**COMPUTER NETWORKS**

**UNIT - 5:** **Principles of Network Applications**

**Principles of Network Applications Principles of Network Applications, The Web and HTTP, Electronic Mail in the Internet, DNS—The Internet’s Directory Service, Peer-to-Peer Applications Video Streaming and Content Distribution Networks (Textbook 2)**

**Network Applications:**

A Network application is an application running on one host and provides a communication to another application running on a different host.A network application development is writing programs that run on different end systems and communicate with each other over the network. In the Web application there are two different programs that communicate with each other:

Browser program running in the user's host.

Web server program running in the Web server host.

**Network Applications – Examples**

**Email**

**Web**

**Remote Login**

**P2P File Sharing**

**Multi-user Network Games**

**Streaming Stored Video (YouTube)**

**Voice Over IP (Skype)**

**Real-time Video Conference**

**Social Networking**

**Network Application Architecture**

**Client-Server architecture**

**P2P (Peer to Peer) architecture**

**Client-Server Architecture**

**Server:**

Its always-on host.

It has a fixed IP address.

Large cluster of host – Data Centers.

E.g. Web Server

**Client:**

It communicate with server.

Its not like continuously connected.

May have dynamic IP addresses.

Do not communicate directly with each other.

E.g. PCs, Mobiles

Client

Server

**P2P Architecture**

Peers (end systems) directly communicate.

Get peers request service from other peers, provide service to other peers.Self Scalability – New peers bring new service capacity, as well as new service demands.

Peers are alternatingly connected and change IP addresses.

Complex management

Peer

Peer

Peer

**Process Communication**

What is Process?

A process is an instance of a program running in a computer.

We can say that process is program under execution.

Within same host, two processes communicate using inter-process communication (IPC).

Process in different hosts communicate by exchanging messages.

Client process: A process that initiates communication.

Server process: A process that waits to be contacted.

**Process P1**

**Process P1**

**Socket**

A process sends messages into and receives messages from; the network through a software interface called a socket.

A process is like a house and its socket is like its door.

Sending process passes message outdoor.

Sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process.

**Internet Applications**

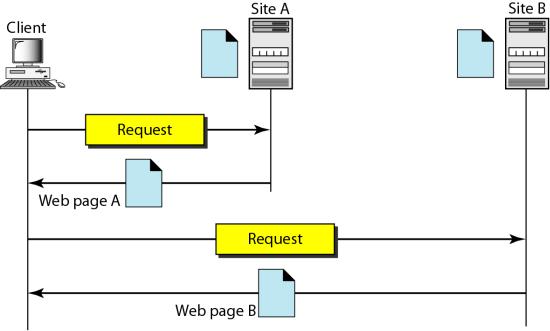
|  |  |  |
| --- | --- | --- |
| **Applications** | **Application-Layer**  **Protocol** | **Underlying Transport**  **Protocol (Service)** |
| Email | SMTP | TCP |
| Remote Terminal Access | Telnet | TCP |
| Web | HTTP | TCP |
| File Transfer | FTP | TCP |
| Streaming Media | HTTP(YouTube), RTP | TCP or UDP |
| Internet Telephony | SIP, RTP(Skype) | Typically UDP |

**5.4 World Wide Web (WWW)**

The **World Wide Web** (WWW) is a repository of information linked together from points all over the world. TheWWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research.

**5.5.1 ARCHITECTURE**

The WWW today is a distributed clientJserver 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,* as shown in Figure.

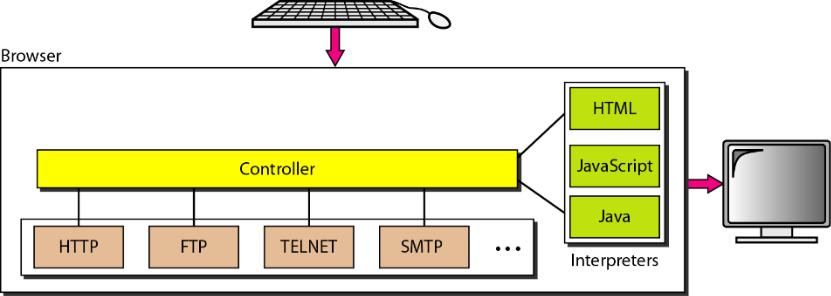


Each site holds one or more documents, referred to as *Web pages.* Each Web page can contain a link to other pages in the same site or at other sites. The pages can be retrieved and viewed by using browsers.

The client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch Web documents. The request, among other information, includes the address of the site and the Web page, called the URL. The server at site A finds the document and sends it to the client. When the user views the document, she finds some references to other documents, including a Web page at site B. The reference has the URL for the new site. The user is also interested in seeing this document. The client sends another request to the new site, and the new page is retrieved.

**5.5.2 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 HTIP. The interpreter can be HTML, Java, or JavaScript, depending on the type of document.



**5.5.3 Server**

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

**5.5.4 Uniform Resource Locator**

A client that wants to access a Web page needs the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. 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, and path



The *protocol* is the client/server program used to retrieve the document. Many different protocols can retrieve a document; among them are FTP or HTTP. The most common today is HTTP.

The host is the computer on which the information is located, although the name of the computer can be an alias. Web pages are usually stored in computers, and computers are given alias names that usually begin with the characters "www". This is not mandatory, however, as the host can be any name given to the computer that hosts the Web page.

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.

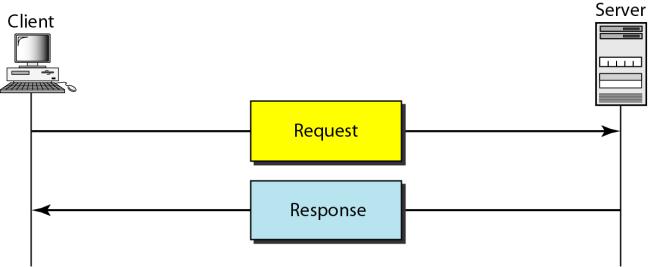
Path is the pathname of the file where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system, separate the directories from the subdirectories and files.

**5.5 HYPERTEXT TRANSFER PROTOCOL (HTTP)**

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. HTTP uses the services of TCP on well-known port 80.

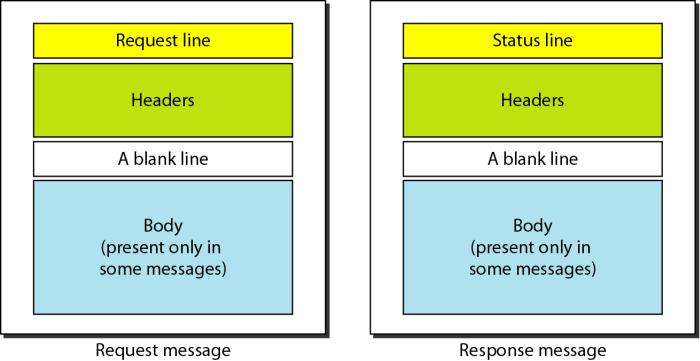
**5.5.1 HTTP Transaction**

Below figure illustrates the HTTP transaction between the client and server. Although HTTP uses the services of TCP, HTTP itself is a stateless protocol. The client initializes the transaction by sending a request message. The server replies by sending a response.

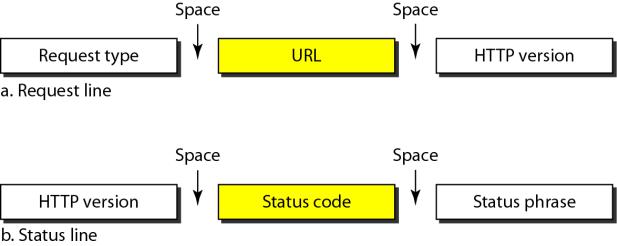


**5.5.2 HTTP Messages**

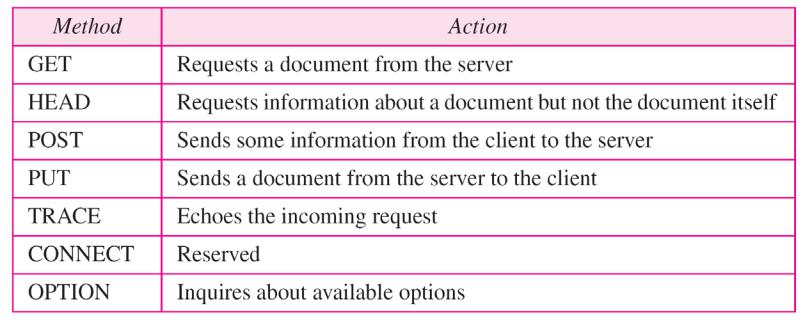
The formats of the request and response messages are similar; both are shown in below figure. A request message consists of a request line, a header, and sometimes a body. A response message consists of a status line, a header, and sometimes a body.



**Request and Status Lines.** The first line in a request message is called a request line; the first line in the response message is called the status line. There is one common field, as shown in below figure.



**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* as defined in below table.

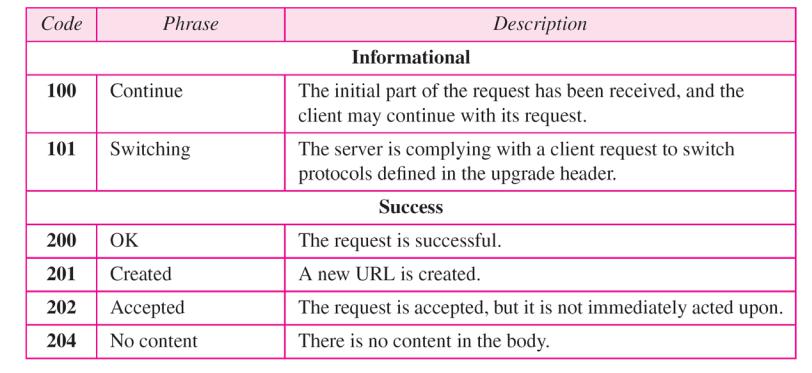


**URL**. The uniform resource locator (URL) is a standard for specifying any kind of information on the Internet..

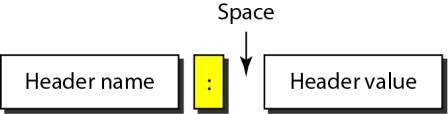
**Version**. The most current version of HTTP is 1.1.

**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. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request.

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

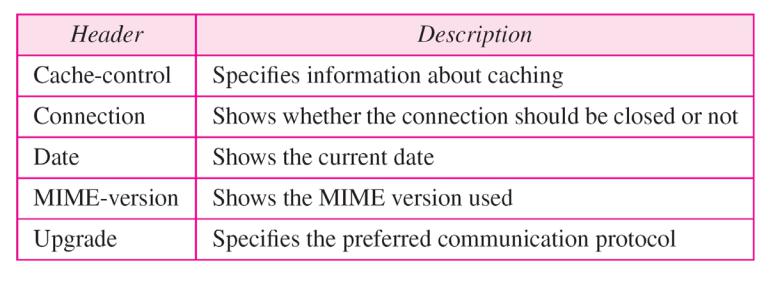


**Header** The header exchanges additional information between the client and the server. For example, the client can request that the document be sent in a special format, or the server can send extra information about the document. The header can consist of one or more header lines. Each header line has a header name, a colon, a space, and a header value.

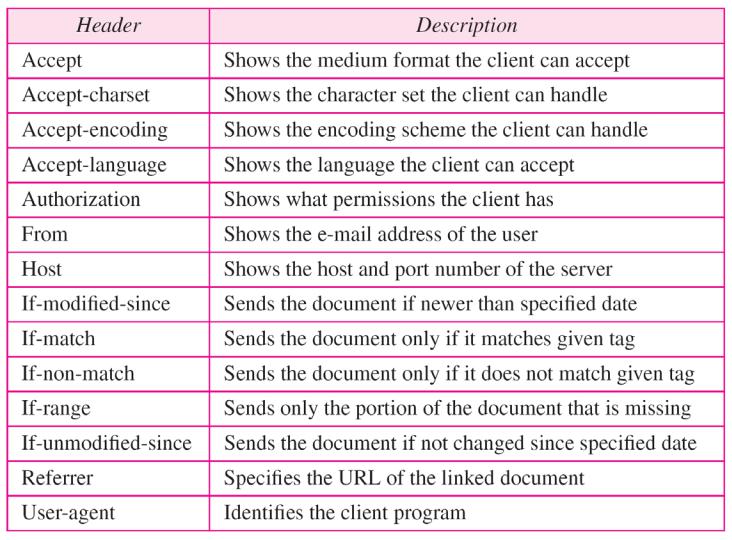


A header line belongs to one of four categories: **general header, request header, response header, and entity header.**

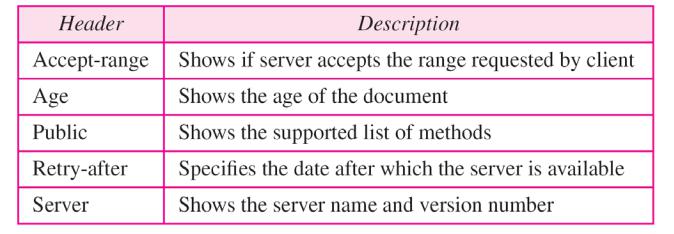
**General header** The general header gives general information about the message and can be present in both a request and a response. Below table lists some general headers with their descriptions.



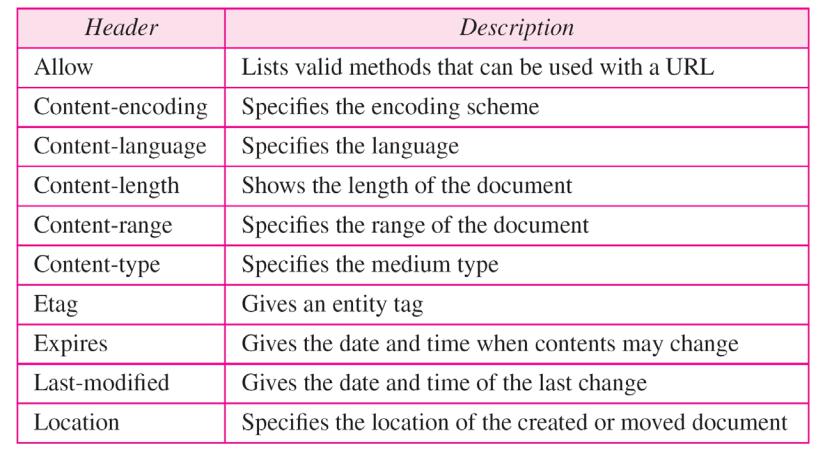
**Request header** The request header can be present only in a request message. It specifies the client's configuration and the client's preferred document format. See below Table for a list of some request headers and their descriptions.



**Response header** The response header can be present only in a response message. It specifies the server's configuration and special information about the request. See below Table for a list of some response headers with their descriptions.



**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, that contain a body also use this type of header. See below Table for a list of some entity headers and their descriptions.



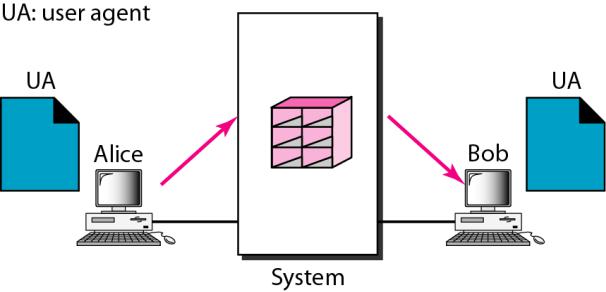
**5.6 ELECTRONIC MAIL**

One of the most popular Internet services is electronic mail (e-mail). At the beginning of the Internet era, the messages sent by electronic mail were short and consisted of text only; they let people exchange quick memos. Today, electronic mail is much more complex. It allows a message to include text, audio, and video. It also allows one message to be sent to one or more recipients.

**5.6.1 Architecture**

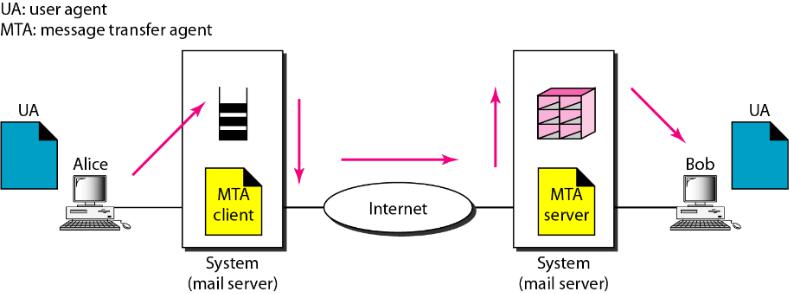
1. *First Scenario*

In the first scenario, the sender and the receiver of the e-mail are users (or application programs) on the same system; they are directly connected to a shared system. The administrator has created one mailbox for each user where the received messages are stored. A *mailbox* is part of a local hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. When Alice, a user, needs to send a message to Bob, another user, Alice runs a *user agent (UA)* program to prepare the message and store it in Bob's mailbox. The message has the sender and recipient mailbox addresses (names of files). Bob can retrieve and read the contents of his mailbox at his convenience, using a user agent.



1. *Second Scenario*

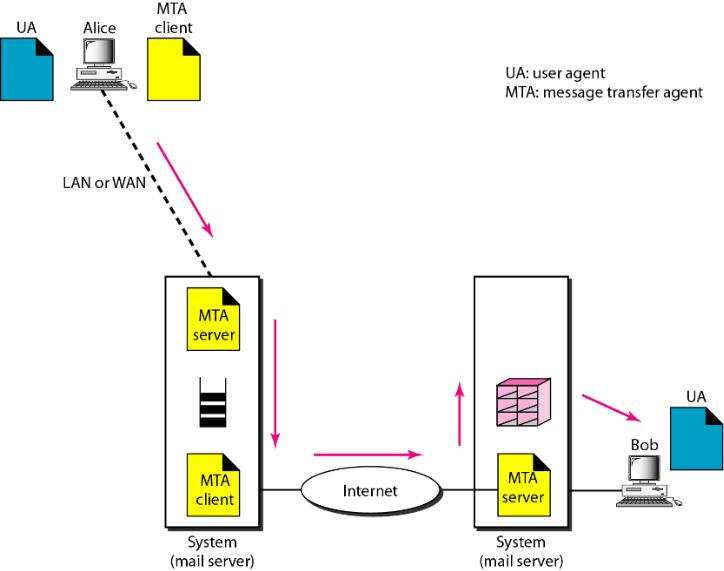
In the second scenario, the sender and the receiver of the e-mail are users (or application programs) on two different systems. The message needs to be sent over the Internet. Here we need user agents (UAs) and message transfer agents (MTAs).



Alice needs to use a user agent program to send her message to the system at her own site. The system (sometimes called the mail server) at her site uses a queue to store messages waiting to be sent. Bob also needs a user agent program to retrieve messages stored in the mailbox of the system at his site. The message, however, needs to be sent through the Internet from Alice's site to Bob's site. Here two message transfer agents are needed: one 'client and one server. Like most client/server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection. The client, on the other hand, can be alerted by the system when there is a message in the queue to be sent.

1. *Third Scenario*

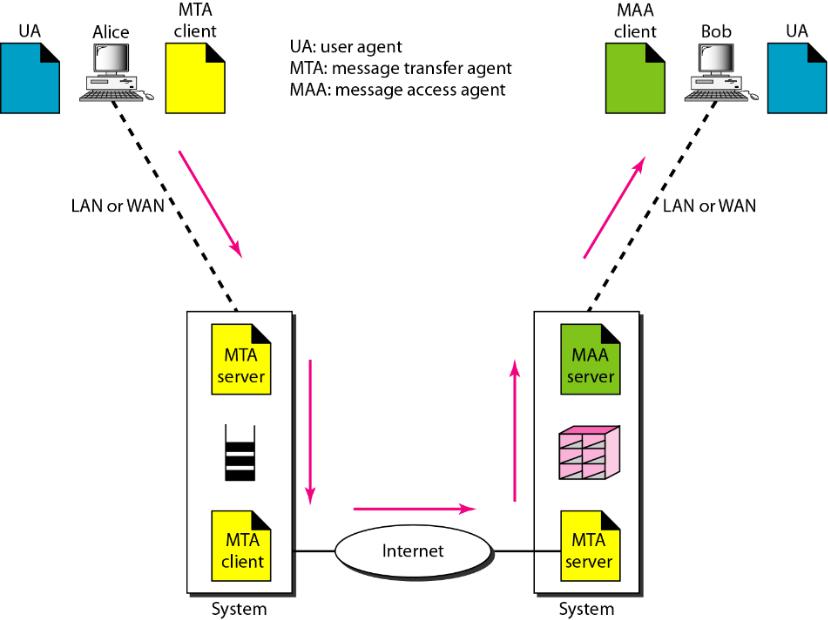
In the third scenario, Bob, as in the second scenario, is directly connected to his system. Alice, however, is separated from her system. Either Alice is connected to the system via a point-to-point WAN, such as a dial-up modem, a DSL, or a cable modem; or she is connected to a LAN in an organization that uses one mail server for handling e-mails-all users need to send their messages to this mail server.



Alice still needs a user agent to prepare her message. She then needs to send the message through the LAN or WAN. This can be done through a pair of message transfer agents (client and server). Whenever Alice has a message to send, she calls the user agent which, in tum, calls the MTA client. The MTA client establishes a connection with the MTA server on the system, which is running all the time. The system at Alice's site queues all messages received. It then uses an MTA client to send the messages to the system at Bob's site; the system receives the message and stores it in Bob's mailbox. At his convenience, Bob uses his user agent to retrieve the message and reads it. Note that we need two pairs of MTA client/server programs.

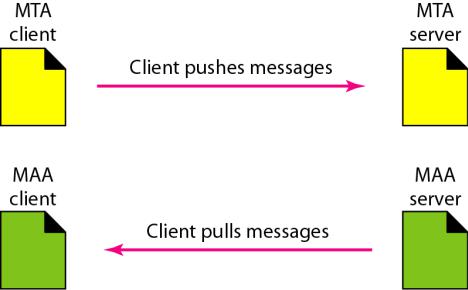
1. *Fourth Scenario*

In the fourth and most common scenario, Bob is also connected to his mail server by a WAN or a LAN. After the message has arrived at Bob's mail server, Bob needs to retrieve it. Here, we need another set of client/server agents, which we call message access agents (MAAs). Bob uses an MAA client to retrieve his messages. The client sends a request to the MAA server, which is running all the time, and requests the transfer of the messages.



There are two important points here. First, Bob cannot bypass the mail server and use the MTA server directly. To use MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN. If he is connected through a-WAN, he must keep the connection up all the time. Neither of these situations is feasible today.

Second, note that Bob needs another pair of client/server programs: message access programs. This is so because an MTA client/server program is a *push* program: the client pushes the message to the server. Bob needs a *pull* program. The client needs to pull the message from the server.

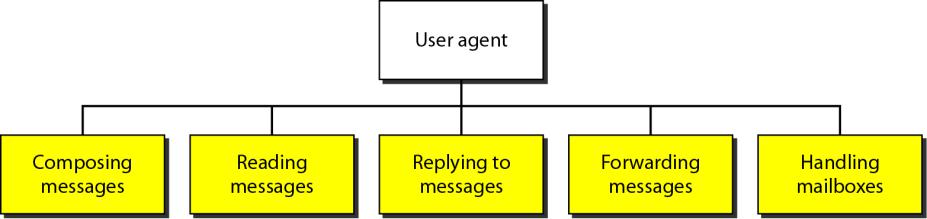


**5.6.2 User Agent**

The first component of an electronic mail system is the user agent *(UA).* It provides service to the user to make the process of sending and receiving a message easier.

*1.* *Services Provided by a 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.

*2.* *User Agent Types*

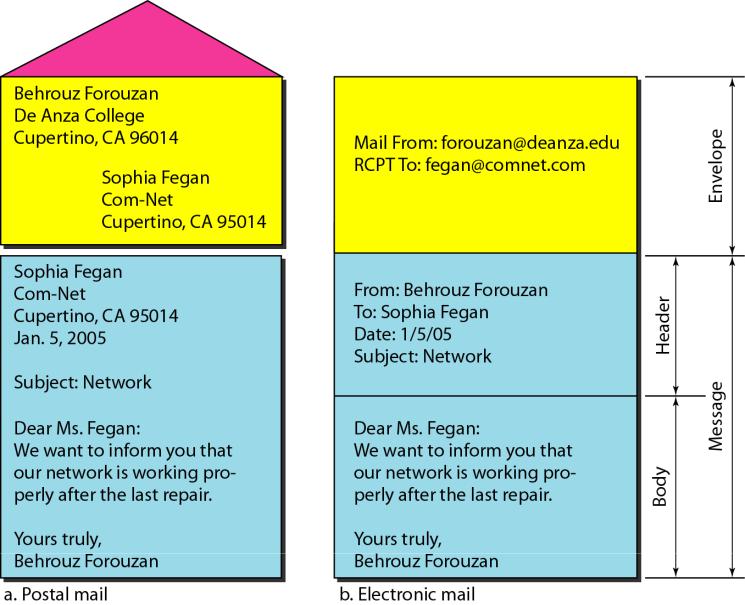
There are two types of user agents: command-driven and GUI-based.

**Command-Driven** user agents belong to the early days of electronic mail. They are still present as the underlying user agents in servers. 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.

**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.

*3.* *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*

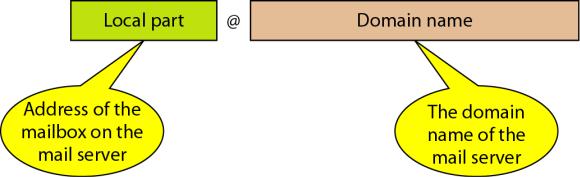
The envelope usually contains the sender and the receiver addresses. Message The message contains the header and the body. The header of the message defines the sender, the receiver, the subject of the message, and some other information (such as encoding type, as we see shortly). The body of the message contains the actual information to be read by the recipient.

*Receiving Mail*

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

*Addresses*

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



**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 theorganization).

*5.* *Mailing List*

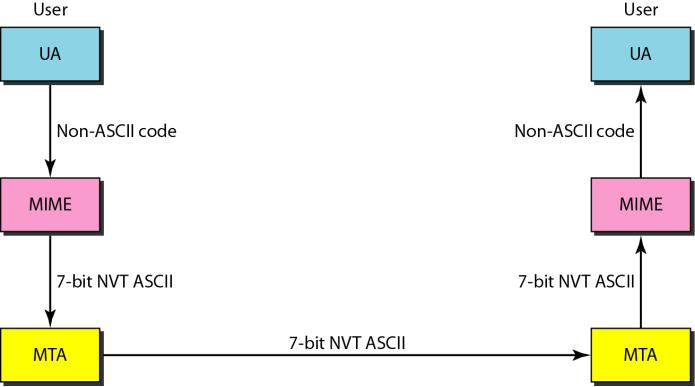
Electronic mail allows one name, an alias, to represent several different e-mail addresses; this is called a mailing list. Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for 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.

**5.6.3 MIME**

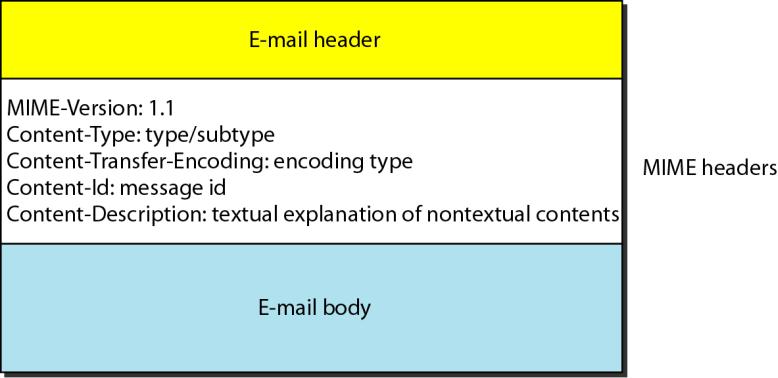
Electronic mail has a simple structure. Its simplicity, however, comes at a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. For example, it cannot be used for languages that are not supported by 7-bit ASCII characters (such as French, German, Hebrew, 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. We can think of MIME as 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

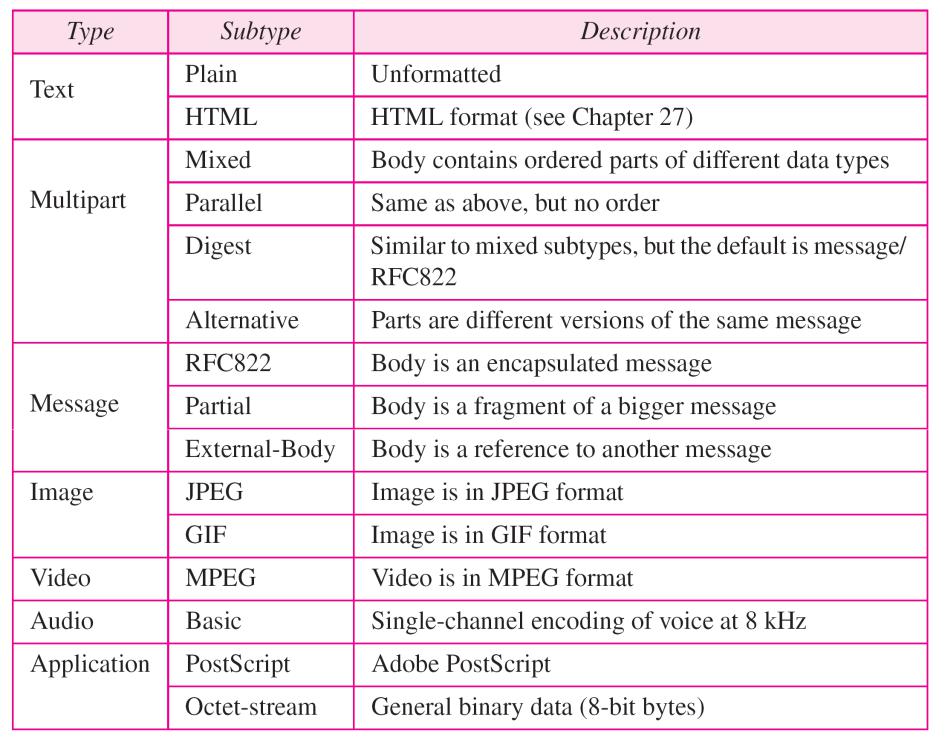


**MIME-Version** this header defines the version of MIME used. The current version is 1.1.

**MIME-Version: 1.1**

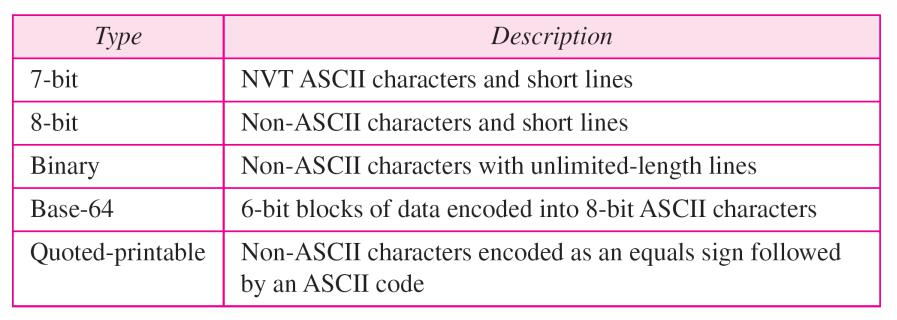
**Content-Type** This header defines the type of data used in the body of the message. The content type and the content subtype are separated by a slash. Depending on the subtype, the header may contain other parameters.

**Content-Type: <type/ subtype, parameter>** MIME allows seven different types of data.



**Content-Transfer-Encoding** This header defines the method used to encode the messages into 0s and 1s for transport:

**Content-Transfer-Encoding: <type>**



**Content-Id** This header uniquely identifies the whole message in a multiple-message environment.

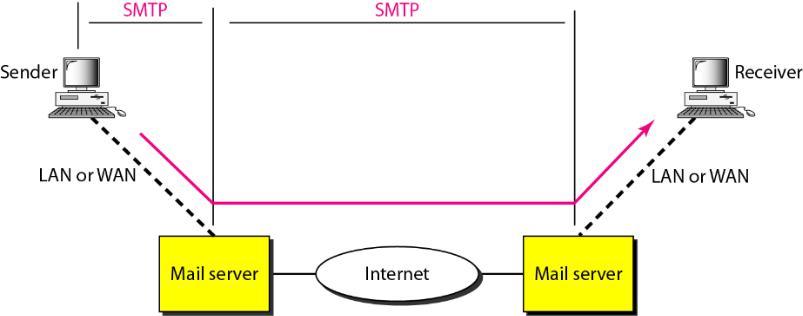
**Content-Id: id =<content-id>**

**Content-Description** This header defines whether the body is image, audio, or video.

**Content-Description: <description>**

**5.6.4 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). As we said before, two pairs of MTA client/server programs are used in the most common situation (fourth scenario).

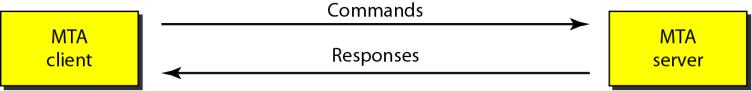


SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. As we will see shortly, another protocol is needed between the mail server and the receiver.

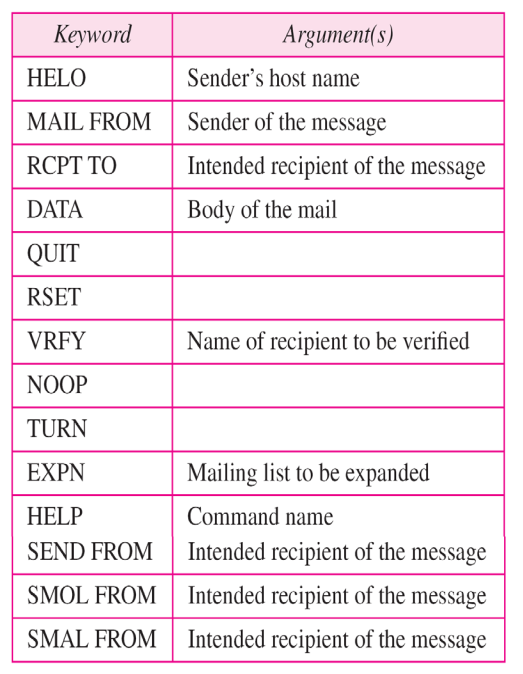
SMTP simply defines how commands and responses must be sent back and forth. Each network is free to choose a software package for implementation. We discuss the mechanism of mail transfer by SMTP in the remainder of the section.

*Commands and Responses*

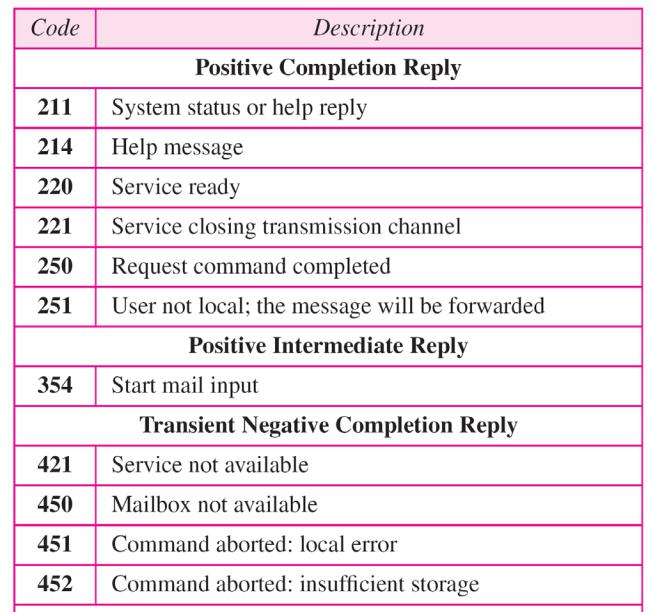
SMTP uses commands and responses to transfer messages between an MTA client and an MTA server.



**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.



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

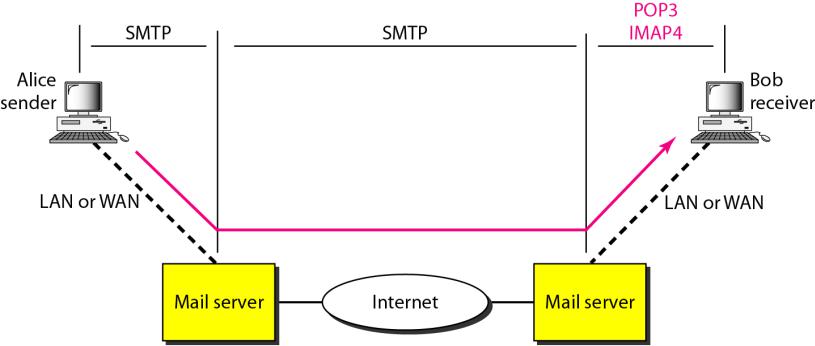


**5.6.5 Message Access Agent: POP and IMAP**

The first and the second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP is a *push* protocol; it pushes the message from the client to the server. In

other words, the direction of the bulk: data (messages) is from the client to the server. On the other hand, the third stage needs a *pull* protocol; the client must pull messages from the server. The direction of the bulk data is from the server to the client. 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). Below figure shows the position of these two protocols in the most common situation (fourth scenario).



*POP3*

**Post Office Protocol, version 3 (POP3)** is simple and 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 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. The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying. The keep mode is normally used when the user accesses her mail away from her primary computer (e.g., a laptop). The mail is read but kept in the system for later retrieval and organizing.

*IMAP4*

Another mail access protocol is **Internet Mail Access Protocol, version 4 (IMAP4)**. IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.

POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. (Of course, the user can create folders on her own computer.) In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.

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. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
* 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.

**5.6.6 Web-Based Mail**

E-mail is such a common application that some websites today provide this service to anyone who accesses the site. Two common sites are Hotmail and Yahoo. The idea is very simple. Mail transfer from Alice's browser to her mail server is done through HTTP The transfer of the message from the sending mail server to the receiving mail server is still through SMTP. Finally, the message from the receiving server (the Web server) to Bob's browser is done through HTIP.

The last phase is very interesting. Instead of POP3 or IMAP4, HTTP is normally used. When Bob needs to retrieve his e-mails, he sends a message to the website (Hotmail, for example). The website sends a form to be filled in by Bob, which includes the log-in name and the password. If the log-in

name and password match, the e-mail is transferred from the Web server to Bob's browser in HTML format.

**5.7 TELENET**

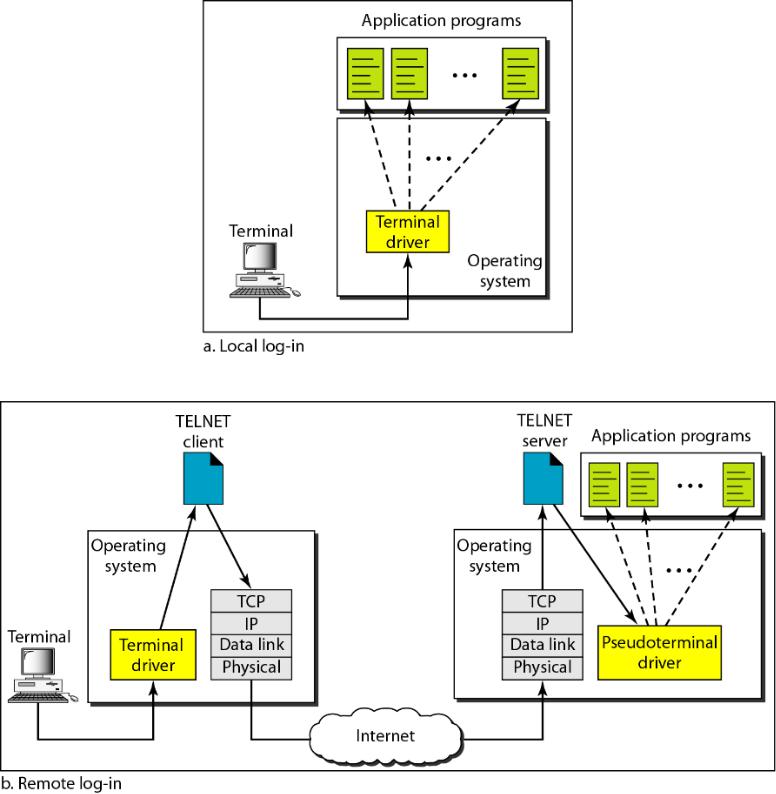
TELNET is an abbreviation for *TErminaL NETwork.* It is the standard TCP/IP protocol for virtual terminal service as proposed by the International Organization for Standards (ISO). TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.

*Timesharing Environment*

TELNET was designed at a time when most operating systems, such as UNIX, were operating in a timesharing environment. In such an environment, a large computer supports multiple users. The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor, and mouse. Even a microcomputer can simulate a terminal with a terminal emulator.

*Logging*

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



When a user logs into a local timesharing system, it is called local log-in. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

When a user wants to access an application program or utility located on a remote machine, she performs remote log-in. Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver, where the local operating system accepts the characters but does not interpret them. The characters are sent to the TELNET client, which transforms the characters to a universal character set called *network virtual terminal (NVT) characters* and delivers them to the local *TCP/IP* protocol stack.

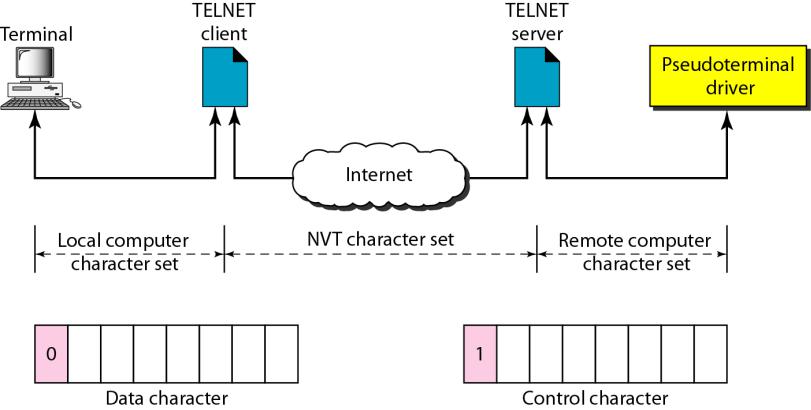
The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the

remote computer. However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server: It is designed to receive characters from a terminal driver.

*Network Virtual Terminal*

The mechanism to access a remote computer is complex. This is so because every computer and its operating system accept a special combination of characters as tokens. For example, the end-of-file token in a computer running the DOS operating system is Ctrl+z, while the UNIX operating system recognizes Ctrl+d.

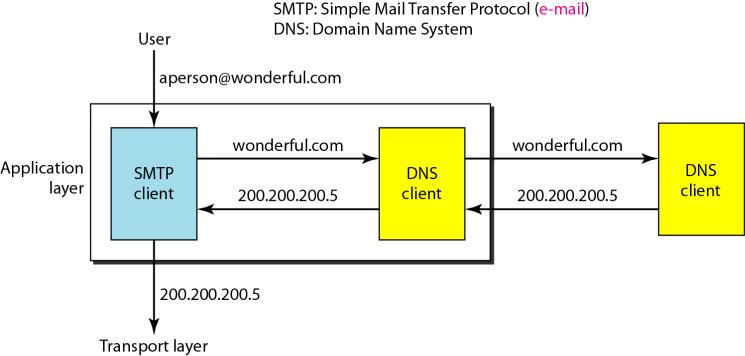
We are dealing with heterogeneous systems. If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer. TELNET solves this problem by defining a universal interface called the network virtual terminal (NVT) character set. Via this interface, the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network. The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer.



**5.8 Domain Name System**

DNS stands for Domain Name System. DNS is a directory service that provides a mapping between the name of a host on the network and its numerical address.

Below figure shows an example of how a DNS client/server program can support an e-mail program to find the IP address of an e-mail recipient. A user of an e-mail program may know the e-mail address of the recipient; however, the IP protocol needs the IP address. The DNS client program sends a request to a DNS server to map the e-mail address to the corresponding IP address.



**5.8.1 NAME SPACE**

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

1. **Flat Name Space**

In a flat name space, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a fiat 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.

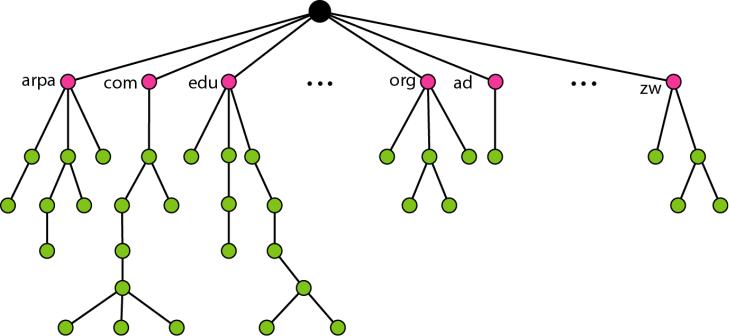
1. **Hierarchical Name Space**

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, and 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 can add suffixes (or prefixes) to the name to define its host or resources.

**5.8.2 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.

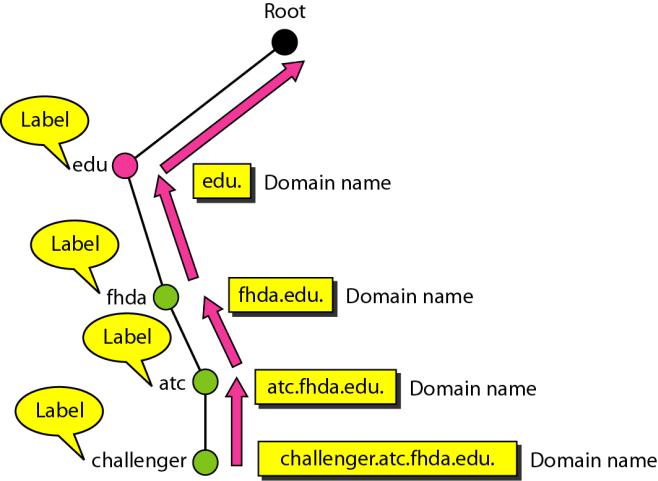


**Label**

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, 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.



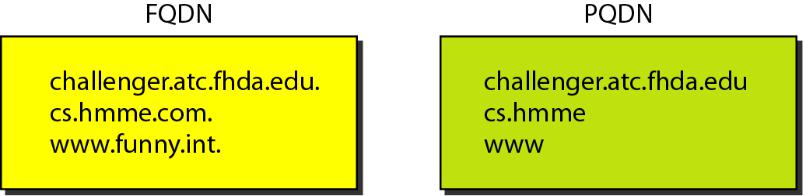
***Fully Qualified Domain Name***

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.

***Partially Qualified Domain Name***

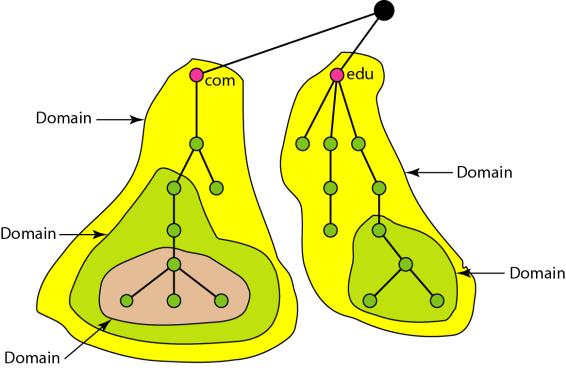
If a label is not terminated by a null string, it is called a partially qualified domain name (PQDN). A PQDN starts from a node, but it does not reach the root. It is used when the name to

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.



**Domain**

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. Figure 25.5 shows some domains. Note that a domain may itself be divided into domains (or **subdomains** as they are sometimes called).

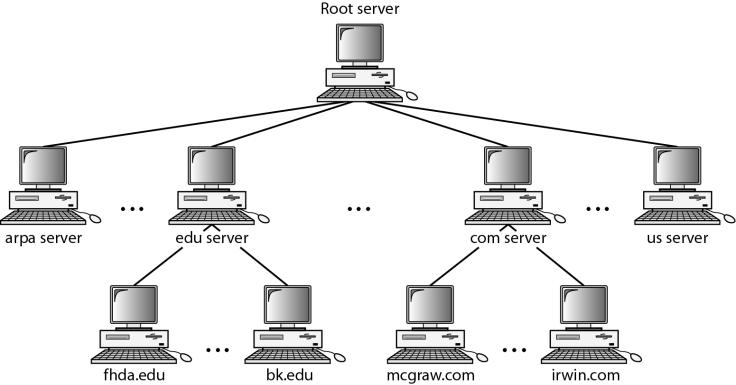


**5.8.3 DISTRIBUTION OF NAME SPACE**

The information contained in the domain name space must be stored. However, it is very inefficient and also unreliable to have just one computer store such a huge amount of information. It is inefficient because responding to requests from all over the world places a heavy load on the system. It is not unreliable because any failure makes the data inaccessible.

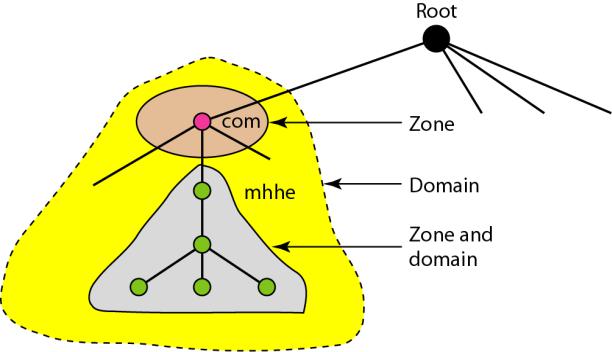
**Hierarchy of Name Servers**

The solution to these problems is to distribute the information among many computers called DNS servers. One way to do this is to divide the whole space into many domains based on the first level. In other words, we let the root stand alone and create as many domains (subtrees) as there are first-level nodes. Because a domain created in this way could be very large, DNS allows domains to be divided further into smaller domains (subdomains). Each server can be responsible (authoritative) for either a large or a small domain.



**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 for every node under that domain.



**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.

**Primary and Secondary Servers**

DNS defines two types of servers: primary and secondary.

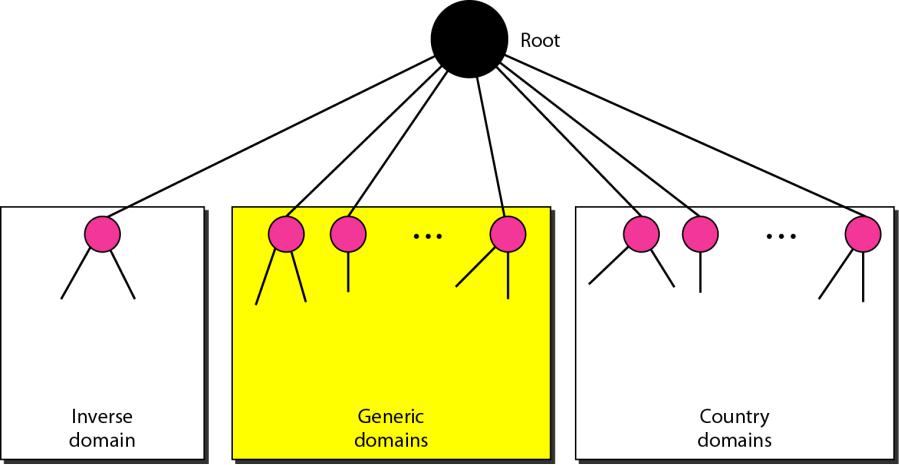
A primary server is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.

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.

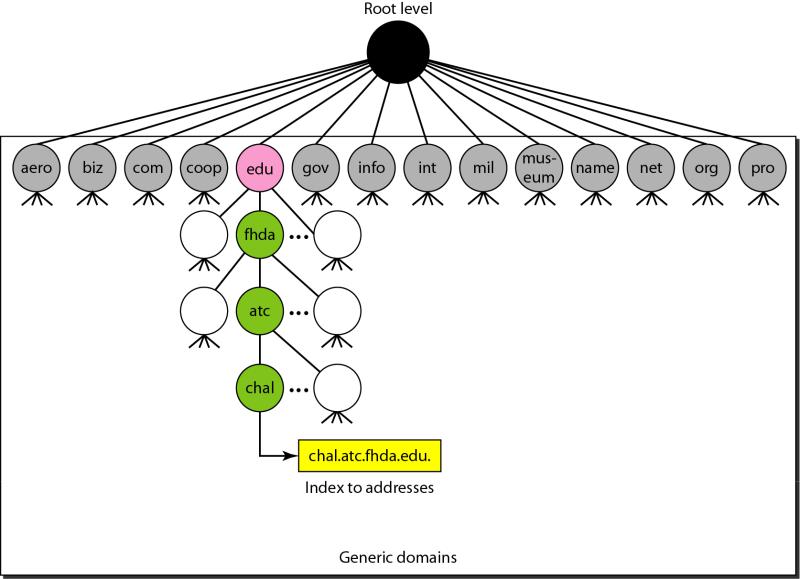
**5.8.4 DNS IN THE INTERNET**

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, and the inverse domain.



**Generic Domains**

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



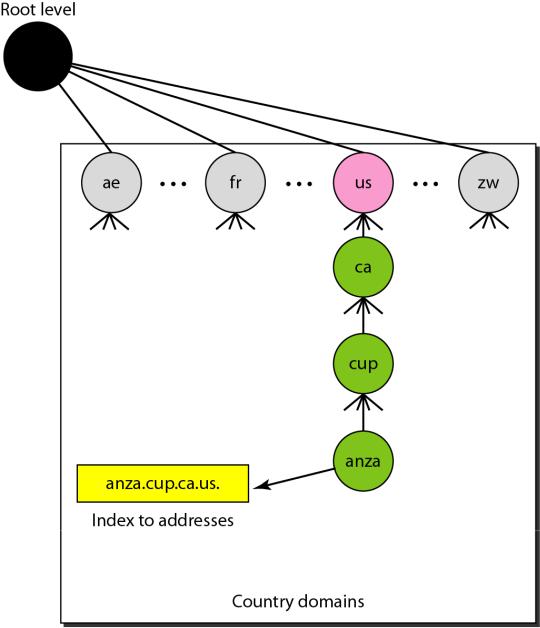
Looking at the tree, we see that the first level in the generic domains section allows 14 possible labels. These labels describe the organization types as listed in Table



**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.).

Below Figure shows the country domains section. The address *anza.cup.ca.us* can be translated to De Anza College in Cupertino, California, in the United States.

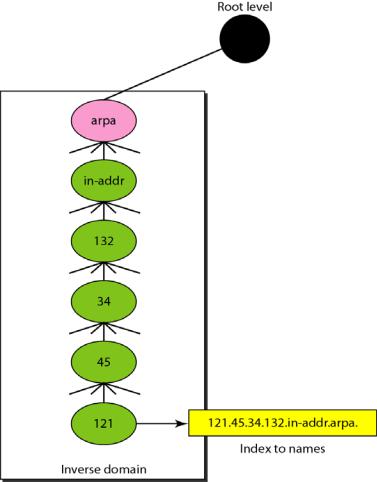


**Inverse Domain**

The inverse domain is used to map an address to a name. 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 IF address such as 132.35.45.121 (a class B address with netid 132.34) is read as 121.45.35.132.in-addr.



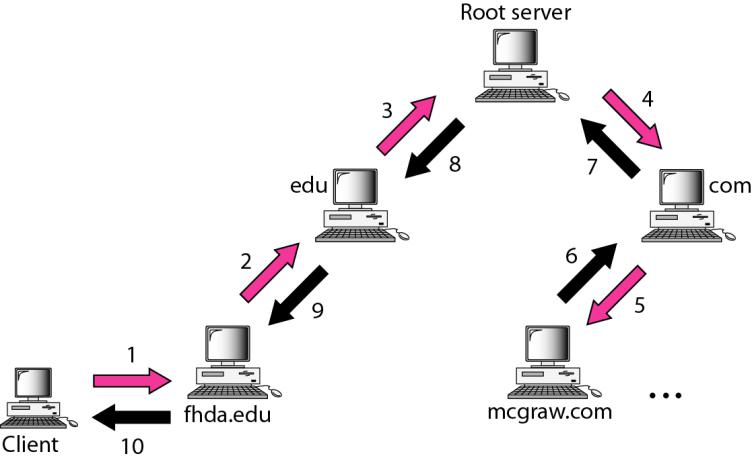
**5.8.5 RESOLUTION**

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

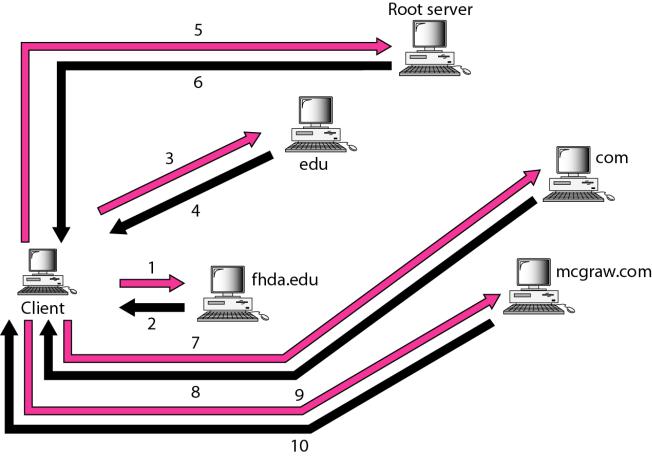
**Recursive Resolution**

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.



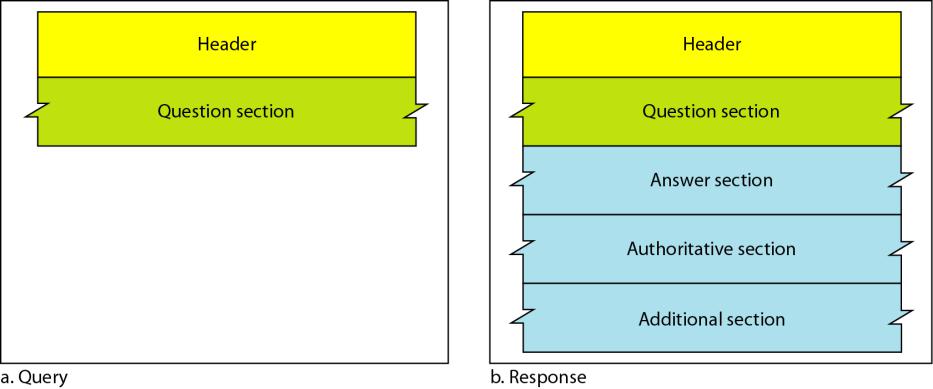
**Iterative Resolution**

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 IP address 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 answers the 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.



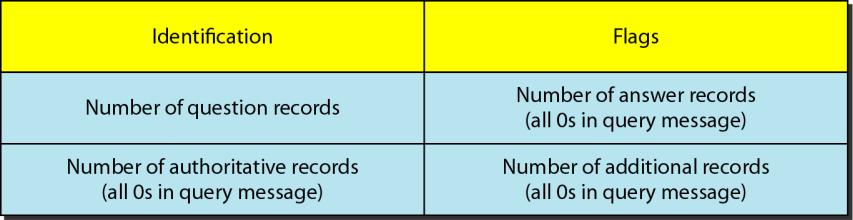
**5.8.6 DNS MESSAGES**

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.



**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 below 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* subfield contains the number of authoritative records in the authoritative sectionof a response message. Its value is zero in the query message. Finally, the *number of additional records* subfield 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. We will discuss the question records in a following section.

***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.

**5.8.7 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 resourcerecords 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 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. Resource records are also what is returned by the server to the client.

**5.8.8 REGISTRARS**

How are new domains added to DNS? This is done through a registrar, a commercial entity accredited by ICANN. 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 at

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

**Peer to Peer Applications**

Peer-to-Peer Live streaming and Video on Demand is the most popular media applications over the Internet in recent years

1.Video streaming

2.Content Distribution networks

**1.P2P Streaming :**

Streaming video and audio content over the Internet is becoming more popular, and the P2P streaming network model can be used for this application.

P2P video streaming does not need support from Internet routers, making it cost effective and simple to deploy. A peer simultaneously acts as a client and a server. A video stream can be downloaded from one set of peers and at the same time uploaded to another set of peers, so P2P streaming can significantly decrease the required bandwidth.

The goal of P2P streaming, or Peer to Peer Streaming, is to maximize the quality of media delivered to individual peers in a scalable fashion, despite the heterogeneity and irregularity of each peer’s capabilities and link speeds. The amount of resources available to the system physically grows with the number of peers. Each peer should be able to continuously receive suitable content from its connected peers, as long as those peers are offering enough outgoing bandwidth.

**P2P Video Streaming**

Compared to file exchange networks, big P2P video streaming networks have to resolve a number of system and networking issues. P2P streaming has stiff timing requirements for efficient audio and video content delivery. Usually, stream data is played and passed to other peers as soon as it’s received. Stream packets that are received with unadaptable delay are considered lost, and that causes visible degradation. Peers usually have limited upload speeds since the Internet was designed for the client/server paradigm and applications. Network congestion can also cause packets to be lost or delayed.

In a P2P network, video content may be forced to go through a number of intermediate receivers which may cause excessive end-to-end delay. Also, the behavior of peers is unpredictable. They may join and leave the service at any time, and therefore disrupt their successors. Receiving peers have to store some amount of data and exchange state information with each other to preserve connectivity. The control overhead at each receiver should be small to prevent excessive use of network resources and not to exceed the resource limitation at each receiver. Organizing the peers into a high quality overlay for distribution is a challenging problem, especially if the broadcasted video requires high bandwidth and low latency.

P2P streaming systems can be categorized as either live-streaming or on-demand.The technology behind them may be tree-based, mesh-based or a hybrid scheme.

**2.Peer-to-Peer Content Distribution**

[Peer-to-peer](https://www.techopedia.com/definition/31702/peer-to-peer-p2p) (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

It is a highly distributed computing model that leverages unused available computer and storage capacity on [endpoint devices](https://www.techopedia.com/definition/29619/endpoint-device) to store and share content. P2P allows users to share data without using a local server infrastructure, and minimizes the amount of traffic that must traverse the [wide area network](https://www.techopedia.com/definition/5409/wide-area-network-wan) (WAN) in getting content and updates to your end users.

Content may include any third-party applications such as Office 365 or Adobe, internally developed line of business applications, or operating systems that run on the enterprise endpoints. Endpoints can be anything that runs an operating system and the associated software, including desktop computers, PCs, kiosks, ATMs, point of sale terminals, etc.

A modern P2P content distribution system can intelligently distribute and pre-stage content so that a single download to a location will serve all the devices in that [subnet](https://www.techopedia.com/definition/9566/subnetwork-subnet). It can also manage the storage of multiple copies of content locally as well as make intelligent decisions and actions.

P2P models have been suggested for decentralized online social network applications, frameworks and architectures as an alternative of the current centralized solutions. Rather than leaving the control of all the users’ information in the hands of the provider, who can do anything with that data as we already know, zero-trust P2P social networks could provide the so-much needed anonymity and resistance against censorship.