UNIT-5
Principles of Network Applications

1. Principles of Network Applications

- At the core of network application development is writing programs that run on different end systems and communicate with each other over the network. For example, in the Web application there are two distinct programs that communicate with each other: the browser program running in the user's host (desktop, laptop, tablet, smart phone, and so on); and the Web server program running in the Web server host.
- The application layer provides services to the user. Communication is provided using a logical connection, which means that the two application layers assume that there is an imaginary direct connection through which they can send and receive messages.

Providing Services: The application layer provides many services, including:

- Simple Mail Transfer Protocol
- File transfer
- Web surfing
- Email clients
- Network data sharing
- Virtual terminals
- Various file and data operations

Network Application Architecture: The application architecture is designed by the application developer and dictates how the application is structured over the various end systems. In choosing the application architecture, an application developer will likely draw on one of the two predominant architectural paradigms used in modern network applications: the **client-server architecture or the peer-to-peer (P2P) architecture**

In a **client-server architecture**, there is an always-on host, called the server, which services requests from many other hosts, called clients. A classic example is the Web application for which an always-on Web server services requests from browsers running on client hosts. When a Web server receives a request for an object from a client host, it responds by sending the requested object to the client host.

Some of the better-known applications with a client-server architecture include the Web, FTP, Telnet, and e-mail.

A new paradigm, called the **peer-to-peer paradigm** (often abbreviated *P2P paradigm*) has emerged to respond to the needs of some new applications. In this paradigm, there is no need for a server process to be running all the time and waiting for the client processes to connect. The responsibility is shared between peers. A computer connected to the Internet can provide service at one time and receive service at another time. A computer can even provide and receive services at the same time.

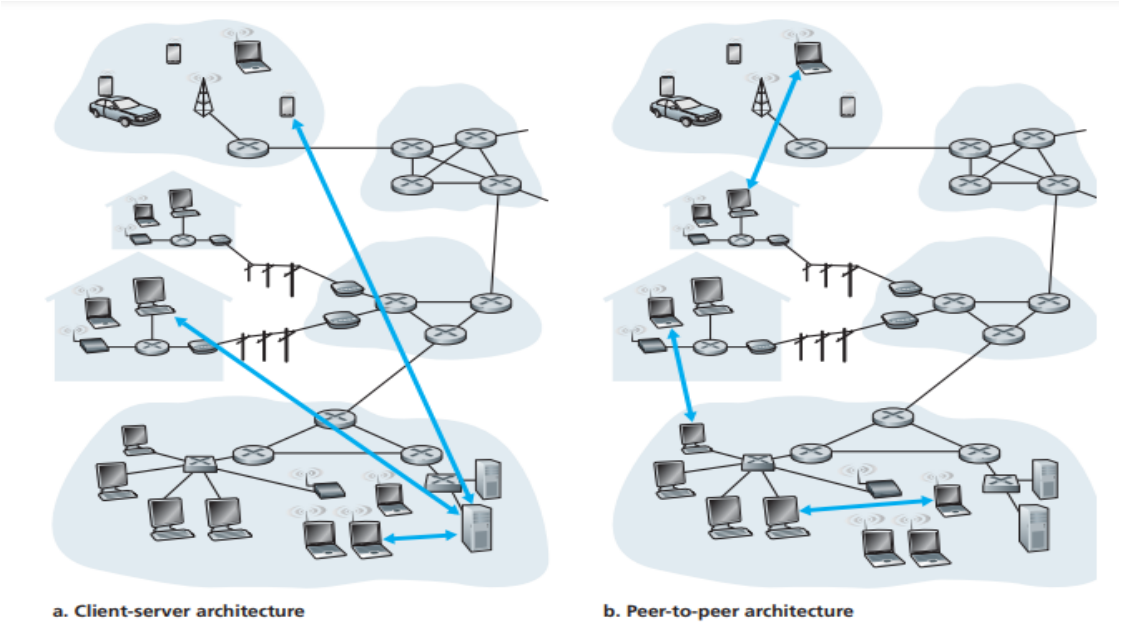


Fig: a) client-server architecture,b) peer-to-peer (P2P) architecture

Processes communication

- In a client-server paradigm, communication at the application layer is between two running application programs called ***processes****:* a *client* and a *server*.
- A client is a running program that initializes the communication by sending a request; a server is another application program that waits for a request from a client.
- The server handles the request received from a client, prepares a result, and sends the result back to the client.
- This definition of a server implies that a server must be running when a request from a client arrives, but the client needs to be run only when it is needed.
- This means that if we have two computers connected to each other somewhere, we can run a client process on one of them and the server on the other.

The Interface Between the Process and the Computer Network

Several APIs have been designed for communication: **socket interface,**

Socket interface is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

A **socket** is the interface between the application layer and the transport layer within a host. It is also referred to as the Application Programming Interface (API) between the application and the network, since the socket is the programming interface with which network applications are built.

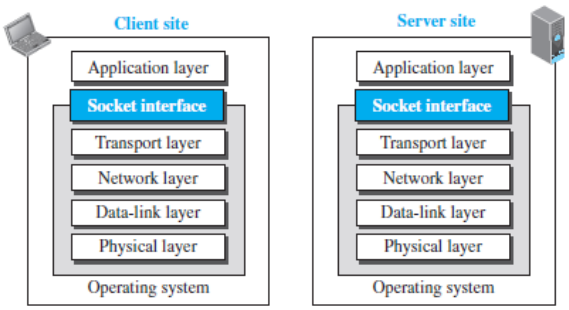


Fig: Position of the socket interface

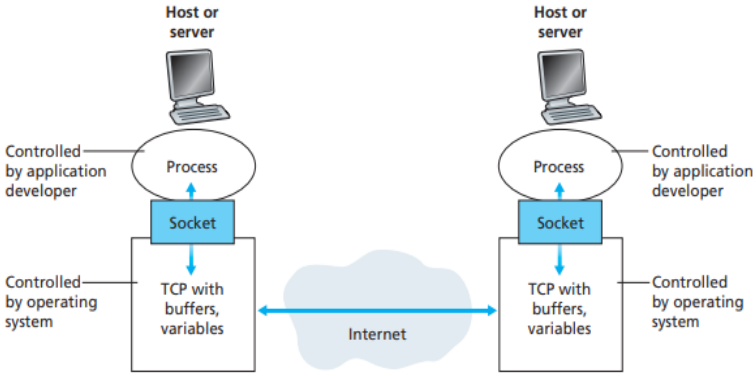


Fig: Application processes, sockets and underlying transpport protocol

2. The Web and HTTP

World Wide Web (WWW)

The **Hyper Text Transfer Protocol (HTTP),** the Web's application-layer protocol, is at the heart of the Web. HTTP is implemented in two programs: a client program and a server program. The client program and server program talk to each other by exchanging HTTP messages.

- A **Web page** (also called a document) consists of objects. An **object** is simply a file—such as an HTML file, a JPEG image, a Java applet, or a video clip—that is addressable by a single URL. Most Web pages consist of a **base HTML file** and several referenced objects.

For example, if a Web page contains HTML text and five JPEG images, then the Web page has six objects: the base HTML file plus the five images.

- The base HTML file references the other objects in the page with the objects' URLs. Each **URL has two components**: the hostname of the server that houses the object and the object's path name. For example, the URL

http://www.someSchool.edu/someDepartment/picture.gif

has **www.someSchool.edu** for a hostname and **/someDepartment/ picture.gif** for a path name.

- The **Web browsers** (such as Internet Explorer and Firefox) implement the client side of HTTP, we will use the words browser. **Web servers**, which implement the server side of HTTP, house Web objects, each addressable by a URL. Popular Web servers include Apache and Microsoft Internet Information Server.

- HTTP defines how Web clients request Web pages from Web servers and how servers transfer Web pages to clients. We discuss the **interaction between client and server** in detail is illustrated in Figure.When a user requests a Web page (for example, clicks on a hyperlink), the browser sends HTTP request messages for the objects in the page to the server. The server receives the requests and responds with HTTP response messages that contain the objects.

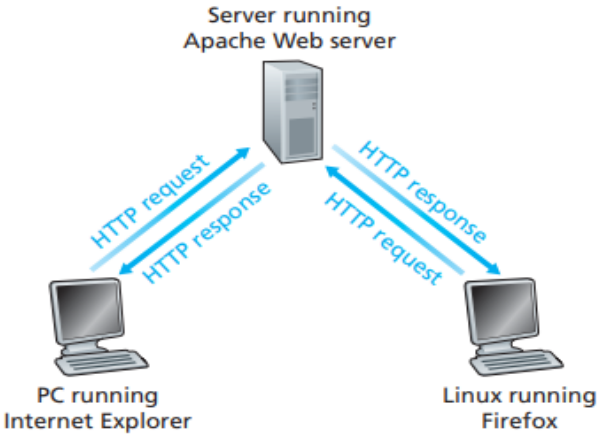


Fig: HTTP request-response behaviour

- ***Web Documents:*** The documents in the WWW can be grouped into three broad categories: ***static, dynamic,*** *and* ***active.***

- ***Static Documents:*** **Static documents** are fixed-content documents that are created and stored in a server. The client can get a copy of the document only.
- ***Dynamic Documents:*** A **dynamic document** is created by a web server whenever a browser requests the document.

When a request arrives, the web server runs an application program or a script that creates the dynamic document. The server returns the result of the program or script as a response to the browser that requested the document.

- ***Active Documents:*** For many applications, we need a program or a script to be run at the client site. These are called ***active documents***. For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user.

HyperText Transfer Protocol (HTTP)

The **HyperText Transfer Protocol (HTTP)** is used to define how the client-server programs can be written to retrieve web pages from the Web.

An HTTP client sends a request; an HTTP server returns a response. The server uses the port number 80; the client uses a temporary port number.

HTTP uses the services of TCP is a connection-oriented and reliable protocol. This means that, before any transaction between the client and the server can take place, a connection needs to be established between them. After the transaction, the connection should be terminated.

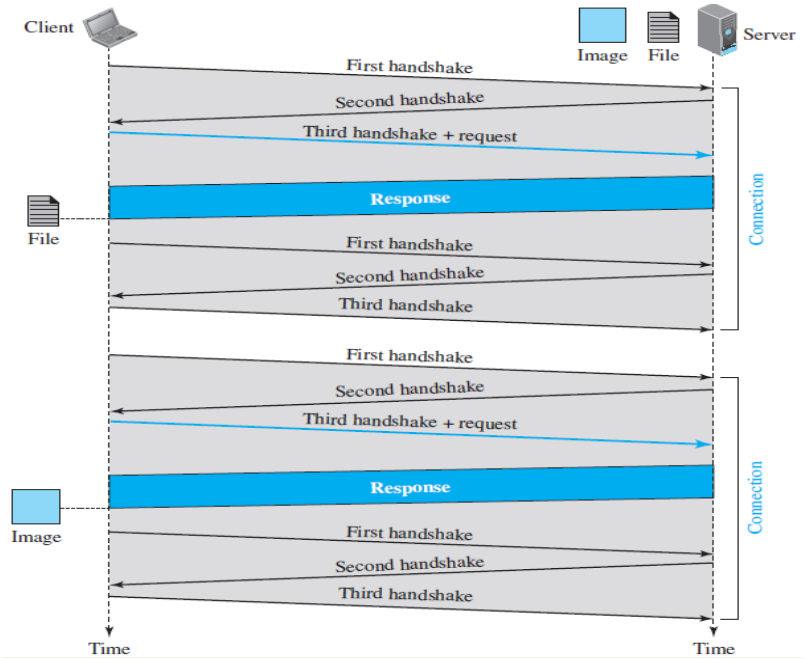
- ***Nonpersistent versus Persistent Connections:***

Nonpersistent Connections

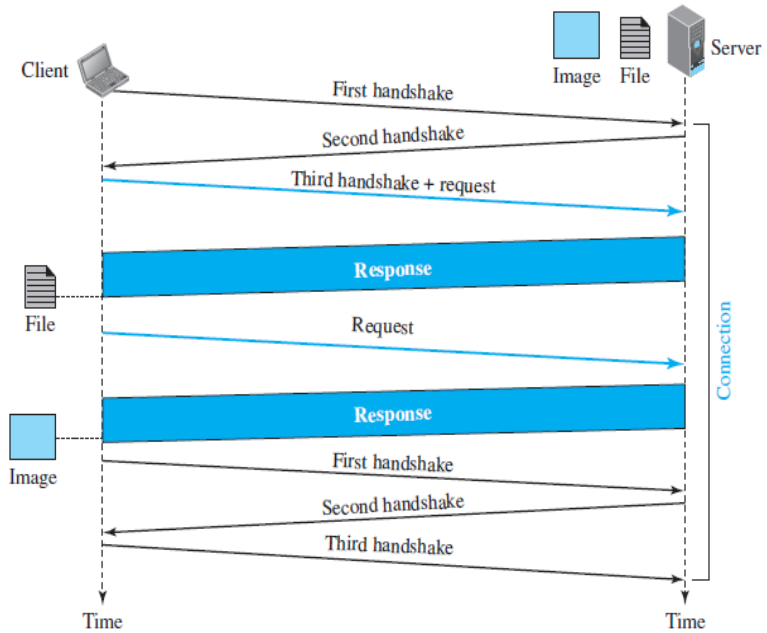
In a **nonpersistent connection**, one TCP connection is made for each request/response.

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 the connection.



Persistent Connections: In a persistent connection, the server leaves the connection open for more requests after sending a response. The server can close the connection at the request of a client or if a time-out has been reached. The sender usually sends the length of the data with each response.



 Message Formats:The HTTP protocol defines the format of the request and response messages. Each message is made of four sections. The first section in the request message is called the ***requestline;*** the first section in the response message is called the ***statusline***. The other three sections have the same names in the request and response messages.

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*.

- The ***methodfield*** defines the request types. In version 1.1 of HTTP, several methods are defined, as shown in Table. The client uses the GET method to send a request.
- The second field, ***URL***. It defines the address and name of the corresponding web page.
- The third field, ***version***, gives the version of the protocol; the most current version of HTTP is 1.1.

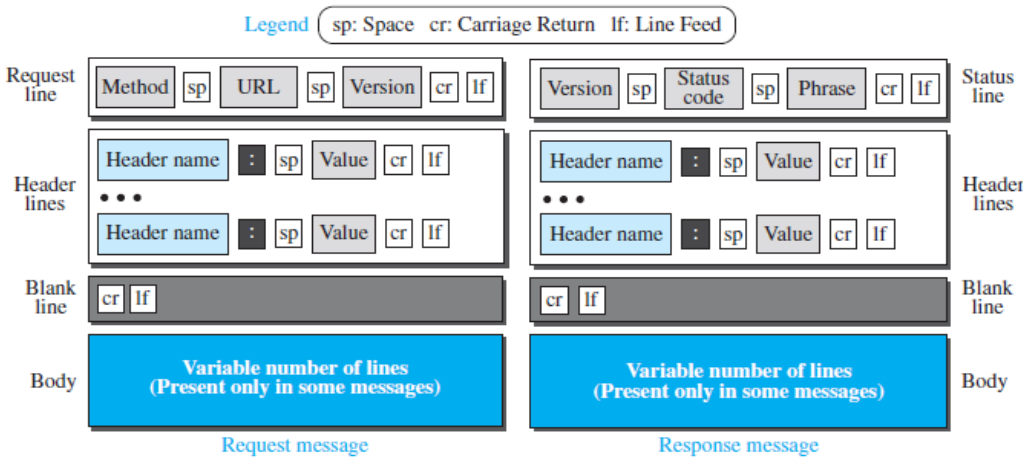


Fig: Formats of the request and response messages

Response Message

The format of the response message is also shown in Figure. The first line in a response message is called the ***statusline***. 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. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request.
- The status phrase explains the status code in text form.

Web Caching: Proxy Servers: 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.
- Note that the proxy server acts as both server and client. When it receives a request from a client for which it has a response, it acts as a server and sends the response to the client.
- The proxy servers are normally located at the client site.

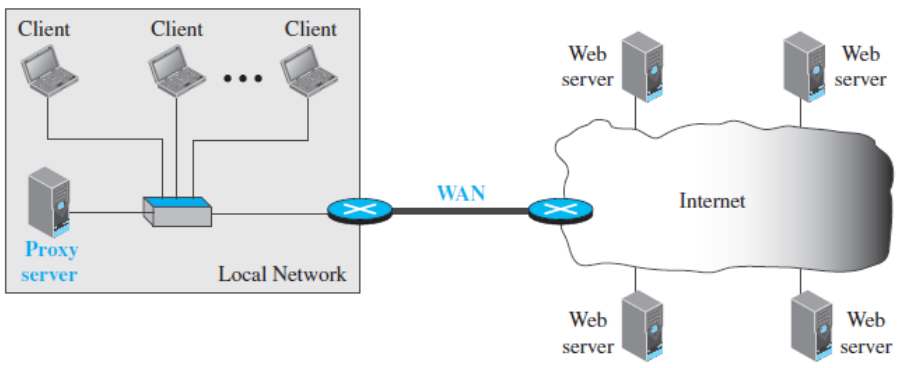


Fig: Example of a proxy server

3. Electronic Mail(e-mail) in the Internet

Electronic mail (or e-mail) allows users to exchange messages. E-mail is considered a one-way transaction. When Sender sends an email to Receiver, she may expect a response, but this is not a mandate. Receiver may or may not respond. If he does

respond, it is another one-way transaction.

 Architecture: To explain the architecture of e-mail, we give a common scenario, as shown in Figure.

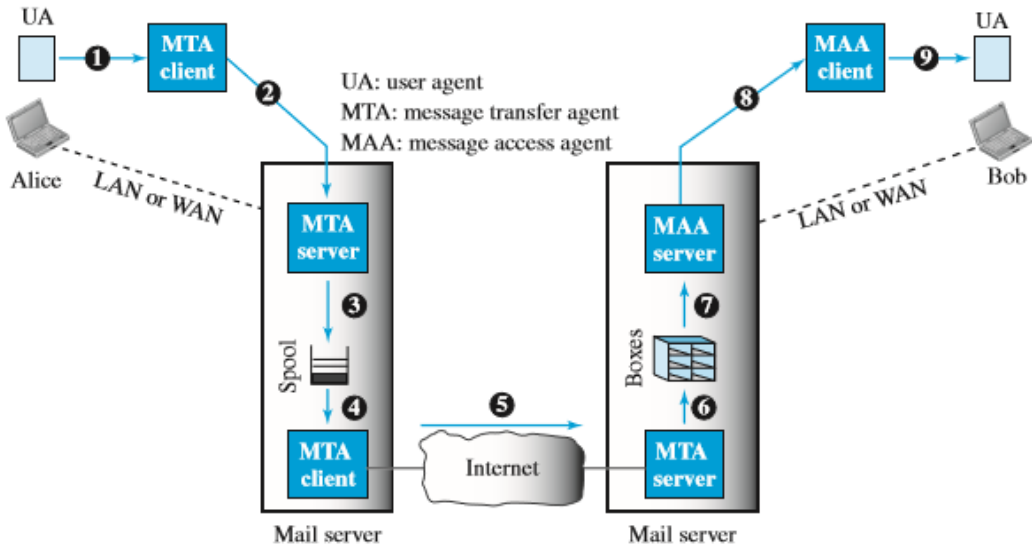


Fig: Common scenario

Sender and receiver use ***three different agents***: a **user agent (UA), a message transfer agent (MTA),** and **a message access agent (MAA).**

When Sender needs to send a message to Receiver, she runs a ***UA program*** to prepare the message and send it to her mail server. The mail server at her site uses a ***queue (spool)*** to store messages waiting to be sent. The message needs to be sent through the Internet from Sender's site to Receiver's site using an **MTA**. Here two message transfer agents are needed: one client and one server.

Client-server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection. The client can be triggered by the system when there is a message in the queue to be sent. The user agent at the Receiver site allows Receiver to read the received message. Receiver later uses an MAA client to retrieve the message from an MAA server running on the second server.

The electronic mail system needs two UAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server).

User Agent(UA):

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.

 Sending Mail: To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an envelope and a message.

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

 Receiving Mail: 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.

The summary usually includes the sender mail address, the subject, and the time the mail was sent or received.

 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 a ***@ sign***

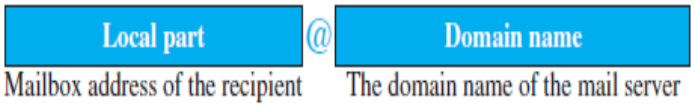


Fig: E-mail address

- The *local part* defines the name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the message access agent.

- The *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).

Message Transfer Agent: SMTP

The e-mail is one of those applications that need three uses of client-server paradigms to accomplish its task. Figure shows these three client-server applications. We refer to the first and the second as Message Transfer Agents (MTAs), the third as Message Access Agent (MAA).

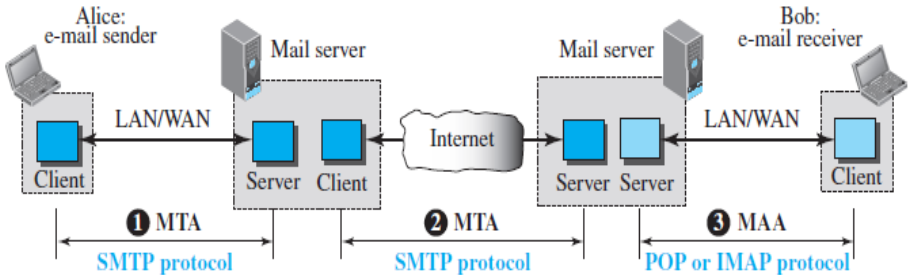


Fig: Protocols used in electronic mail

The formal protocol that defines the MTA client and server in the Internet is called ***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. The command is from an MTA client to an MTA server; the response is from an MTA server to the MTA client.

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

Keyword: argument(s)

Keyword	*Argument(s)*	*Description*
HELO	Sender's host name	Identifies itself
MAIL FROM	Sender of the message	Identifies the sender of the message
RCPT TO	Intended recipient	Identifies the recipient of the message
DATA	Body of the mail	Sends the actual message
QUIT		Terminates the message
RSET		Aborts the current mail transaction

Table: *SMTP commands*

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

Table: *Responses*

 Mail Transfer Phases: The process of transferring a mail message occurs in three phases: connection establishment, mail transfer, and connection termination.
Connection Establishment: After a client has made a TCP connection to the well-known port 25, the SMTP server starts the connection phase.
Message Transfer: After connection has been established between the SMTP client and server, a single message between a sender and one or more recipients can be exchanged.
Connection Termination: After the message is transferred successfully, the client terminates the connection.

 Message Access Agent (MAA): The third stage uses a message access agent. Currently two message access protocols are available: ***Post Office Protocol***, version 3 (POP3) and ***Internet Mail Access Protocol***, version 4 (IMAP4).
POP3

 Post Office Protocol, version 3 (POP3):is simple but limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.

- Mail access starts with the client when the user needs to download its e-mail from the mailbox on the mail server.
- The client opens a connection to the server on TCP port 110. It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one.
- POP3 has ***two modes***: the *delete mode* and the *keep mode*. In the delete mode, the mail is deleted from the mailbox after each retrieval. In the keep mode, the mail remains in the mailbox after retrieval.

 IMAP4: Internet Mail Access Protocol, version 4 (IMAP4) is a mail access protocol. IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.
IMAP4 provides the following extra functions:

- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail.
- A user can create, delete, or rename mailboxes on the mail server.
- A user can create a hierarchy of mailboxes in a folder for e-mail storage.

4. Domain Name System (DNS)or DNS—The Internet's Directory Service

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. People prefer to use names instead of numeric addresses. Therefore, the Internet needs to have a directory system that can map a name to address.

Domain Name System directory system in the Internet can map names to IP addresses.***Figure*** 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.*

However, 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 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.

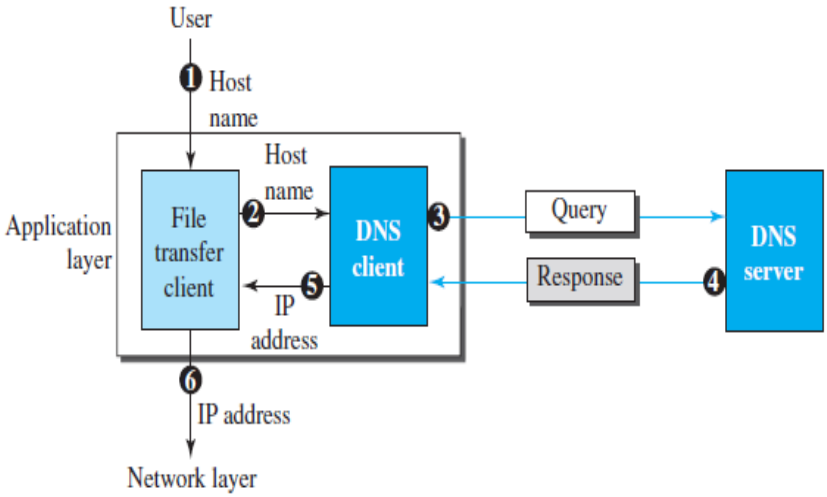


Fig: Purpose of DNS

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

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

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 can have only 128 levels: level 0 (root) to level 127

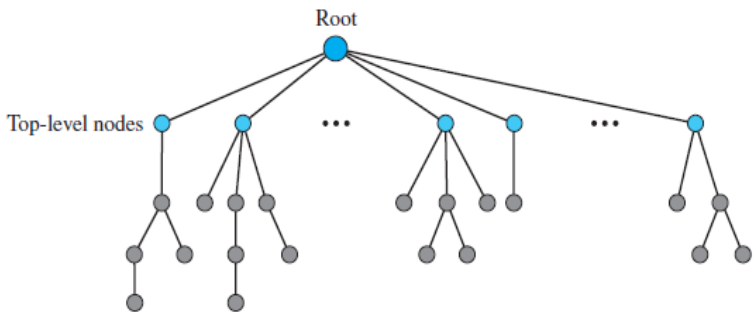


Fig: *Domain name space*

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 from the same node) have different labels

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).

If a label is terminated by a null string, it is called a fully qualified domain name (FQDN). 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.

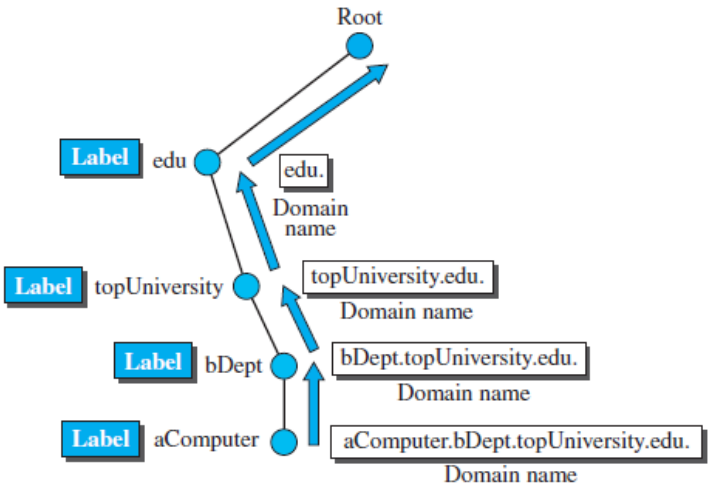


Fig: Domain names and labels

- **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 shows some domains. Note that a domain may itself be divided into domains.

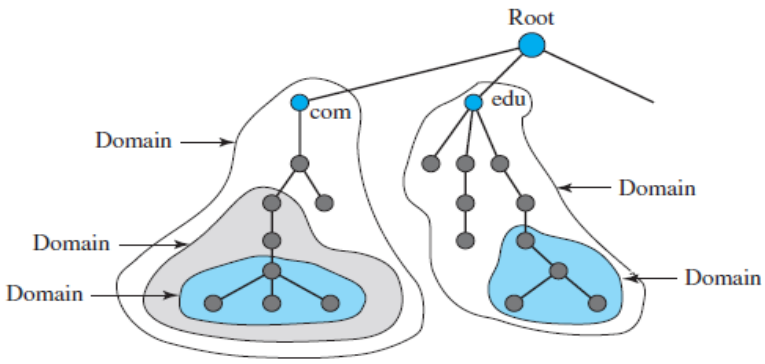


Fig: *Domains*

- **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.

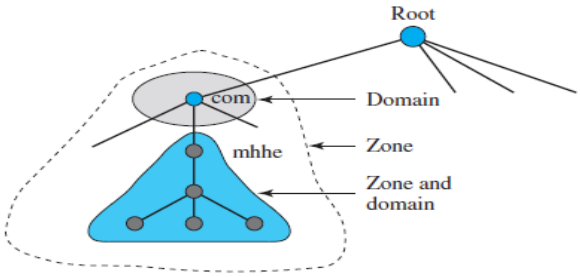


Fig: Zone

- ***Root Server:*** A root server is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers. The root 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.

- **DNS in the Internet:** DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) was originally divided into three different 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.

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

Table: *Generic domain labels*

Country Domains: The country domains section uses two-character country abbreviations (e.g., us for United States).

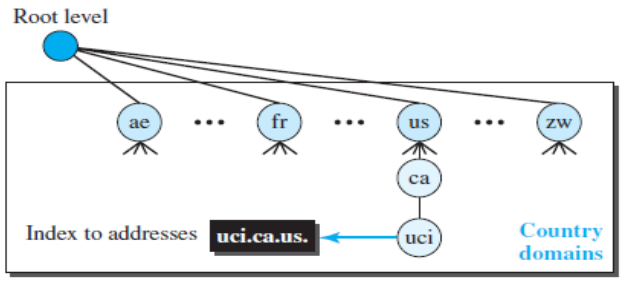


Fig: Country domains

Inverse Domain: The inverse domain is used for mapping an address to a name. When the server has received a request from the client, and the server contains the files of only authorized clients. To determine whether the client is on the authorized list or not, it sends a query to the DNS server and ask for mapping an address to the name.

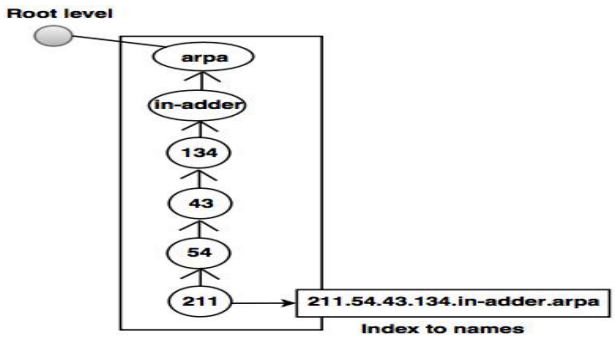


Fig. Inverse domain

5. FTP (File Transfer Protocol)

File Transfer Protocol (FTP) is the standard protocol provided by TCP/IP for copying a file from one host to another.

- **For example,** two systems may use different file name conventions. Two systems may have different ways to represent data. Two systems may have different directory structures. All of these problems have been solved by FTP in a very simple and elegant approach.
- Although we can transfer files using HTTP, FTP is a better choice to transfer large files or to transfer files using different formats.

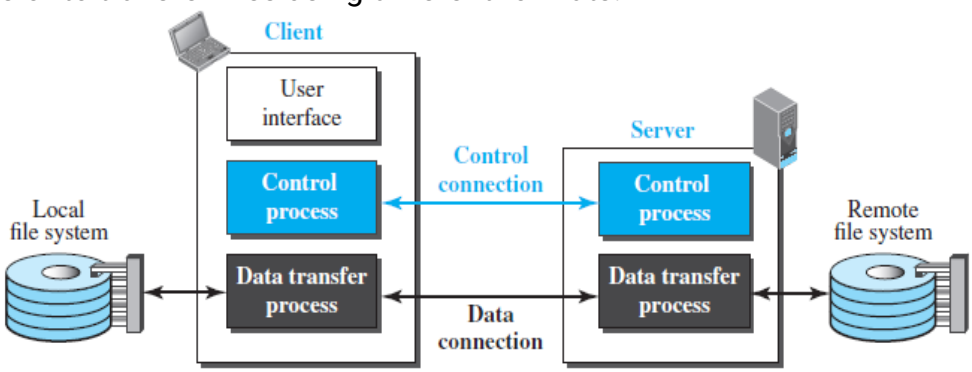


Fig: FTP

- The **client** has three components: the user interface, the client control process, and the client data transfer process.
- The **server** has two components: the server control process and the server data transfer process.
- The **controlconnection** is made between the control processes. The **dataconnection** is made between the data transfer processes.

Two Connections

- The two connections in FTP have different lifetimes. The *control connection* remains connected during the entire interactive FTP session.
- The *data connection* is opened and then closed for each file transfer activity. It opens each time commands that involve transferring files are used, and it closes when the file is transferred.
- FTP uses two well-known TCP ports: port 21 is used for the control connection, and port 20 is used for the data connection.

*Control Connection:*During this control connection, commands are sent from the

client to the server and responses are sent from the server to the client.
Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument. Some of the most common commands are shown in Table.

Command	*Argument(s)*	*Description*
ABOR		Abort the previous command
CDUP		Change to parent directory
CWD	Directory name	Change to another directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
MKD	Directory name	Create a new directory
PASS	User password	Password
PASV		Server chooses a port
PORT	Port identifier	Client chooses a port
PWD		Display name of current directory
QUIT		Log out of the system

Table: Some FTP commands

Every FTP command generates at least one response. A response has two parts: a three-digit number followed by text.

Code	*Description*	*Code*	*Description*
125	Data connection open	**250**	Request file action OK
150	File status OK	**331**	User name OK; password is needed
200	Command OK	**425**	Cannot open data connection
220	Service ready	**450**	File action not taken; file not available
221	Service closing	**452**	Action aborted; insufficient storage
225	Data connection open	**500**	Syntax error; unrecognized command
226	Closing data connection	**501**	Syntax error in parameters or arguments
230	User login OK	**530**	User not logged in

Table: Some responses in FTP

 Data Connection:The data connection uses the well-known port 20 at the server site. The purpose and implementation of the data connection are different from those of the control connection. We want to transfer files through the data connection. The client must define the ***type of file*** to be transferred, the ***structure of the data***, and the ***transmission mode***.

Three attributes of communication: file type, data structure, and transmission mode.
- ***File Type:*** FTP can transfer one of the following file types across the data connection: ASCII file, EBCDIC file, or image file.

- ***Data Structure:*** FTP can transfer a file across the data connection using one of the following interpretations of the structure of the data: *file structure, record structure,* or *page structure*. The ***file structure*** 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.
- ***Transmission Mode:*** FTP can transfer a file across the data connection using one of the following three transmission modes: *stream mode, block mode,* or *compressed mode*.

File Transfer: File transfer occurs over the data connection under the control of the commands sentover the control connection. However, we should remember that file transfer in FTPmeans one of ***three things***: *retrieving a file* (server to client)*, storing a file* (client toserver), and *directory listing* (server to client).

Security for FTP

The FTP protocol was designed when security was not a big issue. Although FTP

requires a password, the password is sent in plaintext (unencrypted), which means it can be intercepted and used by an attacker.
The data transfer connection also transfers data in plaintext, which is insecure. To be secure, one can add a ***Secure Socket Layer*** between the FTP application layer and the TCP layer. In this case FTP is called ***SSL-FTP***.

6. Peer-to-Peer Applications: Video Streaming and Content Distribution Networks

Peer-to-Peer Applications: Video Streaming

Peer-to-Peer (P2P) technology has recently become a promising approach to provide live streaming services or Video-on-Demand (VoD) services to a huge number of the concurrent users over a global area. P2P streaming systems can be classified into P2P live streaming systems and P2P VoD systems.
Peer-to-peer (P2P) is open-source and requires a direct connection between two or more devices to create a real-time audio and video data stream without any cloud or outside servers. The device of the meeting host acts as a server and can initiate a call by sending its IP address directly to the attendee's computer.
There are several ways audio and video calls can form a stable connection, one being peer-to-peer (P2P). This real-time communication solution is a popular choice among users who prioritize security, bandwidth, and a lightweight architecture but is less so among those who wish to scale their number of meeting participants.

How Peer-to-Peer (P2P) Conferencing Works

P2P relies upon a simple architecture that is easy for developers to create and the end-user to enjoy. Instead of broadcasting your live stream to a server that must transmit the data to the other meeting participants, peer-to-peer communication bypasses the need for an authentication server. It streams audio and video from one computer directly to another.

Video Streaming Technology

Video streaming technology is one way to deliver video over the Internet. Using streaming technologies, the delivery of audio and video over the Internet can reach many millions of customers using their personal computers, PDAs, mobile smartphones or other streaming devices.

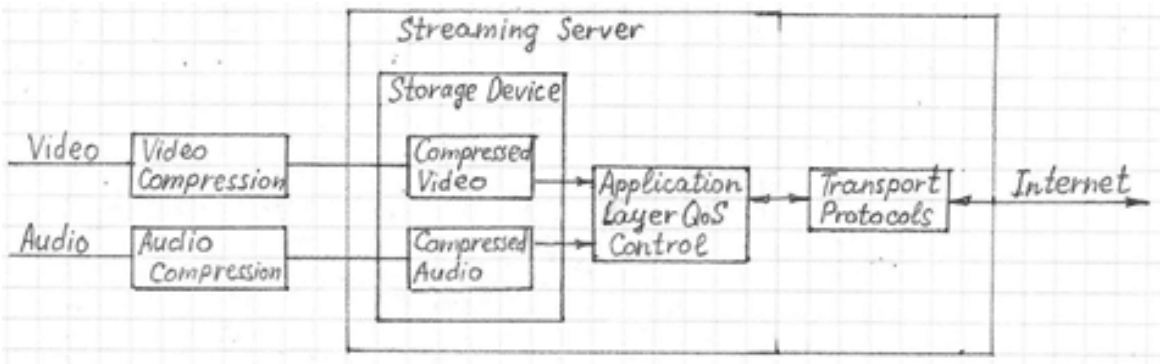


Fig: Architecture of streaming video on the transmitter side

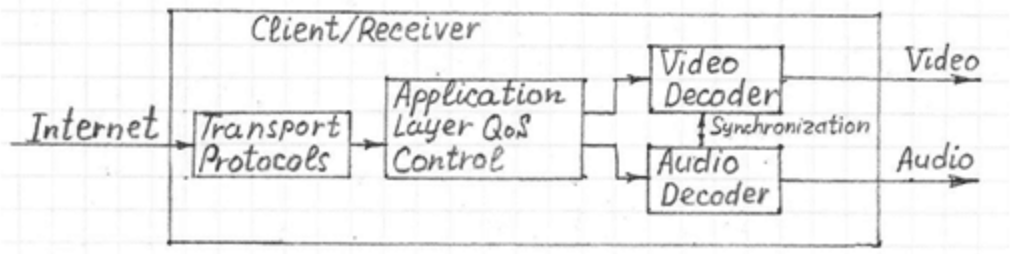


Fig: Architecture of streaming video on the receiver side

Peer-to-Peer Applications: Content Distribution Networks

Peer-to-peer (P2P) content distribution is a model that allows the distribution of files, videos, software or other applications

The Peer-to-Peer method entails the distribution of files and information directly between users (terminals) without going through a server.

- A **CDN** (**Content Distribution Network**). In it, a provider sets up a distributed collection of machines at locations inside the Internet and uses them to serve content to clients. This is the choice of the big players. An alternative architecture is a **P2P** (**Peer-to-Peer**) network
- P2P CDNs work by distributing content delivery across a network of peer nodes rather than relying on a single server or origin point.
- This approach can improve performance and reliability by reducing the dependence on any one node in the network.
- Additionally, P2P CDNs often allow for more efficient bandwidth and resources, as content is shared among the nodes rather than being sent to each node separately.

A Content Delivery Network, or a CDN as it is commonly called, is an essential part of any modern website and application. The content that you view on your phones today, on any website or app, videos or images, or any other kind of content, is very likely to be delivered using a content delivery network.

- **CDNs** (**Content Delivery Networks**) turn the idea of traditional Web caching on its head. Instead, of having clients look for a copy of the requested page in a nearby cache, it is the provider who places a copy of the page in a set of nodes at different locations and directs the client to use a nearby node as the server.
- An example of the path that data follows when it is distributed by a CDN is shown in Fig. It is a tree. The origin server in the CDN distributes a copy of the content to other nodes in the CDN, in Sydney, Boston, and Amsterdam, in this example. This is shown with dashed lines. Clients then fetch pages from the nearest node in the CDN. This is shown with solid lines. In this way, the clients in Sydney both fetch the page copy that is stored in Sydney; they do not both fetch the page from the origin server, which may be in Europe.

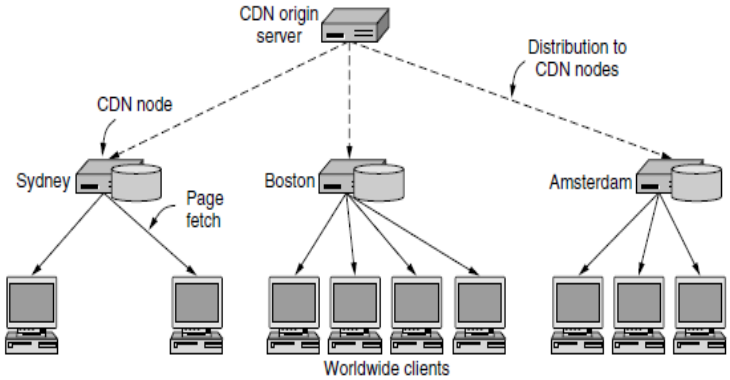


Fig: Content Distribution tree

Some examples of a file-sharing peer-to-peer network include **BitTorrent, uTorrent, Ares Galaxy, FrostWire, and BitComet**.