2.1 Principles of Computer Applications

GTU : Summer-14

- Application layer provides the interface between the applications, use to communicate and underlying network.
- A process is a program in execution. When communicating processes are running on the same system, they communicate with each other using interprocess communication.
- Processes on two different end systems communicate with each other by message passing between computer networks.
- *Fig. 2.1.1 shows communication for network application at application layer.*

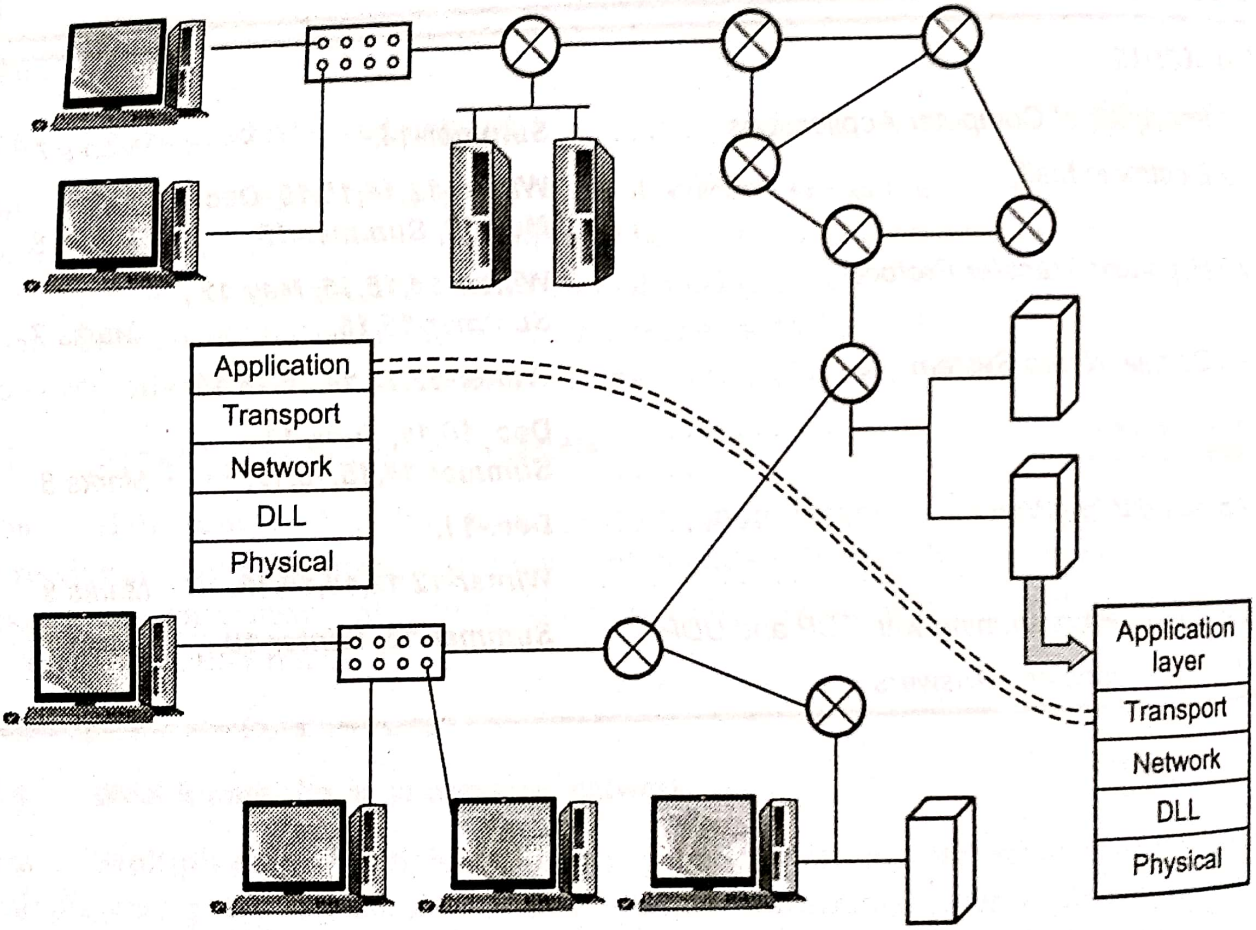


Fig. 2.1.1

- Networking applications have application layer protocols. They decide the rules for communication along with message format.

2.1.1 Application Layer Protocols

- World Wide Web, electronic mail system and domain name system are the traditional application of the application layer network.

- Applications need their own protocols. These applications are part network protocol and part traditional application program.
- Here we study some of the most popular network applications available today. Two of the most popular applications are World Wide Web and Email system.
- Both of these applications use the request/reply method. The users send requests to servers, which then respond accordingly.
- These two application uses various protocol while exchanging the information. So it is important to distinguish between application protocols from application programs.
- Hyper Text Transport Protocol (HTTP) is an application protocol. HTTP is used to retrieve Web pages from remote servers.
- A web client uses application programs like Internet Explorer, Chrome, Firefox and Mozilla. All of them use the HTTP protocol for communication with web server.
- Widely used standardized application protocols are SMTP and HTTP
  1. Simple Mail Transfer Protocol is used to exchange electronic mail.
  2. HTTP is used to communicate between Web browsers and Web servers.

Client and Server Side Application

- Web browser is client side application of HTTP. Web server is server side application of HTTP.
- In electronic mail system, sending mail server implements the client side of SMTP and the receiving mail server implements the server side of SMTP.

Processes Communicating Across Network

- Two processes communicate with each other by sending and receiving messages. Socket is used for process communication. Socket is an interface between the application layer and transport layer within a machine.
- Socket is also referred as the application programmer's interface (API) between the application and the network. Multiple sockets might exist in each host. A port number identifies each such socket in each host.
- **Client process :** Process that initiates communication.
- **Server process :** Process that waits to be contacted.
- Application layer protocols used by both the source and destination device during a communication session. The protocols implemented on the source and destination host must match.

- The client process begins the exchange by requesting data from the server. Server responds by sending one or more streams of data to the client.

Addressing Processes

- A host (sender) uses the address of the destination host to specify where the message should sent. When data is received at the host, the port number is examined to determine which application or process is the correct destination for the data.
- Each device on a network must be uniquely defined. In Internet applications, the destination host is identified by its IP address.
- Fig. 2.1.2 shows application processes, socket and transport protocol.

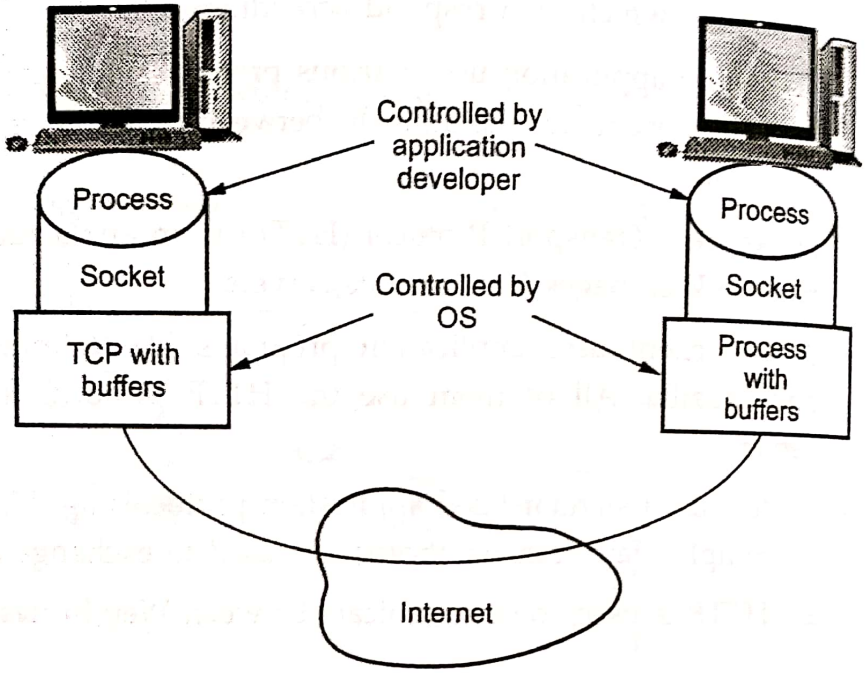


Fig. 2.1.2 Application processes, socket and transport protocol

2.1.2 Types of Services Required for Application

- Application service requirements are as follows :
  1. Data Loss
  2. Bandwidth
  3. Timing
- Following application requires reliable data transfers :
  a. File transfer b. E-mail
  c. Remote host access d. Instant messaging
  e. Financial application f. Web document transfer
- Loss of file data creates problem. Loss of data strongly depends upon the application and coding scheme used.

Bandwidth

- Some applications like multimedia require a minimum amount of bandwidth to be effective.

- Many current multimedia applications are bandwidth sensitive.
- The application which requires little or less bandwidth, they are called *elastic applications*. Example of elastic applications are electronic mail, file transfer and remote access and web transfers.

Timing

- Some application requires low delay.
- Internet telephony, teleconferencing, multiplayer games requires tight timing constraints on data delivery.

Review Question

1. *Explain any two application layer protocol.* **GTU : Summer-14, Marks 7**

2.2 Electronic Mail

GTU : Winter-12,14,15,19, Dec.-10,11, May-12, Summer-15

- E-mail is an asynchronous communication medium. Electronic mail is used for sending a single message that includes text, voice, video or graphics to one or more recipients.
- Electronic mail is fast, easy to distribute and inexpensive.
- Simple Mail Transfer Protocol (SMTP) is the standard mechanisms for electronic mail in the internet. SMTP is the TCP/IP mail delivery protocol.
- E-mail is not a real-time service in that fairly large delays can be tolerated.
- It is also not connection oriented in that a network connection does not need to be setup expressly for each individual message.
- Fig. 2.2.1 shows the high-level view of the internet e-mail system.
- Mail server handles incoming and outgoing mails.
- The Post Office Protocol (POP) servers store incoming mail while SMTP servers relay outgoing mails.
- The Internet Service Provider (ISP) probably runs both an SMTP server and POP server for its customers.

Following are the ways to access the e-mail.

1. Web based e-mail service.
2. E-mail through a LAN.
3. Unix shell account.
4. Using mail client.

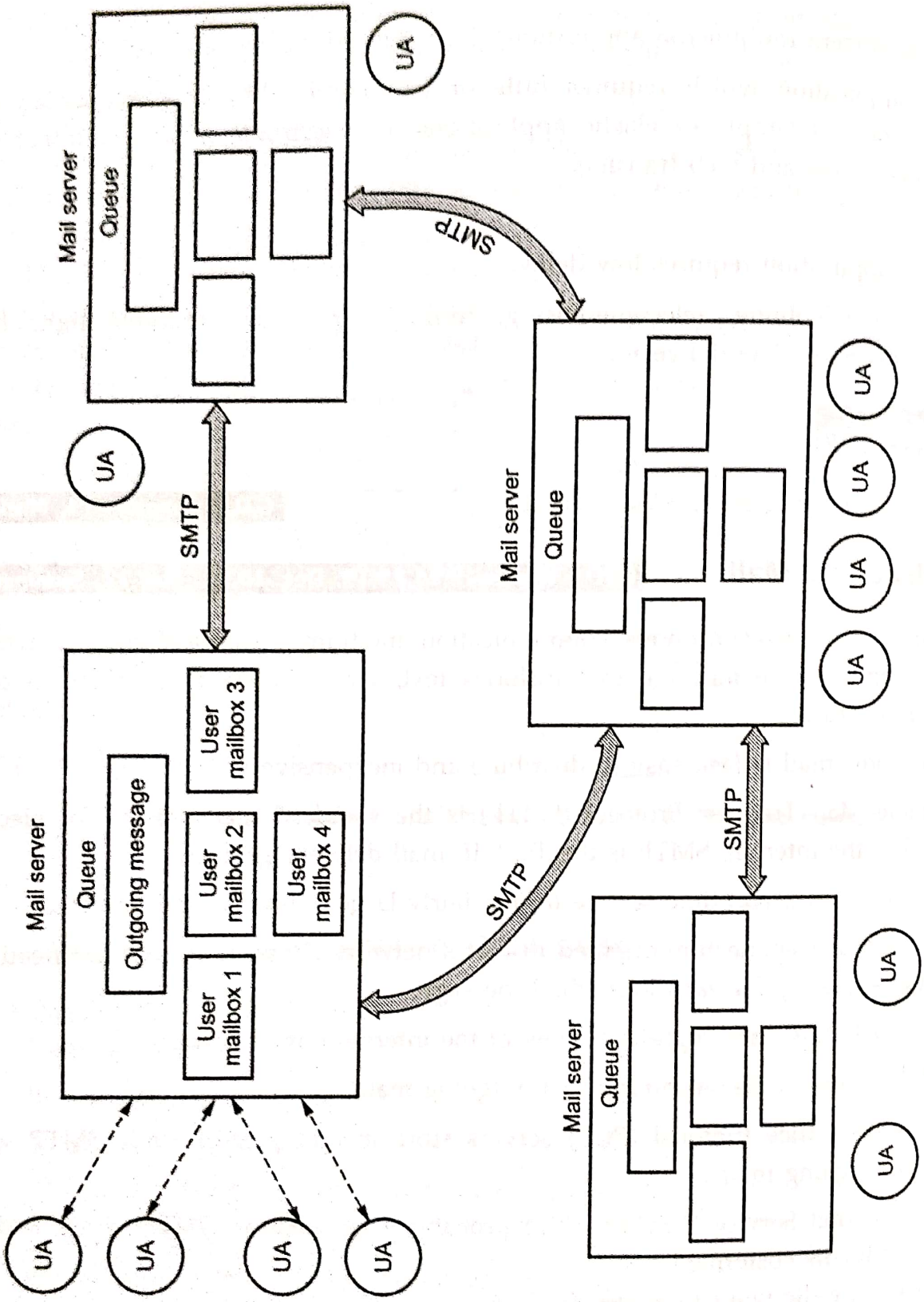


Fig. 2.2.1 View of e-mail system

Components

Three major components are

1. User agents.
2. Mail servers.
3. SMTP.

Fig. 2.2.2 shows the components of an e-mail system.

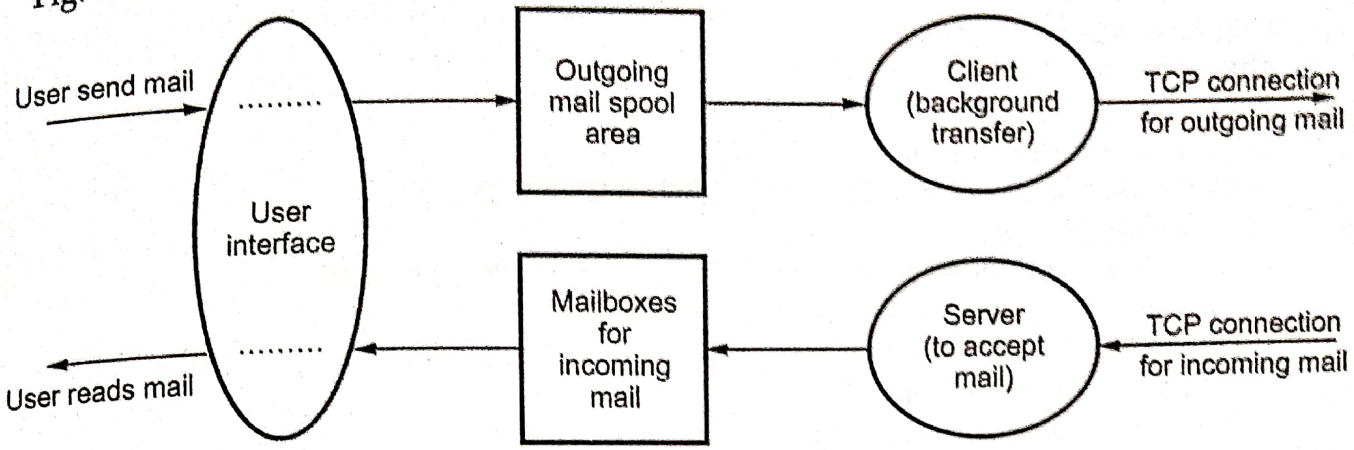


Fig. 2.2.2 Component of e-mail system

2.2.1 E-mail Addressing

- To send e-mail to some one, the Internet e-mail address must be known to sender. E-mail addresses look like this : vilas@hotmail.com.
- The e-mail address has two main parts, joined by @. In this example, vilas is the username. Username can contain numbers, underscores, periods and some other special characters. Commas, spaces and parentheses are not allowed.
- Hotmail.com is the host or domain name. E-mail address is case insensitive. **Vilas@hotmail.com** works just the same as **vilas@hotmail.com**. E-mail addresses do not have punctuation marks around them.

2.2.2 Message Headers

The message headers include the addresses of the receiver and the sender. Each header consists of the type of header, a colon, and the content of the header. Following is the sample of the complete header for a message.

Table shows the list of standard header.

```
Received:from del2.vsnl.net.in(del2.vsnl.net.in [202.54.15.30])
by giaspn01.vsnl.net.in (8.9.0/8.9.0) with ESMTP id MAA22885
for<siitpune@giaspn01.vsnl.net.in>;wed,19 jul 2000
      12:42:33+0530 (IST)
Received: from oemcomputer ([202.54.109.165])
by del2.vsnl.net.in (8.9.2/8.9.2)with SMTP id MAA12595
for<siitpune@giaspn01.vsnl.net.in>;wed, 19 Jul 2000
12:47:52-0500 (GMT)
Reply-To: <kanohar@del2.vsnl.net.in>
From:"SachinMahadik" <kanohar@del2.vsnl.net.in>
To: <siitpune@giaspn01.vsnl.net.in>
Subject: admission
Date: Wed, 19 Jul 2000 12:43:31 +530
```

```
Message-ID:<LPBBKDKDNBJBIDPNDOLHOECOCBAA.kanohar@del2.vsnl.net.in>
MIME-version: 1.0
Content-Type: text/plain;
charset="iso-8859-1"
X-Priority: 3 (Normal)
X-MSMail-Priority: Normal
X-Mailer:Microsoft outlook IMO, Build 9.0.2416 (9.0.2910.0)
Importance: Normal
X-MimeOLE : Produced by Microsoft MimeOLE V5.00.2314.1300
Disposition-Notification-To:"BrijeshSinghal"
<kanohar@del2.vsnl.net.in>
Content-Transfer-Encoding: 8bit
X-MIME-Autoconverted:from quoted-printable to 8bit by
giaspn01.vsnl.net.in id MAA22885
X-UIDL: 69c2d7f9f63fef91eaf7c61f05d2b550
X-Mozilla-Status: 8003
```

2.2.3 Formatted E-mail

- E-mail that supports formatting such as boldface and underlining is a recent development. In the past, e-mail consists only of text characters. If both side supports the formatted e-mail, then both sides use send/receive formatted e-mail. The formatted e-mail comes in the following type.
  a) HTML
  b) Rich text
  c) Multipurpose Internet Mail Extension (MIME)
  d) MS word format
- HTML tags are just like web pages. It can include text formatting, numbering, bullets, horizontal lines, backgrounds, hyperlinks and HTML styles. It uses MIME protocol for sending. Rich text can be read by most word processing applications. MIME formatting are created just for e-mail.
- MIME formatting can include text formatting, pictures, video, sound, and other information. MS word format uses microsoft word and all of its features as your e-mail editor.
- To allow transmission of non-ASCII data through e-mail, the MIME used. MIME does not change SMTP or replace it. MIME allows arbitrary data that is to be encoded in ASCII and then transmitted in a standard e-mail message.
- Fig. 2.2.3 shows the MIME message that contains a photograph in standard GIF representation. The GIF image has been converted to a 7-bit ASCII representation using the base 64 encoding.

```
From : Ravindra@hotmail.com
To : Avinash@hotmail.com
MIME-version : 1.0
Content-Type : imag/gif
Content-Transfer-Encoding : base64
.....data for the image.....
- - - - - - - - - - - - - - - - - - - - - - -
```

Fig. 2.2.3 Example MIME message

- The MIME-version declares that the message was composed using version 1.0 of the MIME protocol. The content-type declaration specifies that the data is GIF image and the content-transfer-encoding header declares that base-64, encoding was used to convert the image to ASCII. To view the image, a receiver mail system must first convert from base 64 encoding back to binary.
- A content-type declaration must contain two identifiers, a content-type and a subtype, separated by a slash. In the example, image is the content type and gif is the subtype. The standard defines seven basic content types.

Sr. No.	Content type	Use
1.	Text	Textual (document)
2.	Image	Photograph
3.	Audio	A sound recording
4.	Video	A video recording including motion
5.	Application	Raw data for program
6.	Multipart	Multiple messages
7.	Message	An entire e-mail message

2.2.4 Functions of E-mail

- E-mail system support five basic functions. They are as follows -
  1. Composition
  2. Transfer
  3. Reporting
  4. Displaying
  5. Disposition

1. Composition : It is a process of creating messages and answers. Any text editor can be used for the body of the message. When answering a message, the e-mail system can extract the originator's address from the incoming e-mail.

2. Transfer : It is moving messages from the originator to the receiver.
3. Reporting : It inform the originator what happened to the message. Whether, email is delivered or not delivered.
4. Displaying : Display is required for reading the email.
5. Disposition is the last step and related what the receiver does with the message after receiving it. It may be read and save or delete or forward the message.

2.2.5 User Agent and Message Transfer Agent

E-mail system consists of two subsystems.

1. User agent
2. Message transfer agent.

1. User Agent (UA)

- User agent is an interface between user and network application.
- It allow user to read and send e-mail. The user agents are local program that provide a command based, menu based or graphical method for interacting with the e-mail system.
- To send an e-mail message, a user must provide the data and the destination address. The destination address should be in proper format and the user agent can deal with destination address. Details of e-mail address, we already studied in email addressing section.
- Most e-mail system support mailing lists, so that a user can send the same message to a list of people with a single command.
- For reading e-mail, the user agent will look at the user's mail box for incoming e-mail before displaying anything on the screen. It display total number of new mail.

2. Message transfer agent

- Message Transfer Agent (MTA) move the messages from the source to the destination. MTA are system program that run in the background and move e-mail through the system.
- After writing the mail, user click of send icon. MTA activates at this time, MTA checks the destination address and transfer the mail to proper destination on the network.
- MTA use different types of protocol for moving the message from source to destination.

1. It must handle temporary failures, if a destination machine is temporarily unavailable, it must spool the message on the local machine for later delivery.
2. MTA must distinguish between local and remote destinations.
3. It may have to deliver copies of a message to several machines.
4. It may allow mixing text, voice and video in a message as well as appending documents and files to a message.

- MTA works in background, while the user usually interacts directly with a user agent.

2.2.6 Simple Mail Transfer Protocol (SMTP)

- SMTP is application layer protocol of TCP/IP model.
- SMTP transfers message from sender's mail servers to the recipients mail servers.
- SMTP interacts with the local mail system and not the user.
- SMTP uses a TCP socket on port 25 to transfer e-mail reliably from client to server.
- E-mail is temporarily stored on the local and eventually transfered directly to receiving server.
- Client / Server interaction follows and commond/reponse paradigm.
  a] Commands are plain ASCII text.
  b] Responses are a status code and an optional phase.
  c] Command and response lines terminated with CRLF.
- Mail client application interacts with a local SMTP server to initiate the delivery of an e-mail message.
- There is an input queue and an output queue at the interface between the local mail system and the client and the server parts of the SMTP.
- The client is concerned with initiating the transfer of mail to another system while server is concerned with receiving mail. Before the e-mail message can be transferred, the application process must be set up a TCP connection to the local SMTP server. The local mail system retains a mailbox for each user into which the user can deposit or retrieve mail. Mail handling system must use a unique addressing system.
- Adderssing system used by SMTP consists of two parts : A local part and a global part. The local part is the user name and is unique only within that local mail system. Global part of the address is the domain name. Domain name is identity of the host, must be unique within the total internet.

- SMTP uses different types of component. They are MIME and POP.

Scenario : Alice sends message to Bob

1. Alice uses User Agent (UA) to compose message and send to bob@technical.org.
2. Alice's UA sends message to her mail server, message placed in message queue.
3. Client side of SMTP opens TCP connection with Bob's mail server.
4. SMTP client sends Alice's message over the TCP connection.
5. Bob's mail server places the message in Bob's mailbox.
6. Bob invokes his user agent to read message.

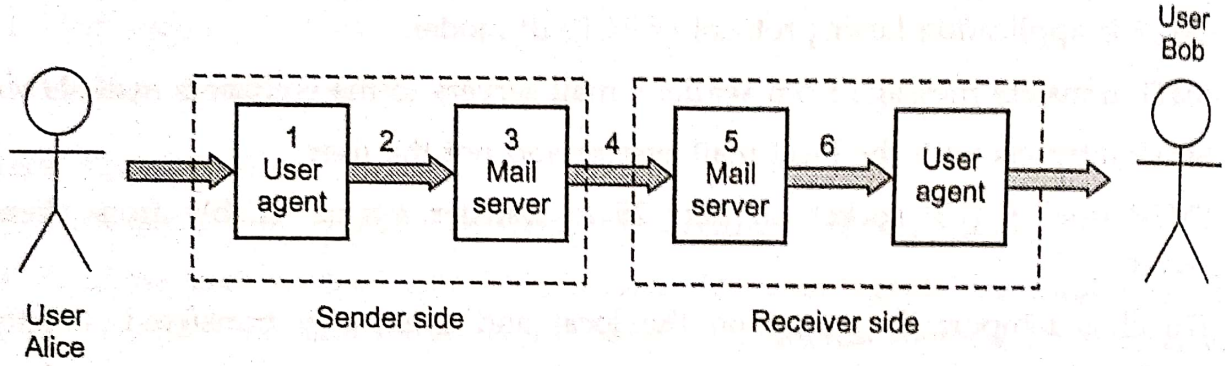


Fig. 2.2.4 Message Scenario

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

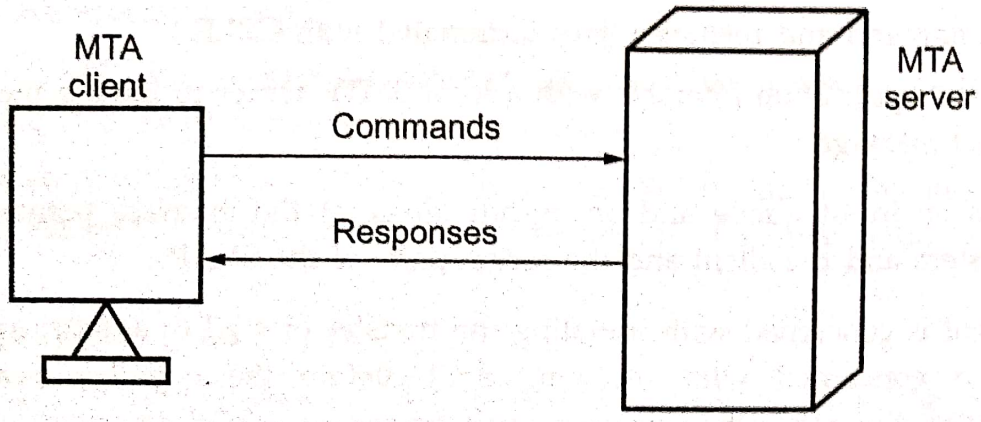


Fig. 2.2.5 Command / Response

- Each command or reply is terminated by a two character end of line token.
- Commands are sent from the client to the server. SMTP defines 14 commands. SMTP commands consist of human readable ASCII strings.

  SMTP commands are,

  i) **HELO :** Initiate a mail transaction, identifying the sender to the recipient.

ii) **MAIL FROM** : Tells the remote SMTP that a new mail transaction is beginning.

iii) **RCPT TO** : The sending SMTP sends a RCPT command for each intended receiver.

iv) **DATA** : If accepted, the sender transfers the actual message. End of message is indicated by sending a " • " on a line by itself.

v) **QUIT** : Terminate the connection.

Sample SMTP Interaction

- Following are messages exchanged between an SMTP client (C) and an SMTP server (S).
- The host name of the client is iresh.fr and the host name of the server is sinhgad.edu.

S : 220 sinhgad.edu

C : HELO iresh.fr

S : 250 Hello iresh.fr, pleased to meet you

C : MAIL FROM : <rupali@iresh.fr>

S : 250 rupali@iresh.fr ... sender ok

C : RCPT TO : < rakshita@sinhgad.edu>

S : 250 rakshita@sinhgad.edu ..... Recipient ok

C : DATA

S : 354 Enter Mail, end with " · " on a line by itself

C : Do you like Apple ?

C : What about school ?

C : ·

S : 250 message accepted for delivery

C : QUIT

S : 221 sinhgad.edu closing connection

2.2.7 Multipurpose Internet Mail Extensions

- MIME is a supplementary protocol that allows non-ASCII data to be sent through SMTP.
- MIME defined by IETF to allow transmission of non-ASCII data via e-mail.
- It allows arbitary data to be encoded in ASCII for normal transmission.

- All media types that are sent or received over the world wide web (www) are encoded using different MIME types.
- Messages sent using MIME encoding include information that describes the type of data and the encoding that was used.
- RFC822 specifies the exact format for mail header lines as well as their semantic interpretations.
- Fig. 2.2.6 shows the working of MIME.

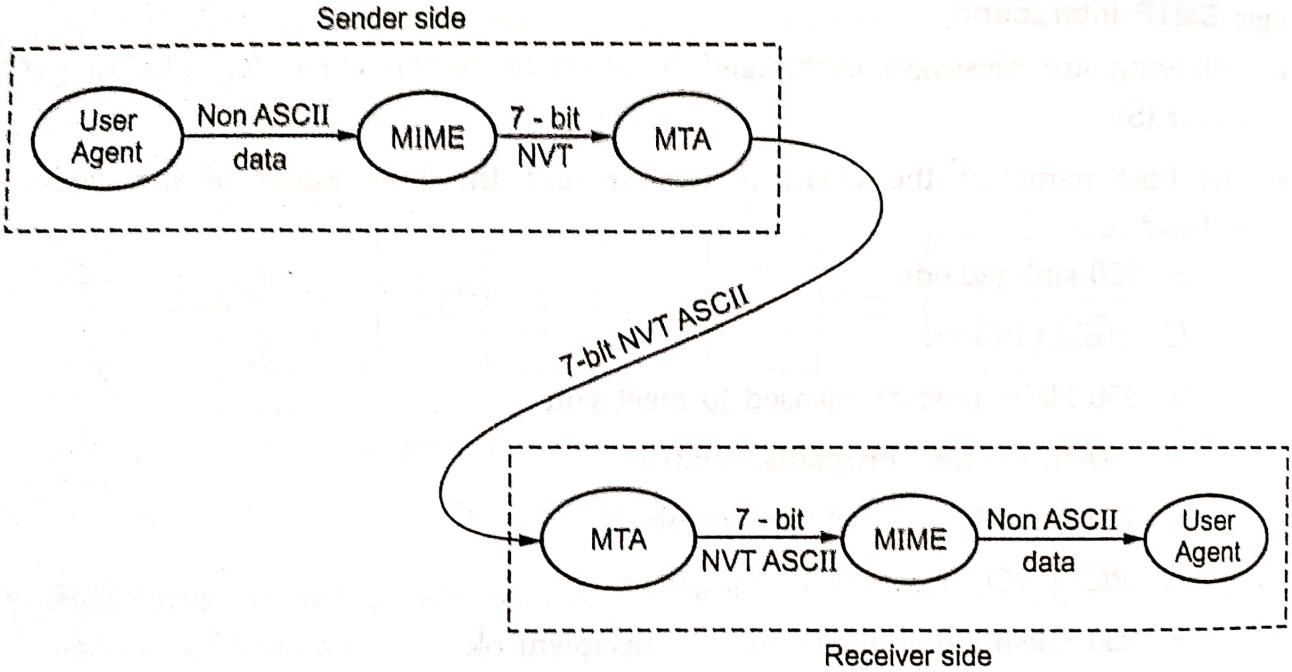


Fig. 2.2.6 MIME

- MIME define five headers.
  1. MIME - Version
  2. Content - Type
  3. Content - Transfer - Encoding
  4. Content - Id
  5. Content - Description

Mail Message Header

- From : iresh@e-mail.com
- TO : rupali@sinhgad.edu
- MIME - Version : 1.0
- Content - Type : image/gif
- Content - Transfer - Encoding : base64

..... data for the image .....

....

....

MIME Types and SubTypes

- Each MIME content - type must contain two identifiers :
  - Content type
  - Content subtype
- There are seven standardized content-types that can appear in a MIME content - type declaration.

Type	Subtype	Description
Text	Plain	Unformatted text.
Multipart	Mixed	Body contains ordered parts of different data types.
	Parallel	Same as above, but no order.
	Digest	Similar to mixed, but the default is message.
	Alternative	Parts are different versions of the same message.
Video	MPEG	Video is in MPEG format.
Audio	Basic	Single channel encoding of voice at 8kHz. (Sound file)
Image	JPEG	Image is in JPEG format.
	GIF	Image is in GIF.
Message	Partial and external body	An entire e-mail message or an external reference to a message.
Application	Postscript	Adobe postscript.
	Octet stream	General binary data.

Content - Transfer Encoding

- This header defines the method to encode the messages into 0 and 1 for transport.

Content-Transfer-Encoding : < Type >

The five types of encoding is listed below.

Type	Description
7-bit	ASCII characters and short lines.
8-bit	Non-ASCII characters and short lines.
Binary	Non-ASCII characters with unlimited length lines.
Base 64	6-bit blocks of data are encoded into 8-bit ASCII characters.
Quoted printable	Non-ASCII characters are encoded as an equal sign followed by an ASCII code.

2.2.8 Post Office Protocol (POP)

- Post Office Protocol 3 (POP3) is used to transfer e-mail messages from a mail server to mail client software.
- Fig. 2.2.7 shows working of POP3.
- POP3 begins when the user agent opens a TCP connection to the mail server on port 110.
- After TCP connection established, POP3 progresses three phases :
  i) Authorization
  ii) Transaction
  iii) Update
- In authorization phase, user agent sends a user name and a password to authenticate the user downloading the mail.
- In transaction phase, the user agent retrieves messages. In this phase, user agent can also mark messages for deletion, remove deletion marks.
- In update phase, it occurs after the client has issued the quit command, ending the POP3 session.
- POP3 has two modes : Delete mode and the keep mode.
- In the delete mode, mail is deleted from the mailbox after each retrieval.
- In the keep mode, the mail remains in the mailbox after retrieval.

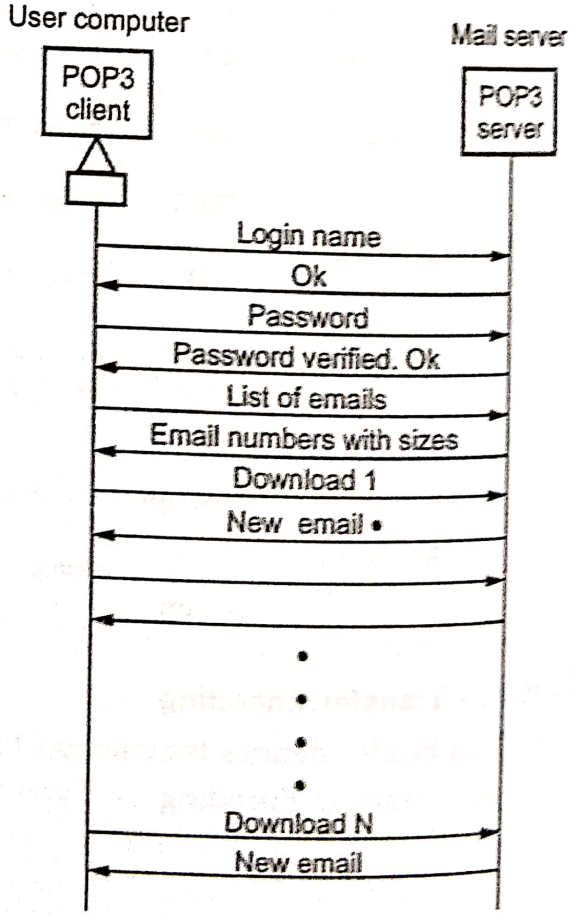


Fig. 2.2.7 POP3

Limitations of POP3

1. POP3 does not allow the user to organize mail on the server, the user cannot have different folders on the server.
2. POP3 does not allow the user to partially check the contents of the e-mail before downloading.

2.2.9 IMAP

Status line	HTTP / 1.1 300 ok
General headers	Date : Wed , 8 Oct 2014 13:00:13 GMT
	Connection : close
Entity headers	Server : Apache / 1.3.27
	Accept-range : bytes
	Content-type : text / html
	Content-length : 200
	Last-modified : 2 Oct 2014 13:00:13 GMT
Blank line	
Message body	<html> <head> <title> Welcome to the India <title> <head> <body>

- IMAP is the Internet Mail Access Protocol. IMAP4 is more powerful and more complex. IMAP is similar to SMTP.
- IMAP allows users to store their email on remote server.
- It was designed to help the user who uses multiple computers.
- IMAP does not copy e-mail to the user's personal machine because the user may have several.
- An IMAP client connects to a server by using TCP.
- IMAP supports the following modes for accessing e-mail messages :
  i) Offline mode ii) Online mode iii) Disconnected mode

Offline mode : A client periodically connects to the server to download e-mail messages. After downloading, messages are deleted from the server. POP3 support this mode.

Online mode : Client process e-mail messages on the server. The e-mail messages are stored on the server itself but are processed by an application on the client's end.

Disconnected mode : In this mode, both offline and online modes are supported.

IMAP4 provides the following extra functions :

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

- The IMAP protocol provides commands to allow users to create folders and move messages from one folder to another.

Fig. 2.2.8 shows IMAP state transition diagram.

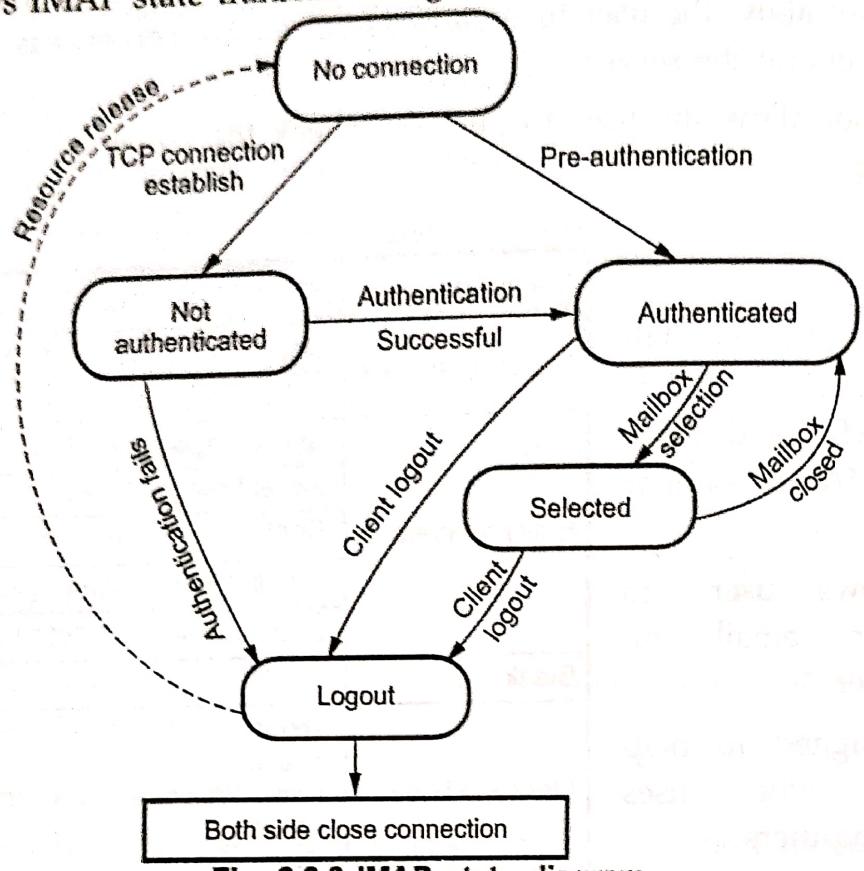


Fig. 2.2.8 IMAP state diagram

1. **Not authenticated :** Client provides authentication information to the server.
2. **Authenticated :** Server verify the information and client is now allowed to perform operations on a mailbox.
3. **Selected :** Client is allowed to access manipulated individual messages within the mailbox.
4. **Logout :** Client send logout command for closing IMAP session.

Review Questions

1. *Write note on following : 1) MIME* **GTU : Winter-14, Marks 3**
2. *Explain the e-mail architecture and services. Write short note on POP3 and MIME.* **GTU : Dec.-10, Marks 7**
3. *What is e-mail ? How it works ? Which protocol it uses ?* **GTU : Dec.-11, Marks 5**
4. *E-mail systems contain which two subsystems ? Write the five basic functions provided by e-mail system.* **GTU : May-12, Marks 7**
5. *Explain the basic functions of the e-mail system.* **GTU : Winter-12, Marks 7**
6. *Explain the working of electronic mail protocols SMTP, IMAP and POP3 in brief with suitable diagram.* **GTU : Summer-15, Marks 8**
7. *Explain the high-level view of Internet e-mail system and its major components.* **GTU : Winter-15, Marks 8**
8. *Why do HTTP, FTP, SMTP, and POP3 run on top of TCP rather than UDP ? Name one application that uses UDP and why ?* **GTU : Winter-19, Marks 4**

2.3 Hypertext Transfer Protocol GTU : Winter-14,15,16, May-12, Summer-15,16

- The standard web transfer protocol is Hyper Text Transfer Protocol (HTTP).
- The HTTP protocol consists of two fairly distinct items: The set of requests from browsers to servers and the set of responses going back the other way.
- All the newer versions of HTTP support two kinds of requests: Simple requests and full requests. A simple request is just a single GET line naming the page desired, without the protocol version. The response is just the raw page with no headers, no MIME, and no encoding. To see how this works, try making a Telnet connection to port 80 of www.w3.org and then type.
  GET /hypertext/www/TheProject.html
  but without the HTTP/1.0 this time. The page will be returned with no indication of its content type. This mechanism is needed for backward compatibility. Its use will decline as browsers and servers based on full requests become standard.
- Full requests are indicated by the presence of the protocol version on the GET request line. Requests may consist of multiple lines, followed by a blank line to indicate the end of the request. The first line of a full request contains the command (of which GET is but one of the possibilities), the page desired, and the protocol/version. Subsequent lines contain RFC 822 headers.
- Although HTTP was designed for use in the Web, it has been intentionally made more general than necessary with an eye to future object-oriented applications. For this reason, the first word on the full request line is simply the name of the **method** (command) to be executed on the web page (or general object).
- When accessing general objects, additional object-specific methods may also be available. The names are case sensitive, so, GET is a legal method but get is not.

HTTP Transaction

- HTTP uses the services of TCP. HTTP is a stateless protocol.
- The client initilizes the transaction by sending a request message. The server replies by sending a response.
- Fig. 2.3.1 shows HTTP transaction

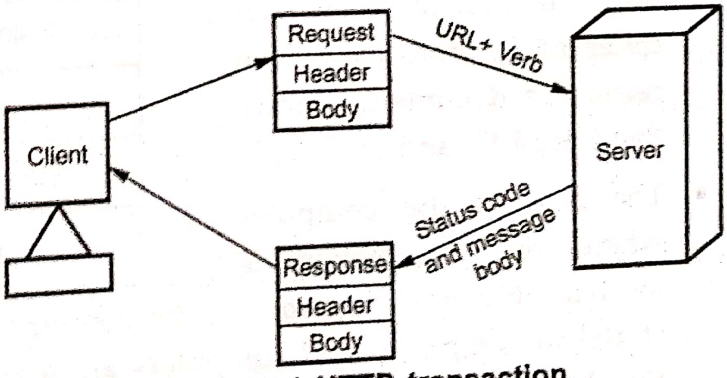


Fig. 2.3.1 HTTP transaction

Message

- HTTP messages are two types
  1. Request 2. Response

- Both message type used same format.
- Request **message** consists of a request line, headers and a body. Fig. 2.3.2 shows request message.

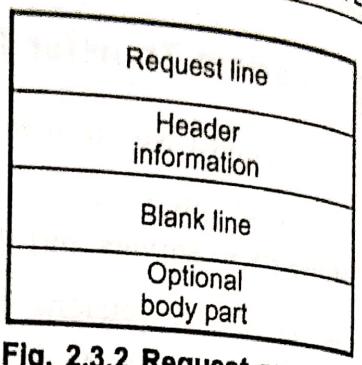


Fig. 2.3.2 Request message

Request line

- Request line defines the
  1. Request type
  2. Resource
  3. HTTP version
- Request type categorizes the request message into several methods for HTTP version 1.1.
- Fig. 2.3.3 shows the request line.

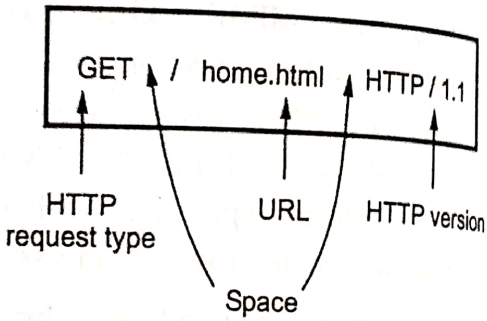


Fig. 2.3.3 Request line

- URL is a standard for specifying any kind of information on the internet. The URL define four things.
  1. Method
  2. Host computer
  3. Port
  4. Path

Fig. 2.3.4 shows the URL.

Fig. 2.3.4 (a) URL

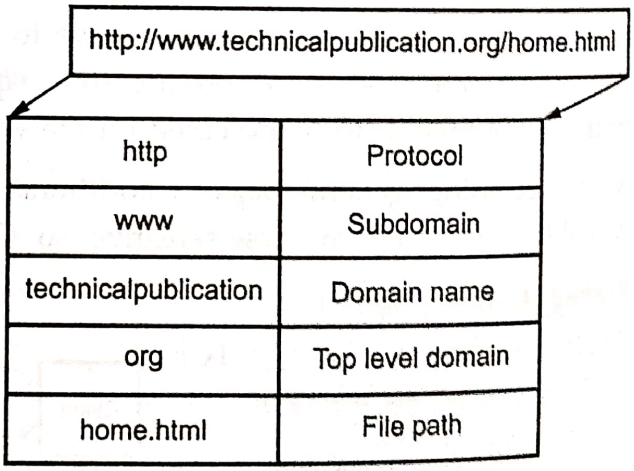


Fig. 2.3.4 (b) URL example

- The method is the prototocol used to retrieve the document. Several different protocols can retrieve a document, among them are FTP and HTTP.
- The host is the computer where the information is located, although the name of the computer can be alias. Web pages are usually stored in computers and computers are given alias names that usually begin with the character www.
- The URL can optionally contain the port number of the server.
- Path is the path name of the file where the information is located.
- The request type field in a request message defines several kinds of messages referred to as methods.

Sr. No.	Method	Purposes
1.	GET	Used when the client wants to retrieve a document from the server. Server responds with the contents of the document.
2.	HEAD	Used when client wants some information about a document but not the document itself.
3.	POST	Used by the client to provide some information to the server i.e. input to the server.
4.	PUT	Used by the client to provide a new or replacement document to be stored on the server.
5.	PATCH	Similar to PUT except that the request contains a list of differences that should be implemented in the existing file.
6.	DELETE	Removes a document on the server.
7.	COPY	Copies a files to another location. URL gives the location of the source file.
8.	MOVE	Move a file to another location.
9.	LINK	Creates a link or links from a document to another location.
10.	UNLINK	UNLINK method deletes links created by the LINK method.
11.	OPTION	This method is used by the client to ask the server about available options.

- The GET method requests the server to send the page (by which we mean object in the most general case) suitably encoded in MIME. However, if the GET request is followed by an If-Modified-Since header, the server only sends the data if it has been modified since the data supplied. Using this mechanism, a browser that is asked to display a cached page can conditionally ask for it from the server, giving the modification time associated with the page. If the cache page is still valid, the server just sends back a status line announcing that fact, thus eliminating the overhead of transferring the page again.
- The HEAD method just asks for the message header, without the actual page. This method can be used to get a page's time of last modification, to collect information for indexing purposes, or just to test a URL for validity. Conditional HEAD request do not exist.
- The PUT method is the reverse of GET : Instead of reading the page, it writes the page. This method makes it possible to build a collection of web pages on a remote server. The body of the request contains the page. It may be encoded using MIME, in which case the lines following the PUT might include content type and authentication headers, to prove that the caller indeed has permission to perform the requested operation.
- Somewhat similar to PUT is the POST method. It too bears a URL, but instead of replacing the existing data, the new data is "appended" to it in some generalized

sense. Posting a message to a news group or adding a file to a bulletin board system are example of appending in this context. It is clearly the intention here to have the web take over the functionality of the USENET news system.

- DELETE does what you might expect; it removes the page. As with PUT authentication and permission play a major role here. There is no guarantee that DELETE succeeds, since even if the remote HTTP server is willing to delete the page, the underlying file may have a mode that forbids the HTTP server from modifying or removing it.
- The LINK and UNLINK methods allow connections to be established between existing pages or other resources.

Response Message

- Fig. 2.3.5 shows the response message. It contains a status line, a header and body.
- Status line defines the status of the response message. It consists of the
  a. HTTP version b. Space
  c. Status code d. Space e. Status phrase

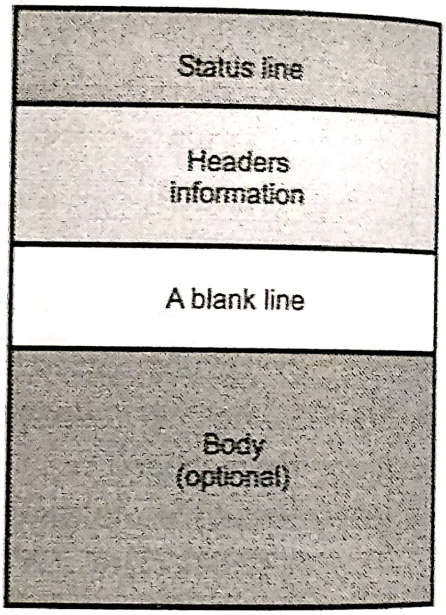


Fig. 2.3.5 Response message

Headers

- Header can be one or more header lines. Each header line is made of a header name, a colon, a space and a header value.
- The header exchange additional information between the client and the server.
- A header line belongs to one of four categories : general header, request header, response header and entity header.
- Fig. 2.3.6 shows the header format.

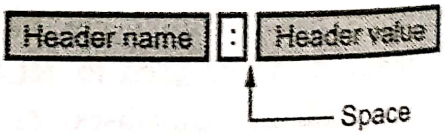


Fig. 2.3.6 Header format

- *General header* includes general information about the message. Request and a response both contains general header.
- *Response header* can be present only in a response message. It specifies the servers configuration and special information about the request.
- Request header can be present only in a request message. It specifies the clients configuration and the client preferred document format.
- *Entity header* gives information about the body of the document. It is mostly present in response messages, some request message, such as POST and PUT methods, that contain a body also use this type of header.

- Fig. 2.3.7 shows the headers.

Status line	HTTP/1.1 300 OK
General Headers	Date : Wed, 8 Oct 2014 13:00:13 GMT
	Connection : close
Response Headers	Server : Apache /1.3.27
	Accept-Ranges : bytes
Entity Headers	Content-Type : text/html
	Content-Length : 200
	Last-Modified : 2 Oct 2014 13:00:13 GMT
Blank Line	
Message Body	<html>
	<head>
	<title> Welcome to the India <title>
	<head>
	<body>

Fig. 2.3.7 Response message header

2.3.1 Persistent and Non-persistent Connection

- HTTP connections are of two types
  1. Persistent HTTP
  2. Non-persistent HTTP

Non-persistent connections

- In this type of connection, one TCP connection is made for each request / response.
- Suppose the page consists of a base HTTP file and ten JPEG images and that all 11 of these objects reside on the same server.
- Suppose the URL for the base HTML file is
  www.vtubooks.com / ITDept / home.index

The sequence of events are as follows :

1. The HTTP client initiates a TCP connection to the server www.vtubook.com on port number 80. It is default port number for HTTP.
2. HTTP client sends an HTTP request message to the server via the socket. Request message includes the path name/ITDept/home.index.
3. HTTP server receives the request message via the socket.

4. HTTP server tells TCP to close the TCP connection.
5. HTTP client receives the response message. The TCP connection terminates.
6. The first four steps are then repeated for each of the referenced JPEG objects.

- As the brower receives the web pages, it displays the page to the user.

Round Trip Time (RTT)

- RTT is the time it takes for a small packet to travel from client to server and then back to the client.
- RTT includes packet propagation delays, packet queuing delays in intermediate routers and switches and packet processing delays.
- Fig. 2.3.8 shows operation when user clicks on a hyperlink.
- Browser to initiate TCP connection between the browser and the web server. It

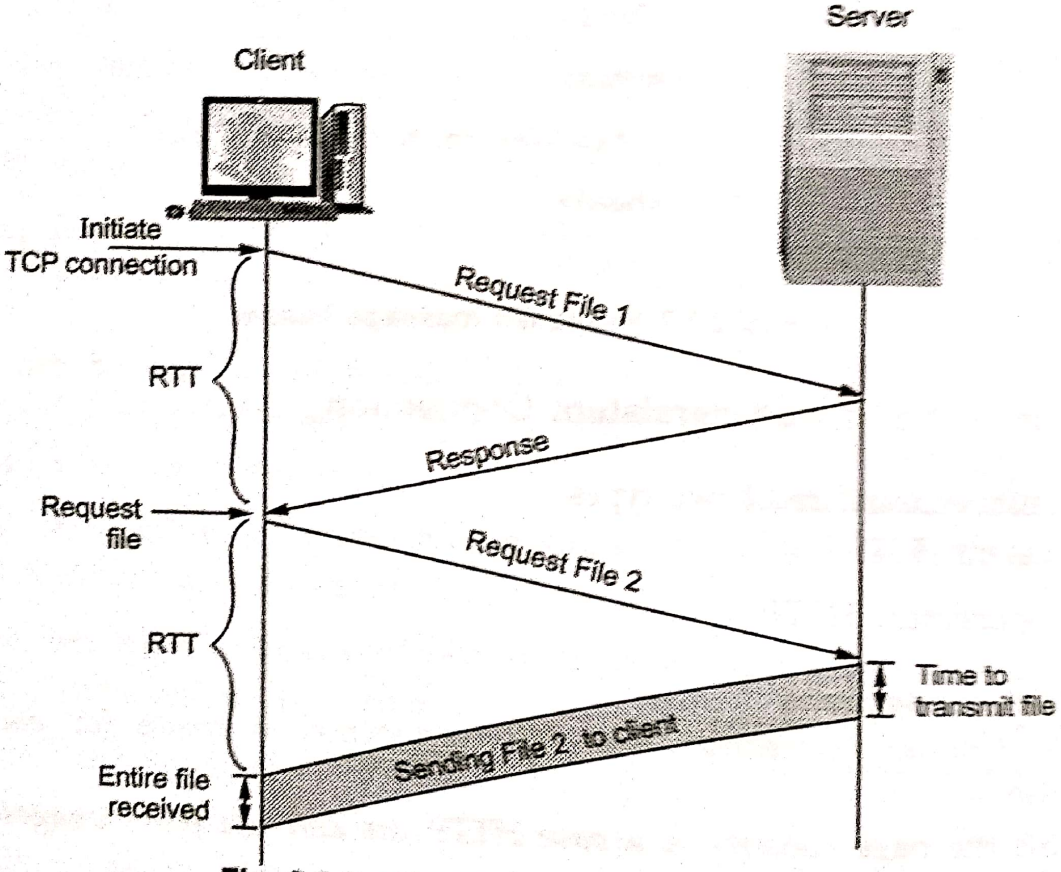


Fig. 2.3.8 Calculation for requesting file

requires three way handshake.

- The client sends a small TCP segment to the server.
- The server acknowledges and responds with a small TCP segment.
- Finally, the client acknowledges back to the server.
- The initial design HTTP 1.0 uses nonpersistent connections. The TCP connection is closed after each request/response interaction.

- Each subsequent request from the same client to the same server involves the setting up and tearing down of an additional TCP connection.

Disadvantages of non-persistent

1. TCP processing and memory resource wasted in the server and the client.
2. It requires delay of 2 RTT associated with the transfer of each object.
3. Each TCP connection setup involves the exchange of three segments between client and server machines.

Persistent connection

- HTTP 1.1. made persistent connections the default mode.
- The server now keeps the TCP connection open for a certain period of time after sending a response.
- This enables the client to make multiple requests over the same TCP connection and hence avoid the inefficiency and delay of the nonpersistent mode.

Types of persistent connections

- There are two versions of persistent connections :
  1. Without pipelining 2. With pipelining

Without pipelining

- The client issues a new request only when the previous response has been received.
- The client experiences one RTT in order to request and receive each of the referenced objects.
- Disadvantage : TCP connection is idle i.e. does nothing while it waits for another request to arrive. This idling wastes server resoruces.

With pipelining

- Default mode of HTTP 1.1. uses persistent connections with pipelining.
- Client issues a request as soon as it encounters a references. The HTTP client can make back to back requests for the referenced objects.
- It can make a new request before receiving a response to a previous request.
- When the server receives the back-to-back requests, it sends the objects back-to-back.
- It uses only one RTT.
- Pipelined TCP connection remains idle for a smaller fraction of time.

- **Persistent HTTP** connections have a number of **advantages**.
  1. By opening and closing fewer TCP connections, CPU time is saved in routers and hosts.
  2. Requests and responses can be pipelined on a connection.
  3. Network congestion is reduced by reducing the number of packets caused by TCP opens.
  4. Latency on subsequent requests is reduced.

Proxy server

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

2.3.2 Difference between Persistent and Non-persistent

Sr. No.	Persistent HTTP	Non-persistent HTTP
1.	Persistent version is 1.1.	Non-persistent HTTP version is 1.0.
2.	It uses one RTT.	It uses two RTT.
3.	TCP connection is not closed.	TCP connection is closed after every request-response.
4.	Client make multiple request over the same TCP connection.	Client make multiple request over the multiple TCP connection.
5.	It is default mode.	It is not default mode.
6.	Request methods are GET, HEAD, POST, PUT, DELETE, TRACE and OPTIONS.	Request methods used are GET, POST and HEAD.

Example 2.3.1 *Consider the following HTTP message and answer the following questions :*

*GBT /cs453/index.html HTTP/1.1<cr><1f>Host : gai a.cs.umass.edu<cr><1f>User-Agent : Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gec ko/20040804 Netscape/7.2 (ax) <cr><1f>Accept:ex t/xml, application/xml, application/xhtml+xml, text /html; q = 0.9, text/plain; q = 0.8, image/png, */*; q = 0.5 <cr><1f>Accept-Language : en-us, en; q=0.5<cr><1f>Accept-Encoding: zip, deflate<cr><1f>Accept-Charset : ISO - 8859-1, utf-8; q=0.7, *; q = 0.7<cr><1f>Keep-Alive: 300<cr> <1f>Connection:keep-alive<cr><1f><cr><1f>*

1) Does browser request a non-persistent or a persistent connection ?

2) Which is the (complete) URL of the document requested by the user ?

3) Which HTML method is used to retrieve the requested URL ?

GTU : Summer-15, Marks 6

Solution :

1) The browser is requesting a persistent connection as indicated by the connection : **Keep-alive.**

2) The document request was http://gaia.cs.umass.edu/cs453/index.html. The Host : field indicates the server's name and /cs453/index.html indicates the file name.

3) GET method is used.

Example 2.3.2 *Consider the following HTTP message and answer the following questions :*

GET/home.asp HTTP/1.1
Host : gtu.ac.in
Accept - Encoding : gzip, deflate, sdch
Accept - Language : en - US, en;q = 0.8
Cookie : OGPC = 5061921 - 11 : 5061952 - 13 : 5061985 - 24 : 5061983 - 27 : 5061968 - 13 : 5062004 - 7 : 5062009 - 6 : 5062022 - 12 :;
SID = DQAAALgBAAA3RAjeUILOOSuH0G91uzL5JOJNUYU2aVOm16jEWVTCo9
- User - Agent : Chrome/49.0.2623.110
X - Client - Data : CIS2yQEIprbJAQjDtskBCP2VygE
Connection : keep - alive|

i) From which browser URL is requested ?

ii) Does browser request a non - persistent or a persistent connection ?

iii) Which is the (complete) URL of the document requested by the user ?

iv) Which HTML method is used to retrieve the requested URL ?

GTU : Summer-16, Marks 7

Solution :

i. Browser URL is Mozilla/5.0

ii. The browser is requesting a persistent connection, as indicated by the Connection : keep-alive.

iii. Complete URL is http://gtu.ac.in/home.asp

iv. GET

Review Questions

1. *Write note on following : 1. HTTP* **GTU : Winter-14, Marks 4**
2. *Describe the built in HTTP request methods.* **GTU : May-12, Marks 7**
3. *What is HTTP ? Differentiate its persistent and non-persistent types with request - response behavior of HTTP.* **GTU : Winter-15, Marks 6**
4. *Explain HTTP GET and HTTP POST method in detail.* **GTU : Summer-16, Marks 4**
5. *What is HTTP ? Explain with respect to persistent and non - persistent connections.* **GTU : Winter-16, Marks 7**

2.4 Domain Name System

GTU : Winter-12,13,14,16,18, Dec.-10,11, June-11, Summer-14,15,16,17

- **Goal :** Assign meaningful high-level names to a large set of machines and handle the mapping of those names to a machine's **IP** address.
- The DNS is a distributed database that resides on multiple machines on the internet and used to convert between names and address and to provide e-mail routing information.
- DNS provides the protocol that allows the client and servers to communicate with each other.
- Domain names are case insensitive so **com** and **COM** mean the same thing.
- The **DNS** protocol runs over **UDP** and uses port 53.
- The **DNS** is specified in **RFC 1034** and **RFC 1035.**
- The **DNS** protocol is the application layer protocol.
- A full domain name is a sequence of labels separated by dots (.).
- The **DNS** name space is hierarchical and it is similar to the unix file system.
- Originally, the internet was small and mapping between names and addresses was accomplished using a centrally-maintained file called *hosts.txt*. To add a name or change an address required contacting the central administrator, updating the table, and distributing it to all the other sites. This solution worked at first because most sites had only a few machines, and the table didn't require frequent changes.

The centrally-maintained table suffered from several drawbacks :

1. The name space was *flat*, and no two machines could use the same machine name.

2. As the internet grew, changes to the database took days to weeks to take effect.
3. The central site became congested with the increase in the number of sites retrieving copies of the current table.
4. The internet grew at an astonishing rate.

The Domain Name System (DNS) is a hierarchical, distributed naming system designed to cope with the problem of explosive growth :

1. It is *hierarchical* because the name space is partitioned into *subdomains*.
2. It is distributed because management of the name space is delegated to local sites. Local sites have complete control (and responsibility) for their part of the name space. DNS queries are handled by servers called *name servers*.
3. It does more than just map machine names to internet addresses. For example, it allows a site to associate multiple machines with a single, mailbox name.

In the DNS, the name space is structured as a tree, with *domain names* referring to nodes in the tree. The tree has a *root*, and a *fully-qualified* domain name is identified by the *components* of the path from the domain name to the root.

Services provided by DNS :

- **Host aliasing :** A host with complicated hostname can have one or more alias names. DNS can be invoked by an application to obtain the canonical hostname for a supplied alias hostname as well as the IP address of the host.
- **Mail server aliasing :** DNS can be invoked by a mail application to obtain the hostname for a supplied alias hostname as well as the IP address of the host.
- **Load distribution :** DNS is also used to perform load distribution among replicated servers.

2.4.1 Components of DNS

DNS includes following components

1. Domain
2. Domain name
3. Name server
4. Name resolver
5. Name cache
6. Zone

1) For example, vtubooks.com is the site for technical publications. Here com is the domain.
2) Domain name is defined by the DNS as being the sequence of names and domain. For example, vtubooks.com could be domain name.
3) In name server, software (program) that maps names to addresses. It does this by mapping domain names to IP addresses.
4) Name resolver is a software that functions as a client interacting with a name server.

5) Name cache is the storage used by the name resolver to store information frequently used.

6) Zone is a contiguous part of a domain.

2.4.2 DNS in the Internet

DNS is divided into three different sections in the internet i.e. Generic domain, Country domain and Inverse domain.

- Fig. 2.4.1 shows the DNS in the internet.

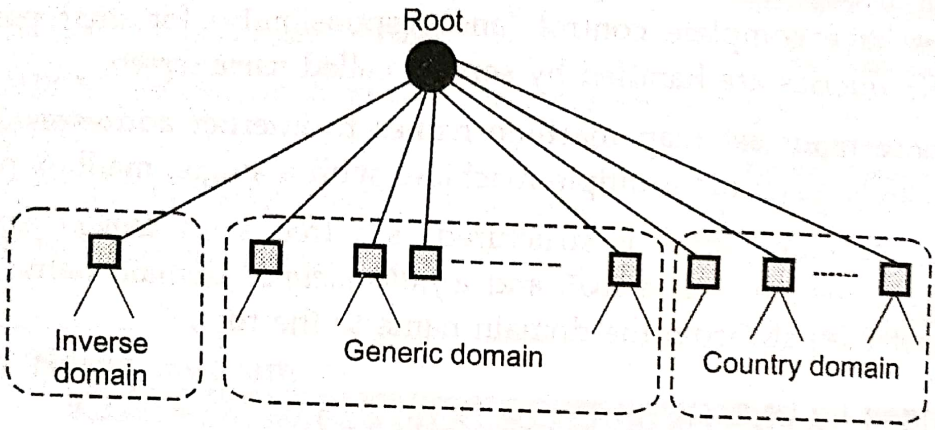


Fig. 2.4.1 DNS in the internet

Generic Domains

- Each node in the tree defines a domain, which is an index to the domain name space database.
- Generic domain labels are as follows

Sr. No.	Label	Description
1.	com	Commercial organization
2.	edu	Educational organization
3.	gov	Government Institutions
4.	int	International organizations
5.	mil	Military group
6.	net	Network support centers
7.	org	Nonprofit organization

- Fig. 2.4.2 shows the generic domains

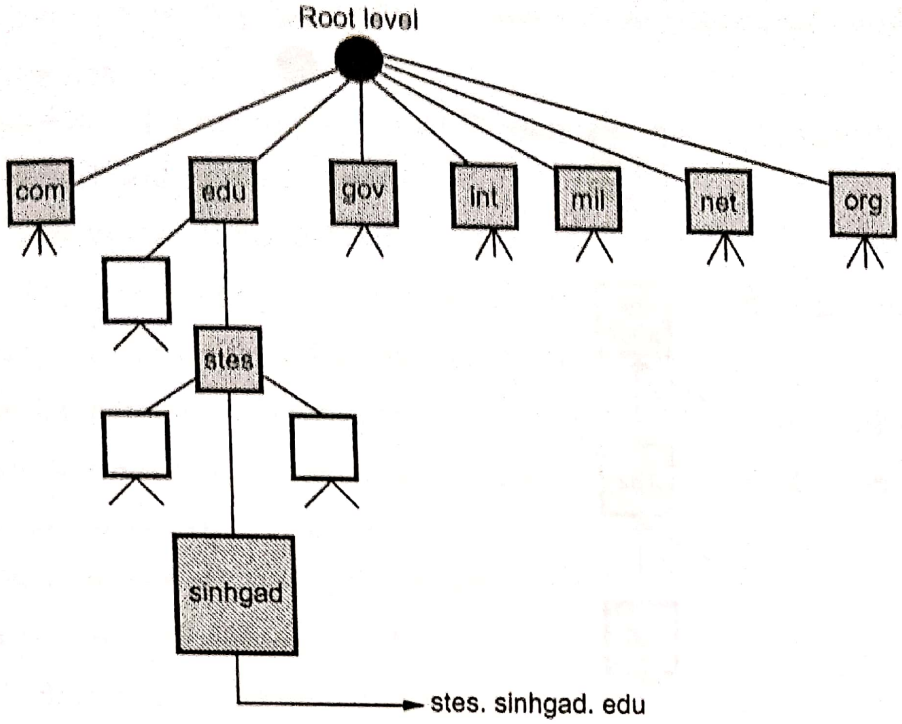


Fig. 2.4.2 Generic domains

Country Domains

- It uses two character country abbreviations at first level. Second level labels can be more specific, national destinations. For India, the country domain is in.
- Fig. 2.4.3 shows country domains.

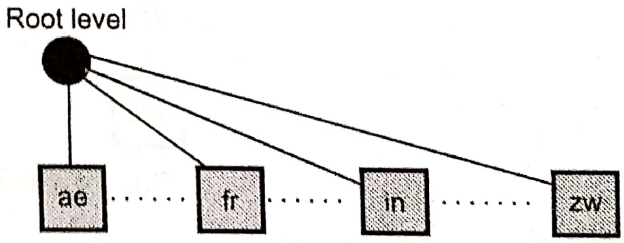


Fig. 2.4.3 Country domains

Inverse Domain

- Used to map an address to a name.
- Example : When a client send a request to the server for doing a particular task, server finds the list of authorized client. The list contains only IP address of the client.
- Server send a query to the inverse DNS server and ask for a mapping of address to name for authorized client list.
- The above query is called an inverse or pointer query.
- The pointer query is handled by the first level node called arpa. The second level is also one single node named in-addr. The rest of the domain defines IP addresses.

Fig. 2.4.4 shows inverse domain.

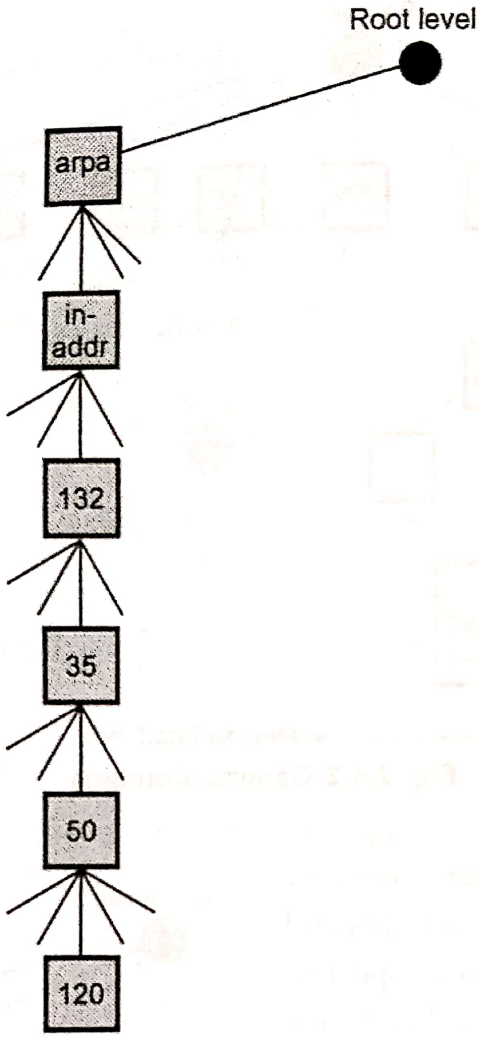


Fig. 2.4.4 Inverse domain

2.4.3 Name Spaces

- Name spaces are of two types : Flat name spaces and Hierarchical names.
- The name assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses.

i) Flat name spaces :

- The original set of machines on the Internet used flat namespaces.
- These namespaces consisted of sequence of characters with no further structure.
- A name is assigned to an address.
- **Advantage :**
  1. Names were convenient and short.
- **Disadvantages :**
  1. Flat name spaces cannot generalize to large sets of machines because of the single set of identifiers.

2. Single central name authority was overloaded.
3. Frequent name-address binding changes were costly and cumbersome.

II) Hierarchical names

- The partitioning of a namespace must be defined in such a way that it :
  - Supports efficient name mapping.
  - Guarantees autonomous control of name assignment.
- Hierarchical namespaces provides a simple yet flexible naming structure.
- The namespace is partitioned at the top level.
- Authority for names in each partition are passed to each designated agent.
- The names are designed in an inverted-tree structure with the root at the top.
- The tree can have only 128 levels.

The top level domains are divided into three areas :

1. Arpa is a special domain used for the address-to-name mappings.
2. The 3 character domains are called the generic domains.
3. The 2 character domains are based on the counter codes found in ISO 3166. These are called the country domains.

- Fig. 2.4.5 shows the hierarchy of DNS.

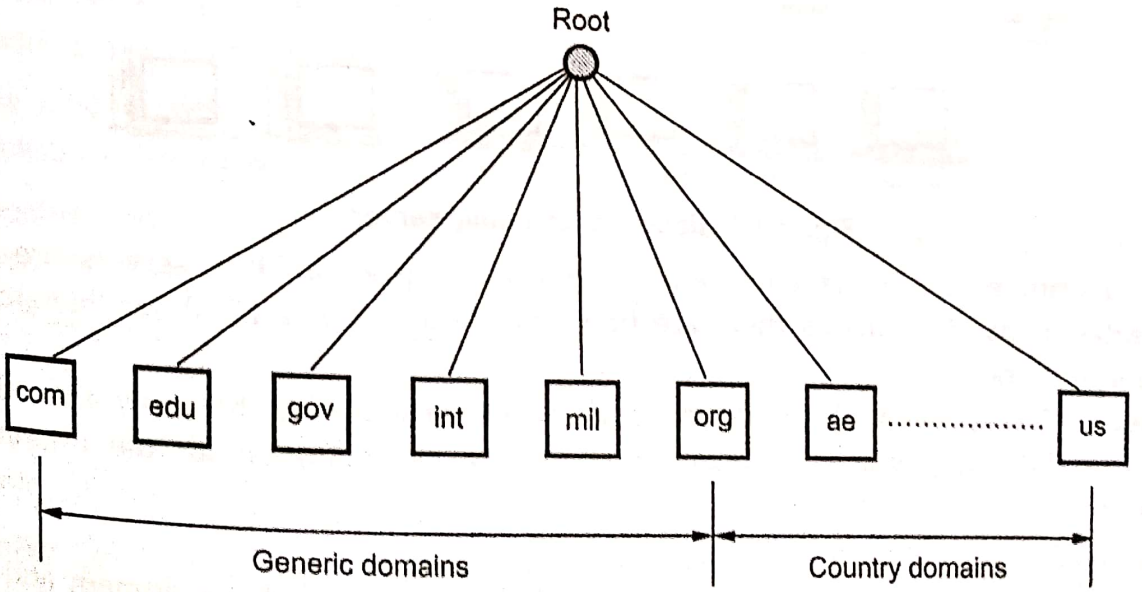


Fig. 2.4.5 Hierarchy of DNS

2.4.4 Domain Name Space

- In DNS, names are defined in an inverted tree structure with the root at the top. The tree can have only 128 levels : Level 0 to Level 127.
- 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 , i.e. empty string.

- Each node in the tree has a domain name, a full domain name is a sequence of labels separated by dots(.). Fig. 2.4.6 shows the domain names and labels.

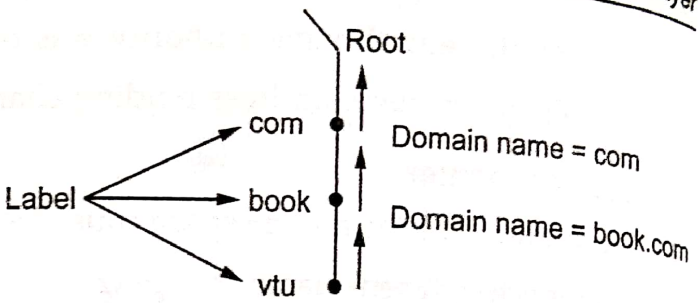


Fig. 2.4.6 Domain names and label

- In fully qualified domain name, label is terminated by a null string. Fully Qualified Domain Name (FQDN) contains the full name of host. All labels are part of FQDN.
- Partially Qulified Domain Name (PQDN) : In this label is not terminated by a null string. It always start from node. A domain name does not include all the levels between the host and the root node. For example, vtu.book.com.

Hierarchy of Name Servers

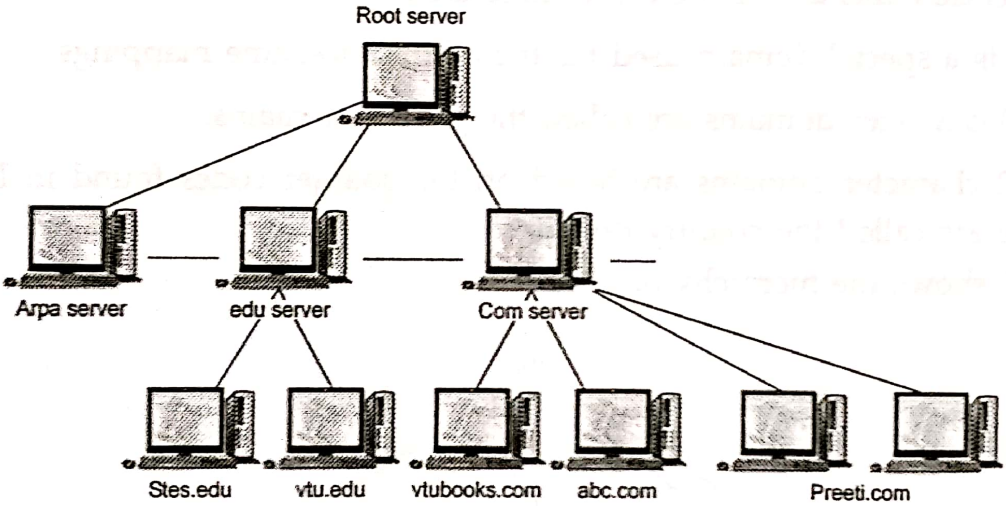


Fig. 2.4.7 Hierarchy of name server

- To distribute the information among many computers, DNS servers are used. Creates many domains as there are first level nodes. Fig. 2.4.7 shows hierarchy of name servers.
- **Zone :** Server have some authority and also responsible for operation. Server creates database, which is called zone file. Server maintain all the information about node of that domain.
- Fig. 2.4.8 shows domain with zone.
- Domain and zone are same if server accepts responsibillity for a domain and does not divide the domain into **subdomian.**
- Domain and zone are different, if a server divides its domain into subdomains and delegates part of its authority to other server.
- **Root server :** If zone consists of the full tree then that zone server is called root server. Root server do not maintain any information about domains.
- DNS uses two types of servers :
  1. Primary server 2. Secondary server

- **Primary server :** This server keeps a file about the zone for which it is responsible and have authority. It performs operation on zone file like create, update and maintaining.
- **Secondary server :** It loads all information from the primary server. Secondary server can not perform any operation on zone file.

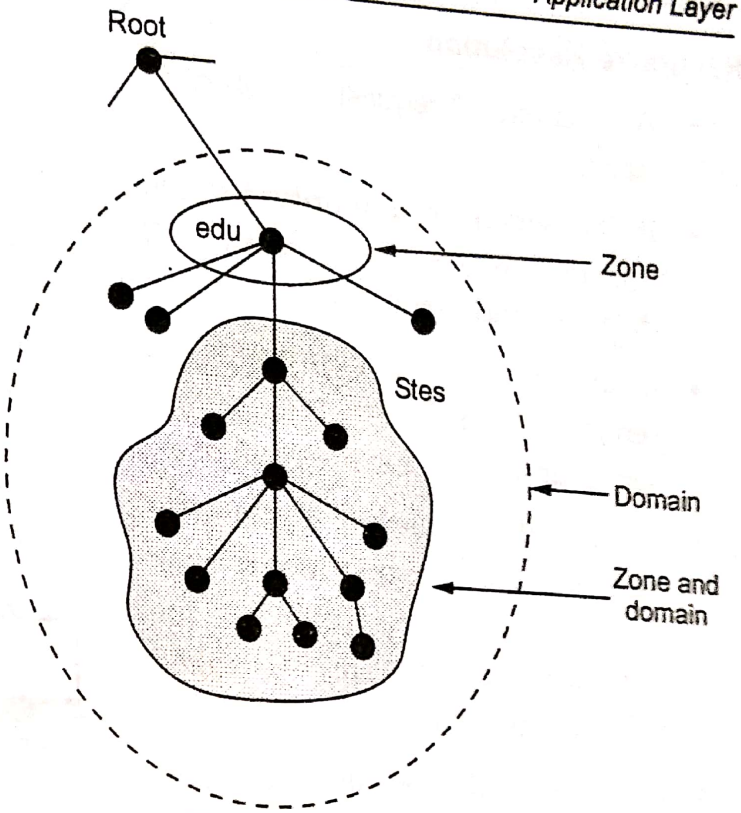


Fig. 2.4.8 Domain and zone

2.4.5 Resolution

- DNS is designed as a client server application. A host that needs to map an address to a name or a name to an address calls a DNS client named a **resolver**.

Working :

- Name resolving must also include the type of answer desired (specifying the protocol family is optional).
- The DNS partitions the entire set of names by class (for mapping to multiple protocol suites).
- Naming items is required since one cannot distinguish the names of subdomains from the names of individual objects or their types.

Mapping Domain Names to Addresses :

a) The DNS also includes an efficient, reliable, general purpose, distributed system for mapping names to addresses using an independent co-operative system called name servers.

b) Names Servers - are server programs that translate names-to-addresses (maps DN => IP addresses) and usually executes on a dedicated processor.

c) Name Resolvers - client software that uses one or more name servers in getting a mapped name.

d) Domain name servers are arranged in a conceptual tree structure that corresponds to the naming hierarchy.

Recursive Resolution

- A client request complete translation.
- If the server is authority for the domain name, it checks its database and responds.
- If the server is not authority, it sends the request to another server and waits for the response.
- When the query is finally resolved, the response travel back until it finally reaches the requesting client. This is called recursive resolution.
- Fig. 2.4.9 shows the recursive resolution.

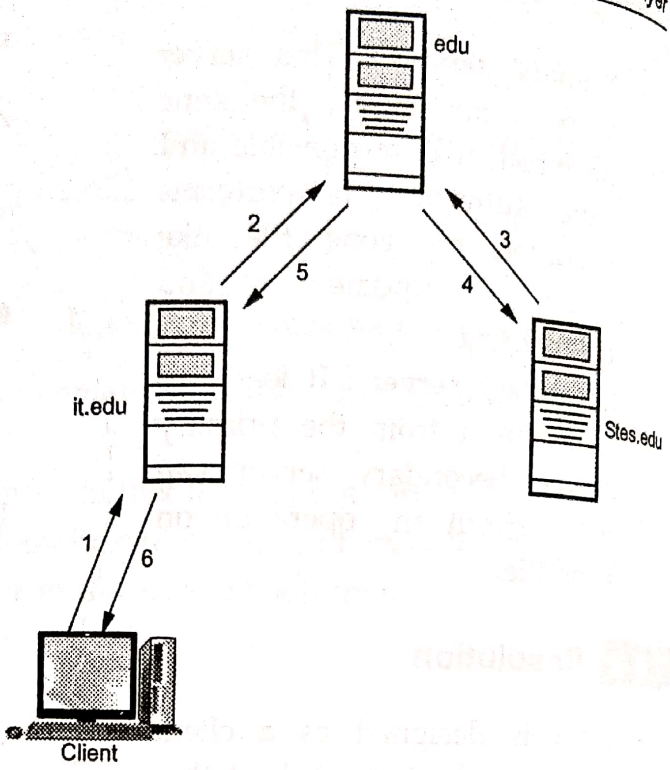


Fig. 2.4.9 Recursive resolution

Iterative Resolution

- Only a single resolution is made and returned (not recursive).
- Client must now explicitly contact different name servers if further resolution is needed.
- If the server is an authority for the name, it sends the answer. If it is not, it returns 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. This process is called iterative resolution because the client repeats the same query to multiple servers.
- Fig. 2.4.10 shows iterative resolution.

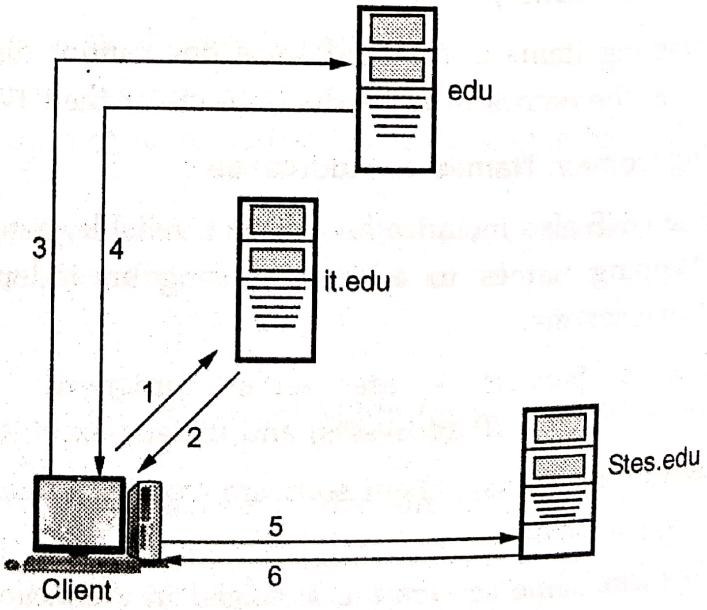


Fig. 2.4.10 Iterative resolution

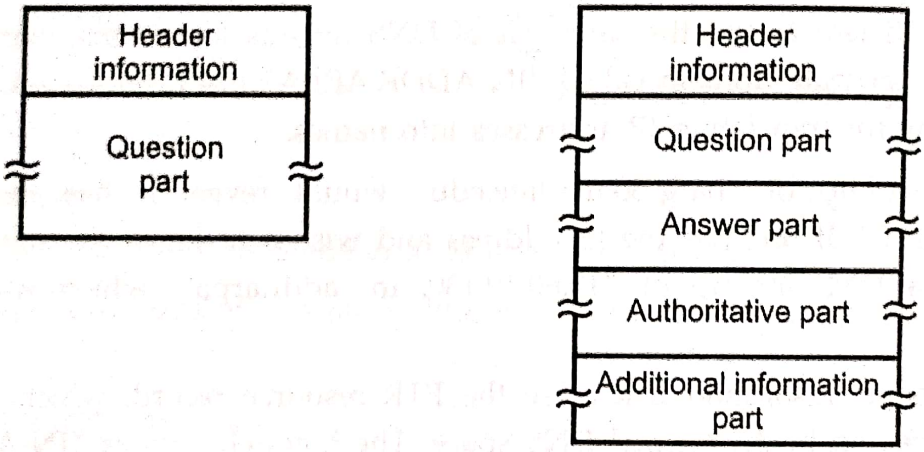
- Conceptually, name resolution proceeds in a top-down fashion.
- Name resolution can occur in one of two different ways : Recursive resolution and Iterative resolution
- Name servers use name caching to optimize search costs.
- Time To Live (TTL) is used to determine a guaranteed name binding during it's time interval. When time expires, the cache name binding is no longer valid, so the client must make a direct name resolution request once again.

Reverse Name Resolution :

- Reverse name resolution is important task of DNS on the internet or the translation of IP addresses back to domain names. For example, servers can determine and record the full domain name of machine connecting to them over the network.
- It is not efficient to use the same set of DNS records for reverse name resolution. Instead, a separate domain called "IN-ADDR.ARPA" has been set aside to provide a hierarchy for translating IP addresses into names.
- A DNS lookup of "barg.oo.msstate.edu" would reveal it has the IP address "130.19.60.10". If one has the IP address and wishes to know the name, one must perform a DNS lookup of "10.60.19.130. in -addr.arpa", which will return the name.
- Reverse name resolution fields use the PTR resource record, which points to the correct position in the normal DNS space. The hierarchy under "IN-ADDR.ARPA" can be delegated of course just like any other domain.
- To obtain the IP address of a named server, each host has a client protocol known as the name resolver. On receipt of the name, the client application protocol posses it to the name resolver using the standard interprocess communication primitive supported by the local operating system.
- The resolver then creates a resolution request message in the standard message format of the domain name server protocol.
- A resolver can have multiple request outstanding at any time. Hence the identification field is used to relate a subsequent response message to an earlier request message.
- The name resolver passes the request message to its local domain name server using TCP/IP. If the request is for a server on this network, the local domain name server obtains the corresponding IP address from its DIB and returns it in a reply message.

2.4.6 Message Format

- Messages are sent between domain clients and domain servers with a specific format.
- All messages of this format are used for name resolution and naming queries.
- Question sent by the client and answers provided by the server are included within different fields of the same message.
- DNS has two types of messages : Query and Response. Both types have the same format.
- The query message consists of the header and the question records, the response message consists of a header, question record, answer record, authoritative record and additional records.
- Fig. 2.4.11 shows the query and response messages.



(a) Query (b) Response

Fig. 2.4.11 Query and response message

- Fig. 2.4.12 shows the header format of the DNS.

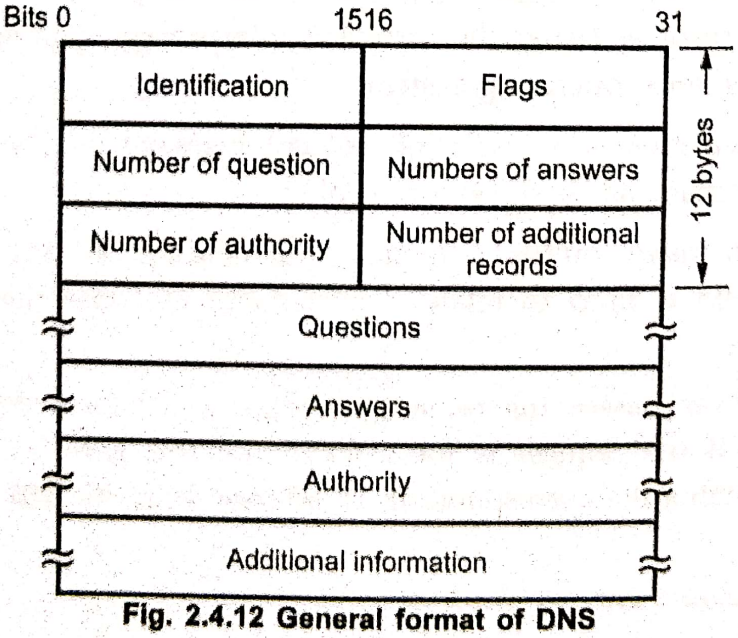


Fig. 2.4.12 General format of DNS

- **Identification :** It is 16 bits fields and unique value used by the client to match responses to queries.
- **Flags :** It is the collection of subfields that define the type of messages and type of the answers requested and so on.
- Number of question record contains the number of queries in the question section of the message.
- Number of answer record contains the number of answer records in the answer section of the response message.
- Number of authority record contains the number of authority records in the authoritative section of the response message.
- Number of additional records contains the number of additional records in the additional section of the response message. The message has a fixed 12-byte header followed by 4 variable length fields. The identification field is set by client and returned by the server. It lets the client, match responses to requests.
- Fig. 2.4.13 flag fields in DNS header.

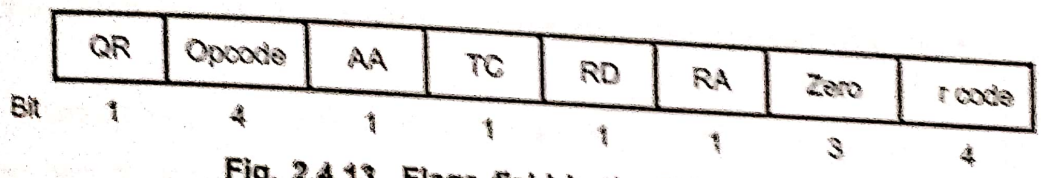


Fig. 2.4.13 Flags field in the DNS header

- The flags field is divided into 8 parts.

QR = 0 For message is a query

= 1 It is response

Opcode = 0 Standard query

= 1 Inverse query

= 2 Server status request

AA = Authoritative answer

TC = Truncated

RD = Recursive query

RA = Recursion available

r code = Return code

- RD field is 1-bit and can be set in a query and is then returned in the response. This flag tells the name server to handle the query itself, called a recursive query.

- RA is a 1-bit field and set to 1 in the response if the server support recursion. There is a 3-bit field that must be zero.
- r code is a 4-bit field. The common value are 0 for no error and 3 for name error. A name error is returned only from an authoritative name server and means the domain name specified in the query does not exist.
- The next four 16-bit fields specify the number of entries in the four variable length fields that complete the record.

2.4.7 Resource Records

- Different types of resource records are used in DNS. An IP address has a type of A and PTR means pointer query.
- There are about 20 different types of resource records available. Some PR are listed below.

1) A = It defines an IP address. It is stored as a 32-bit binary value.

2) CNAME = "Canonical name". It is represented as a domain name.

3) HINFO = Host information, two arbitrary character strings specifying the CPU and operating system (OS).

4) MX = Mail exchange records. It provide domain willing to accept e-mail.

5) PTR = Pointer record used for pointer queries. The IP address is represented as a domain name in the in-addr.arpa domain.

6) NS = Name Server record. These specify the authoritative name server for a domain. They are represented as domain names.

A) Configuration of DNS :

The DNS server can be configured manually by editing files in the default WINNT installation path \% SYSTEM ROOT %\ SYSTEM 32 \ DNS. Administration is identical to administration in traditional DNS. These files can be modified using a text editor. The DNS service must then be stopped and restarted.

2.4.8 Name Servers

- When a resolver has a query about a domain name, it passes the query to one of the local name servers. If the domain is remote and no information about the requested domain is available locally, the name server sends a query message to the top level name server for the domain requested.
- Fig. 2.4.14 shows the eight steps for resolving the remote name.
- A resolver on flits.cs.vu.nl wants to know the IP address of the host linda.cs.yale.edu.

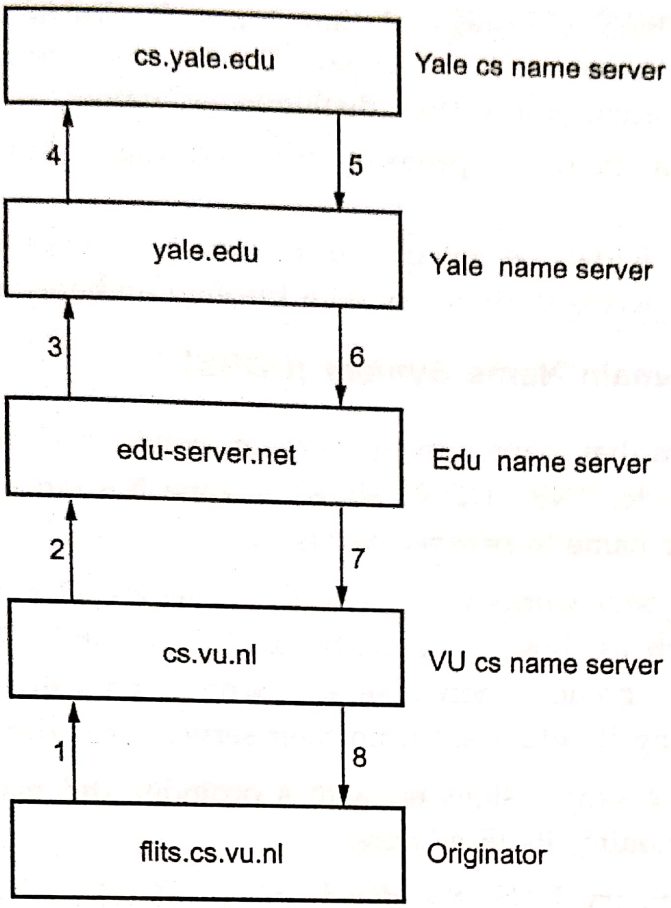


Fig. 2.4.14 Remote name resolve

Steps

1. It sends a query to the local name server cs.vu.nl. This query contains the domain name, sought, the type (A) and the class (IN).
2. and 3. Suppose the local name server has never had a query for this domain before and knows nothing about it. It may ask a few other nearby name servers, but if none of them know, it sends a UDP packet to the server for edu given in its database, edu-server.net. This server knows all its children, so it forwards the request to the name server for yale.edu.
4. This forwards the request to cs.yale.edu, which must have the authoritative resource records.
5. to 8. Each request is from a client to a server, the resource record requested works its way back in these steps.

2.4.9 LDAP

- LDAP is Lightweight Directory Access Protocol. It provides X-500 features. LDAP is an application-level protocol that is implemented directly on top of TCP.
- It stores entries, which is similar to objects. Each entry must have a distinguished name, which un-equally identifies the entry. Entries can also have attributes.

- LDAP provides binary, string and time types. It allows the defination of object classes with attribute name of types. Entries are organized into a directory information tree, according to their distinguished names.
- LDAP defines a network protocol for carrying out data defination and manipulation.
- LDAP has been widely adopted, particularly for internet directory services. It provides secured access to directory data through authentication.

2.4.10 Dynamic Domain Name System (DDNS)

- DDNS is a service that maps internet domain names to IP addresses. DDNS serves a similar purpose to DNS : DDNS allows anyone hosting a Web or FTP server to advertise a public name to prospective users.
- Unlike DNS that only works with static IP addresses, DDNS works with dynamic IP addresses, such as those assigned by an ISP or other DHCP server. DDNS is popular with home networkers, who typically receive dynamic, frequently-changing IP addresses from their service provider.
- To use DDNS, one simply signs up with a provider and installs network software on their host to monitor its IP address.
- Compared to ordinary DNS, the disadvantage of DDNS is that additional host software, a new potential failure point on the network, must be maintained.

Review Questions

1. *What is the role of Domain Name Server (DNS) in Internet ? Explain the hierarchy of various domain names.* **GTU : Winter-14,18, Marks 7**
2. *How does DNS work ? Explain.* **GTU : Dec.-10, Marks 7**
3. *Explain DNS in detail with example.* **GTU : June-11, Marks 7**
4. *Explain : DNS and its advantages.* **GTU : Dec.-11, Marks 5**
5. *Explain the domain name system.* **GTU : Winter-12, Marks 7**
6. *What is a resource record ? How it is useful for DNS ?* **GTU : Winter-13, Marks 7**
7. *Explain : DNS and its use.* **GTU : Summer-14, Marks 4**
8. *Why distributed database design is more preferred over centralized design to implement DNS in the Internet ? Justify. Also explain the way of DNS servers to handle the recursive DNS query using suitable diagram.* **GTU : Summer-15, Marks 8**
9. *What is the purpose of Domain Naming System (DNS) ?* **GTU : Summer-16, Mark 1**
10. *Discuss the DNS services in detail.* **GTU : Winter-16, Marks 7**
11. *Write short note on DNS.* **GTU : Summer-17, Marks 7**

2.5 World Wide Web

GTU : Dec.-11, Winter-12,13,15,16,18

- World wide web is collection of millions of files stored on thousands of servers all over the world. These files represent documents, pictures, video, sounds, programs, interactive environments.
- Following are hardware, software and protocols that make up the web.
  1. A web server is a computer connected to the Internet that runs a program that takes responsibility for storing, retrieving and distributing some of the web files. A web client (web browser) is a computer that requests files from the web.
  2. Well-defined set of languages and protocols that are independent of the hardware or operating system are required to run on the computers.
  3. The Hyper Text Markup Language (HTML) is the universal language of the web.
  4. Java is a language for sending small applications over the web. Java script is a language for extending HTML to embed small programs called scripts in web pages. The main purpose of Java and scripts is to speed up the interactivity of web pages.
  5. VB script and Activex controls are microsoft system that work with IE.
  6. Pictures, drawings, charts and diagrams are displayed on web using image formats such as JPEG and GIF formats.
  7. The Virtual Reality Modeling Language (VRML) is the web's way of describing three-dimensional objects.
- A web page is an HTML document that is stored on a web server. A web site is a collection of web pages belonging to a particular organization.
- URL of these pages share a common prefix, which is the address of the home page of the size. Search engines are a bottom-up approach for finding your way around the web. Some search engines search only the titles of web pages. While other search every word. Keywords can be combined with Boolean operations, such as AND, OR and NOT, to produce rather complicated queries.
- Home page is the front door of a web site. When a person or organization says "My web site is at www.sangeeta.com", the URL to which they refer is the URL of the site's home page. The home page introduces the rest of the web site and provides links that leads to other pages on the site.

2.5.1 Web Browsers

- A web browser is a program. Web browser is used to communicate with web servers on the Internet, which enables it to download and display the webpages. Netscape Navigator and Microsoft Internet Explorer are the most popular browser softwares available in market. Browser interact with web as well as computer operating system and with other programs.

- Internet explorer is the default browser in newly installed window 98 systems. Most browser windows have the same basic layout. Some of the basic elements are -
  - Menu bar
  - Tool bar
  - Address or location window
  - Viewing window
  - Status bar

Some web pages are divided into independent pages, called frames.

- The purpose of the web browser is to display web pages, which may either arrive over the Internet. Web browser can be used to view files of any common web format that are stored on the user system. Window, Macintosh and some Unix desktops support the default web browser.
- There are different ways for opening the web page.
  1. Enter its URL into the address or location box of a web browser.
  2. Select it from the list that drops down from the address.
  3. Link to it from another web page.
  4. Link to it from a mail message or newsgroup article.

2.5.2 Working of WWW

- www uses client-server interaction. The browser program acts as a client that uses the Internet to contact a remote server for a copy of the requested page. The server on the remote system returns a copy of the page along with the additional information.
- The additional information a www server returns tells the browser two important things.

  1) It describes how to display the information.

  2) It gives a URL for each selectable item on the page.
- When a browser receives a page from a remote server, it displays the page and then waits for the user to select one of the highlighted items. Once a user makes a selection, the browser consults the hidden information that arrived with the page to find the URL that corresponds to the selection. The browser then uses the Internet to obtain the newly selected page of information.
- Each URL uniquely identifies a page of information by giving the name of a remote computer, a server on that computer and a specific page of information available from the server. Fig. 2.5.1 illustrates how the URL encodes the information.

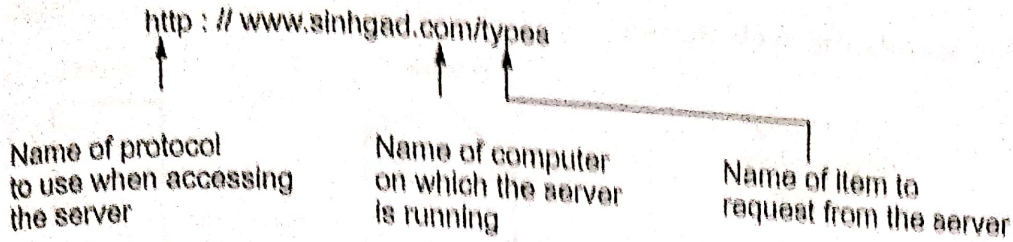


Fig. 2.5.1 Format of URL

- World Wide Web was developed using the client server architecture, which ensured cross-platform portability. The www is officially described as a "Wide area hypermedia information retrieval initiative". It is an information system that links data from many different Internet services under one set of protocols.
- Web clients, called browsers, interpret Hyper Text Markup Language documents delivered from web servers.
- The world wide web is a distributed, multimedia, hyper text system. It is distributed since information on the web can be located on any computer system connected to the Internet around the world.
- It is multimedia because the information it holds can be in the form of text, graphics, sound and video.
- Hyper text means that the information is available using hyper text technique, which involves selecting highlighted phrases or images that one selected retrieve information related to the selected highlighted subject.
- The information being retrieved can be information located any where in the world. The normal way to provide information on the world wide web is by writing documents in HTML (Hyper Text Markup Language).

1. The Client Side

- When a user clicks on a hyperlink, the brower carries out a series of steps in order to fetch the page pointed to.
  1. The brower determines the URL.
  2. The brower asks DNS for the IP address of www.vtubooks.com.
  3. DNS replies with 172.16.16.1.
  4. The brower makes a TCP connection to port 80 on 172.16.16.1.
  5. It then sends over a request asking for file/home/index.html.
  6. The www.vtubooks.com server sends the file/home/index.html.
  7. TCP connection is released.
  8. The brower displays all the text in home/index.html.
  9. The brower fetches and displays all images in this file.

- Fig. 2.5.2 shows the web model.

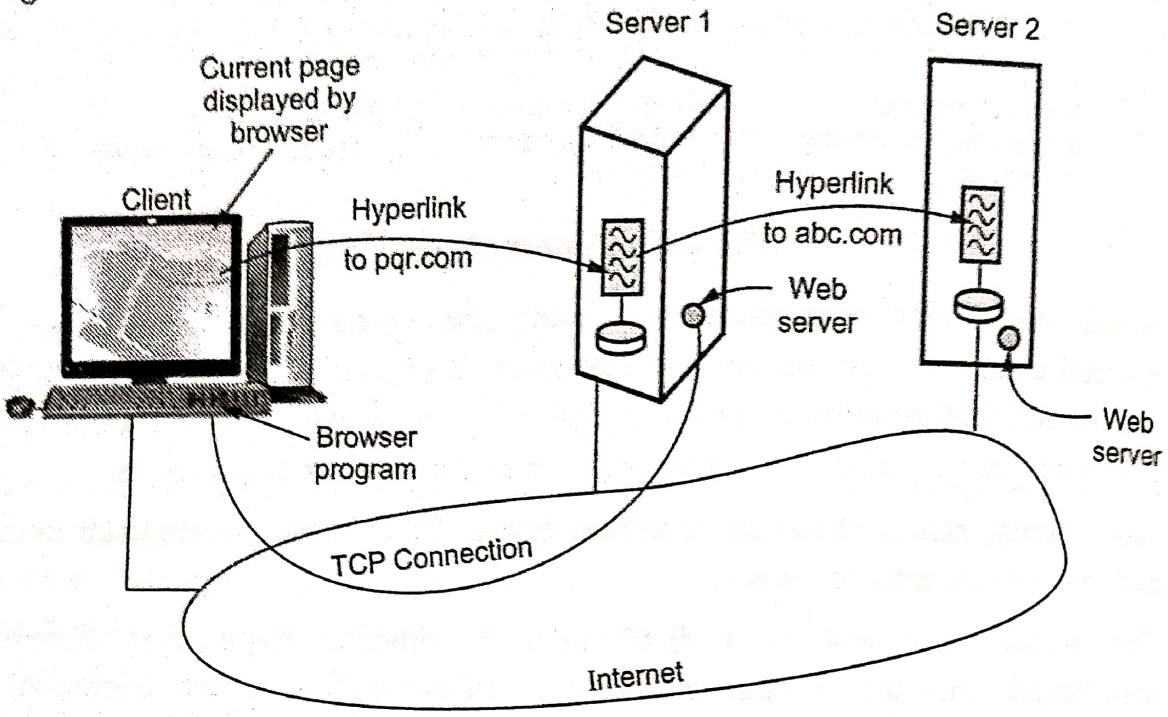


Fig. 2.5.2 Web model

2. The Server Side

- The steps that the server performs.
  1. Accept a TCP connection from a client browser.
  2. Get the name of the file required.
  3. Get the file.
  4. Return the file to the client.
  5. Release the TCP connection.

2.5.3 Statelessness and Cookies

- The web is basically stateless. There is no concept of a login session. The browser sends a request to a server and gets back a file. Then the server forgets that it has ever seen that particular client.
- When a client requests a web page, the server can supply additional information along with the requested page. This information may include a cookie, which is a small file. Browsers store offered cookies in a cookies directory on the client's hard disk unless the user has disabled cookies.
- Cookies are just files or strings, not executable programs. In principle, a cookie could contain a virus, but since cookies are treated as data there is no official way for the virus to actually run and do damage.
- A cookie may contain upto five fields.
  a) Domain

b) Path

c) Content

d) Expires

e) Secure

a) **Domain :** It tells where the cookies came from. Browsers are supposed to check that servers are not lying about their domain. Each domain may store no more than 20 cookies per client.

b) **Path :** The path is a path in the server's directory structure that identifies which parts of the server's file tree may use the cookie. It is often 1, which means the whole tree.

c) **Content :** It takes the form name = value. Both name and value can be anything the server wants. This field is where the cookies content is stored.

d) **Expires :** The expires field specifies when the cookies expires. If this field is absent, the browser discards the cookies when it exits. Such a cookies is called a **nonpersistent cookie.** If a time and date are supplied, the cookie is said to be **persistent** and is kept until it expires.

e) **Secure :** This field can be set to indicate that the browser may only return the cookie to a secure server. This feature is used for e-commerce, banking and other secure applications.

Examples of cookies

Domain	Path	Content	Expires	Secure
toms-casino.com	/	customer ID=497793521	15-10-02, 17:00	Yes
joes-store.com	/	cart=1-00501; 1-07031; 2-13721	11-10-02, 15:20	No
sneaky.com	/	user ID=3456789	30-12-06, 11:00	Yes

2.5.4 Static Web Documents

- Hyper Text Markup Language (HTML) is intended as a common medium for typing together information from widely different sources. HTML documents are the Standard Generalized Markup Language (SGML) documents with generic semantic that are appropriate for representing information from a wide range of applications.

- HTML documents are in plain text format that contain embedded HTML tags. Documents can be created in any text editor. There are also many other tools, including editors, designed specifically to assist in creating HTML documents. To view an HTML document, the user needs a browser.
- HTML defines the structural elements in a document such as headers, and addresses, layout information and the use of inline graphics together with the ability to provide hyper text links. Web pages were written in HTML level 0.
- The most basic element in the HTML document is the paragraph. The web browser flows all the contents of the paragraph together from left to right and from top to bottom gives the current window.
- A document will be ready by both graphical and character based web browser. The three basic tagging pairs used to create the highest level of structure in an HTML documents are as follows :

```
<HTML> HTML documents </HTML>
<HEAD> Header information of document </ HEAD>
<BODY> Body of the HTML document </ BODY>
```

The general structure of the HTML is

```
<HTML>
  <HEAD>
   <TITLE>
        Title here
      </ TITLE>
      </ HEAD>
      <BODY>
           Body element and content
      </ BODY>
           </ HTML>
```

A simple HTML document is given below.

```
<HTML>
<HEAD>
<TITLE> Communication Network </ TITLE>
</ HEAD>
<BODY>
<H/> Information about the communication network </H/>
<P> Information about the communication network is available
<A HREF :"http ://www.technicalpublicationspune.com"></A></P>
</BODY>
</HTML>
```

- Structural elements in the document are identified by Start and End tags. For example the <TITLE> and </TITLE> tags are used to specify the title of the document.

- The <H/> and </H/> tags are used to define the first level heading. Headings are generated by an <Hn> tags, where n is a digit in the range 1 to 6. <H/> is the most important heading and <H6> is the less important. Typically the lower numbered heading will be displayed in a larger and heavier font.
- The browser may also choose to use different colors for each level of heading. Typically <H1> headings are large and bold face with at least one blank line above and below.
- In contrast <H2> headings are in a smaller font, and with less space above and below. The
, <P> and <HR> tags all indicate a boundary between sections of text.
- The precise format can be determined by the style sheet associated with the page. The
 tag just forces a line break. <P> starts a paragraph, which might for example, insert a blank line and possibly some indentation. <HR> (horizontal-rule) tag forces the browser to generate a horizontal rule or line, across the display. It breaks pages into logical sections and is useful when creating forms. There is no equivalent vertical rule.

Advantages and Disadvantages of HTML

A) Advantages of HTML :

1. Applications are quickly developed, requiring substantially less time than is required when creating programs with languages such as C and Pascal.
2. Web applications are easy to maintain and update without disrupting the network data traffic.
3. Developing applications in HTML takes advantage of HTML general compatibility. Web applications can access other company data servers, such as FTP and WAIs databases.
4. Collecting information with HTML.
5. Web viewer request a document from a web site, its server sends the data and the connection between the two computers is dropped. This is the client server relationship. This relationship reduces the amount of time a server spends serving a client freeing it to serve other users.

B) Disadvantages :

1. **Locking :** HTML is not a compiled data format. Web pages cannot be locked. Users have free and open access to look at HTML sources.
2. **Security :** Information is easily accessible and travels unimpeded between hosts and desktops. The cost of this freedom is lack of inherent security.

2.5.4.1 XML and XSL

- HTML does not provide any structure to web pages. HTML mixes the content with the formatting with web pages in HTML, it is very difficult for a program to search particular word.
- To overcome this problem, two new language Extensible Markup Language (XML) and Extensible Style Language (XSL) are used. The XML and XSL specifications are much stricter than HTML specification.
- Web pages in XML and XSL are still static since they simply contain instructions to the browser about how to display the page, just as HTML pages do. XML allows the web site designer to make up definition files in which the structures are defined in advance. Defination files can be included, making it possible to use them to build complex web pages.

2.5.4.2 XHTML

- XHTML is new web standard and should be used for all new web pages to achieve maximum portability across platforms and browsers.
- Difference between XHTML and HTML are as follows :
  1. XHTML pages and browsers must strictly conform to the standard.
  2. All tags and attributes must be in lower case.
  3. Closing tags are required even for </p>.
  4. Attributes must be contained within quotation marks.
  5. Tags must nest properly.
  6. Every document must specify its type.

2.5.5 Dynamic Web Documents

Server side dynamic web page generation. Fig. 2.5.3 shows processing of information in HTML.

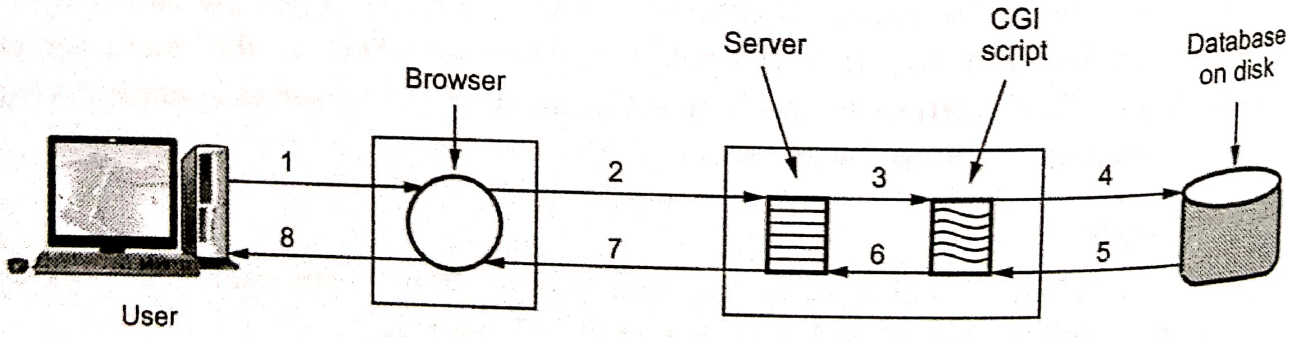


Fig. 2.5.3 Steps in processing information

1. User fills in form.
2. Form sent back.

3. Handed to CGI.
4. CGI queries database.
5. Record found.
6. CGI builds pages.
7. Page returned.
8. Page displayed.

2.5.5.1 Common Gateway Interface

- CGI makes dynamic computation of web pages possible. It allows a web server to associate some URLs with computer program instead of static documents on disk.
- When a browser request one of the special URLs the server runs the associated computer program and sends the output from the program back to the user. A server can have an arbitrary number of CGI programs that perform different computations.

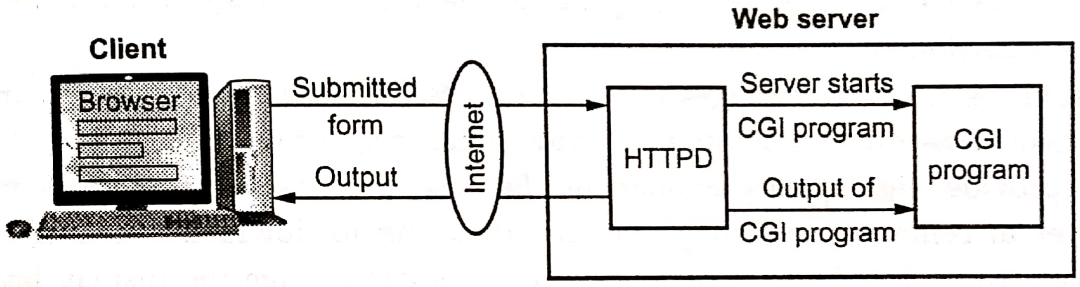


Fig. 2.5.4 CGI working

- The server uses the URL in the incoming request to determine which CGI program to run. CGI working is as follows : CGI program is part of a web server.
- The browser sends a request to the server. If the requested URL corresponds to a CGI program, the server starts the appropriate program and passes to the program a copy of the request.
- The server then sends the output from the CGI program back to the browser in the form of reply.
- From a browser's point of view, there is no difference between a URL that corresponds to a static document and one that corresponds to a CGI program. Requests for both static documents and CGI output have the same syntatic form.

2.5.5.2 Java Technology

- Sun Microsystems, Incorporated has developed a popular active document technology called Java, the technology can be used to create animated web pages, pages that interact with the user, or pages that use the screen in unexpected ways.

- Java calls an active web page an *apple* ; the terminology is so widespread that most other vendors have either adopted it or chosen to use a minor variation.
- Java became popular for four reasons.
  1) The designers chose to make the Java language similar to a widely-used programming language, meaning that professional programmers could learn to write Java applets easily.
  2) No other active document technology was available.
  3) Because the Java system includes software to handle common tasks such as controlling the screen, a programmer can use predefined pieces to create a Java applet quickly.
  4) Java is so powerful that it provides more functionality than most other technologies. For example, Java can handle direct user interaction better than forms, can fetch a sequence of pages better than client-pull, can control multiple areas of the screen better than frames, and can manipulate a variety of data formats better than plugins. Thus, Java can substitute these by other technologies.
- Despite its many advantages over existing technologies, the strongest motivation for Java came from its ability to provide functionality that other technologies could not provide high quality animations. Because they use a computer's processing power to compute new images instead of trying to download them from a Web server, active document technologies like Java can change the display fast enough to present the illusion of smooth motion. Because none of the older Web technologies can provide the same functionality, many Web sites have been eager to use Java.

2.5.6 Browser Architecture

- Each browser usually consists of three parts.

1. Controller 2. Client programs 3. Interpreters

Fig. 2.5.5 shows browser architecture.

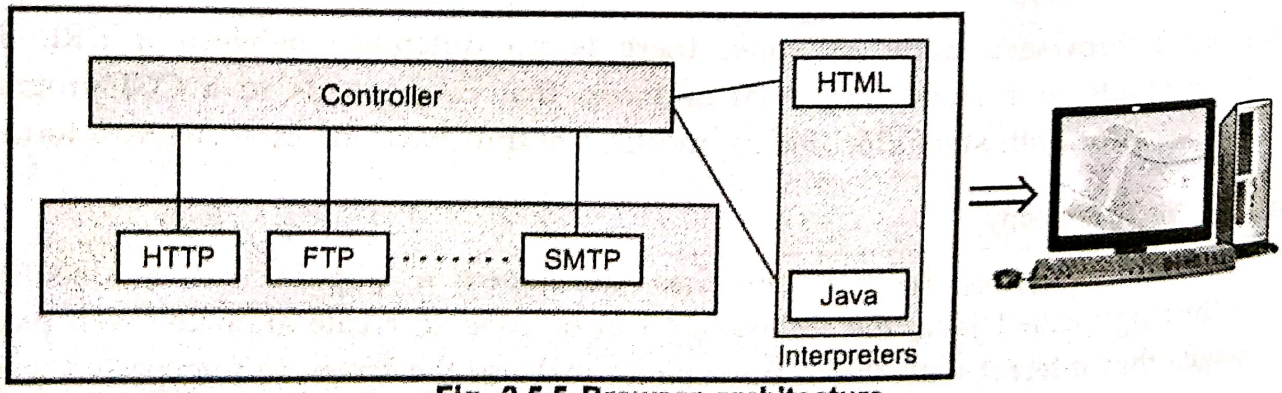


Fig. 2.5.5 Browser architecture

- The controller receives input from the keyboard or 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. Client program uses protocol such as HTTP, FTP or SMTP. The interpreter can be HTML or Java.

2.5.7 Caching in Web Browser

- Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. DNA handles this with a mechanism called caching.
- When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client.
- If the same or another client asks for the same mapping, it can check its cache memory and resolve the problem.
- To inform the client that the response is coming from the cache memory and not from an authoritative source, the server marks the response as unauthoritative.
- Caching speed up the resolution.
- Problem - if a server caches a mapping for a long time, it may send an outdated mapping to the client.

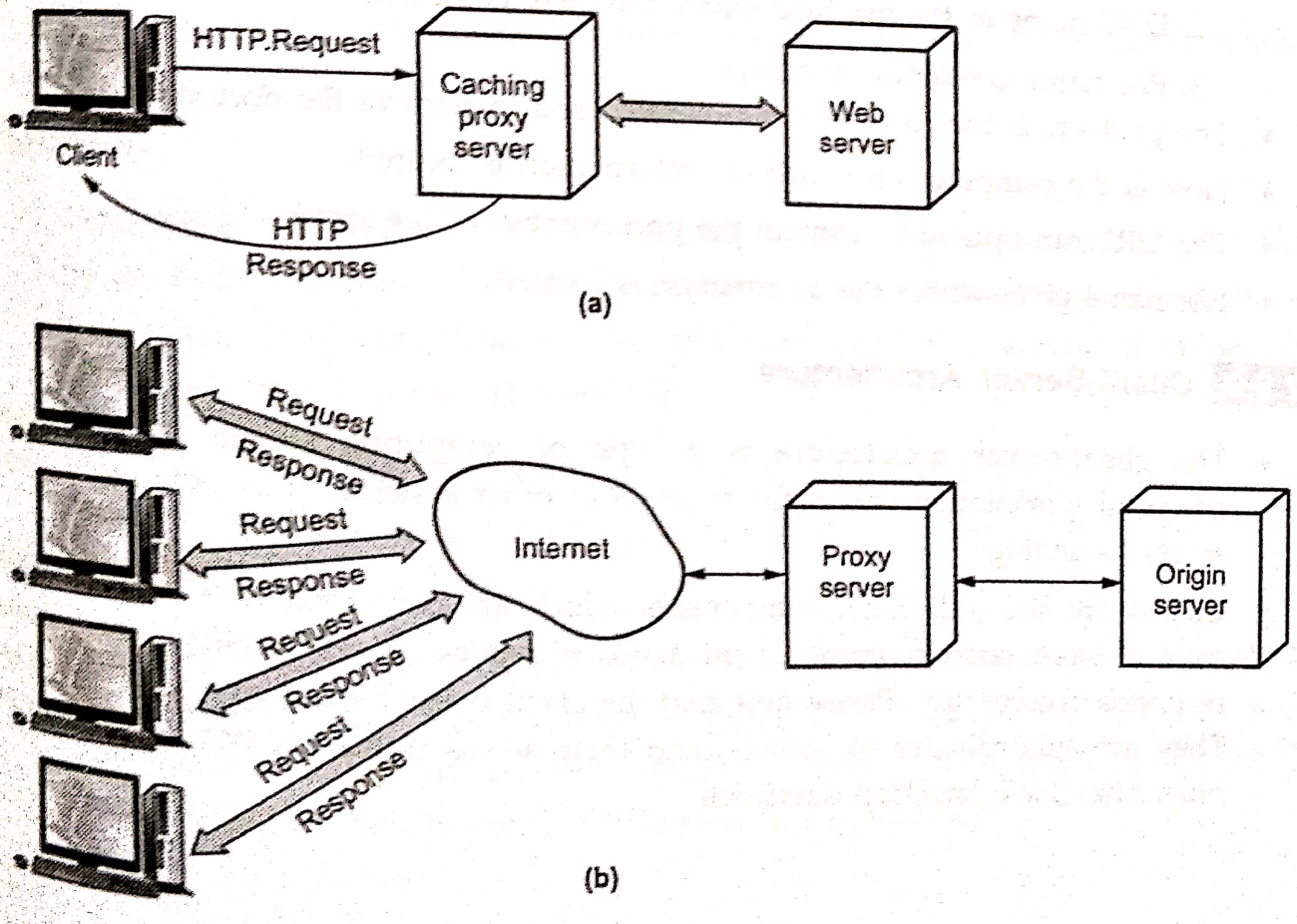


Fig. 2.5.6 Web Caching

- Above problem is solved by two methods.
  1. Authoritative server always adds a piece of information to the mapping called time to live (TTL).
  2. DNS requires that each server keeps a TTL counter for each mapping it caches. The cache memory must be searched periodically and those mappings with an expired TTL must be purged.
- It satisfies the client request without involving origin server.
- User sets brower i.e. web accesses via cache.
- Brower sends all HTTP request to cache. If the object is in cache, it returns the object otherwise cache request the object from origin server.
- Fig. 2.5.6 shows the caching.

2.5.8 Uniform Resource Locators

- The Uniform Resource Locator (URL) is a standard for specifying any kind of information on the Internet.
- URL has three parts
  1. The protocol
  2. DNS name of the machine where the page is located.
  3. File name containing the page.
- The protocol is the client-server program used to retrieve the document.
- Host is the computer on which the information is located.
- The URL can optionally contain the port number of the server.
- File name gives where the information is located.

2.5.9 Client-Server Architecture

- The client-server architecture is a type of computing system in which one powerful workstation serves the requests of other systems, is an example of client server technology.
- Clients are the individual components which are connected in a network. They have a basic configuration. Client sends a request/query to server and server responds accordingly. Please note that the client doesn't share any of its resources. They are subordinates to servers, and their access rights are defined by servers only. They have localized databases.

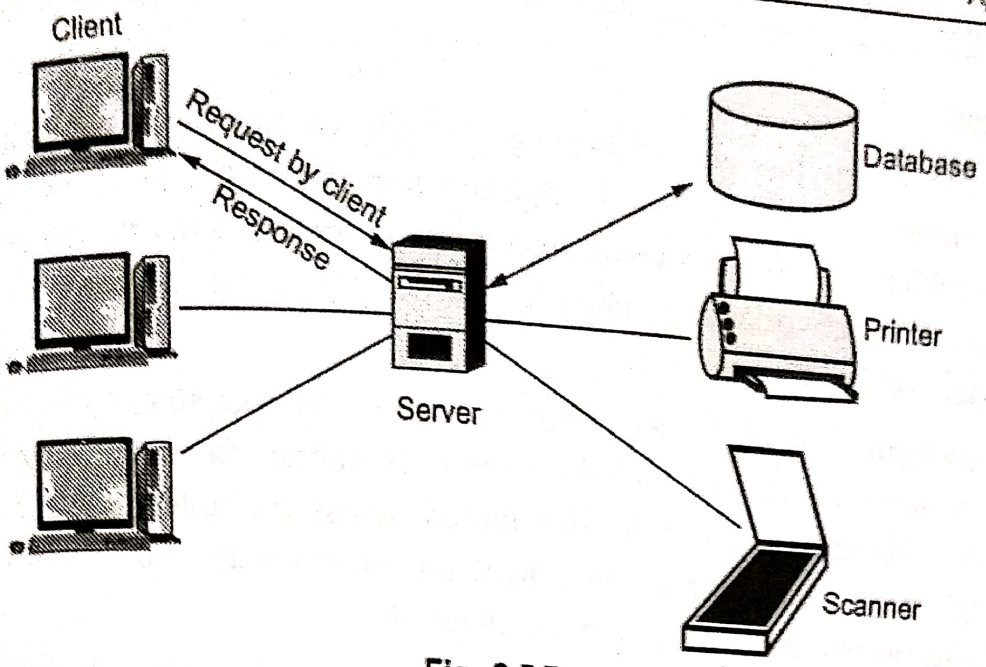


Fig. 2.5.7

Components of Client Server Network :

1) Clients or Workstations. 2) Servers.

3) **Network Devices :** They connect the clients and servers, and at the same time ensure proper collision free routing of information.

4) Other components like scanner, printer, etc can also be connected to network architecture.

Advantages

1. **Centralization of control :** Access, resources and integrity of the data are controlled by the **dedicated server** so that a program or unauthorized client cannot damage the system. This centralization also facilitates task of updating data or other resources (better than the networks P2P).

2. **Scalability :** You can increase the capacity of clients and servers separately. Any element can be increased (or enhanced) at any time, or you can add new nodes to the network (clients or servers).

3. **Easy maintenance :** distribute the roles and responsibilities to several standalone computers, you can replace, repair, upgrade, or even move a server, while customers will not be affected by that change (or minimally affect). This independence of the changes is also known as encapsulation.

Disadvantages

1. Traffic congestion has always been a problem in the paradigm of C/S. When a large number of simultaneous clients send requests to the same server might cause many problems for this (to more customers, more problems for the server). On the contrary, P2P networks each node in the network server also makes more nodes, the better bandwidth you have.
2. The paradigm of C/S Classic does not have the robustness of a network P2P. When a server is down, customer requests cannot be met. In most part, P2P networks resources are usually distributed across multiple nodes of the network. Although some out or abandon download, others may still end up getting data download on rest of the nodes in the network.
3. The software and hardware of a server are usually very decisive. A regular computer hardware staff may not be able to serve a certain number of customers. Usually you need specific software and hardware, especially on the server side, to meet the work. Of course, this will increase the cost.
4. The client does not have the resources that may exist on the server. For example, if the application is a Web, we cannot write the hard disk of the client or print directly on printers without taking before the print preview window of the browser.

Review Questions

1. *Explain WWW and HTTP.* **GTU : Dec.-11, Marks 5**
2. *Explain the architectural overview of the world wide web.* **GTU : Winter-12, Marks 7**
3. *Give architectural overview of WWW.* **GTU : Winter-13, Marks 7**
4. *Explain the concept of cookies and its components with suitable example.* **GTU : Winter-15, Marks 8**
5. *What is client - server architecture ? Discuss its merits and demerits.* **GTU : Winter-16,18, Marks 3**

2.6 Socket Programming with TCP and UDP **Summer-15, Winter-19**

Socket

- Socket interface is a protocol independent interface to multiple transport layer primitives. In order to write applications which need to communicate with other applications.

- Socket is an abstraction that is provided to an application programmer to send or receive data to another process.
- Data can be sent to or received from another process running on the same machine or a different machine.
- It is like an endpoint of a connection. It exists on either side of connection and identified by IP Address and Port number.
- Sockets works with UNIX I/O services just like files, pipes and FIFO.
- API stands for Application Programming Interface. It is an interface to use the network. Socket API defines interface between application and transport layer.
- The API defines function calls to create, close, read and write to/from a socket.

Advantages of using Socket Interface

- Syntax of the API functions is independent of the protocol being used. Ex:- TCP/IP and UNIX domain protocols can be used by applications using a common set of functions.
- Gives way to better portability of applications across protocol suites.
- Hides the finer details of the protocols from application programs thereby yielding faster and bug free application development
- Sockets are referenced through socket descriptors which can be passed directly to UNIX system I/O calls. File I/O and socket I/O are exactly similar from the programmer perspective.

Sockets versus File I/O

- Working with sockets is very similar to working with files. The socket ( ) and accept ( ) functions both return handles (file descriptor) and reads and writes to the sockets requires the use of these handles (file descriptors).
- In Linux, sockets and file descriptors also share the same file descriptor table. That is, if you open a file and it returns a file descriptor with value say 8, and then immediately open a socket, you will be given a file descriptor with value 9 to reference that socket.
- Even though sockets and files share the same file descriptor table, they are still very different. Sockets have addresses associated with them whereas files do not; notice that this distinguishes sockets form pipes, since pipes do not have addresses with which they associate.
- You cannot randomly access a socket like you can a file with lseek ( ). Sockets must be in the correct state to perform input or output.

Socket Abstraction

- Socket is the basic abstraction for network communication in the socket API. Socket defines an endpoint of communication for a process.

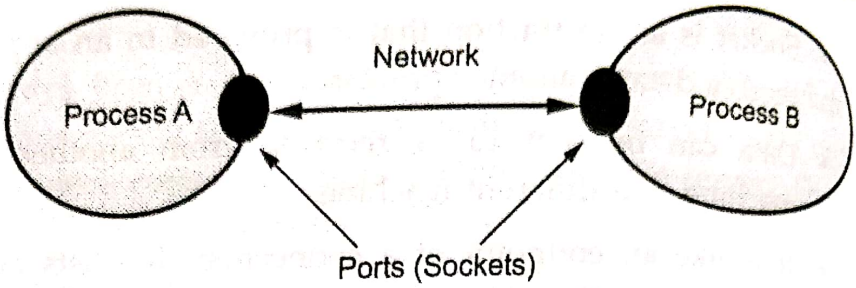


Fig 2.6.1 Socket and process

- Operating system maintains information about the socket and its connection. Fig. 2.6.1 shows the socket and process.

Socket Creation

int socket (int family, int type, int protocol);

Parameters:

1. family : AF_INET or PF_INET (These are the IP4 family)
2. type : SOCK_STREAM (for TCP) or SOCK_DGRAM (for UDP)
3. protocol : IPPROTO_TCP (for TCP) or IPPROTO_UDP (for UDP) or use 0

- If successful, socket ( ) returns a socket descriptor, which is an integer, and – 1 in the case of a failure.
- An example call:

```
if ((sd = socket(AF_INET, SOCK_DGRAM, 0) < 0)
{
        printf(socket() failed.);
        exit(1);
}
```

- Creating a socket is in some ways similar to opening a file. This function creates a file descriptor and returns it from the function call. You later use this file descriptor for reading, writing and using with other socket functions.
- Remember that the sockets API are generic. There must be a generic way to specify endpoint addresses. TCP/IP requires an IP address and port number for each endpoint address. Other protocol suites (families) may use other schemes.

2.6.1 TCP Socket

- In UNIX, whenever there is a need for IPC within the same machine, we use mechanism like signals or pipes. When we desire a communication between two applications possibly running on different machines, we need **Sockets.**

- Sockets are treated as another entry in the UNIX open file table.
- Sockets provide an interface for programming networks at the transport layer.
- Network communication using Sockets is very much similar to performing file I/O. In fact, socket handle is treated like file handle.
- Socket-based communication is programming language independent.
- To the kernel, a socket is an endpoint of communication. To an application, a socket is a file descriptor that lets the application read/write from/to the network.
- A server (program) runs on a specific computer and has a socket that is bound to a specific port. The server waits and listens to the socket for a client to make a connection request.
- To review, there are five significant steps that a program which uses TCP must take to establish and complete a connection. The server side would follow these steps :
  1. Create a socket.
  2. Listen for incoming connections from clients.
  3. Accept the client connection.
  4. Send and receive information.
  5. Close the socket when finished, terminating the conversation.
- In the case of the client, these steps are followed:
  1. Create a socket.
  2. Specify the address and service port of the server program.
  3. Establish the connection with the server.
  4. Send and receive information.
  5. Close the socket when finished, terminating the conversation.
- Only steps two and three are different, depending on if it's a client or server application.
- Fig. 2.6.2 shows a timeline of the typical scenario that takes place between a TCP client and server.

(Refer Fig. 2.6.2 on next page.)

2.6.2 Socket Function

- To perform network I/O, process call the socket function specifying the type of communication protocol desired.
- The *socket ( )* system call creates an endpoint for communication and returns a descriptor.

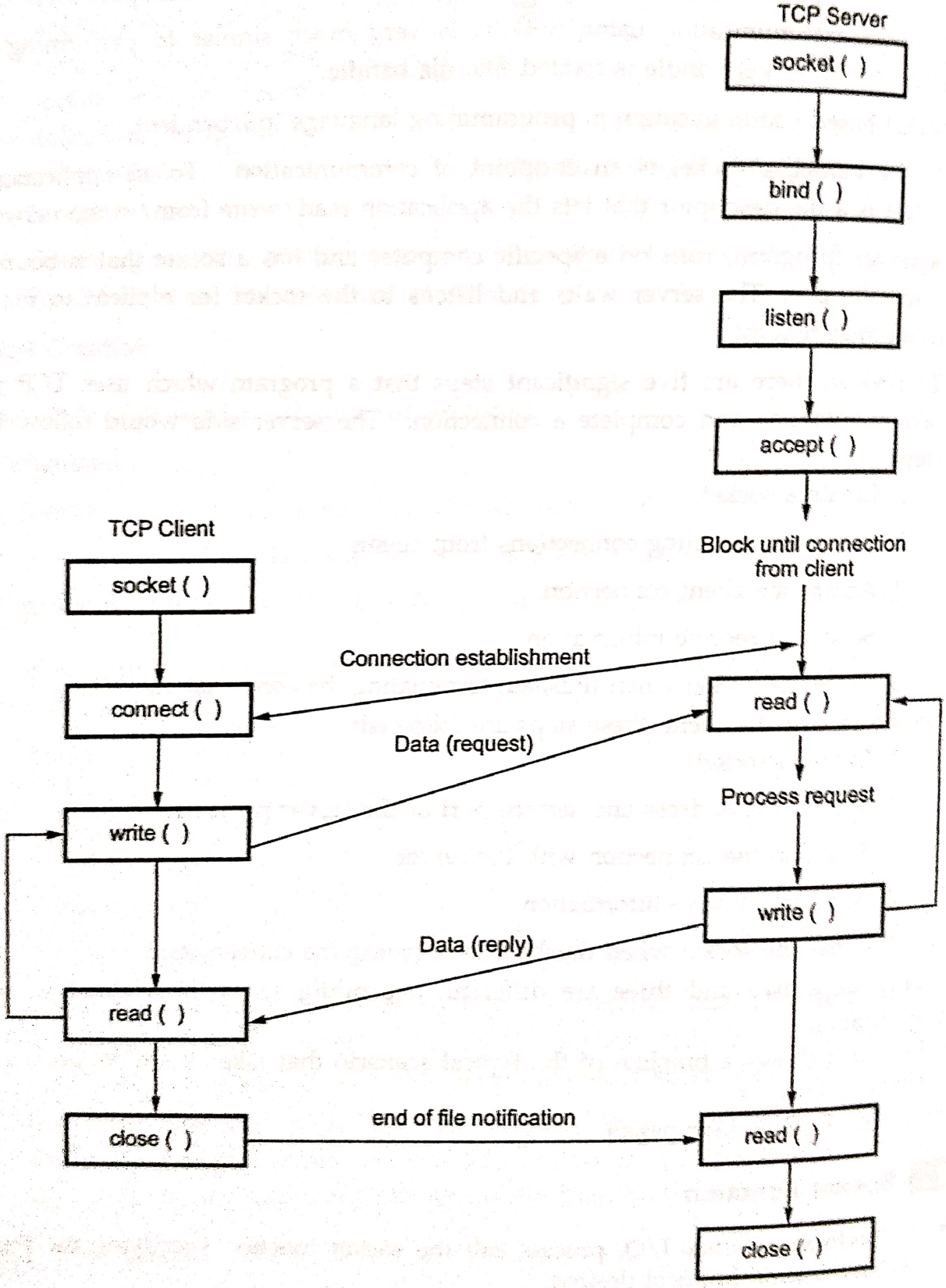


Fig. 2.6.2 Socket function for elementary TCP client server

- *Socket( )* allocates resources needed for a communication endpoint - but it does not deal with endpoint addressing.
- System calls for elementary TCP sockets :

```
include <sys/types.h>
#include <sys/socket.h>
    int socket(int family,int type,int protocol );
returns on success: socket descriptor {a small nonnegative integer}
        on error:       -1
```

1. family : This selects the protocol family which should be used. These families are defined in the include file <sys/socket.h>. Protocol family AF_INET for Internet, PF_INET for TCP/IP).

2. type : The socket has the indicated type, which specifies the semantics of communication. Currently defined types are:

SOCK_STREAM	stream socket	TCP
SOCK_DGRAM	datagram socket	UDP
SOCK_RAW	raw socket	

3. protocol : The protocol argument specifies a particular protocol to be used with the socket. Normally only a single protocol exists to support a particular socket type within a given protocol family. However, it is possible that many protocols may exist, in which case a particular protocol must be specified in this manner.

- A SOCK_STREAM type provides sequenced, reliable, two-way connection based byte streams.
- A SOCK_DGRAM socket supports datagrams.
- A SOCK_SEQPACKET socket may provide a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length;
- SOCK_RAW sockets provide access to internal network protocols and interfaces.
- Protocol family constants for socket function.

Family	Description
AF_INET	IPv4 protocols
AF_INET6	Ipv6 Protocols
AF_ROUTE	Unix domain protocol
AF_ROUTE	Routing sockets
AF_KEY	Key Socket

- Type of socket for socket function

Type	Description
SOCK_STREAM	Stream socket
SOCK_DGRAM	Datagram socket
SOCK_RAW	Raw socket

- Combination of family and type for the socket function.

	AF-INET	AF_INET6	AF_LOCAL	AF_ROUTE	AF_KEY
SOCK_STREAM	TCP	TCP	YES	–	–
SOCK_DGRAM	UDP	UDP	YES	–	–
SOCK_RAW	IPv4	IPv6	–	YES	YES

- The AF_prefix stands for "address family" and the PF_ prefix stands for "protocol family".
- PF_ was supposed to be used to create the socket that might support multiple address families, and AF_ value was used in socket address structures.
- But in actuality, a protocol family supporting multiple address families has never been supported and the <sys/socket.h> header defines the PF_value for a given protocol to be equal to the AF_ value for that protocol.
- Example:

```
if (( sd = socket (AF_INET, SOCK_STREAM, 0)) < 0)
        err_sys ("socket call error");
```

2.6.3 Connect Function

- The connect function is used by a TCP client to establish a connection with a TCP server.

```
#include <sys/socket.h>
int connect(int sockfd,const struct sockaddr *servaddr, socklen_t addrlen);
                                        Returns: 0 if OK, -1 on error
```

```
sockfd:   a socket descriptor returned by the socket function
*servaddr: a pointer to a socket address structure
addrlen:   the size of the socket address structure
```

sockfd is a socket descriptor returned by the socket function. The second and third arguments are a pointer to a socket address structure and its size.

- connect( ) only returns when a connection is established or when an error occurs.
- The socket address structure must contain the IP address and port number of the server.
- The client does not have to call bind before calling connect: the kernel will choose both an ephemeral port and the source IP address if necessary.
- In the case of a TCP socket, the connect function initiates TCP's three-way handshake. The function returns only when the connection is established or an error occurs.
- There are several different error returns possible.
  1. If the client TCP receives no response to its SYN segment, ETIMEDOUT is returned. For example, in 4.4BSD sends one SYN when connect is called, another 6 seconds later, and another 24 seconds later. If no response is received after a total of 75 seconds, the error is returned.
  2. ECONNREFUSED : If the server's response to the client's SYN is an RST. RST indicates that no process is waiting for connections on the specified port. Considered hard error , i.e. error is returned to the client as soon as the RST is received. RST is a type of TCP segment that is sent by TCP when something is wrong. Three conditions that generates a RST:

- A SYN arrives for a port that has no listening server.
- A TCP wants to abort an existing connection.
- A TCP receives a segment for a connection that does not exist.

  3. EHOSTUNREACH or ENETUNREACH: If the SYN elicits an ICMP destination unreachable, the client kernel saves the message; i.e. considered soft error. It should not terminate connection attempt, keep trying. If no response after some fixed amount of time (say 75 secs), the saved message is returned to the process.

- Example :

```
if(connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))!= 0)
        err_sys("connect call error");
```

2.6.4 Bind Function

- The servers bind to a particular port on their machines using the bind system call. This function has to be called only after a socket has been created and has to be passed the socket descriptor returned by the socket call. Again this binding on

both the machines need not be in any particular order. Moreover the binding procedure on the client is entirely optional.

- The *bind* system call requires the address family, the port number and the IP address. The address family is known to be AF_INET, the IP address of the client is already known to the operating system. All that remains is the port number.
- The programmer can specify which port to bind to, but this is not necessary. The binding can be done on a random port as well and still everything would work fine. The way to make this happen is not to call *bind* at all. Alternatively *bind* can be called with the port number set to 0. This tells the operating system to assign a random port number to this socket. This way whenever the program tries to connect to a remote machine through this socket, the operating system binds this socket to a random local port. This procedure as mentioned above is not applicable to a server, which has to listen at a standard predetermined port.
- When a socket is created, it does not have any notion of end points addresses. An application calls bind to specify the local endpoint address for a socket. That is the bind function assigns a local port and address to a socket.

```
#include <sys/socket.h>
int bind (int sockfd, const struct sockaddr *myaddr,
          socklen_t addrlen)
```

- The second argument is a pointer to a protocol specific address and the third argument is the size of this address structure. Servers bind their well known port when they start.
- A process can bind a specific IP address to its socket. The IP address must belong to an interface host.

Uses of bind ( )

- Server would like to bind to a well known address (port number).
- Client can bind to a specific port.
- Client can ask the O.S. to assign any available port number.

2.6.5 Listen Function

- The listen function is called only by TCP server and it performs following functions.
  a. The listen function converts an unconnected socket into a passive socket, indicating that the kernel should accept incoming connection requests directed to this socket. In terms of TCP transmission diagram the call to listen moves the socket from the CLOSED state to the LISTEN state.

b. The second argument to this function specifies the maximum number of connections that the kernel should queue for this socket.

```
#include <sys/socket.h>
int listen (int sockfd,int backlog);  returns 0 if OK -1on
                                                    error.
```

- This function is normally called after both the socket and bind functions and must be called before calling the accept function. The kernel maintains two queues and the backlog is the sum of these two queues. These are :
  a. An *Incomplete Connection Queue*, which contains an entry for each SYN that has arrived from a client for which the server is awaiting completion of the TCP three way handshakes. These sockets are in the SYN_RECD state.
  b. A *Completed Connection Queue* which contains an entry for each client with whom three handshakes has completed. These sockets are in the ESTABLISHED state.
- Fig. 2.6.3 .shows the two queues for a given listening socket.

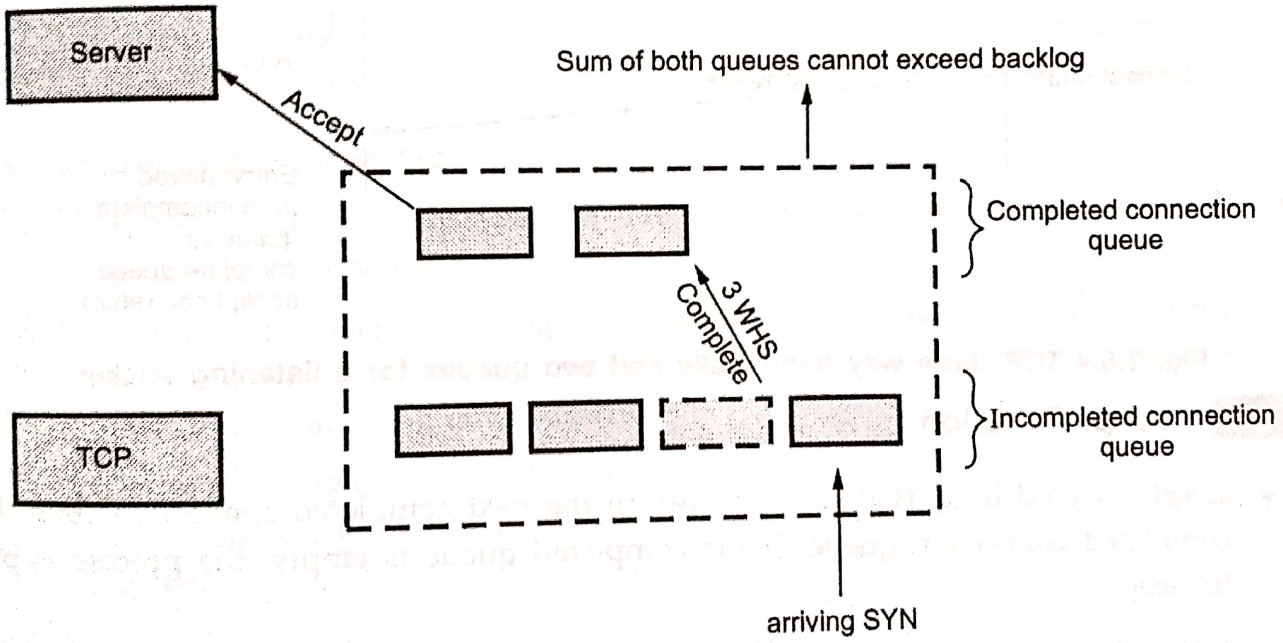


Fig. 2.6.3 Two queues maintained by TCP

- When a SYN arrives from a client, TCP creates a new entry on the incomplete queue and then responds with the second segment of the three way handshake. The server's SYN with an ACK of the clients SYN.
- This entry will remain on the incomplete queue until the third segment of the three-way handshake arrives or the entry times out. If the three-way hand shake completes normally, the entry moves from the incomplete queue to the completed queue.

- When the process calls accept, the first entry on the completed queue is returned to the process or, if the queue is empty, the process is put to sleep until an entry is placed onto the completed queue. If the queues are full when a client arrives, TCP ignores the arriving SYN, it does not send an RST. This is because the condition is considered temporary and the client TCP will retransmit its SYN with the hope of finding room in the queue.
- Fig. 2.6.4 shows the TCP three way handshake and two queues for a listening socket.

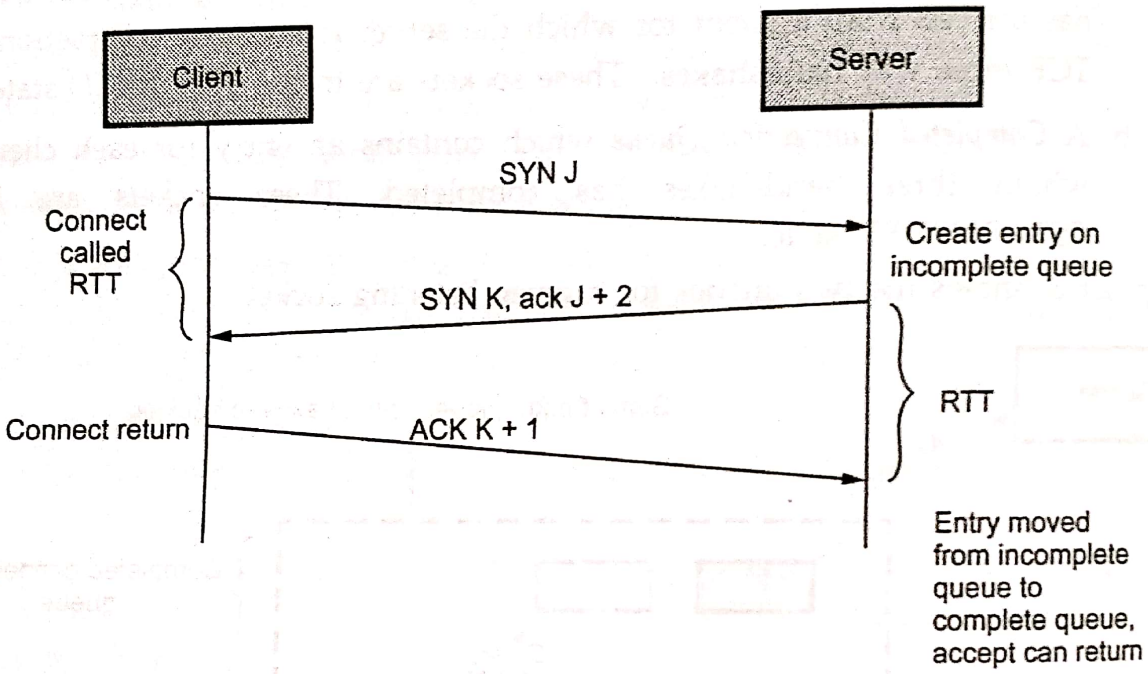


Fig. 2.6.4 TCP three way handshake and two queues for a listening socket

2.6.6 Accept Function

- *accept* is called by a TCP server to return the next completed connection from the completed connection queue. If the completed queue is empty, the process is put to sleep.

```
# include <sys/socket.h>
int accept ( sockfd, struct sockaddr * cliaddr, socklen_t *addrlen) ;
                return non negative descriptor if OK, -1 on error.
```

- The *cliaddr* and addrlen arguments are used to return the protocol address of the connected peer process. *addrlen* is a value-result argument before the call, we set the integer value pointed to by **addrlen* to the size of the socket address structure pointed to by *cliaddr* and on return this integer value contains the actual number of bytes stored by the Kernel in the socket address structure.
- If *accept* is successful, its return value is a brand new descriptor that was automatically created by the Kernel. This new descriptor refers to the TCP

connection with the client. This function accepts extracts a connection on the buffer of pending connections in the system, creates a new socket with the same properties as skfd, and returns a new file descriptor for the socket.

- The *accept* call is a blocking system call. In case there are requests present in the system buffer, they will be returned and in case there aren't any, the call simply blocks until one arrives.

Accept styles

There are basically three styles of using accept:

1. Iterating server : Only one socket is opened at a time. When the processing on that connection is completed, the socket is closed, and next connection can be accepted.
2. Forking server : After an accept, a child process is forked off to handle the connection. Variation: the child processes are preforked and are passed the socketId.
3. Concurrent single server : Use select to simultaneously wait on all open socketIds, and waking up the process only when new data arrives.

Pro and Con of Accept styles

1. Iterating server is basically a low performance technique since only one connection is open at a time.
2. Forking servers enable using multiple processors. But they make sharing state difficult, unless performed with threads. Threads, however present a very fragile programming environment.
3. Concurrent single server : Reduces context switches relative to forking processes and complexity relative to threads. But it does not benefit from multiprocessors systems.

2.6.7 fork and exec Function

- The fork ( ) function is the only way in UNIX to create a new process. It is defined as follows:

```
#include <unist.h>
pid_t fork(void);
Returns: 0 in child, process ID of child in parent, 1 on error
```

- The new process created by fork is called the child process. This function is called once but returns twice.

- The only difference in the returns is that the return value in the child is 0, whereas the return value in the parent is the process ID of the new child.
- The reason the child's process ID is returned to the parent is that a process can have more than one child, and there is no function that allows a process to obtain the process IDs of its children.
- The reason fork returns 0 to the child is that a process can have only a single parent, and the child can always call getppid to obtain the process ID of its parent.
- Both the child and the parent continue executing with the instruction that follows the call to fork. The child is a copy of the parent. For example, the child gets a copy of the parent's data space, heap, and stack.
- There are two typical uses of fork:
  1. A process makes a copy of itself so that one copy can handle one operation while the other copy does another task. This is typical for network servers.
  2. A process wants to execute another program. Since the only way to create a new process is by calling fork, the process first calls fork to make a copy of itself, and then one of the copies (i.e. child process) calls exec to replace itself with the new program.

exec function

- Fork function is to create a new process that then causes another program to be executed by calling one of the exec functions.
- When a process calls one of the exec functions, that process is completely replaced by the new program, and the new program starts executing at its main function.
- The process ID does not change across an exec, because a new process is not created; exec merely replaces the current process its text, data, heap, and stack segments with a brand new program from disk.
- There are six different exec functions. These six functions round out the UNIX system process control primitives.
- With fork, we can create new processes; and with the exec functions, we can initiate new programs. The exit function and the wait functions handle termination and waiting for termination.

```
#include <unistd.h>
int execl(const char *pathname,constchar *arg0,... /* (char*)0*/ );
int execv (const char *pathname, char *const argv[]);
int execle (const char *pathname, const char *arg0, ...
                    /* (char *) 0, char *const envp[] */ );
int execve (const char *pathname,char *const argv[],char *const envp[]);
```

```
int execlp(const char *filename,const char *arg0,.../*(char*)0*/);
int execvp (const char *filename, char *const argv[]);
```

All six return : 1 on error, no return on success

- The differences in the six exec functions are:
  1. Whether the program file to execute is specified by a filename or a pathname;
  2. Whether the arguments to the new program are listed one by one or referenced through an array of pointers;
  3. Whether the environment of the calling process is passed to the new program or whether a new environment is specified.
- These functions return to the caller only if an error occurs. Otherwise, control passes to the start of the new program, normally the main function.

2.6.8 Close Function

- The *close* ( ) function deallocate the file descriptor indicated by *fildes*. To deallocate means to make the file descriptor available for return by subsequent calls to *open* ( ) or other functions that allocate file descriptors. All outstanding record locks owned by the process on the file associated with the file descriptor shall be removed (i.e. unlocked).
- If *close*( ) is interrupted by a signal that is to be caught, it shall return -1 with *errno* set to EINTR and the state of *fildes* is unspecified. If an I/O error occurred while reading from or writing to the file system during *close*( ), it may return -1 with *errno* set to EIO; if this error is returned, the state of *fildes* is unspecified.
- When all file descriptors associated with a pipe or FIFO special file are closed, any data remaining in the pipe or FIFO shall be discarded. When all file descriptors associated with an open file description have been closed, the open file description shall be freed.
- If the link count of the file is 0, when all file descriptors associated with the file are closed, the space occupied by the file shall be freed and the file shall no longer be accessible.
- The normal UNIX close ( ) is also used to close a socket and terminate a TCP connection.

```
#include<unistd.h>
int close (int sockfd);
```

- The default action of close with a TCP socket is to mark the socket as closed and return to the process immediately. The socket descriptor is no longer usable by the process. i.e. it can not be used as an argument to read or write.

getsockname ( ) and getpeername():

- These two functions return either the local protocol address associated with a socket or the foreign address associated with a socket.

```
#include <sys/socket.h>
int getsockname(int sockfd, struct sockaddr *localadddr,
                socklen_t  *addlen);
int getpeername(int sockfd, struct sockaddr *peeradddr,
                socklen_t  *addlen);
                both return 0 if OK and –1 on error.
```

- The *getsockname( )* function retrieves the locally-bound name of the specified socket, stores this address in the **sockaddr** structure pointed to by the *address* argument, and stores the length of this address in the object pointed to by the *address_len* argument.
- If the actual length of the address is greater than the length of the supplied **sockaddr** structure, the stored address will be truncated. If the socket has not been bound to a local name, the value stored in the object pointed to by *address* is unspecified.
- Upon successful completion, 0 is returned, the *address* argument points to the address of the socket, and the *address_len* argument points to the length of the address. Otherwise, – 1 is returned and *errno* is set to indicate the error.
- **The** *getpeername( )* function retrieves the peer address of the specified socket, stores this address in the **sockaddr** structure pointed to by the *address* argument, and stores the length of this address in the object pointed to by the *address_len* argument.
- If the actual length of the address is greater than the length of the supplied **sockaddr** structure, the stored address will be truncated. If the protocol permits connections by unbound clients, and the peer is not bound, then the value stored in the object pointed to by *address* is unspecified.
- Upon successful completion, 0 is returned. Otherwise, – 1 is returned and *errno* is set to indicate the error.
- These functions are required for the following reasons.
  a. After connect successfully returns a TCP client that does not call **bind()**, **getsocketname()** returns the local IP address and local port number assigned to the connection by the Kernel
  b. After calling bind with a port number of 0, **getsockname()** returns the local port number that was assigned
  c. When the server is exceed by the process that calls **accept()**, the only way the server can obtain the identity of the client is to call **getpeername()**.

2.6.9 UDP Socket Programming

- There are some instances when it makes to use UDP instead of TCP. Some popular applications built around UDP are DNS, NFS and SNMP.
- Fig. 2.6.5 shows the interaction between a UDP client and server.
- Initially client does not establish a connection with the server. Instead, the client just sends a datagram to the server using the send to function which requires the address of the destination as a parameter.
- Similarly, the server does not accept a connection from a client. Instead, the server just calls the recvfrom function, which waits until data arrives from some client.

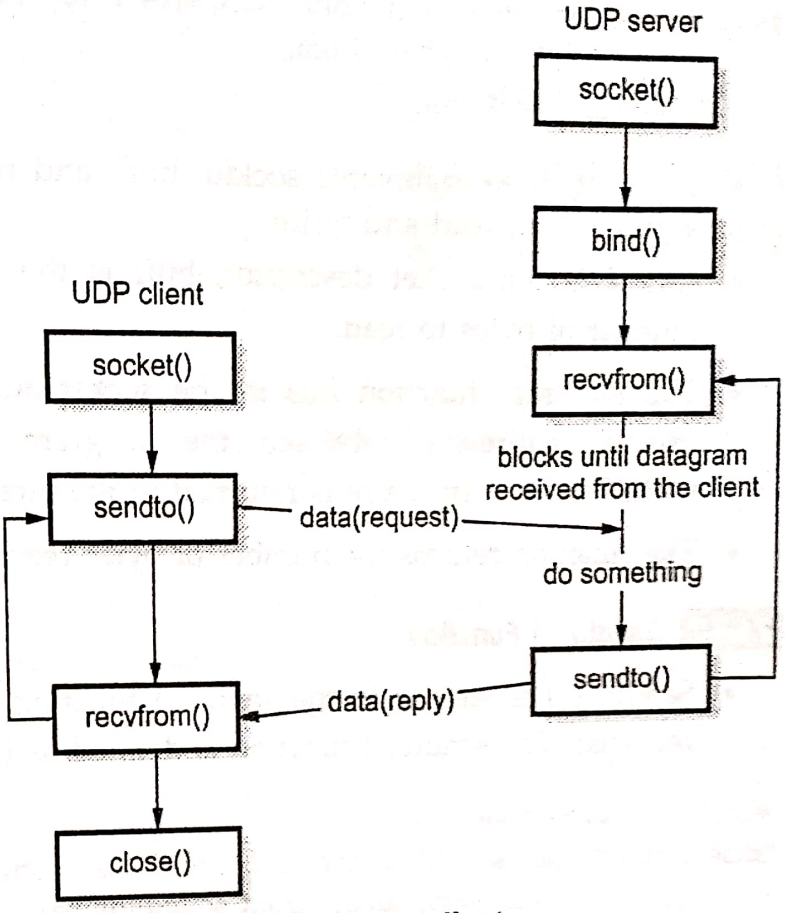


Fig. 2.6.5 UDP client-server

- *recvfrom* returns the protocol address of the client, along with the datagram, so the server can send a response to the client.

Steps on the client side are as follows:

1. Create a socket using the socket( ) function;
2. Send and receive data by means of the recvfrom ( ) and sendto( ) functions.

Steps on the server side are as follows:

1. Create a socket with the socket( ) function;
2. Bind the socket to an address using the bind( ) function;
3. Send and receive data by means of recvfrom( ) and sendto ( ).

2.6.9.1 The recvfrom ( ) Function

- This function is similar to the read ( ) function, but three additional arguments are required. The recvfrom ( ) function is defined as follows:

```
#include <sys/socket.h>
ssize_t recvfrom (int sockfd, void* buff, size_t nbytes, int flags,
                  struct sockaddr* from,
      socklen_t *addrlen);
```

- The first three arguments sockfd, buff, and nbytes, are identical to the first three arguments of read and write.
- sockfd is the socket descriptor, buff is the pointer to read into, and nbytes is number of bytes to read.
- The recvfrom function fills in the socket address structure pointed to with the protocol address of who sent the datagram. The number of bytes stored in the socket address structure is returned in the integer pointed by addrlen.
- The function returns the number of bytes read if it succeeds, -1 on error.

2.6.9.2 Sendto ( ) Function

- Sendto ( ) is similar to the send ( ) function, but three additional arguments are required. The sendto( ) function is defined as follows:

```
#include <sys/socket.h>
ssize_t sendto (int sockfd, const void *buff, size_t nbytes,
                int flags, const struct sockaddr *to,
                socklen_t addrlen);
```

- The first three arguments sockfd, buff, and nbytes, are identical to the first three arguments of recv.
- sockfd is the socket descriptor, buff is the pointer to write from, and nbytes is number of bytes to write.
- The sendto argument is a socket address structure containing the protocol address (e.g., IP address and port number) of where the data is sent. addrlen specified the size of this socket.

The function returns the number of bytes written if it succeeds, -1 on error.

Example 2.6.1 *Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segmen to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts ?*

GTU : Summer-15, Marks 4

Solution : Yes, both segments will be directed to the same socket. For each received segment, at the socket interface, the operating system will provide the process with the IP address to determine the origins of the individual segments.

Review Questions

1. *Demonstrate socket programming flow for a simple client-sever application using TCP. why must the server program be executed before client program ? For the client-server application over UDP, why may the client program be executed before the server program ?*

GTU : Winter-19, Marks 7

2. *Explain in brief socket multiplexing and demultiplexing.*

GTU : Winter-19, Marks 4

Fill in the blanks with Answers

1 *HTTP stands for* ———— **[Ans. : Hyper Text Transfer Protocol]**

2 *HTTP is a* ———— *protocol.* **[Ans. : stateles]**

3 *The DNS protocol runs over UDP and uses port* ———. **[Ans. : 53]**

4 *The DNS name space is* ———— *and it is similar to the unix file system.* **[Ans. : hierarchical]**

5 *The 3 character domains are called the* ————*domains.* **[Ans. : generic]**

6 *LDAP is an* ———— *protocol that is implemented directly on top of TCP.* **[Ans. : (application-level]**

7 *SMTP uses a TCP socket on port* ——— *to transfer e-mail reliably from client to server.* **[Ans. : 25]**

8 ———— *handles incoming and outgoing mails.* **[Ans. : Mail server]**

9 *MIME stands for* ————. **[Ans. : Multipurpose Internet Mail Extensions]**

10 ———— *is used to transfer e-mail messages from a mail server to mail client software.* **[Ans. : Post Office Protocol 3]**

11 *The* ———— *is the universal language of the web.* **[Ans. : HTML]**

12 *World wide web uses* ———— *interaction.* **[Ans. : client-server]**

13 *If a time and date are supplied, the cookie is said to be* ———— *and is kept until it expires.* **[Ans. : persistent]**

14 ——— *is an abstraction that is provided to an application programmer to send or receive data to another process.* **[Ans. : Socket]**

15 *SOCK_STREAM sockets are used by* ———— *processes.* **[Ans. : TCP]**

16 *The* ——— *system call requires the address family, the port number and the IP address.* **[Ans. : bind]**

17 *SOCK_STREAM sockets are usued by* ——— *processes.* **Summer-16, Mark 1**

a) UDP b) IP c) TCP d) HTTP **[Ans. : c]**

Short Questions and Answers

Q.1 Briefly explain the working of SMTP. Winter-2016

Ans. : SMTP : SMTP is application layer protocol of TCP/IP model. SMTP transfers message from sender's mail servers to the recipient's mail servers. SMTP interacts with the local mail system and not the user. SMTP uses a TCP socket on port 25 to transfer e-mail reliably from client to server.

Q.2 How DNS is useful in Internet ? Winter-2016

Ans. : DNS : DNS is a distributed database that resides on multiple machines on the internet and used to convert between names and address and to provide e-mail routing information. DNS provides the protocol that allows the client and servers to communicate with each other.

Q.3 Give an example of URL and explain its components. Winter-2016

Ans. : URL : URL is a standard for specifying any kind of information on the internet. The URL defines four things. 1. Method 2. Host computer 3. Port 4. Path

Each URL uniquely identifies a page of information by giving the name of a remote computer, a server on that computer and a specific page of information available from the server.

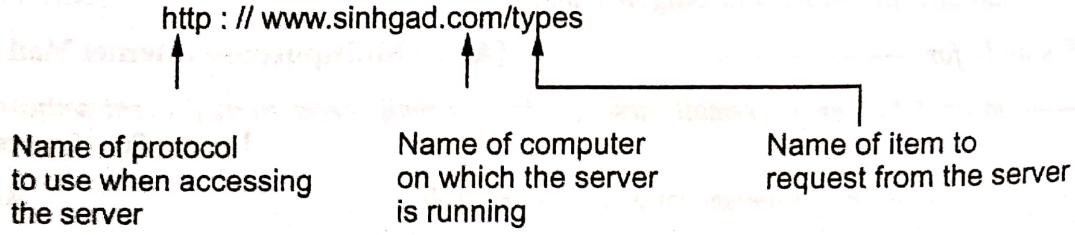


Fig. 2.1 Format of URL