Module-1: Application layer.

1.1 PRINCIPLES OF NETWORK APPLICATION.

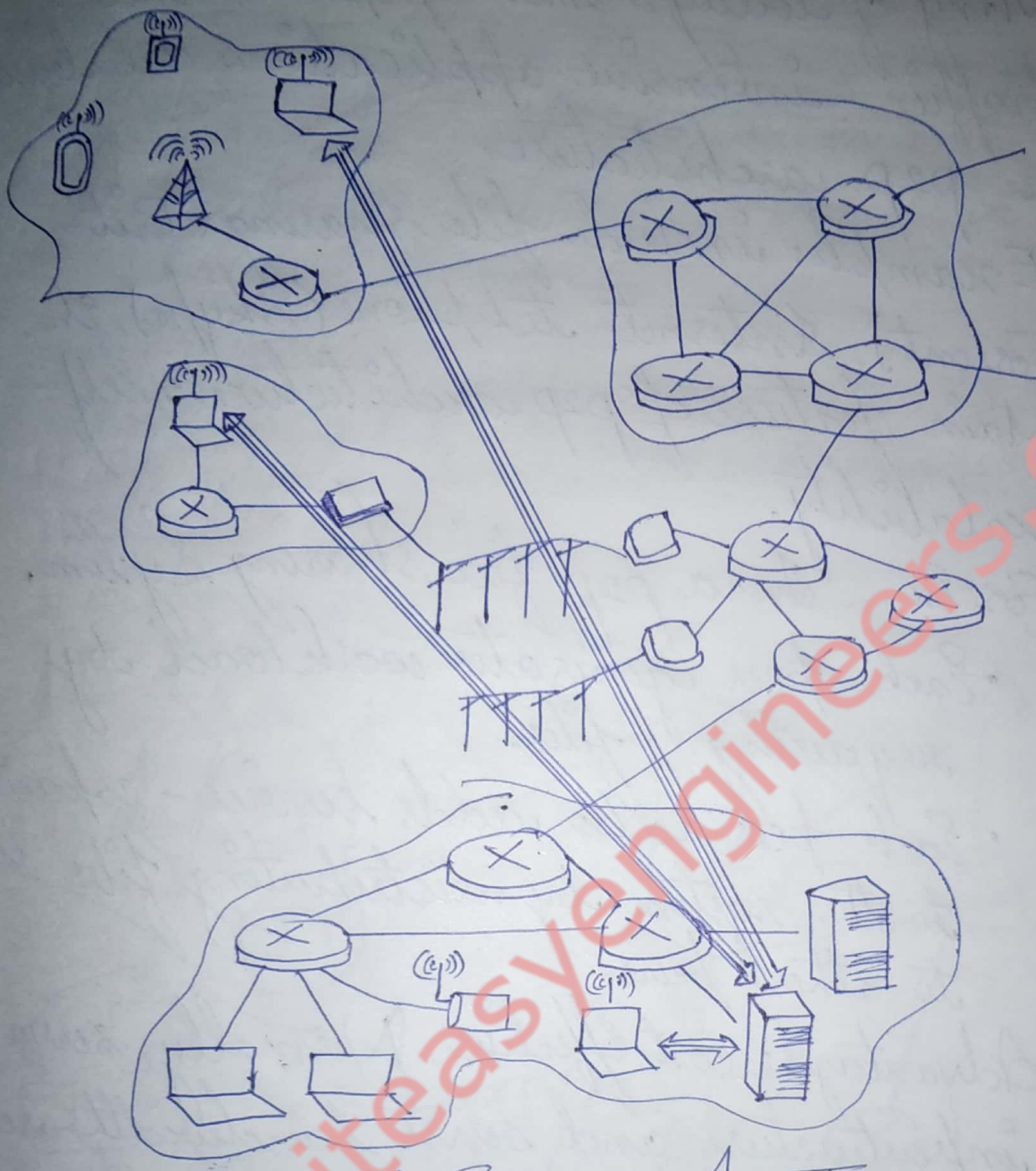
- Network-applications are the driving force for the explosive development of the internet.
- Ex. of network application
  1) Web
  2) File transfers.
  3) E-mail
  4) P2P file sharing
  5) Social network (facebook, Twitter).
  6) Video distribution (youtube).
  7) Real time video conferencing (Skype).
  8) On-line games (world of warcraft).
- In network-applications, programs usually needs to
  → run on different end-systems and
  → communicate with one another over network
- For Ex. In web application, there are 2 different programs
  1) The browser program running in the user's host (laptop or smartphone)
  2) The web-server program running in the web-server host.

1.1.1 NETWORK APPLICATION ARCHITECTURES.

- Two approaches for developing an application.
  1) client - server architecture.
  2) p2p (peer to peer architecture).

1.1.1.1 CLIENT - SERVER ARCHITECTURE

→ In this architecture, there is a server and many clients distributed over the network (Fig 1.1a).
→ The server is always-on while a client can be randomly ewn.
→ The server is listening on the network and a client initializes the communication.
→ Upon the request from a client, the server provides certain services to the client.
→ Usually, there is no communication between 2 clients.
→ The server has a fixed IP address
→ A client contacts the server by sending a packet to server's IP address.
→ A server is able to communicate with many clients.
→ The applications such as FTP, telnet, web, E-mail etc use the client - server architecture.



a. CLIENT-SERVER ARCHITECTURE

1.1.1.2 P2P ARCHITECTURE

→ There is no dedicated server
→ pairs of host is called peers.
→ The peers communicate directly with each other.
→ The ~~pee~~ peers are not owned by the service-provider. Rather, the peers are laptops controlled by users.

→ Many of today's most popular and traffic-intensive applications are based on p2p architecture.
→ Examples include file sharing (BitTorrent), Internet telephone (Skype), etc.
→ Main feature of p2p architecture: self-scalability.
→ For Ex. In a p2p file sharing System,
- Each peer generates workload by requesting files.
- Each peer also adds service-capacity to the system by distributing files to other peers.

→ Advantages: cost effective. Normally, server-infrastructure and server bandwidth are not required.
→ Three challenges of p2p applications
1) ISP friendly
⇒ Most residential ISPs have been designed for asymmetrical bandwidth usage.
⇒ Asymmetrical bandwidth means there is more downstream traffic than upstream traffic.
⇒ But p2p applications shift upstream traffic from server to residential ISP's

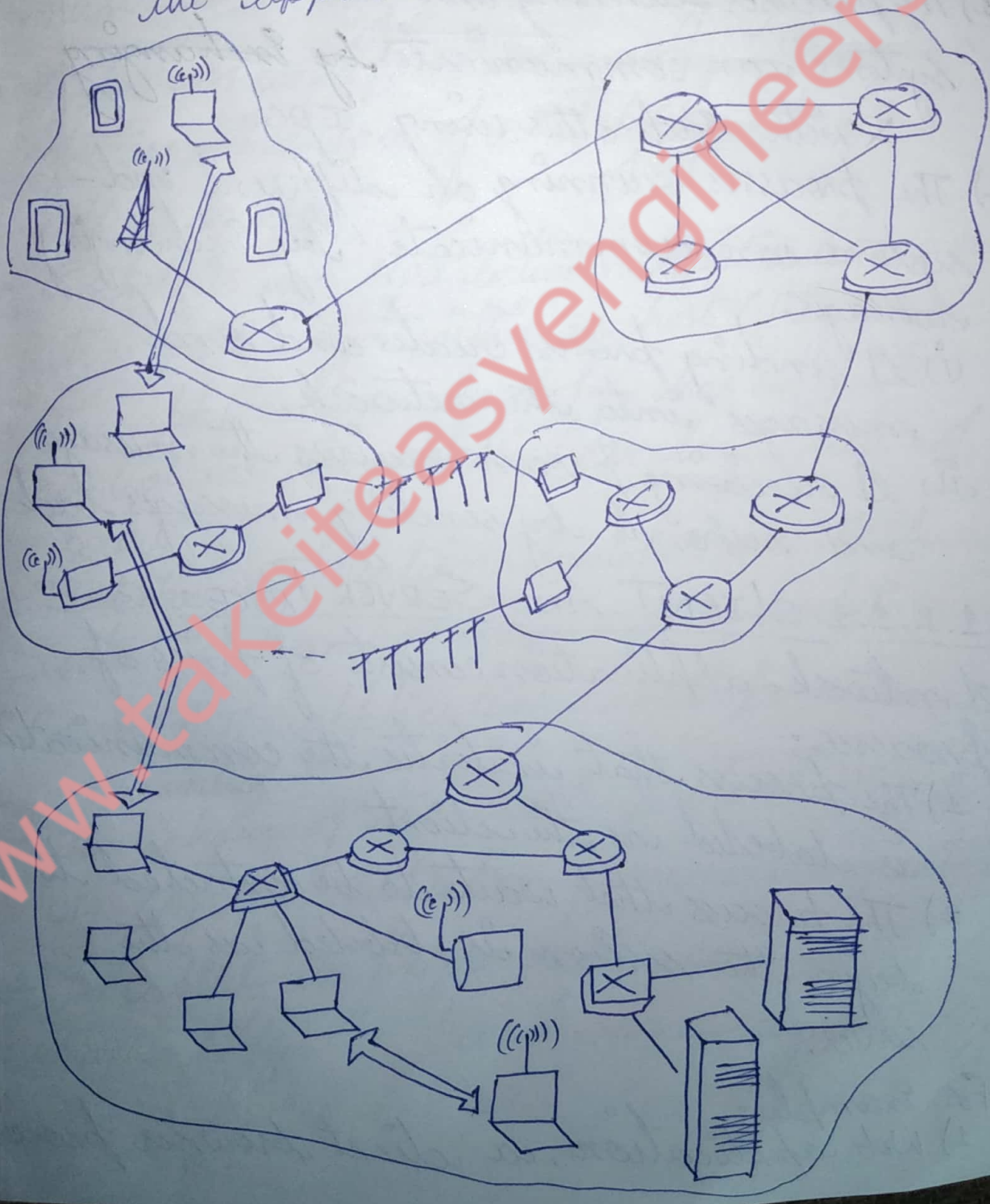
which stress on the ISP's.

2) Security

> Since the highly distribution and openness, p2p application can be a challenge to security.

3) Incentive

> Sucess of p2p depends on convincing users to volunteer bandwidth and resources to the application.



1.1.2 PROCESSES COMMUNICATION

1.1.2.1 PROCESS

- A process is an instance of a program running in a computer (IPC → Inter process communication)
- The process may run on the 1) same System
  2) different System

1) The processes running on the same End systems can communicate with each other using IPC
2) The processes running on different End-systems can communicate by Exchanging messages.
   (i) A sending process creates and sends messages into the network.
   (ii) A receiving process receives the messages and responds by sending messages back.

1.1.2.1.1 CLIENT AND SERVER PROCESSES

- A network-application consists of pairs of processes:
  1) The process that initiates the communication is labeled as the client.
  2) The process that waits to be contacted to begin the session is labeled as the server.
- For Example:
  1) Web application, a client browser proces

communicates with a web-server-process.

2) In p2p file system, a file is transferred from a process in one peer to a process in another peer.

1.1.2.1.2 INTERFACE BETWEEN THE PROCESS AND THE COMPUTER NETWORK SOCKET.

- Any message sent from one process to another must go through the underlying network.
- A process sends/receives message through a software-interface of underlying network is called socket.
- Socket is an API between Application and transport layer within a host (Fig 1.2).
- The application-developer has complete control at the application layer side of socket
- But the application developer has little control of transport layer side of socket.

For Ex: The application-developer can control

1) The choice of transport-protocol: Tcp or UDP (API → Application programming interface)
2) The ability to fix parameters such as max-buffer and max-segment-size.

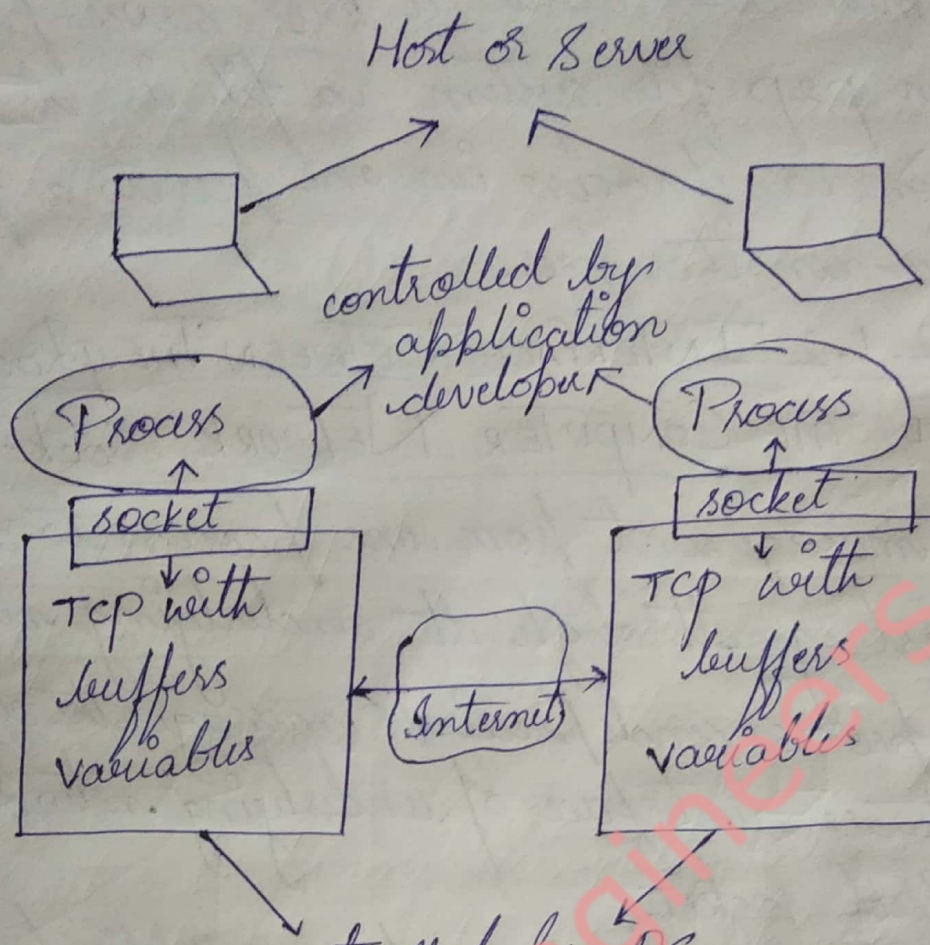


Fig 1.2 Application processes, sockets and transfer - protocol.

1.1.2.1.3 ADDRESING PROCESSES

- To identify the receiving - process, two pieces of information need to be specified
  1) IP address of destination - host
  2) Port - number that specifies the receiving process in destination - host.
- In the internet, the host is identified by IP address.
- An IP address is a 32 - bit that uniquely identify the host.
- Sending - Process needs to identify receiving process, a host may run several

network - application
- For this purpose, a distination port-no is used.
- For example,
  1) A web server is identified by port-no 80
  2) A mail-server is identified by port-number 20

1.1.3 TRANSPORT SERVICES AVAILABLE TO APPLICATIONS

- A networks usually provide more than one transport-layer protocols for different applications
- An application-developer should choose certain protocol according to the type of application
- Different protocols may provide different services.

1.1.3.1 RELIABLE DATA TRANSFER

- Reliable means ~~ga~~ guaranteeing the data from the sender to receiver is delivered correctly.
  For Ex: Tcp provides reliable service to an application
- Unreliable means data from the sender to receiver may never arrive
  For Ex. UDP provides unreliable service to an application.

- Unreliable may be acceptable for loss-tolerant applications, such as multimedia application
- In multimedia application, the lost data might result in a small glitch in the audio / video.

1.1.3.2 THROUGHPUT.

- Throughput is the rate at which the sending process can deliver bits to the receiving-process
- Since other hosts are using the network the throughput can fluctuate with time
- Two types of application
  1) Bandwidth Sensitive Applications
  → These application need a guaranteed throughput
  For Ex. Multimedia application
  → Some transport-protocol provides guaranteed throughput at some specified rate (x bits / sec)
  2) Elastic Application
  → These application may not need a guarantee throughput.
  For Ex - Electronic mail, file transfer & Web transfer.

1.1.3.3 TIMING

- A transport layer protocol can provide timing - guarantees.
- For Ex: guaranteeing ~~Reception~~ Every bit received at the receiver is less than 100ms
- Timing constraints are useful for real-time application such as
  - → Internet telephony
  - → Virtual Environments
  - → Teleconferencing and
  - → Multiplayer games.

1.1.3.4 SECURITY

- A transport - protocol can provide one or more security services.
- For Example,
  1) In the sending host, a transport - protocol can Encrypt all transmitted - data.
  2) In the receiving host, the transport protocol can decrypt the received data.

1.1.4 TRANSPORT SERVICES PROVIDED BY THE INTERNET

- The internet makes two transport - protocols available to application. UDP and TCP

- An application – developer who creates a new network-application must use either: UDP & TCP.
- Both UDP and TCP offers a different set of services to the invoking application.
- Table shows the service requirements for some selected applications.

Application	Data loss	Throughput time	Sensitive
File transfer/ download	no loss	elastic	NO.
E-mail	no loss	elastic	NO.
web documents	no loss	elastic (few kbps)	NO.
Internet telephony /video-conferencing	loss-tolerant	Audio: few kbps – 1Mbps Video: 10kbps – 5Mbps	yes: 100s of ms
streaming stored video/ audio	loss-tolerant	same as above	yes: few sec
Interactive games	loss-tolerant	few kbps – 10kbps	yes: 100s of ms.
Instant messaging	No loss	elastic	yes and no.

1.1.4.1 TCP SERVICES

- An application using transport-protocol TCP receives following 2 services.

1. connection - Oriented Service.

→ Before the start of communication, client and server need to exchange control-information.

→ This phase is called handshaking phase.

→ Then, the 2 processes can send messages to Each other over the connection.

→ After the End of communication, the application must tear down the connection.

2. Reliable Data transfer Service.

→ The communicating processes must deliver all data sent without Error and in the proper order.

Tcp also includes a congestion - control. The congestion - control throttles a sending process when the network is congested.

1.1.4.2 UDP SERVICES

→ UDP is a lightweight transport - protocol, providing minimal services.

→ UDP is connectionless, so there is no handshaking before the 2 processes start to communicate.

→ UDP provides an unreliable data transfer service.

→ Unreliable means providing no guarantee that the message will reach the receiving - process.

→ Not Furthermore, messages that do arrive at the receiving - process may arrive out - of order.
→ UDP does not include a congestion - control.
→ UDP can pump data into the network layer at any rate.

1.2 THE WEB AND HTTP

→ The appearance of web dramatically changed the internet.
→ Web has many advantages for a lot of applications.
→ It operates on demand so that the users receive what they want when they want it.
→ It provides an Easy way for Everyone make information available over the world
→ Hyperlinks and search Engines help us navigate through an ocean of web-sites.
→ Forms, Javascript, Java Applets, and many other devices Enables us to interact with pages and sites.
→ The Web serves as a platform for many killer applications including youtube, gmail and facebook.

1.2.1 OVERVIEW OF HTTP

1.2.1.1 WEB

→ A web-page consist of objects (HTML)
→ An objects is a file such as an HTML file, a JPEG image, a Java applet, a video clip.
→ The object is addressable by a single URL
→ Most web-pages consist of a base HTML file and server referenced objects.
→ For Example:
If a web page contains ~~a~~ HTML text and 5 JPEG images; then the web pages has 6 objects.
1) Base HTML file and
2) 5 images
→ The base HTML file references the other objects in the page with the objects URLs
→ URL has 2 components
1) The hostname of the server that houses the object and
2) The object's path name.
→ For Example
"http://www.someSchool.edu/someDept/picture.gif"
In the above URL,
1) Hostname = "www.someSchool.edu

2) pathname = "/someDept/picture.gif".

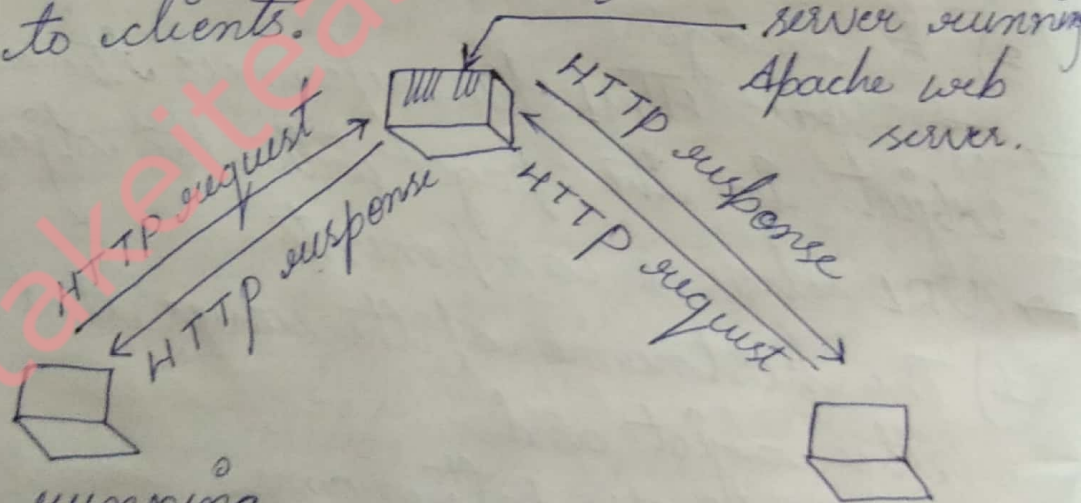
→ The web browsers implement the client side of HTTP,
For Ex: Google chrome, Internet Explorer.

→ The web-servers implement the server side of HTTP,
For Ex. Apache.

1.2.1.2 HTTP

→ HTTP is web's application layer protocol (Fig 1.3).

→ HTTP defines
  → how clients requests web pages from server and
  → how servers transfer web-pages to clients.



→ When a user request a web page, the browser sends HTTP request to the server.

→ Then, the server responds with HTTP response that contains the requested objects.
→ HTTP uses Tcp as its underlying transport protocol.
→ The HTTP client first initiates a Tcp connection with server.
→ After connection setup, the browser and the server - processes access Tcp through their sockets
→ HTTP is a stateless protocol
→ Stateless means the server sends requested object to client w/o storing state info about the client.
→ HTTP uses the client server architecture:
1) client
Browser that requests receive and display web objects.
2) Server
Web server sends objects in response to requests.

1.2.2. NON-PERSISTENT & PERSISTENT CONNECTIONS

→ In many internet applications, the client and server communicate for an extended period of time.

→ when this client - server interactions takes place over TCP, a decision should be made:

1) should each request/response pair be sent over a seperate TCP connection

or

2) should all request and their corresponding response be sent over same TCP connection?

→ These different connection are called non-persistent connections (1) or persistent connection (2).

→ Default mode: HTTP uses persistent connections.

1.2.2.1 HTTP WITH NON-PERSISTENT CONNECTIONS

→ A non-persistent connection is closed after the server sends the requested-object to the client.

→ In other words, the connection is used ~~for~~ exactly for one request and one response

→ For downloading multiple objects, multiple connections must be used

→ Suppose user Enters URL:
"http://www.someSchool.edu/someDept/home.index"

→ Assume above link contains text and references to 10 jpeg images.

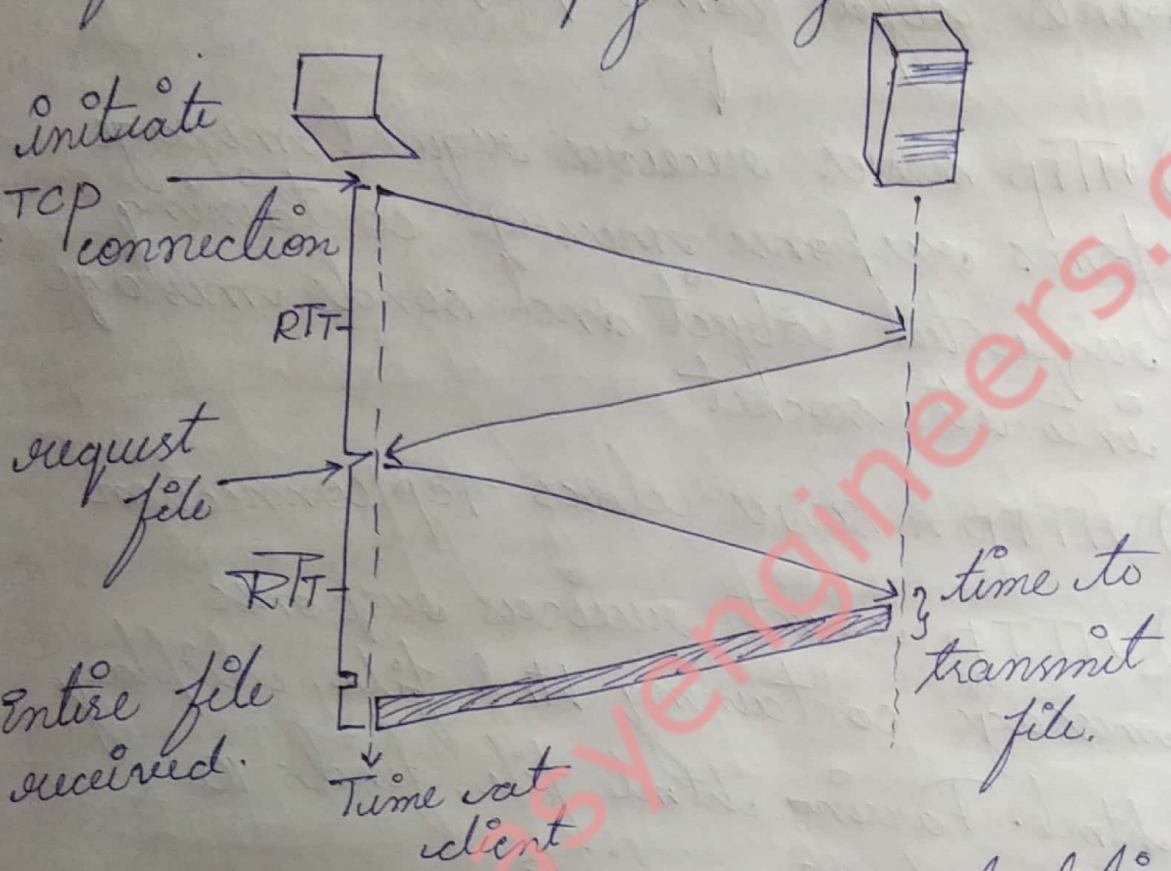


Fig 1.4: Back-of-the-Envelope calculation for the time needed to request and receive an HTML file.

→ Here is how it works:

1a) HTTP client initiates TCP connection to HTTP server (process) at www.someschool.edu on port 80

↓

1b.) HTTP server at host www.someschool.edu waiting for TCP connection at port 80. 'accepts' connection, notifying client

↓

2) HTTP client sends HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object someDept/home.index

3.) HTTP server receives request message forms response message containing requested object and sends message into its socket

4.) HTTP server closes TCP connection

5.) HTTP client receives response message containing html file, display html. Parsing html file, finds 10 referenced jpeg objects.

6.) Steps 1-5 repeated for each of 10 jpeg objects.

→ RTT is the time taken for a packet to travel from client to server and then back to the client.

→ The total response time is sum of following (fig 1.4):

(i) One RTT to initiate TCP connection (RTT → round trip time)

(ii) One RTT for HTTP request and first few bytes of HTTP response to return.

(iii) File transmission time.

i.e total response time = (i) + (ii) + (iii)

= 1RTT + 1RTT + file transmission time

= 2RTT + File transmission time.

1.2.2.2 HTTP WITH PERSISTENT CONNECTIONS

→ Problem with non-persistent connections:

1) A new connection must be Established and maintained for each requested object.
   → Hence, buffers must be allocated and state info must be kept in both the client and server.
   → This result in a significant burden on the server.
2) Each object suffers a delivery delay of 2RTTs:
   → (i) one RTT to Establish the TCP connection and
   → (ii) one RTT to request and receive an object.

→ Solution : Use persistent connections.
→ With persistent connections, the server leaves the TCP connections open after sending responses.
→ Hence, subsequent requests and response b/w same client and server can be sent over same connection.
→ The server closes the connections only when the connection is not used for a certain amount of time.
→ Default mode of HTTP : persistent connections with pipelining.
→ Advantages:
1) This method requires only one RTT for all the referenced - objects.
2) The performance is improved by 20%.

1.2.3 HTTp MESSAGE FORMAT.

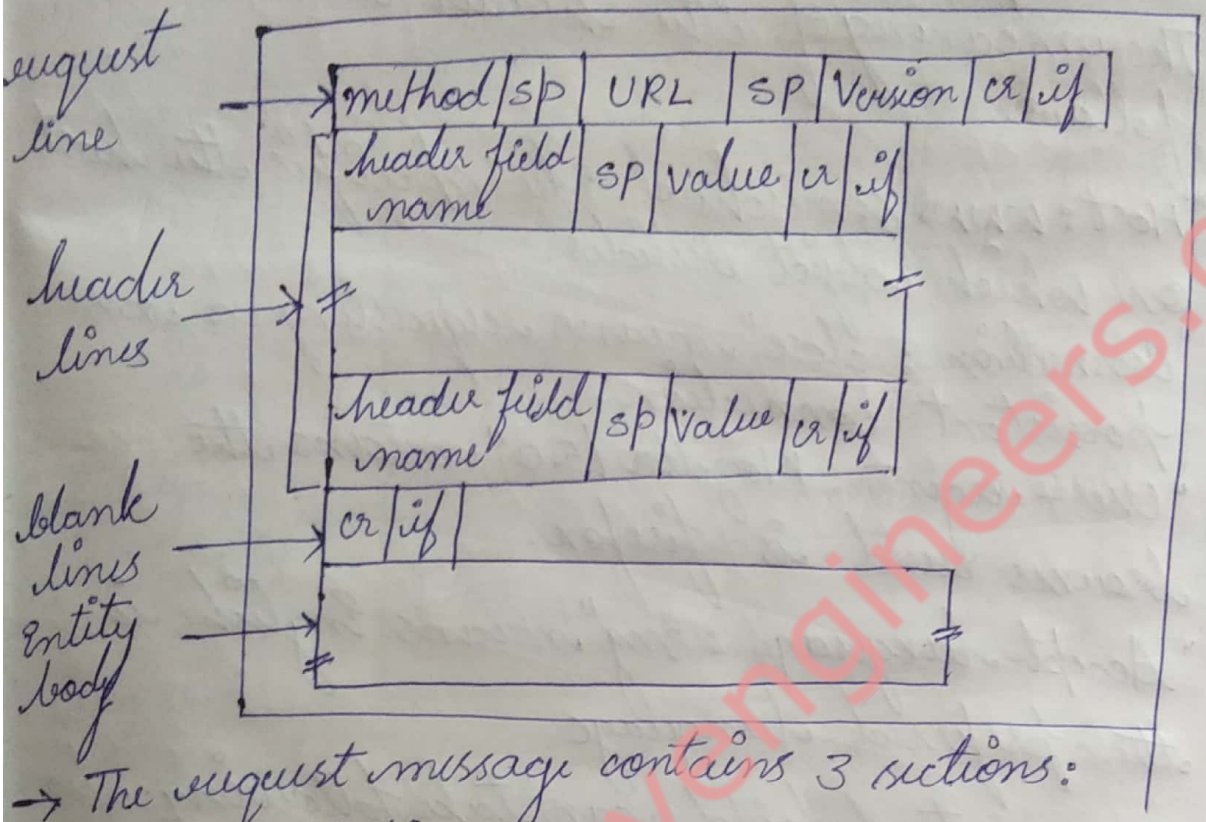
→ Two types of HTTp messages
1) Request message 2) Response messag

1.2.3.1 HTTp REQUEST MESSAGE

→ An example of request message is as follows:

```
GET /somedir/page.html HTTp/1.1
Host: www.someschool.edu
```

connection : close
User-agent : Mozilla/5.0
Accept-language : Eng



→ The request message contains 3 sections:
- Request line
- Header line
- carriage return

→ The first line of message is called the request line. The subsequent lines are called header lines

→ The request line contains 3 fields. The meaning of fields is as follows:

1) Method
"GET": This method is used when the browser requests an object from the server

2) URL
"/somedir/page.html": This is the object requested by browser.

3) Version
"HTTP/1.1": This version is used by browser.

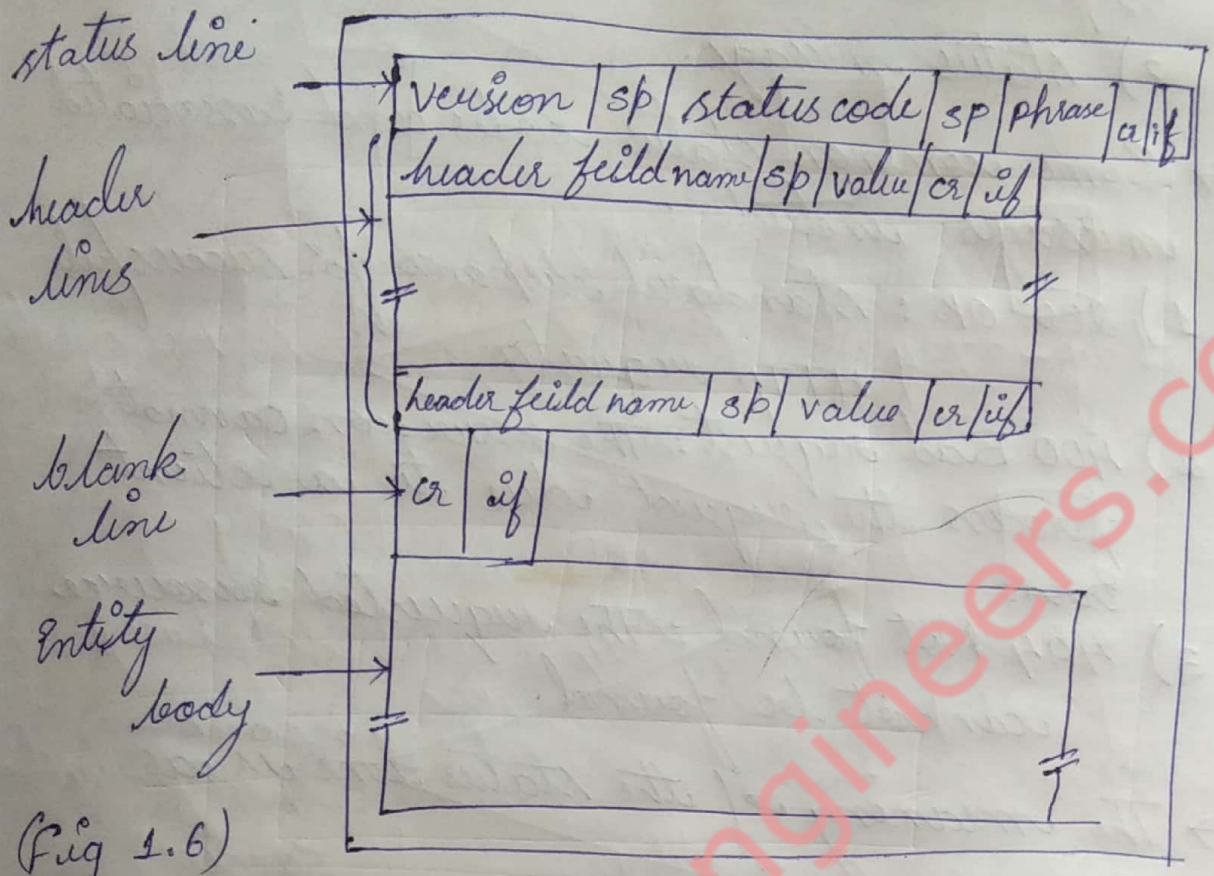
→ The request message contain 4 header line The meaning of the header lines is as follows.

1) "Host: www.someschool.edu" specifies the host on which object resides.
2) "Connection: close" means requesting a non persistent connection.
3) "User-agent: Mozilla/5.0" means the browser used is firefox.
4) "Accept-language: Eng" means English is the preferred language.

→ The method feild can take following values: GET, POST, HEAD, PUT, DELETE

1) GET: is used when the browser request an object from the server
2) POST is used when user fills out a forms and sends to server
3) HEAD is identical to GET Except the server must return as message-body in response
4) PUT is used to upload objects to servers
5) DELETE allows an application to delete an object on server.

1.2.3.2 HTTP Response Message



(Fig 1.6)

→ An example of response message is as follows

HTTP/1.1 200 OK
connection : close
Date : Tue, 09 Aug 2020 15:44:16 GMT.
Server : Apache/2.2.3 (CentOS)
Last-Modified : Tue, 09 Aug 2020 15:50:04 GMT
content-Length : 6821
content-Type : text/html.
(data data data data ---)

→ The response message contains 3 section (fig 1.6)
1) status line 2) Header line 3) Data (Entity body.

→ The status line contains 3 fields:
1) protocol version 2) status-code
3) status message.

→ Some common status codes and associated messages include:
1) 200 OK: standard response for successful HTTP requests
2) 400 Bad request: The server cannot process the request due to a client Error
3) 404 Not found: The requested resource cannot be found.

→ The meaning of the status line is as follows:
"HTTP/1.1 200 OK": This line indicates the server is using HTTP/1.1 and that Everything is OK.

→ The response-message contains 6 header-lines. The meaning of header-lines is as follows:
1) connection: This line indicates browser requesting a non-persistent connection
2) Date: This line indicates the time & date when the response was sent by server.
3) Server: This line indicates that the message was generated by an Apache

web-server.

4) Last-Modified: This line indicate the time and date when the object was last modified

5) content-length: This line indicate the no. of bytes sent in object.

6) content-type: This line indicates that the object in the Entity body in HTML text.

1.2.4 USER-SERVER INTERACTION: COOKIES.

→ Cookies refer to a small text file created by a web-site that is stored in user's computer.

→ Cookies are stored either temporarily for that session only or permanently on hard disk.

→ Cookies allow web-sites to keep track of users.

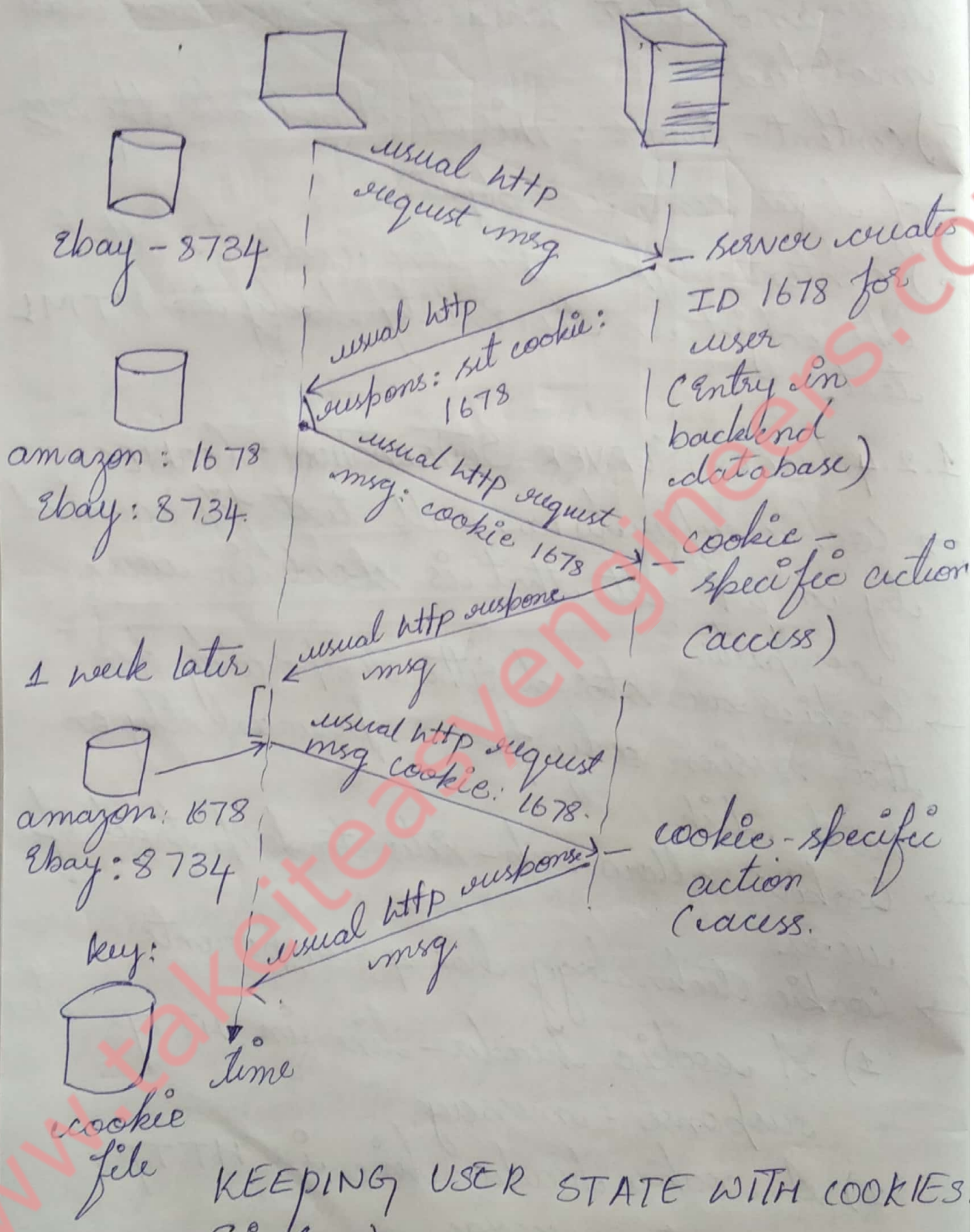
→ Cookie technology has 4 components

1) A cookie header-line in HTTP response-message

2) A cookie header-line in HTTP request-message

3) A cookie file kept on user's end system and managed by user's browser.

4) A back-end database at web-site



KEEPING USER STATE WITH COOKIES.
Fig (1.7)

→ Here is how it works (fig 1.7)

1) when a user first time visits a site, the server

→ creates a unique identification number (1678) and
→ creates an entry in its back-end database by identification number.

2) The server then responds to user's browser

3) → HTTP response includes Set-cookie header which contains the identification number into cookie-file.

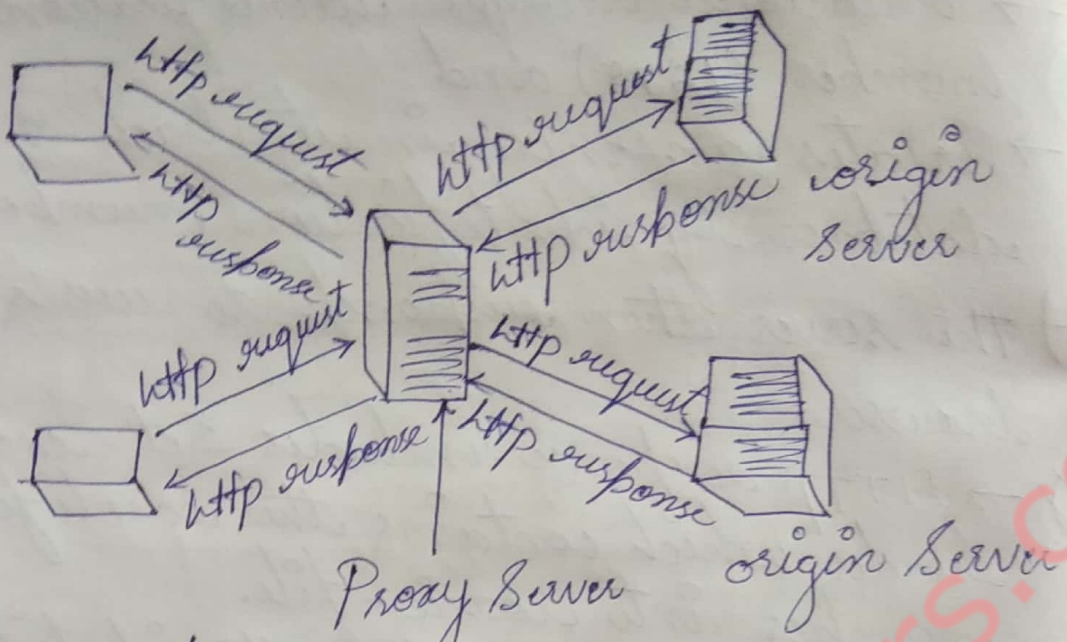
3) The browser then stores the identification number into cookie-file.

4) Each time the user requests a web page the browser
→ Extracts the identification number from the cookie file and
→ puts the identification number in the HTTP request.

5) In this manner, the server is able to track user's activity at the web site.

1.2.5 WEB CACHING

→ A web-cache is a network Entity that satisfies HTTP requests on the behalf of an orgenal web-server.
→ The web-cache has disk-storage
→ The disk-storage contains copies of recently requested-objects.



(Fig: 1.8) : CLIENTS REQUESTING OBJECTS THROUGH A WEB-CACHE (OR PROXY SERVER)

→ Here is how it works (fig 1.8)

1) The user's HTTP requests are first directed to the web-cache.
2) If the cache has the object requested the cache returns the requested-object to the client.
3) If the cache does not have the requested-object, then the cache
   → connects to the original server and
   → asks for the object.
4) When the cache receives the object, the cache
   → stores a copy of object in local storage
   → sends a copy of the object to the client.

→ A cache acts as both a server and a client at the same time

1) The cache acts as a server when the cache
   → receives requests from a browser.
   → sends response to the browser.
2) The cache acts as a client when the cache
   → requests to an original server and
   → receives responses from the origin server

→ Advantages of caching
   1) To reduce response time for client request
   2) To reduce traffic on a institution's access - link to internet.
   3) To reduce web traffic in internet.

1.2.6 THE CONDITIONAL GET

→ Conditional GET refers a mechanism that allows a cache to verify that the objects are up to date.

→ An HTTP request-message is called conditional GET if
   1) Request-message uses the GET method and
   2) Request-message includes an If-Modified-Since: header-line.

→ The following is an example of using conditional GET:

GET /fruit/kiwi.fig HTTP 1.1
Host: www.exoriquecuisine.com
If-Modified-since: wed, 7 sep 2020 9:23:24

→ The response is:

HTTP/1.1 304 Not Modified
Date: Sat, 15 oct 2011 15:39:29

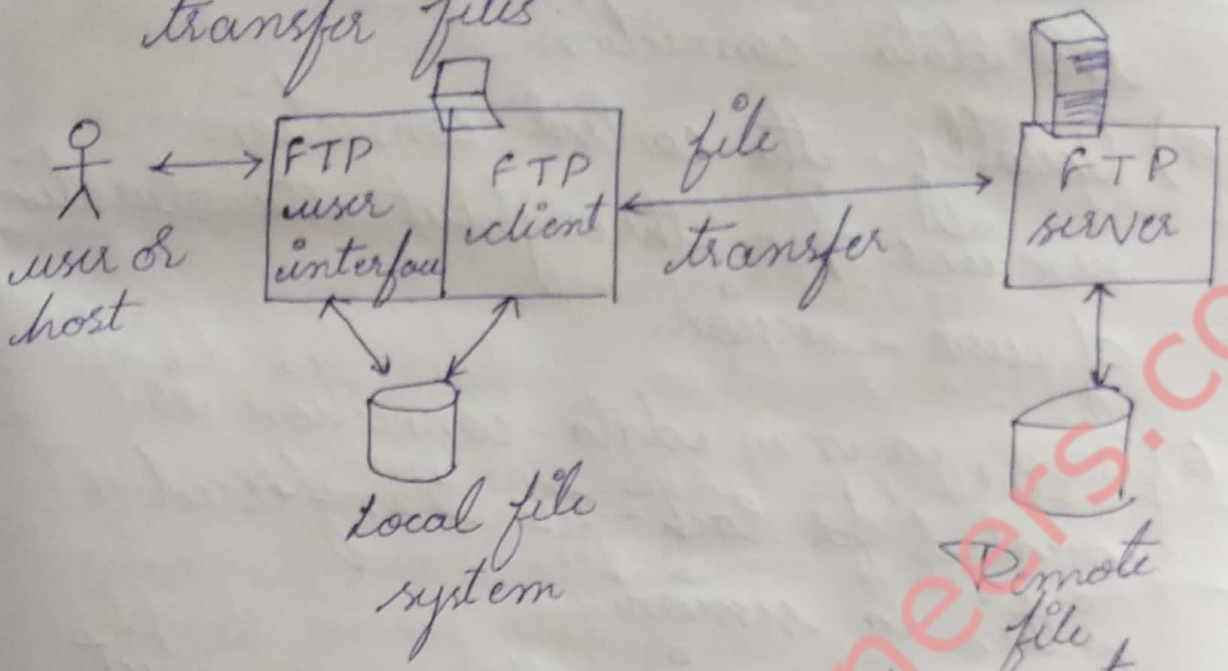
1.3 FILE TRANSFER: FTP

→ FTP is used by the local host to transfer files to or from a remote-host over the network.
→ FTP uses client-server architecture
→ FTP uses 2 parallel TCP connection

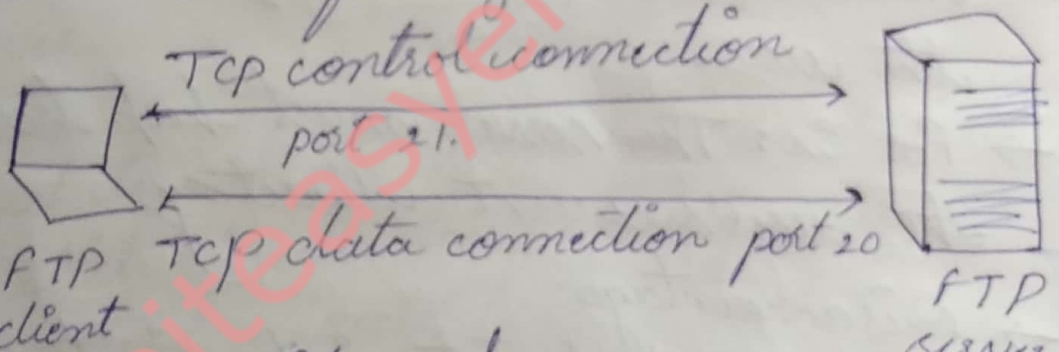
1) Control connection
→ The control-connection is used for sending control-information b/w local and remote-hosts.
→ The control-information includes:
  → User identification
  → password
  → commands to change directory
  → commands to put and get files.

2) Data Connection

→ The data connection is used to transfer files



FTP moves files b/w local and remote file system.



→ Here is how it works:

1) When session starts, the client initiates a control-connection with server on port 21

2) The client sends user identity and password over the control-connection

3) Then, the server initiates data connection to the client on port 20

4) FTP sends exactly one file over the data-connection and then closes the data-connection

5) Usually, the control-connection remains open throughout the duration of user-session.

6) But, a new data-connection is created for each-file transferred within a session.

→ During a session, the server must maintain the state-information about the user.

→ For Ex: The server must keep track of the user's current directory.

→ Disadvantage:
Keeping track of state-info limits the no. of sessions maintained simultaneously by a server.

1.3.1. FTP commands and replies

→ The commands are sent from client to server

→ The replies are sent from servent to client.

→ The command and replies are sent across the control connection in 7 bit ASCII format.

→ Each command ~~send~~ consist of 4 - uppercase ASCII characters followed by optional argument.

→ For Ex:

1) USER username
Used to send the user identification to the server

2) PASS password
Used to send the user password to the server.

3) LIST
Used to ask the server to send back a list of all the files in current remote directory.

4) RETR filename
Used to retrieve a file from the current directory of the remote host

5) STOR filename
Used to store a file into the current directory of remote - host.

→ Each reply consist of 3-digit numbers followed by optional message.

→ for Ex:

1) 331 Username OK, password required

2) 125 Data connection already open.

1.4 ELECTRONIC MAIL IN THE INTERNET.

→ E-mail is an asynchronous communi-cation medium in which people send and read messages.

→ E-mail is fast, easy to distribute and inexpensive.

→ e-mail has feature such as
- → messages with attachments
- → hyperlinks
- → HTML-formatted text
- → Embedded photos.

→ Three major components of an E-mail system (fig 1.11)

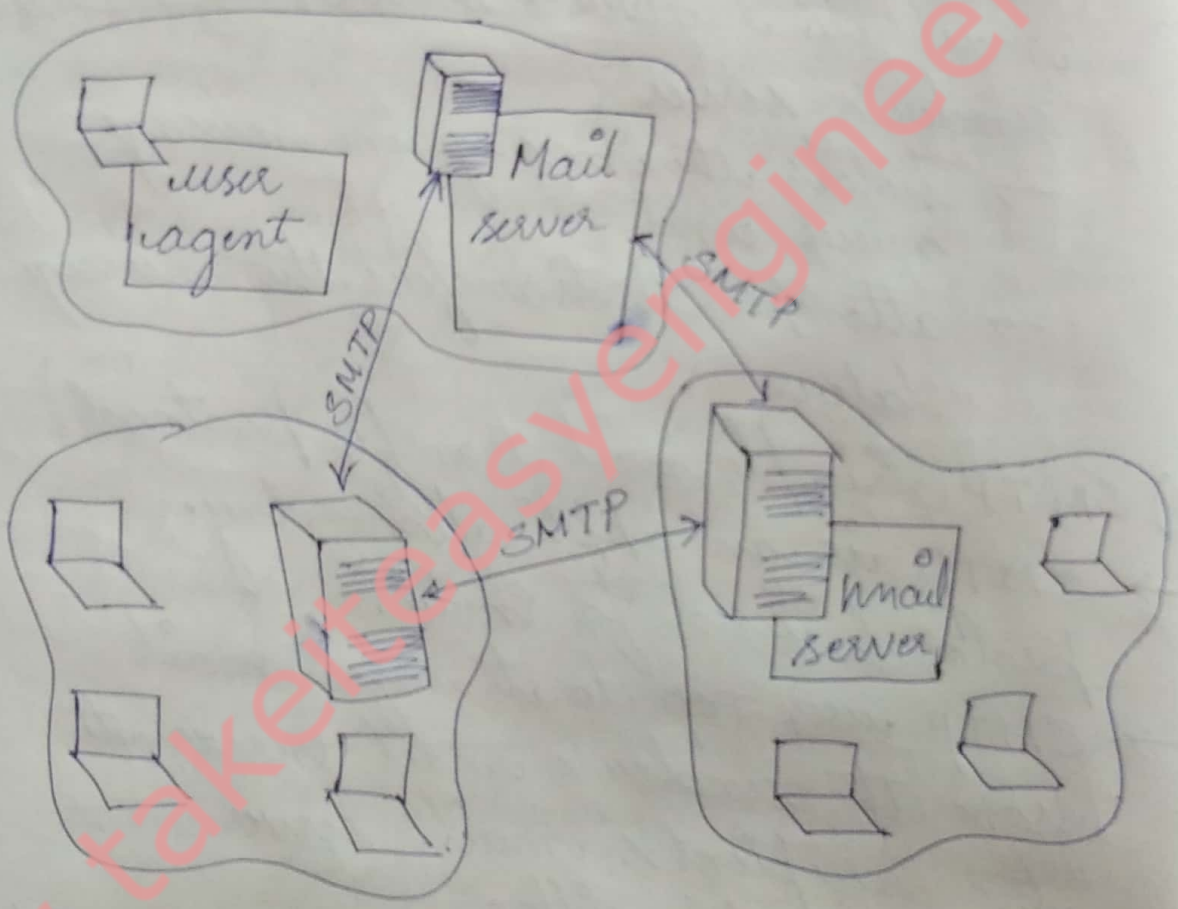
1) User agents
- → User-agents allow users to read, reply to, forward, save and compose messages
- → For Ex: Microsoft Outlook and Apple Mail.

2) Mail Servers
→ Mail-servers contain mailbox for users.
→ A message is first sent to the sender's mail server.
→ Then, the sender's mail-server sends the message to the receiver's mail server.
→ If the sender's server cannot deliver mail to receiver's server, the sender's server
→ holds the message in message queue and
→ attempts to transfer the message later.

3) SMTP (Simple mail transfer protocol)
→ SMTP is an application-layer protocol used for email.
→ SMTP uses TCP to transfer mail from the sender's mail-server to the recipient's mail-server
→ SMTP has two sides:
1) A client-side, which Executes on the sender's mail-server
2) A Server-side, which Executes on the recipient's mail-server.

→ Both the client and server-sides of SMTP run on every mail-server.

→ When a mail-server receives mail from other mail-servers, the mail-server acts as as server, when a mail server sends mail to other mail-server, the mail server acts as a client.



1.4.1 SMTP

→ SMTP is the most important protocol of Email System

→ 3 characteristics of SMTP (that differs from other application)

1) Message body uses 7-bit ASCII code only
2) Normally, no intermediate mail-servers used for sending mail.
3) Mail transmission across multiple network through mail relaying.

→ Here is how it works:

1) Usually, mail servers are listening at port 25.
2) The sending server initiates a TCP connection to the receiving mail-server
3) If the receiver's server is down, the sending server will try later.
4) If connection is established, the client and the server perform application layer handshaking.
5) Then, the client indicates the E-mail address of the sender and the recipient.
6) Finally, the client sends the message to the server over the same TCP connection.

1.4.2 COMPARISION OF SMTP WITH HTTP

1) HTTP is mainly a pull protocol. This ~~is~~ is because,

→ someone loads information on a web server and
→ users use HTTP to pull the information from the server.

On the other hand, SMTP is primarily a push protocol, This because
→ the sending mail server pushes the file to receiving mail server.

2.) SMTP requires each message to be in 7 bit ASCII format.
- If message contain binary data, the message has to be encoded into 7 bit ASCII format.
- HTTP does not have this restrictions.

3) HTTP encapsulates each object of message in its own response - message.
- SMTP places all of the message's objects into one message.

2.4.3 MAIL ACCESS PROTOCOLS

→ It is not realistic to run the mail servers on PC & laptop. This is because
→ mail - servers must be always on and
→ mail - servers must have fixed IP addresses.

→ Problem : How a person can access the email using pc or laptop?
solution : Use mail access protocols.
→ Three mail access protocols :
1) post office protocol (POP)
2) Internet Mail access protocol (IMAP)
3) HTTP.

2.4.3.1 POP

→ POP is an extremely simple mail access protocol.
→ pop server will listen at port 110
→ Here is how it works:
• The user agent at client's computer opens a TCP connection to the main server.
• pop then progresses through 3 parts:
1) Authentication:
• The user-agent sends user name and password to authenticate the user.
2) Transaction
• The user-agent retrieves messages.
• Also, user-agent can
→ mark messages for deletion
→ remove deletion marks &
→ obtain mail statistics.

- The user agent issue commands, and the server responds to each command with a reply
- There are 2 responses:
  (i) +OK: used by server to indicate that the previous command was fine.
  (ii) –ERR: used by server to indicate that something is wrong.

3) Update:
After user issues a quit command, the mail-server removes all messages marked for deletion.

Disadvantages:
The user cannot manage the mail at remote mail server, for Ex- user cannot delete messages.

1.4.3.2 IMAP

→ IMAP is another mail access protocol, which has more features than POP.
→ An IMAP server will associate each message with a folder.
→ When a message first arrives at server, the message is associated with recipient's INBOX # folder.

→ Then, the recipient can
  → move the message into a new, user-created folder.
  → read the message
  → delete the message and
  → search remote folder for messages matching specific criteria.
→ An IMAP server maintains user state-information across IMAP session.
→ IMAP permits a user-agent to obtain components of messages
  for Ex: a user-agent can obtain just the message header of a message.

1.4.3.3 WEB-BASED E-MAIL

→ HTTPs are now used for web-based e-mail accessing.
→ The user-agent is an ordinary web browser.
→ The user communicates with its remote-server via HTTP
→ Now, web server emails are provided by many companies including Google, Yahoo Etc.

1.5 DNS

→ DNS is an internet service that translates domain-names into IP addresses.
For Ex: the domain name "www.google.com" might translate to IP address "198.105.232.4".

→ Because domain names are alphabetic they are easier to remember for human being.

→ But, the internet is really based on IP addresses (DNS - Domain Name System)

1.5.1 SERVICES PROVIDED BY DNS

→ The DNS is
1) A distributed database implemented in a hierarchy of DNS servers.
2) An Application layer protocol that allows hosts to query the distributed database

→ DNS servers are often UNIX machines running BIND software

→ The DNS protocol runs over UDP and uses port 53 (BIND → Berkely Internet Name Domain)

→ Assume a browser requests the URL www.someschool.edu/index.html
→ Next, the user's host must first obtain the IP address of www.someschool.edu
→ This is done as follows
1.) The same user machine runs the client-side of DNS application
2.) The browser
→ Extracts the hostname "www.someschool.edu" from URL and
→ passes the hostname to the client side of DNS application...
3.) The client sends a query containing the hostname of DNS server.
4.) The client Eventually receives a reply, which includes the IP address for the hostname
5.) After receiving the IP address, the browser can initiate a TCP connection to HTTP server
→ DNS also provides following Services.
1.) Host Aliasing
A host with a complicated host name can have one or more alias name
2.)

2) Mail Server Aliasing:
For obvious reason, it is highly desirable that Email addresses be mnemonics.

3) Load distribution
→ DNS is also to perform load distribution among replicated servers.
→ Busy sites are replicated over multiple servers & each server runs on a different System.

1.5.2 OVERVIEW OF HOW DNS WORKS

→ Distributed database design is more preferred over centralized ~~des~~ design because

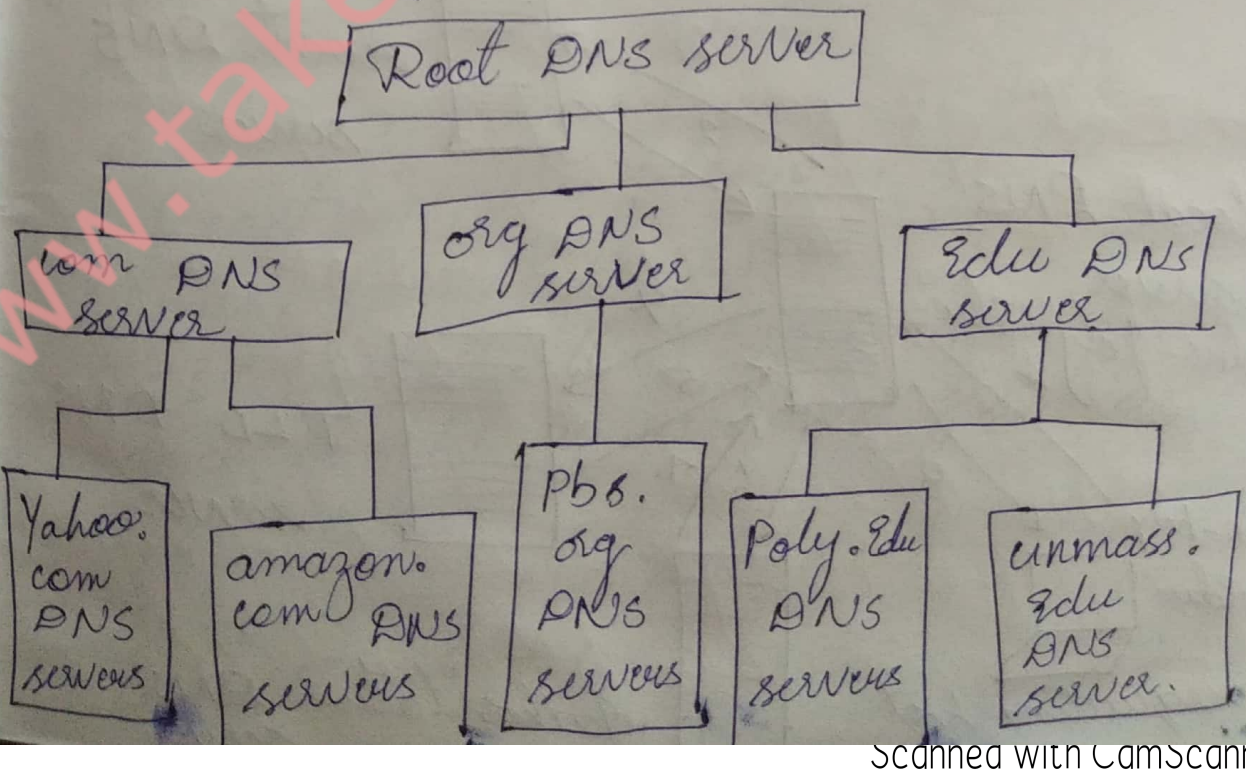
1) A single Point of failure
If the DNS server crashes then the entire Internet will not stop.

2) Traffic Volume
A single DNS Server cannot handle the huge global DNS traffic.
But with distributed system, the traffic is distributed and reduces overload on Server.

3) Distant Centralized Database.
A single DNS server cannot be "close to" all the querying clients.
If we put the single DNS server in Mysore, then all queries from USA must travel to the other side of the globe.
This can lead to significant delays.

4) Maintenance
The single DNS server would have to keep records for all Internet hosts
This centralized database has to be updated frequently to account for every new host.

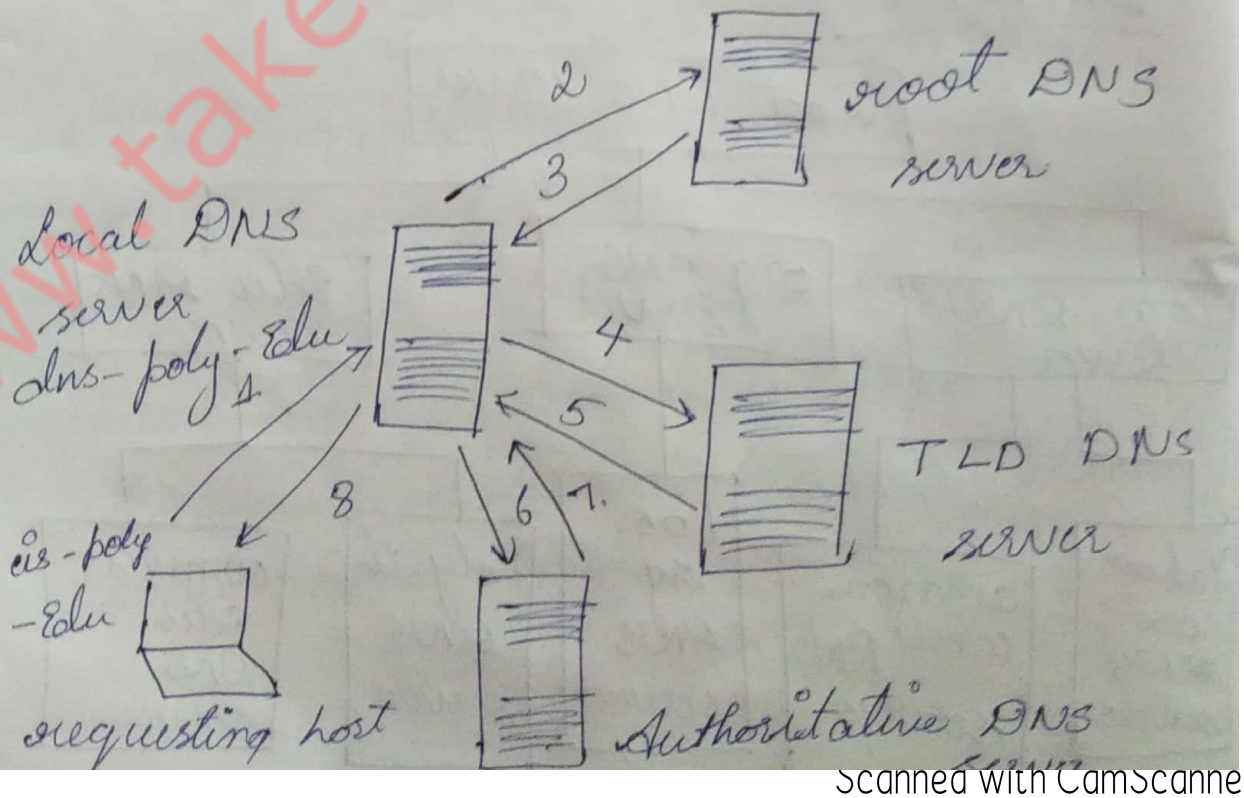
1.5.2.1 A DISTRIBUTED, HIERARCHICAL DATABASE



→ Suppose a client wants to determine IP address "www.amazon.com".

1) The client first contacts one of the root servers, which returns IP address for TLD servers
2) Then, the client contacts one of these TLD servers
3) The TLD server returns the IP address of an authoritative - server for "amazon.com"
4) Finally, the client contacts one of the authoritative - server for amazon.com
5) The authoritative - server returns the IP address for hostname "www.amazon.com".

1.5.2.1.1 RECURSIVE QUERIES & ITERATIVE QUERIES.



→ The Example shown in figure makes use of both recursive and iterative queries.
→ The query 1 sent from cis-poly-edu to dns-poly-edu is recursively query. This is because, the query asks dns-poly-edu to obtain the mapping on its behalf.
→ But the subsequent three queries 2, 4, 6 are iterative. This is because, all replies are directly returned to dns-poly-edu

1.5.3 DNS-RECORDS AND MESSAGES.

→ The DNS server stores resource-records (RRs)
→ RRs provide hostname-to-IP address mappings.
→ Each DNS reply message carries one or more resource-records
→ A resource-record is a 4-tuple that contains the following feilds:
(Name, Value, Type, TTL)
→ TTL (Time to live) determines when a resource should be removed from a cache.
→ The meaning of Name and value depend on type.

1) If Type-A, then name is a host name and value is the IP address for the hostname.

2) If Type-NS then
   (i) Name is a domain (such as foo.com) and
   (ii) Value is the hostname of an authoritative DNS server.

3) If Type = CNAME, then value is a canonical hostname for the alias hostname Name.

4) MX records allow the hostnames of mail-servers to have simple aliases.

1.5.3.1 DNS MESSAGES

→ Two types of DNS messages: 1) query
2) reply.

→ Both query and reply messages have the same format.

→ The various fields in a DNS message are as follows (fig 1.14).

1) Header section

→ The first 12 bytes is the header session

→ This section has following fields:

(i) Identification
→ This field identifies the query.
→ This identifier is copied into the reply messages to a query.
→ This identifier allows the client to match received replies with sent queries.

(ii) Flag
→ This field has following 3 flag bit.
a) Query / Reply.
This flag-bit indicates whether the message is a query (0) or a reply (1).
b) Authoritative
This flag bit is set on a reply message when a DNS server is an authoritative-server
c) Recursion Desired.
This flag-bit is set when a client desires that the DNS server perform recursion.

(iii) Four Number-of-feilds
These feilds indicate the no. of occurences of 4 types of data section that follow the header.

2) Question Section
→ This section contains information about the query that is being made.
→ This section has following feilds:
(i) Name
This feild contains the domain name that is being queried.
(ii) Type
This feild indicates the type of question being asked about the domain - name
3) Answer Section
→ This section contains a reply from a DNS server
→ This section contains the resource-records for the name that was originally queried.
→ A reply can return multiple RRs in the answer, since a hostname can have multiple IP addresses
4) Authority Section
→ This section contains records of other authoritative - servers
5) Additional Section
→ This section contain other helpful records.

Identification	Flags	12 bytes
Number of Questions	Number of answer RRs	
Number of authority RRs	Number of additional RRs	
Questions (Variable number of Questions)		- Name, type fields for a query
Answer (Variable number of Answers)		- RRs in response to query
Authority (Variable number of RRs)		- records for authoritative servers
Additional information (Variable number of RRs)		- Additional "helpful" info that may be used.

1.6 PEER-TO-PEER APPLICATIONS

~~→ peer-to-peer~~

→ P2P architecture is different from client-server architecture

→ In P2P each node (called peers) acts as a client and server at the same time

→ The peers are not owned by a service provider

→ The peers not supposed to be always listening on the internet

→ The peers are dynamic, i.e, some peers will join some peers will leave from time to time.

1.b.1 P2P FILE DISTRIBUTION

→ One popular p2p file distribution protocol is BitTorrent.

→ Consider the following scenario: Suppose a server has a large file and 'N' computers want to downloa -d the file (fig. 1.15)

1) In client - server architecture, Each of the N computers will
   → connect to server
   → download a copy of the file to local host

2) In p2p architecture, a peer need not necessarily download a copy from the server
   → Rather, the peer may download from other peers.

→ Let us define the following
   Length of file = f
   Upload rate of for the server = u_s
   Upload rate for ith computer = u_i

Download rate for ith computer = d_i
Distribution - time for the client - server architecture = D_{cs}
Server must transmit = N_F bits.

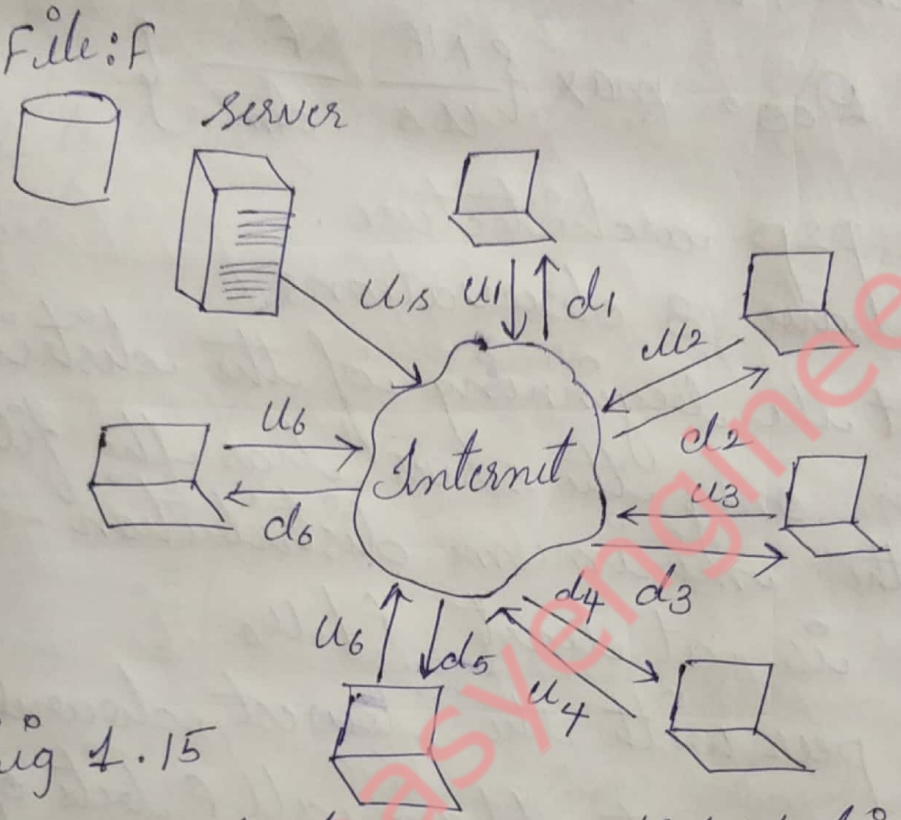


Fig 1.15

An illustrative file distribution prblm

Case 1 : client - Server architecture

→ we have a observation

1) The server must transmit one copy of the file to Each of N peers

→ Since the server's upload rate is u_s the distribution - time is at least N_F / u_s

2) The peer with the lowest download rate cannot obtain all F - bits of the

file is less than F/dmin.

→ Thus the minimum distribution time is at least F/dmin.

→ putting above 2 observations together, we have

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

case 2: P2P architecture.

→ we have 3 observations

1) At the beginning of the distribute - only the server has the file. So the minimum distribution time is at least F/u_s.

2) The peer with the lowest download rate cannot obtain all F bits of the file in less than F/dmin. Thus, the minimum distribution time is at least F/dmin

3) The total upload capacity of the system as a whole is $u_{total} = u_s + u_1 + u_2 \cdots + u_N$

The system must deliver F bits to each of N peers

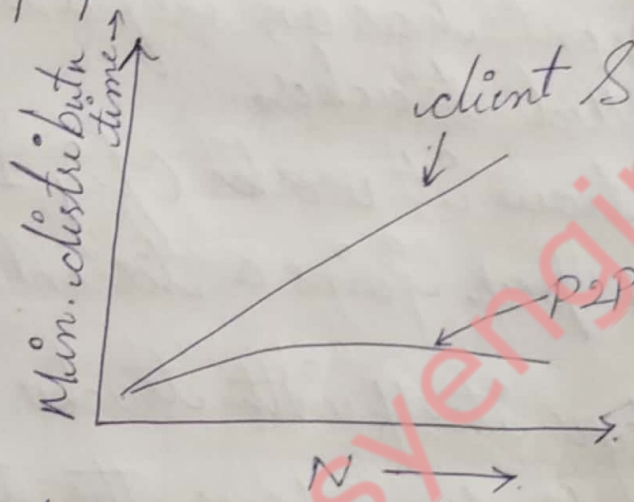
Thus, the min distribution time

is at least $NF/(u_s + u_1 + u_2 \cdots + u_N)$.

→ putting above 3 observations together we have

$$D_{p2p} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

→ Fig 1.16 compares the minimum distribution-time for the client-server and p2p architecture.



→ The above comparision shows that when N is large

(i) p2p architecture is consuming less distribution time

(ii) p2p architecture is self-scaling.

1.6.1.1 BIT TORRENT

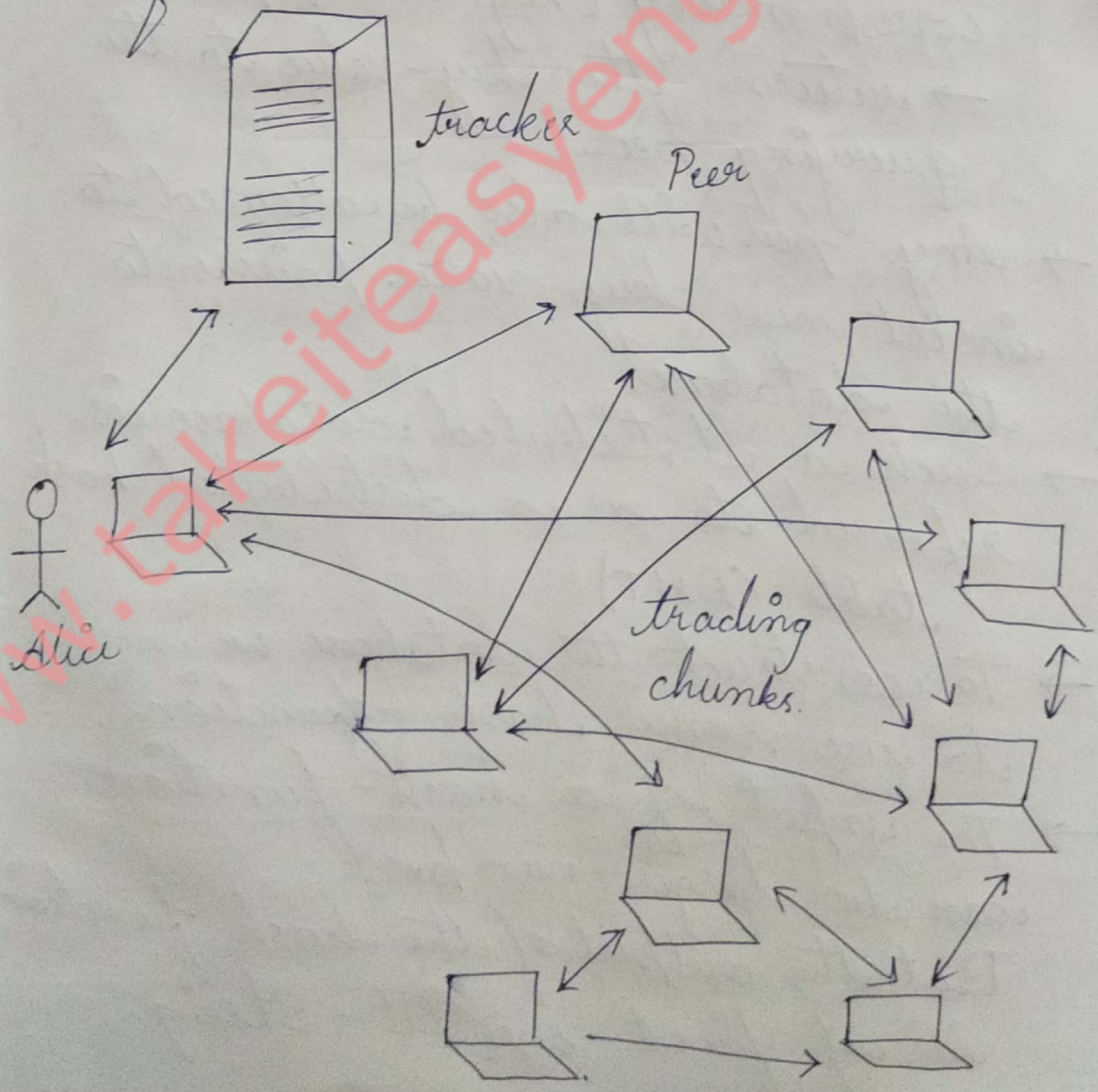
→ The collection of all peers participating in the distribution of a particular file is called a torrent.

→ peers download Equal size chunks of the file from one another. chunk size = 256 KBytes.
→ The peer also uploads chunks to other peers.
→ Once a peer has acquired the entire file, the peer may leave the torrent or remain in torrent.
→ Each torrent has an infrastructure node called tracker.
→ Here is how it works (Fig 1.17)
1) when a peer joins a torrent, the peer
→ registers itself with tracker and
→ periodically informs the tracker that it is in the torrent.
2) when a new peer joins the torrent, the tracker.
→ randomly selects a subset of peers from the set of participating peers and
→ sends the IP addresses of these peers to the new peer.
3) Then, the new peer tries to Establish concurrent TCP connections with all peers on this list.

4) periodically, the new peer will ask each of the neighbouring peers for the set of chunks.

→ To choose the chunks to download, the peer uses a technique called rarest-first.

→ Main idea of rarest-first:
  → Determine the chunks that are the rarest among the neighbors and
  → Request then those rarest chunks first.



1.6.2 DISTRIBUTED HASH TABLE

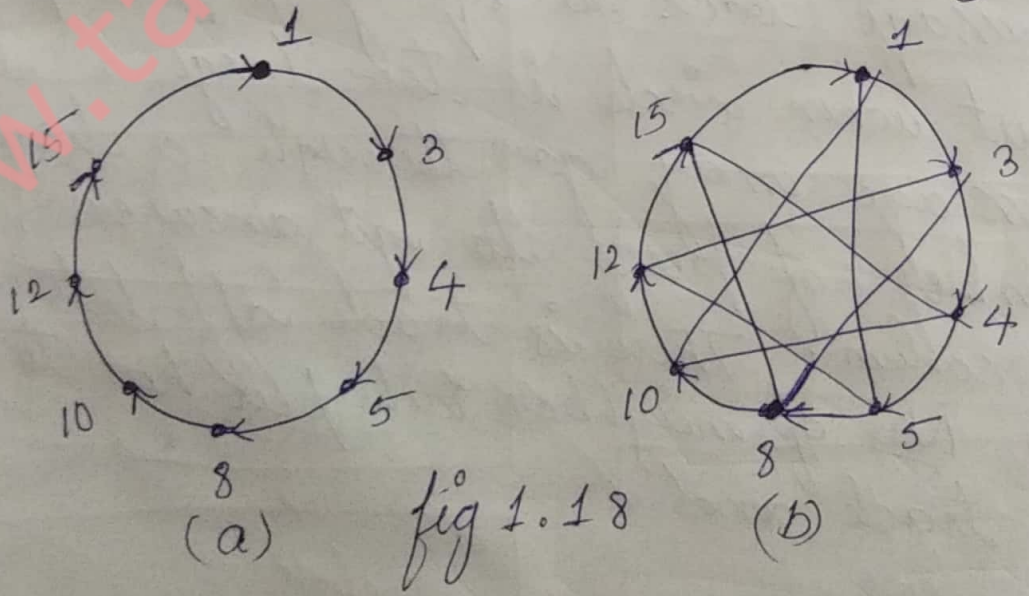
→ we can use p2p architecture to form a distributed database.
→ we consider a simple database, which contains (key, value) pairs.
→ Each peer will only hold a small subset of data.
→ Any peer can query the distributed database with a particular key.
→ Then, the database will.
  → locate the peers that have the corresponding (key, value) pairs and
  → return the key-value to the querying peer.
→ Any peer will also be allowed to insert new key-value pairs into the database.
→ Such a distributed database is referred to as a distributed hash table (DHT)
→ To construct the database, we need to use some hash-function
→ The input of a hash-function can be larger number.
  But the output of the hash function is of fixed-size bit-String.

→ The outline of building a DHT is as follows:

1) Assign an identifier to each peer where the identifier is an n-bit string.
So we can view the identifier as an integer at the range from 0 to $2^n - 1$.

2) For a data pair the hash value of the key is computed.
Then the data is stored in the peer whose identifier is closet to the key.

3) To insert or retrieve data, first we need to find the appropriate peer.
Problem: It is not realistic to let the peer to store all of the other peer's identifiers.
solution: Use a circular arrangement.



(a) fig 1.18 (b)

1.6.2.1 CIRCULAR ARRANGEMENT.

→ In this Example, the identifiers are range (0, 15) (4-bit strings) and there are 8 peers.

→ Each peer is only aware of its immediate successor and predecessor (fig 1.18 a)

→ So, Each peer just needs to track 2 neighbors

→ When a peer asks for a key, the peer sends the message clockwise around the circle

→ For Example:

- The peer 4 sends a message saying "who is responsible for key 11?"
- The message will forward through peers 5, 8, 10 and reach peer 12.
- The peer 12 determines that it is the closest peer to key 11
- At this point, peer 12 sends a message back to querying peer 4.

→ But when circle is too large, a message may go through a large number of peers to get answer.

→ problem? There is trade off between

(i) No. of neighbors each peer has to track and

(ii) No. of message to be sent to resolve a single query.

→ Solution, to reduce the query time, add "shortcuts" to the circular arrangement. (fig 1.18b)

1.7 SOCKET PROGRAMMING : CREATING NETWORK APPLICATION

→ Two types of network-application

1) First type is an implementation whose operation is specified in a protocol standard (RFC). Such an application is referred to as "open".
~~2)~~ The client and server programs must conform to the rule dictated by the RFC.

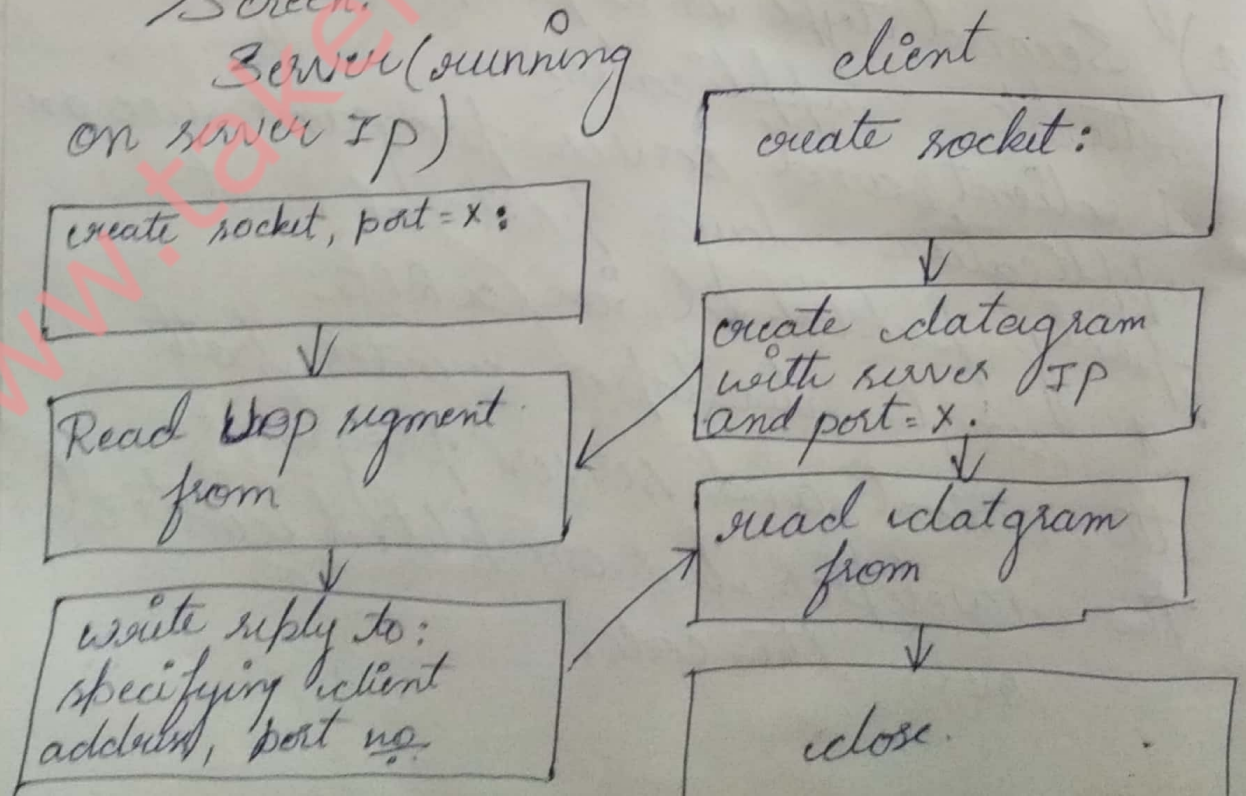
2) Second type is a proprietary network-application
A client and server programs use an application-layer protocol not openly published in a RFC.
A single developer creates both the client and server programs
The developer has completed control over the code.

→ During development phase, the developer must decide whether the application uses TCP or UDP.

1.7.1 SOCKET PROGRAMMING WITH UDP

→ Consider client-server application in which following Events occurs:

1) The client
   - reads a line of characters (data) from keyboard and
   - sends the data to Server.
2) The server receives the data and converts the characters to uppercase.
3) The server sends modified data to client
4) The client receives the modified data and displays the line on Screen.



1.7.2 Socket Programming with TCP

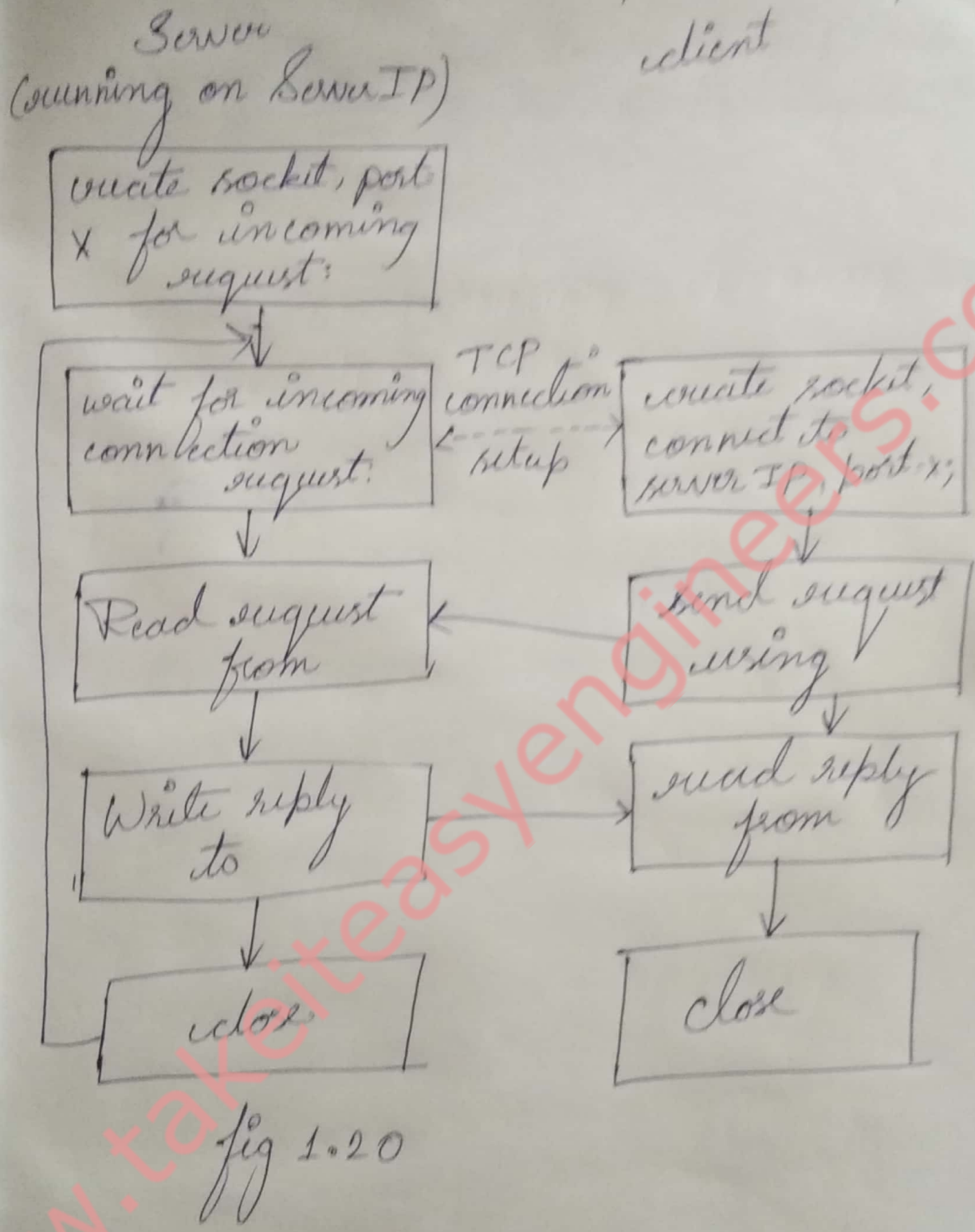


fig 1.20