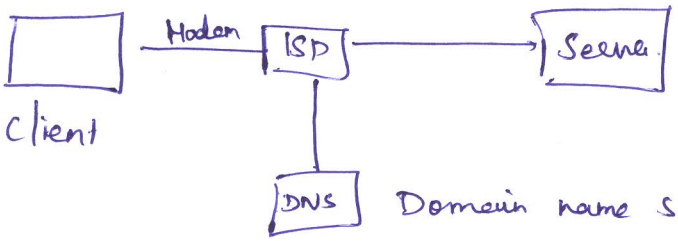
Computer Networks

Module - 1

Application Layer.

Principles of Nlw Applications



Client - Pgmg in
Seerver - different languages.

Nlw App' Architecture

i) Client / Seerver

ii) Peer - Peer → eg: Uploading (Seerver) Downloading (client)
→ not secure.

Processes Communicating

i) Process → client / Seerver process
↓
initiates comm'

ii) Socket → Interface blw App' & Trans' Layer.

iii) Addressing → Host → IP add' Process → Port no'
Phy' add' → 6 bytes
Logical add' → 4 bytes
Port no' → 2 " [eg: SMTP - 25].

Transport Services available to Internet Applications

i) Reliability. → Less Tolerent App'. (no adjust).

ii) Throughput → no' of bits/sec. Bandwidth Constrained app'/
Elastic app'.

iii) Timing → Delay 100 msec.

iv) Security → Encryption / Decryption → Confidentiality
is also supported. ← Authentication, Integrity

Transport Services provided by the Internet

Specific → UDP & TCP.

TCP → Connection oriented Service & reliable data transfer service, Congestion ctrl.

TCP / UDP doesn't do encryption / decryption

So SSL (Secure Socket Layer) libraries are added

TCP enhanced with SSL. Enhancements implemented in 'app' Layer.

App Layer → SSL Socket (encrypt) – TCP Socket (Transport Layer).

UDP → lightweight, connectionless.

No congestion ctrl mechanism.

Services Not provided by Internet Transport Protocols

Throughput & Timing.

E-mail, Remote Terminal Access, Web, File Transfer → TCP

Internet Telephony → UDP. [To stop cong' ctrl of TCP & tolerate loss]

But if Firewall blocks UDP then TCP should be in backup.

Application Layer Protocols

Protocols define

i) The types of msg exchanged (req', res')

ii) Syntax of msg (fields)

iii) Semantics (meaning of the inf' in the field)

iv) Rules for determining when & how a process sends msgs & responds to msgs.

App' Layer protocol is only a one piece of a n/w app'.
eg Web App' -> HTML, Browser, Server & App' Layer protocol
HTTP → defines format & sequence of msgs enchanged b/w client & server.

The Web & HTTP

Until Early (Before) 1990s → Internet is used in Universities → Researchers, academics → sharing files, e-mails, remote access.

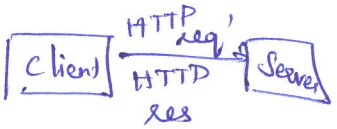
Early 1990. → World wide Web → Browsing. on-demand unless like radio & TV.

Overview of HTTP

Client/Server communicates by HTTP protocol ie by HTTP msgs.

Web Page: → consists of objects → simple file (eg: HTML file, JPEG image, video clip) + addressable by a single URL.

URL: https://www.cmrit.ac.in/ise/picture.gif
host name — path name.



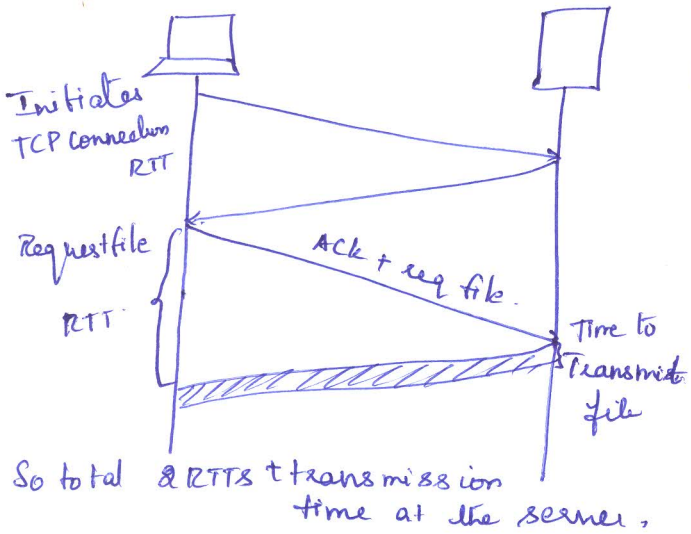
HTTP uses TCP as its transport protocol.
TCP will take care of reliable data transfer.
HTTP → is a stateless protocol ie. The server won't remember Previous client req'.

Non-persistent & Persistent Connections

Non-persistent

i). HTTP client sends HTTP req' through client socket by TCP connection
ii) HTTP server receives the req' & send backs the response via socket
iii) TCP connection will be closed.

For one HTTP req/res' one TCP connection will be there (for one object)



So total 2 RTTs + transmission time at the server.

RTT (Round Trip Time). → Time blw req's & res!

HTTP with Persistent Connection

Single TCP connection (connection is open) for all req' from (a single) client. same.

HTTP Message Format

HTTP Request Msg

Method URL Version → Request Line

GET /somedir/page.html HTTP/1.1

header field name: value ← Host: www.cmrit.ac.in

Connection: close → if persistent also.

User-agent: Mozilla/5.0

Accept-language: fr → french. else default.

} Header lines

Blank line

Entity body → empty only for POST it will be there.

Method: GET, POST, PUT, HEAD, DELETE → Delete file from server

i/p in URL → in body → same as GET but no i/p in URL.

→ publishing, uploading

GET, POST, HEAD → requesting a web page from server.

HTTP Response Message.

Version Status code phrase. → Status line

HTTP/1.1 200 ok.

header field name: value ← Connection: close

Date: Mon, 06 Aug 2017 15:44:04 GMT

Server: Apache/2.2.3 (CentOS) → server name.

Last-Modified: Tue, 02 Aug 2017 15:11:03 GMT

Content-Length: 6821 → bytes in response object

Content-Type: text/html. → html file.

(Cdata data data data data ...)

Blank Line

Entity body

} Header lines

when the response is object creation / created / modified date

200 ok - Request success, info' returned.

301 Moved Permanently - The object moved. New URL specified in Location: header of the response msg. Client will automatically get the new URL.

400 Bad Req' - Server didn't understand the req'.

404 Not Found - Document doesn't exist in the server

505 HTTP Version Not Supported: eg: HTTP 1.0. that's y updation.

Cookies: User-Server Interaction

HTTP → Stateless.

But in shopping sites - Shopping cart is there,

Address, Phone no' → once registered is there.

→ Achieved by Cookie.

Components:

1) A Cookie header line in the HTTP response msg.

2) " in " request "

3) A Cookie file kept on the user's end s/m & managed by the user's browser.

4) A back-end DB at the web-site.

eg: Amazon [Business Analytics/ Intelligence]

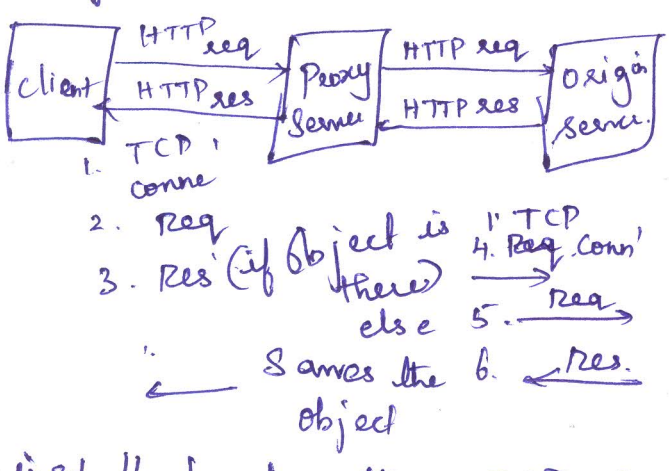
If a user visits then the server will create a unique id & send Set-Cookie: 1678 to the browser.

So the user's activity will be monitored.

If the user registers then Address will be updated for all purchases.

Web Caching → also called a proxy server — is a n/w entity that satisfies HTTP requests on behalf of an origin web server.

Proxy server / web cache → will have its own disk - storage.



Browser need to be configured to send the req' to the proxy.

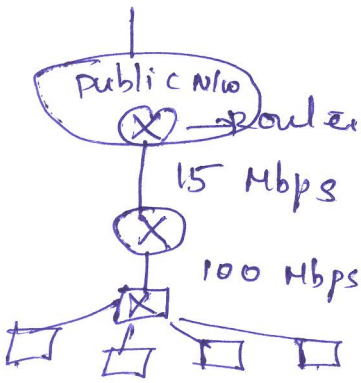
The cache acts both as client / server.

Web Cache is purchased & installed by the ISP. eg: Universities can have Cache.

Reasons:

i) Fast Response Time. ii) Reduce Traffic.

eg: Ins' need not provide high bandwidth as the req' objects are provided by the cache.



Average object size - 1 Mbits
" Request rate from insitiution's browser to server - 15/s.
Internet delay → Req & Res (2 s).
from Router to reach the server.

Req & Res the object. ← Response time - LAN delay + Access delay (Clw routers) + Internet delay.

Traffic Intensity. on the LAN.

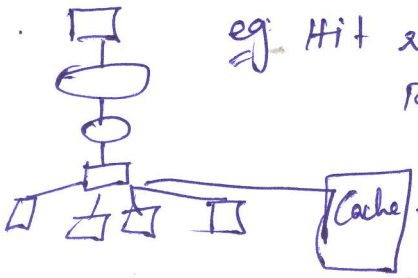
(15 req/s) . (1 Mbits/req)/(100 Mbps) = 0.15 → Tens of milliseconds of delay.

Traffic Intensity on the Access link

(15 req/s) . (1 Mbits/req)/(15 Mbps) = 1 → Very large.

Solution:

i) 15 Mbps to 100 Mbps but it is costly.

ii). 

eg Hit rate 0.4. → 40% req' in 10 milliseconds by cache
Remaining 60% to the server.

Traffic Intensity - 0.6.
" less than 0.8 → Small. Less delay. tens of millisec.

This delay is negligible compared with 2 s Internet delay.

Average delay. 0.4 . (0.01 s) + 0.6 . (2.01 s). → slightly greater than 1.2 s.

Cost is low. → Many caches use public-domain slw.

Through Content Distribution Nlws (CDNs) → Web caches (Google) are playing an important role in the Internet.

The Conditional GET.

Issue with Cache: If the object in the server get modified, it won't be reflected in the cache.

To check whether object is updated or not conditional GET is used.

eg: After Req getting req' from the browser.

The cache will send Conditional GET message.

Req: GET /cmrit/ise/pic.gif HTTP 1.1

Host: www.cmrit.ac.in

If-modified-since: wed, 7 Aug 17 09:23:24

Res:

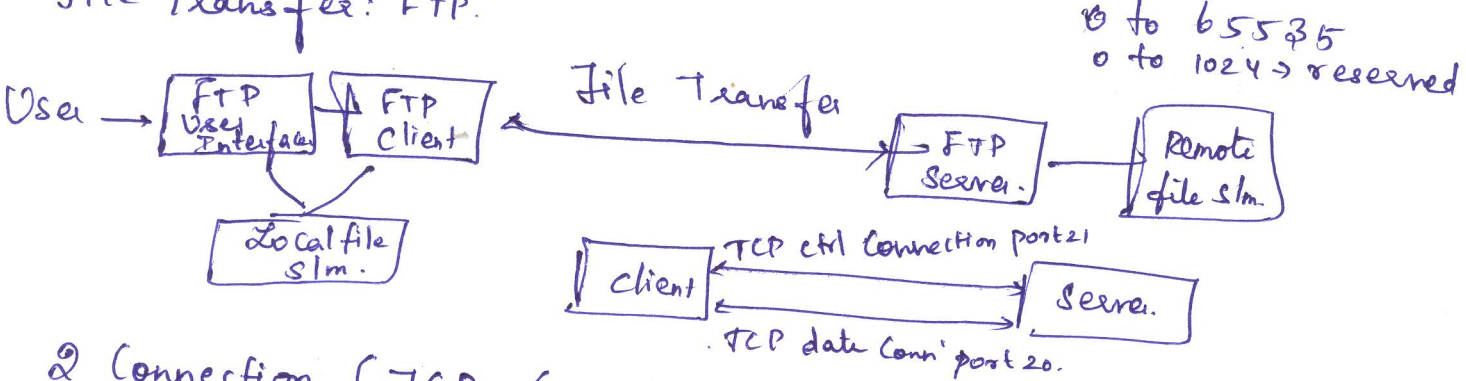
HTTP 1.1 304 : Not Modified

Date: Sat, 15 Oct 16 15:39:29

Server: Apache 11.3.0 (Linux)

empty entity body.

File Transfer: FTP.



2 Connection (TCP Connection)

1. Ctrl Connection - (user identification, Pwd, Cmds to change remote directory, " put & get files).
2. Data Connection - Actually send a file.

FTP - send its ctrl inf' out-of-band.

But HTTP & SMTP - in-band.

i) Client side of FTP initiates a ctrl TCP connection with the server side on server port no' 21. (by sending ID & Pwd)

ii) Server side initiates Data Connection to the client side

iii) FTP client sends exactly one file over the conn' & closes the conn'. (20)

Ctrl Conn' - remain open until comm'.

Data Conn' - non-persistent.

FTP server must maintain state about the user. So that can restrict the no' of session. (Current directory).

FTP Commands & Replies

Request

USER username eg.

PASS password

List → To ask the server to send back list of files.

RETR filename → get particular file

* STOR filename - put a file.

All req' & res' msg in ctrl connection.

Reply msg: eg:

331 Username ok, pwd required → optional

125 Data conn' already open; transfer starting

425 Can't open data connection

452 Error writing file.

Electronic Mail in the Internet

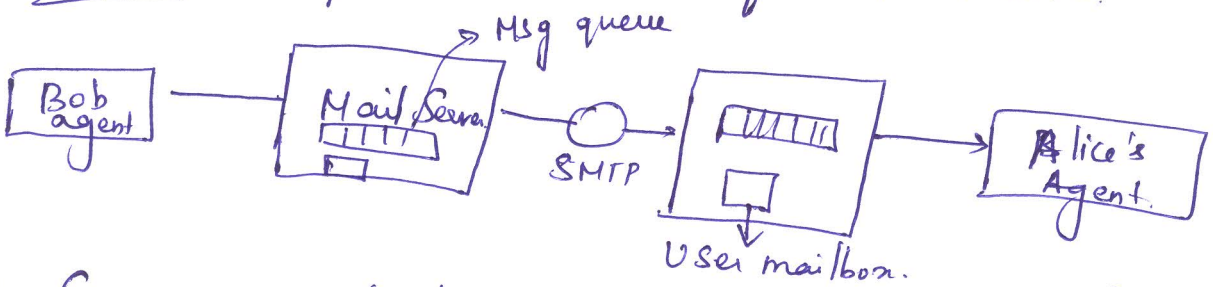
SMTP uses TCP.

SMTP in Sender side: Client SMTP & client

Receive side: SMTP Server.

Msg queue → attempts to sent msg every 30 min else will notify & delete

SMTP (Simple Mail Transfer Protocol)



1. Compose mail by user agent.
2. In msg queue of Mail Server.
3. Mail Server (Bob) → TCP Conn' with Mail Server (Alice).
4. Send mail to Alice web Mail Server.
5. Places mail in Alice's Mailbox.
6. Can read using Agent.

SMTP - 25 port no'.

Intermediate server is not used.

Step 3: Handshaking. The SMTP client indicates e-mail addl' of both sender & receiver.

The client can use the same connection to send other msgs else can also close the connection (Persistent connections)

Msg should be in 7-bit ASCII format.

eg: SMTP Client (c), Server (s).

client host name. gmail.com. Receiving host name: Cmrit.ac.in

S: 220 cmrit.ac.in

C: HELO gmail.com

S: 250 Hello cmrit.ac.in, pleased to meet u.

C: MAIL FROM: < bob@gmail.com >

S: 250 bob@cmrit.ac.in ... Sender ok

C: RCPT TO: < alice@cmrit.ac.in >

S: 250 alice@cmrit.ac.in ... Receipient ok

C: DATA. S: 250 Msg accepted.

220 - Service ready
250 - Requested mail action ok, Completed
354 - Start Mail i/p. end with <CRLF>.<CRLF>
221 → Service closing transmission channel.

S: 354 Enter mail, end with "." on a line by itself
C: Do you like Noodles?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 cmrit.ac.in closing connection

CR - Carriage return
LF - Line feed.

Comparison with HTTP

Uniqueness:

i) SMTP → Transfer files (mail) b/w 2 mail servers
ii) HTTP → " (object) " client & server.

Both use persistent connections.

HTTP	SMTP
i) Pull Protocol: Info' in server, whenever client want can get from server.	Push Protocol: The mail is delivered receipient's mail server.
ii) No restriction on the msg.	Should be in 7-bit ASCII
iii) All data (text, images) are handled as objects.	Seperately.

Mail Message formats

From: bob@google.com
To: alice@cmrit.ac.in
Subject: Computer Network: A Top Down Approach → Optional.

Message header

Message Body.

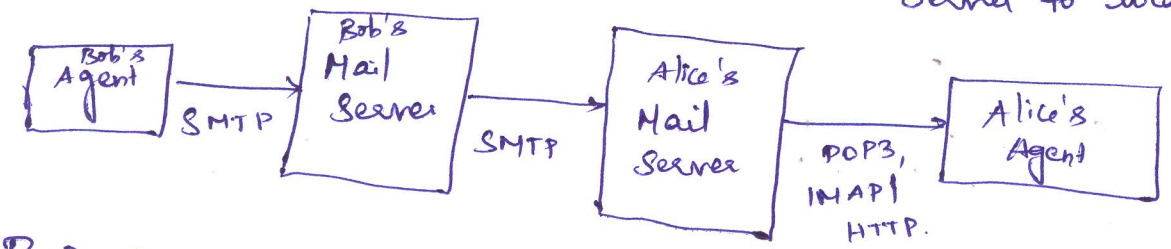
Mail Access Protocols.

If the sender's mail server sends a mail to the receiver's mail server, the receiver's mail server should be always in online. i.e. If Local PC or Mobile phone is used then it can't be always online to receive mails.

In this case, the mail server is considered to be in the PC/ mobile itself. If the receiver is offline then the sender's mail server will try in every 30 min & will delete. So a shared mail server is used that is maintained by ISP.

Once the Mobile/PC comes online it can't request mail from (its mail server) because it is push Protocol (SMTP).

So, to overcome this issue mail access protocols like POP3, IMAP & HTTP are used. [To access mail from Alice's Mail Server to local PC/Mobile]



POP3 (Post Office Protocol version 3)

SMTP is used to send mail from sender's mail server to receiver's mail server. POP3 is used to transfer mail from recipient's receiver's mail server to recipient's user agent. [Port no: 110]

3 Phases:

i) Authorization: User agent sends user name & password.

ii) Transaction: " retrieves msg, mark msgs for deletion, Unmark,..

iii) Update: The deleted msg for deletion will get deleted.

Authorization Phase commands & responses.

+ok POP3 server ready
user alice
+ok → else -ERR.
pass chi
+ok user successfully logged on.

+ok & -ERR- responses.
user uname } - Command.
pass pwd

Transaction Phase → i) Download & delete mode.
ii) " keep mode.

C- User agent S: mail server.

```
C: list          → size of stored msgs.
S: 1 498
S: 2 912
S: .
C: retr 1        → reading the msg.
S: ( CN CN ...
S: ..... CN)
S: .
C: dele 1        → deleting the msg.
C: quit
S: +Ok POP3 server signing off.
```

Problem with download & delete mode is that if a mail is deleted in PC then it can't be viewed in laptop, Mobile... But it is possible in download & keep mode.

POP3 maintain some state info' about who have marked delete but not maintain across POP3 sessions.

IMAP (Internet Mail Access Protocol) [Port no: 143].

In POP3 the mail is copied to the local machine. The mail is stored in the local machine. Without Internet connection also it can be accessed. Useful for users using single machine to access the mail.

In IMAP the email can be configured in many machines. The mail is not downloaded rather it is accessed remotely from the server. Once the header is loaded if the user wants read the msg then it is (loaded) read by from the server. (i.e downloaded from the server).

In a e-mail only part of mail eg: image, text file, video only text file can be downloaded. Internet connection is needed.

Changes made in local machine will be reflected in the server.

Web-Based E-Mail

Accessing e-mail through web browsers. The user agent in a web browser (communicates with) the remote mailbox by HTTP. (access msg from) (i.e mailserver) rather than POP3 or IMAP protocol. eg: Hot Mail

DNS - The Internet's Directory Service.

As humans are identified by their name & adhar no'. host are (is) identified by their it's ip address & domain name.

eg: 121.7.106.8 => 4 bytes range from 0 to 255.

Processed from left to right.

Services provided by DNS Port no: 53

Human will identify host by Domain name, whereas routers will identify host by IP address. Inorder, to convert domain name to IP add' Domain Name System (DNS) has been introduced.

The DNS is 1. A distributed DB implemented in a hierarchy of DNS servers.
2. An app'-layer protocol that allows hosts to query the distributed DB.

Runs over UDP. Employed by HTTP, SMTP & FTP to translate hostnames (user-supplied) to IP add'.

Steps:

1. The user machine runs the client side of the DNS app'.
2. The browser extracts the host name www.cmrit.ac.in from the URL & passes it to the client side of the DNS app'.
3. The DNS client sends a query containing the hostname to a DNS server.
4. The DNS client receives the corresponding IP address.
5. Once the browser receives the IP add', it initiates TCP connection to the HTTP server process located at port 80 at that IP add'.

DNS Services

Host aliasing

Complicated hostname can have one or alias name.

eg: Canonical (original/full) hostname: relay1.west-coast.enterprise.com.

Alise name: enterprise.com or www.enterprise.com.

DNS will be invoked by an app' to obtain the canonical hostname & IP add' from the alias name.

Mail Server aliasing

The mail'd should be mnemonic (easy to remember)

eg: bob@enterprise.com → is alias name but the canonical name of the mail server is relay1.west-coast.enterprise.com

DNS is used to obtain the canonical mail server name & IP add'.

Load distribution

↳ Load dis' among replicated web servers can be achieved by DNS. For replicated web servers different IP add' will be there but only one canonical host name is associated.

When the client sends a DNS query then the DNS server will send the entire list of IP add' but will rotate the order within each reply.

Because a client typically sends its HTTP req' msg to the IP add' that is listed first in the set, DNS rotation distributes the traffic among the replicated servers.

Working of DNS

If a browser needs to translate a host name into IP add' it invokes the DNS client & the client will send the query to the DNS server. This process is very simple. But the complexity is with distributed DNS servers.

To overcome this complexity single centralized server having all mapping but it won't suits to today's Internet.

Problems with centralized design:

i) A single point of failure: If the server fails.?

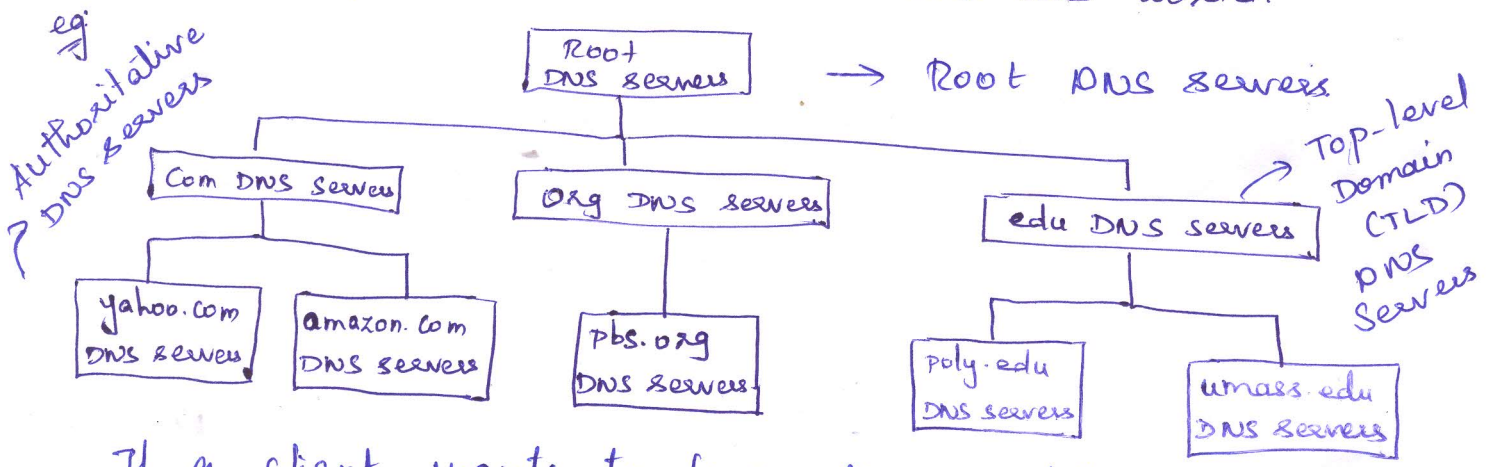
ii) Traffic volume: All queries to a single server.

iii) Distant centralized DB: If the server is in NewYork then it is far away from India.

iv) Maintenance: Have to be updated for every new host.

A Distributed, Hierarchical DB.

DNS uses a large no' of servers organized in a hierarchical fashion & distributed around the world.



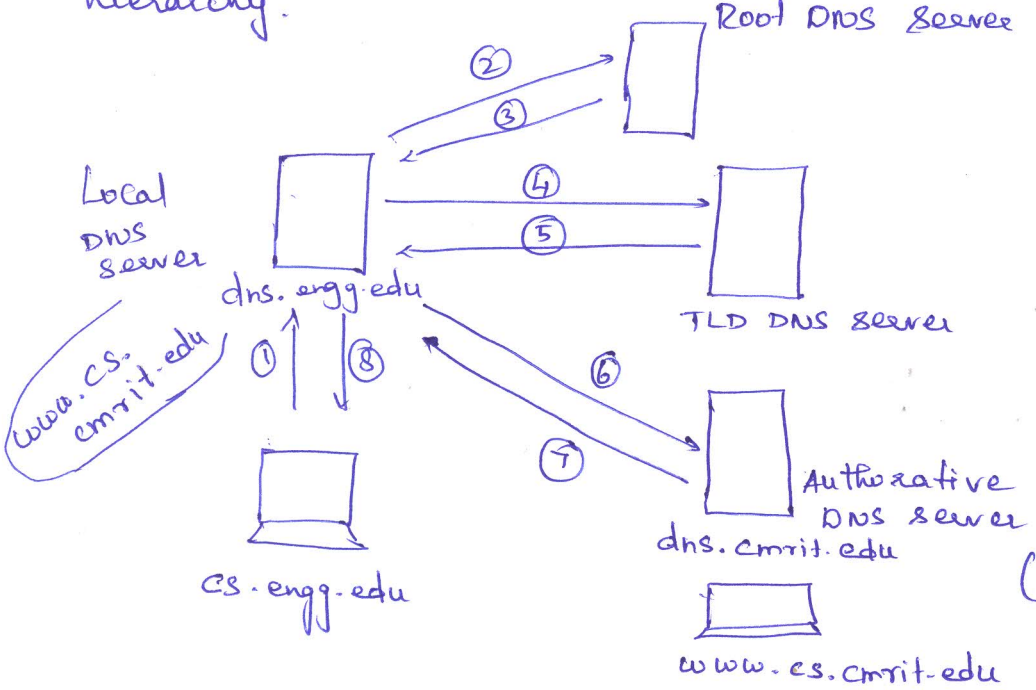
If a client wants to know the IP add' of www.amazon.com It first contacts the Root server which returns IP for the TLD servers for the top-level domain com. Then the client contact TLD server, it gives IP add' of the authoritative server for amazon.com (amazon.com) Then the client contacts auth' server, which gives/returns IP add' for the hostname www.amazon.com.

i) Root DNS servers: 13 root servers, replicated as 24/7.

ii) Top-level domain (TLD) servers: responsible for com, org, edu, gov & country top-level domains like in, uk, fr, ...

iii) Authoritative DNS servers: Every org' with web & mail servers must have auth' DNS servers that houses DNS records. An auth' org' can have its own auth' server else the org' can pay to have these records to some service providers.

Local DNS server → doesn't belongs to the hierarchy. It acts as proxy, for forwarding the query into the DNS server hierarchy.



The query from requesting host to local DNS server is Recursive query, others are iterative query (as reply is coming to itself)

(for cs again it goes for one more server)

DNS Caching

8 messages have been enchanged to get the IP add', inorder to avoid this DNS caching is used.

The local DNS server will acts as proxy such that if it gets the reply, then it will store it in th its m/y (2 days), if it receives the same request then it won't query the root server, rather will reply back soon.

DNS Records & Messages

The DNS servers stores resource records (RRs).

A resource record is a 4-tuple.

Fields: (Name, Value, Type, TTL).

- Name ↓ hostname
- Value ↓ IP address (for Type A)
- TTL ↓ Time to Live (determines when a record should be removed from a cache)

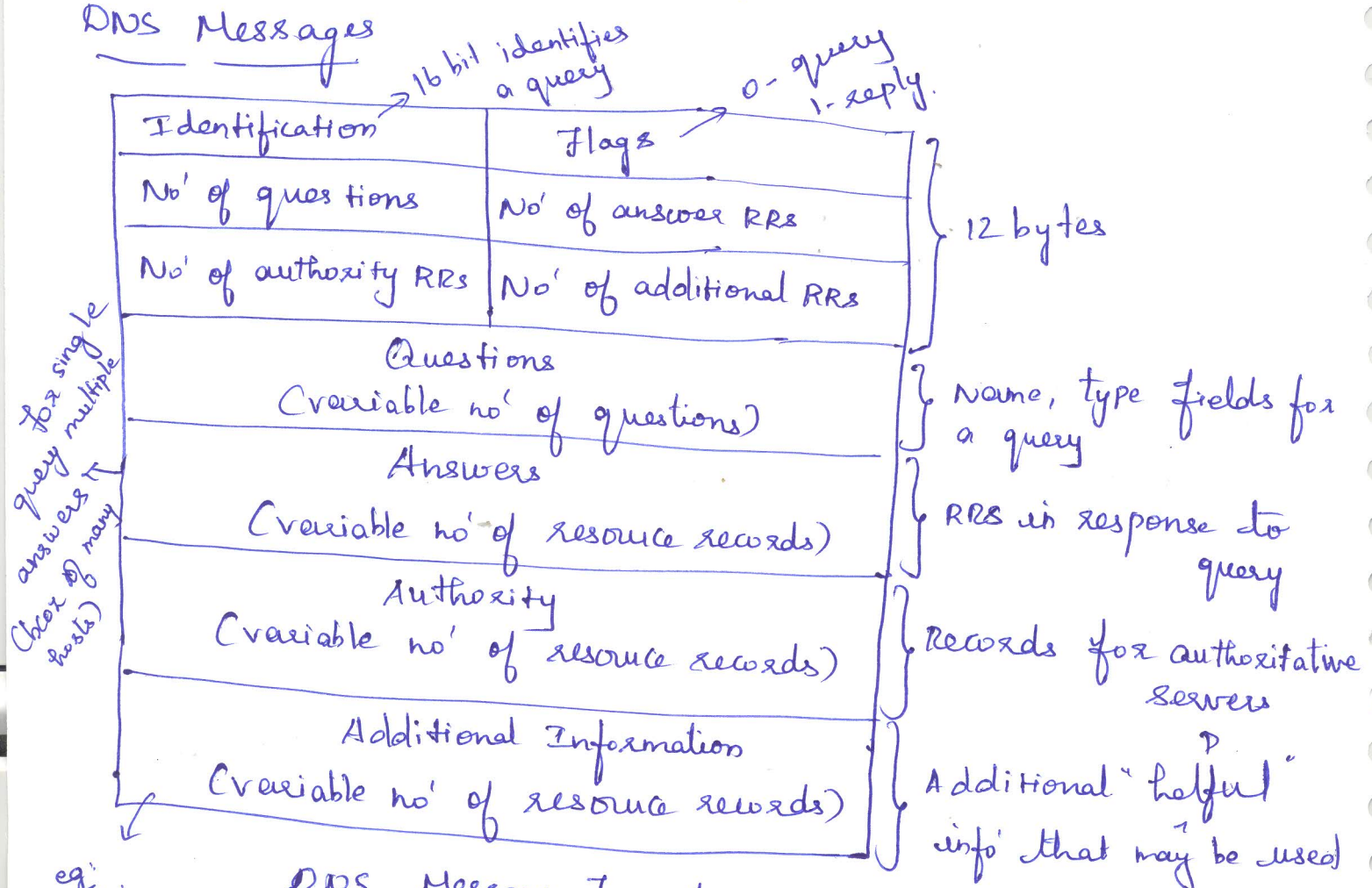
i) Type = A → provides std hostname to IP add' mapping.

ii) Type = NS · Name → Host name Value = host name of an authoritative DNS. (knows IP add' of the host)

iii) Type = CNAME → Value = canonical hostname.

iv) Type = MX → Canonical name of a mail server.

DNS Messages



eg: DNS Message Format

If in questions if aliase name of the host is given then in the reply (TYPE = CNAME) Canonical name will be send. The add' info' will give the corresponding IP address.

Inserting Records into the DNS DB.

→ If a new site is being created then the domain name cmrit.ac.in at a registrar. A registrar is a commercial entity that verifies the uniqueness of the domain name, enters the domain name into the DNS DB & collects or small fee for the services provided.

→ IP address & host names of Primary & Secondary authoritative servers need to be provided.

eg:

(cmrit.com, ise.cmrit.com, NS)

(ise.cmrit.com, 212.212.212.1, A)

→ UPDATE option has been added to the DNS protocol to allow data to be dynamically added / deleted from the DB via DNS msgs.

Peer-to-Peer Applications

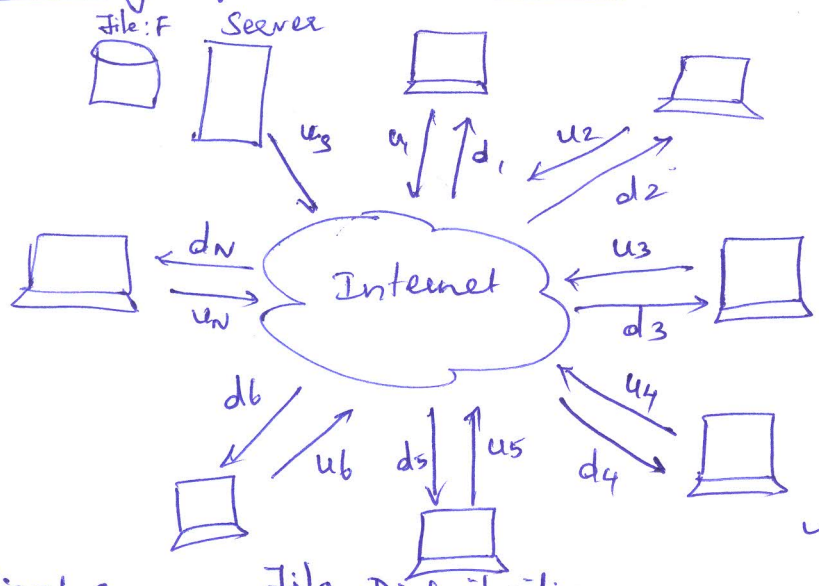
The app' employed like Web, e-mail & DNS - all employ client-server architectures with significant reliance on always-on infrastructure servers.

In P2P arch' there is no need for this always-on servers arch' instead 2 hosts / peers, communicate directly with each other The peers are not owned by a service provider i.e. simply laptops, desktops controlled by users.

P2P File Distribution

Any Peer can re-distribute any portion of the file it has received to any peers, thereby assisting the server in the distribution process.

Scalability of P2P Architectures



Download rate - d_i
Upload rate - u_i
Size of the file - F
No' of peers - N
Distribution time → time takes to get a copy of the file to all N peers

i) Client server arc' File - Distribution

Distribution time - D_{cs}

→ The server must transmit one copy of the file to N peers. Thus NF bits need to be transmitted. The time to distribute the file must be at least NF/u_s.

→ Let d_{min} denote the lowest download rate, that is $d_{min} = \min\{d_1, d_2, \ldots d_N\}$. Thus the min' distribution time is at least F/d_{min}

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

Actual distribution time $D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$

If the no' of peers (N) increases then the distribution time will also increases.

ii) P2P archi'

Calculation of distribution time is complex, since the distribution time depends on how much each peer distributes portions of the file to the other peers.

→ At the beginning of the distribution, only the server has the file. So, this file should be sent atleast once. Thus min' distr' time is atleast F/u_s.

→ In case of peer download rate, the min dis' time is atleast F/d_{min}

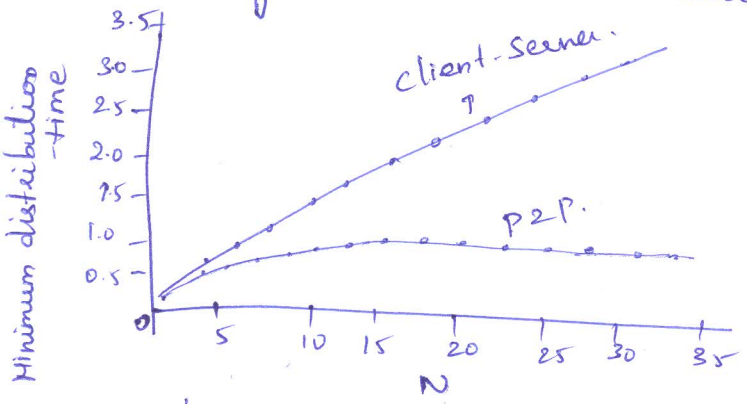
→ The total upload capacity is equal to the upload rate of the server plus the upload rates of each of the individual peers, i.e. $u_{total} = u_s + u_1 + \cdots u_N$. Delivering NF bits. The min dis' time is atleast $NF/(u_s + u_1 + \cdots + u_N)$.

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

if $F/u = 1$ hour, $u_s = 10u$, $d_{min} \geq u_s$.

↓ is always less than one hour for any no' of peers N.

The server trans' rate is 10 times the peer upload rate, so if peer download rate are set large doesn't have any effect.



Dist' time for P2P & client-server architecture.

Bit Torrent

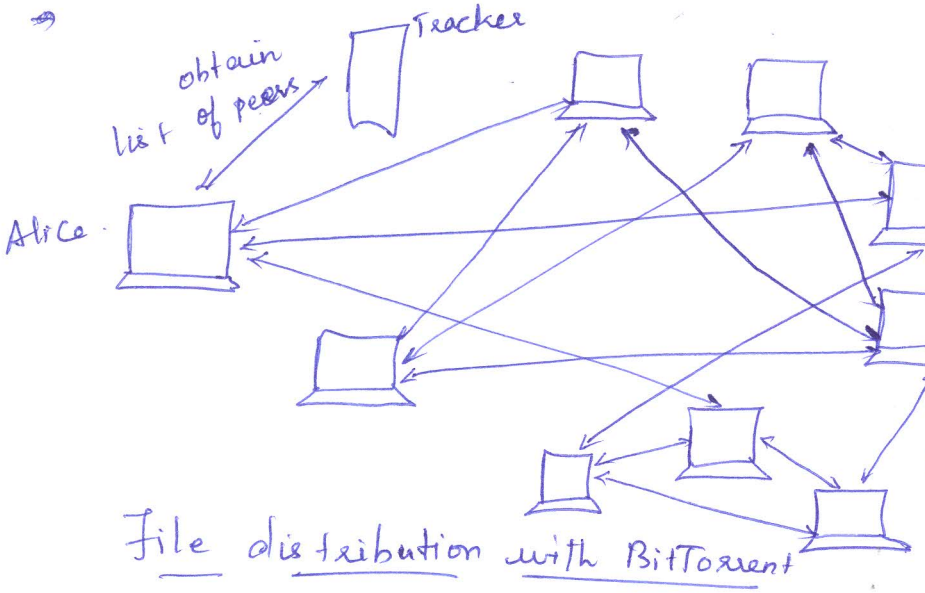
→ Is a popular P2P protocol for file distribution. In Bit Torrent the collection of peers participating in the distribution of a particular file is called as torrent. Peers in a torrent download equal-size chunks of the file from one another, with a typical chunk size of 256 KBytes.

When a peer first joins a torrent, it has no chunks. Over time it accumulates more & more chunks. While it downloads chunks it also uploads chunks to other peers.

→ Once a peer has acquired the entire file, it may leave the torrent or remain in the torrent & continue to upload chunks to other peers. Any peer can leave & rejoin the torrent at any time.

→ In BitTorrent each Torrent has an infrastructure node called as tracker. When a peer joins a torrent, it registers itself with the trackers & periodically informs the tracker that it is still in the torrent. A torrent can have more than 1000 peers participating at a time.

→



When, Alice joins the torrent, the tracker will send IP add' of 50 peers (randomly selected) to Alice. Alice attempts to establish TCP conn' with these peers. The neighboring peers will fluctuate over time.

File distribution with BitTorrent

Alice will have a subset of chunks from the file, with different peers having different subsets. Alice will ask each of her neighboring peers for the list of chunks they have.

If Alice has l different neighbors, she will obtain l lists of chunks. With this knowledge, Alice will issue requests for chunks she currently does not have.

2 Decisions Alice have to make.

1. Which chunks should she request ist from her neighbors.
2. To which of her neighbors should she send requested chunks.

For (1), Alice uses a technique called rarest first. The idea is to determine, from among the chunks she does not have, the chunks that are the rarest among the neighbors., hence equalize the copies in the torrent.

for (2) BitTorrent uses a clever trading alg'. The basic idea is that Alice gives priority to the neighbors that are currently supplying her data at the highest rate.

Alice selects 4 peers with highest rate & every 10 sec, she recalculates the rates & change the peers. These 4 peers are called as unchoked.

Every 30 sec, she also picks one additional neighbor at random & sends it chunks. This neighbor is called as optimistically unchoked.

Other peers other than these five peers, who didn't receive chunks are called as "choked". This mechanism is called as til-for-tat mechanism.

Distributed Hash Tables (DHTs)

A simple DB can be table with key-value pairs. eg: key-hostname value - IP add'. In case of peer-to peer arch' any peer can query the distributed DB & get back the results. The peer can also insert key-value pairs into the DB. Such a distributed DB is referred to as distributed hash table (DHT).

eg: key www.cmrit.ac.in , value 192.128.2.1.
the distributed DB

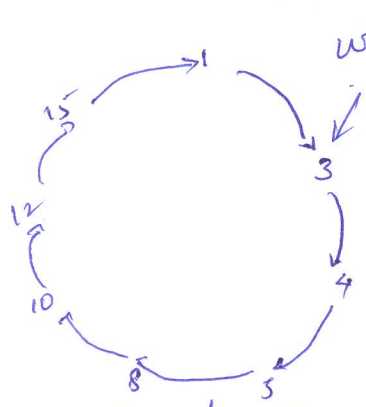
Problem: Alice will get query & get the web page from 192.128.2.1. The query will go to all the peers. The peer having the IP add' will respond.

Hash fun' is applied (many-to-one fun') → 2 different i/p's can have the same o/p. The key is converted to hash value. Instead of referring the key, the hash value of the key is used.

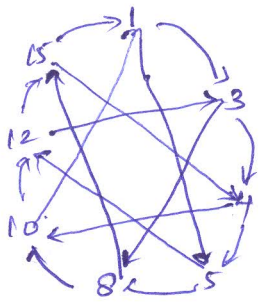
To store the (key, value) pair the peer with key value closest to the key is identified. By identifying this is an issue

Circular DHT

To address this pblm, the peers are organized into a circle. In this, each peer only keeps track of its immediate successor & predecessor (modulo 2^n).



Circular DHT



A circular DHT with shortcuts

→ the msg is sent clockwise around the circle. If a peer receives it can determine whether it is responsible for the key in the query if else then it will forward.

This process continues until the msg arrives at peer 12, peer 12 can send back a msg to the querying peer (3), indicating that it is responsible for key 11. But here, the msg need to be travel over the circle; N/2 messages are sent on average.

Soln: A peer not only keep tracks of only its predecessor & successor peers, but also of a realit relatively small no of shortcut peers scattered over the circle.

The no of neighbor peers & the no of msgs per query is O(log N).

Peer churn

As any peer can leave or come at any time without warning, so there is an issue with maintaining overlay.

Every peer need to track its 1st & 2nd successors by sending ping msgs. eg: peer 3 will track peer 4 & 5. If peer 5 leaves 8 will be the 1st successor of 5, &
8 will be the 2nd successor of 3.

Socket Programming : Creating Nlw applications

Client & server processes communicate through each other by sockets.

* Nlw app' -> 1. App' (Open app') that follow std protocol. (eg. FTP rules)
2. Proprietary App' - Developed by individual team.

Before writing code, one have to decide the app' is to run over TCP or over UDP.

Socket Programming with UDP.

In UDP, the sender will send the packet along with the IP add' (destination add') & port no' of the receiver.

eg:

client

```
import java.net.*;
import java.util.Scanner;
public class UDPClient
{
public static void main(String args[]) throws Exception
{
DatagramSocket ds = new DatagramSocket(); //Constructs a datagram socket
Scanner sc=new Scanner(System.in);
System.out.println("Enter your message");
String str = sc.nextLine();
InetAddress ip = InetAddress.getByName("localhost"); length
DatagramPacket dp= new DatagramPacket(str.getBytes(); str.length(), packet
ip, 3000);
ds.send(dp); ds.close();
}}
```

Server.

```
import java.net.*;
public class UDPServer
{
public static void main(String args[]) throws Exception {
DatagramSocket ds = new DatagramSocket(3000);
byte[] buf = new byte[1024];
DatagramPacket dp = new DatagramPacket(buf, 1024);   // 1024 -> length
ds.receive(dp);
String str = new String(dp.getData(), 0, dp.getLength());   // 0 -> offset
System.out.println("Message from client");
S.o.p(str);
ds.close();
}}
```

Send Output:

client.

Enter your Message
Welcome

Server.

Message from client
Hi Welcome.

The message that is typed in client side has been sent to the server and is also viewed in server side.

Socket Programming using TCP

TCP is connection-oriented protocol.

Handshaking between client & server need to be done before transferring messages. The handshaking process will be invisible to the program.

Once a message is received, the server can send the response to the client.

eg: Client

```
import java.io.*;
import java.net.*;
public class TCPClient{
PSVM() throws Exception
{
Socket s=new Socket("localhost", 6666);
DataOutputStream dout=new DOS(s.getOutputStream());
DataInputStream din=new DIS(s.getI/PStream());
BufferedReader br=new BR(new I/PStreamReader(S/m.in));
S.O.P("Enter the msg to the server);
String str=br.readline();
dout.writeUTF(str);
dout.flush();
S.O.P("Acknowledgment from the server);
String str1=din.readUTF(); S.O.P(str1); dout.close();
s.close(); }}
```

Server

```
import java.io.*;
import java.net.*;
Public class TCPServer {
Public static void main (String argsIJ) throws Exception
{
 ServerSocket ss= new ServerSocket (6666);
  Socket s: ss. accept ();
  DataOutputStream dout= new DOS (s. getOutputStream ());
  DataInputStream din = new DIS (s. getInputStream ());
  BufferedReader br = new BR(new IIPStreamReader (Slm. in));
  String str=dis. readUTF();
  S.O.p ("Msg from client:" + "\t" +str);
  S.O.p (" Enter the msg to the client");
  String str1 = br. readLine ();
   dout. writeUTF (str1);
   dout. flush ();
   din. close ();
   s. close ();
   ss. close ();
 }}
```

Output:

Client

Enter the message to the Server
Welcome
Acknowledgment from the Server
Received the msg.

Server

Msg from client: welcome.
Enter the msg to the client
Received the msg