate control connection only data are transferred between the client and the server HTTP is like SMTP because the data transferred between the client and the server look like SMTP messages. In addition, the format of the messages is controlled by MIME-like headers. **(Multipurpose Internet Mail Extensions)**Unlike SMTP, the HTTP messages are not destined to be read by humans; they are read and interpreted by theHTTP server and HTTP client (browser).SMTP messages are

stored and forwarded, but HTTPmessages are delivered immediately. The commands from the client to the server areembedded in a request message. The contents of the requested file or other information are embedded in a response message. HTTP uses the services of TCP on well-known port 80.

HTTP Transaction

HTTP uses the services of TCP, HTTP itself is a stateless protocol(HTTP server need not keeptrack of any state information). The client initializes the transaction by sending a request message. The server replies by sending a response



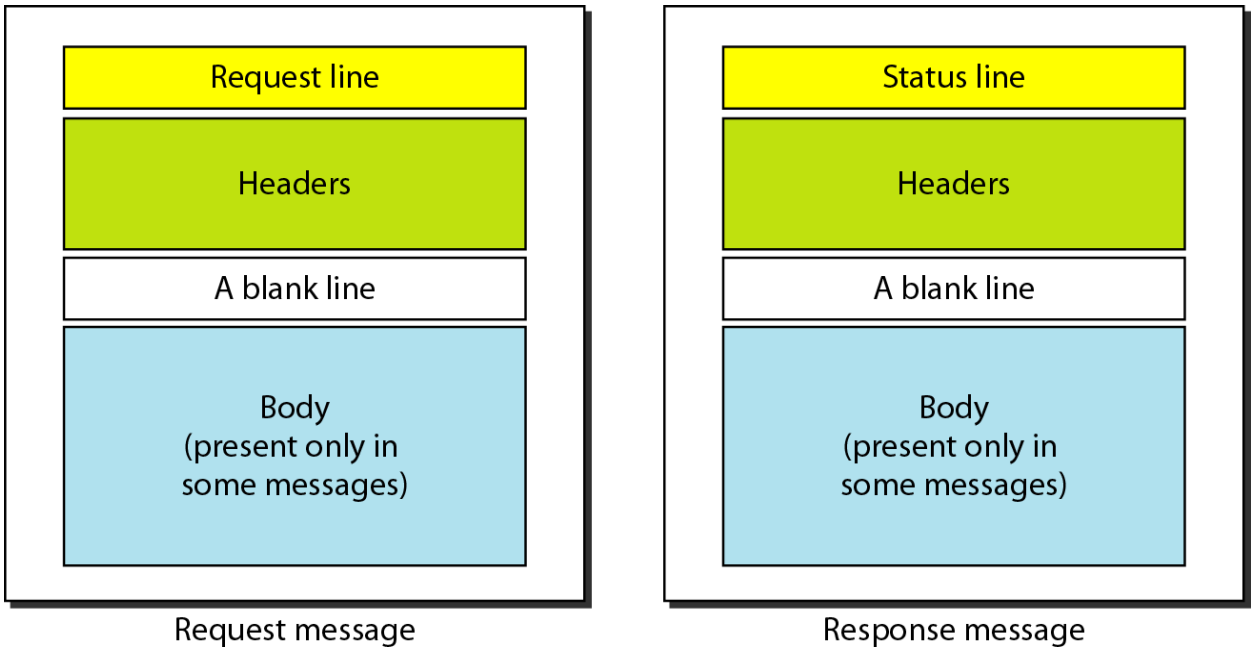
Messages

The formats of the request and response messages are similar; The request message consists of arequest line, a header, and sometimes a body.

A response message consists of a status line, a header, and sometimes a body.

Request and Status Lines

The first line in a request message is called a request line; The first line in the response messageis called the status line



Request type.

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

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

Uniform Resource Locator

A client that wants to access a Web page needs the address. To facilitate the access of documentsdistributed throughout the world, HTTP uses locators. The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet. The URL defines four things:

- Protocol
- Host computer
- Port
- Path

The protocol is the client/server program used to retrieve the document. Many different protocolscan retrieve a document; among them are FTP or HTTP. The host is the computer on which the information is located. The URL can optionally contain the port number of the server.

If the **port is** included, it is inserted between the host and the path, and it is separated from the host by a colon. Path is the pathname of the file where the information is located.

Version. The most current version of HTTP is 1.1.

Status code.

This field is used in the response message.

The status code field is similar to those in the FTP and the SMTP

protocols.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
- 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

Status phrase.

This field is used in the response message. It explains the status code in text form

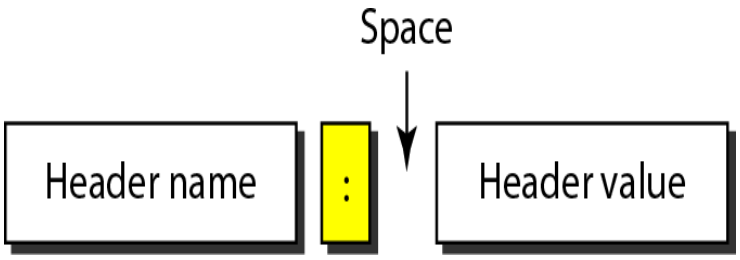
Status Codes

Code	*Phrase*	*Description*
	Informational	
100	Continue	The initial part of the request has been received, and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
	Success	
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.

Header

The header exchanges additional information between the client and the server.The header canconsist of one or more header lines. Each header line has a header name, a colon, a space, and aheader value

HEADER FORMAT



A header line belongs to one of four categories:

- General header
- Request header
- Response header
- Entity header.

A request message can contain only general, request, and entity headers.

A response message, on the other hand, can contain only general, response, and entity headers.

General header

The general header gives general information about the message and can be present in both arequest and a response.

Request header

The request header can be present only in a request message. It specifies the client's configuration and the client's preferred document format.

Header	*Description*
Accept	Shows the medium format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the server
If-modified-since	Sends the document if newer than specified date
If-match	Sends the document only if it matches given tag
If-non-match	Sends the document only if it does not match given tag
If-range	Sends only the portion of the document that is missing
If-unmodified-since	Sends the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

Response header :

The response header can be present only in a response message. It specifies the

server'sconfiguration and special information about the request.

Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

Entity header

The entity header gives information about the body of the document. Although it is mostly present in response messages, some request messages, such as POST or PUT methods, thatcontain a body also use this type of header.

Header	*Description*
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

Body

The body can be present in a request or response message. Usually, it contains the document tobe sent or received

Persistent Versus Nonpersistent Connection

HTTP prior to version 1.1 specified a nonpersistent connection, while a persistent connection isthe default in version 1.1.

Nonpersistent Connection

In a nonpersistent connection, one TCP connection is made for each

request/response.The following lists 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 theconnection.

In this strategy, for N different pictures in different files, the connection must be opened and closed N times. The nonpersistent strategy imposes high overhead on the server because the server needs N different buffers and requires a slow start procedure each time a connection isopened

Persistent Connection

HTTP version 1.1 specifies a persistent connection by default. 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. This is the case 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

Proxy Server

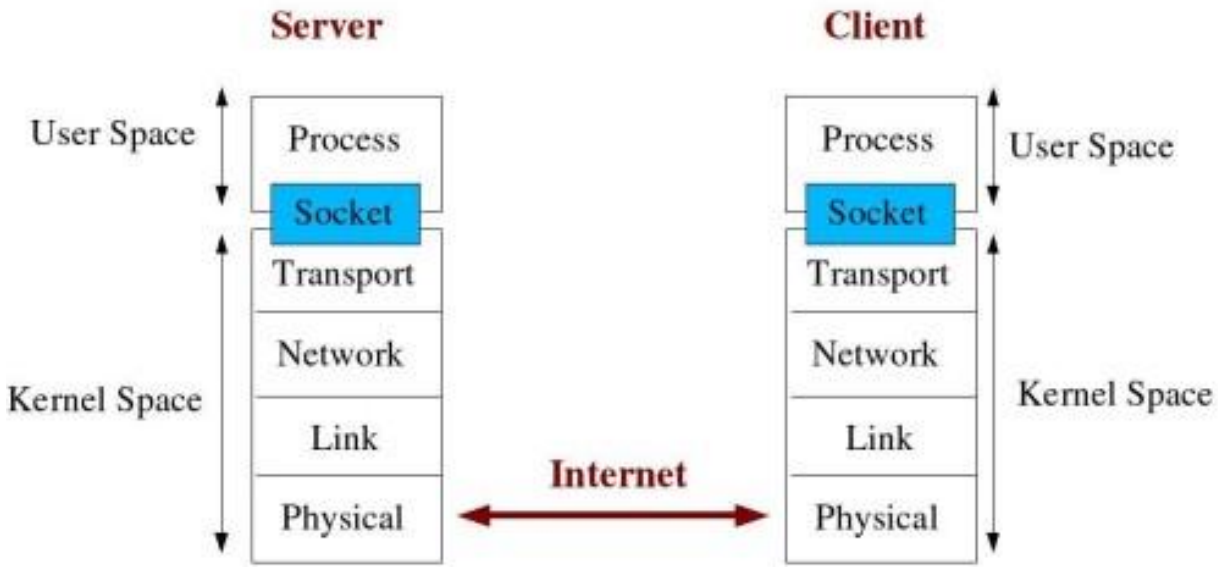
HTTP supports proxy servers. A proxy server is a computer that keeps copies of responses to recent requests. The HTTP client sends a request to the proxy server. The proxy server checks its cache. If the response is not stored in the cache, the proxy server sends the request to the corresponding server. Incoming responses are sent to the proxy server and stored for future requests from other clients. The proxy server reduces the load on the original server, decreases traffic, and improves latency. However, to use the proxy server, the client must be configured toaccess the proxy instead of the target server.

SOCKET PROGRAMMING

Socket:

An interface between an application process and transport layer. The application process can send/receive messages to/from another application process via a socket. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.

Socket Description



Types of Sockets

- Internet Sockets,
- unix sockets,
- X.25 sockets etc

Types of Internet Sockets

● Stream Sockets (SOCK_STREAM)

A stream socket uses the Transmission Control Protocol (TCP) for sending messages

● Datagram Sockets (SOCK_DGRAM)

A datagram socket uses the User Datagram Protocol (UDP) for sending messages. UDP is a much simpler protocol as it does not provide any of the delivery guarantees that TCP does.

Using Internet Sockets, we can identify a host using an IP address and a port number. The IP address uniquely identifies a machine while the port number identifies the application we want tocontact at that machine.

Creating a Socket

Now for the socket() function.

```
SOCKET socket(
    int af,       // address family
    int type,     // socket type
    int protocol // protocol type
);
```

socket() function

Binding a Socket

The purpose of binding is to associate a local address with a socket

```
int bind(
    SOCKET s,                       // socket to bind to
    const struct sockaddr *name, // local address info
    int namelen                     // length of *name
);
```

bind() function

Connecting to a Socket

For a client to send messages to a remote host (i.e. the server) we need to connect to thathost.

```
int connect(
    SOCKET s,                      // socket to connect to
    const struct sockaddr *name, // remote address info
    int namelen                    // length of *name
);
```

connect() function

Server need to listen for new connections requests from clients

```
int listen(
    SOCKET s,    // socket to listen on
    int backlog // max. incoming queue length
);
```

listen() function

The backlog integer is used to specify the maximum length of the incoming

connectionqueue

When a connection request is received the server to needs to accept it to service theclient.

```
SOCKET accept(
   SOCKET          s,       // socket the server listens on
   struct sockaddr *addr,   // incoming connection details
   int             *addrlen // length fo *addr
);
```

accept() function

Now we are at the point where our server has accepted a client's connection request andcreated a new socket for it to accept and send messages.