Dinesh Naik

Unit 2

Application layer

Overall glance of topics

* principles of network applications
* Web and HTTP
* FTP
* electronic mail (SMTP, POP3, IMAP)
* DNS
* P2P Applications
* Socket programming with UDP and TCP.

Principles of network applications

Creating a network app

* write programs that:
  * run on end systems
  * Communicate over network
    ex:- Google server and app in mobile

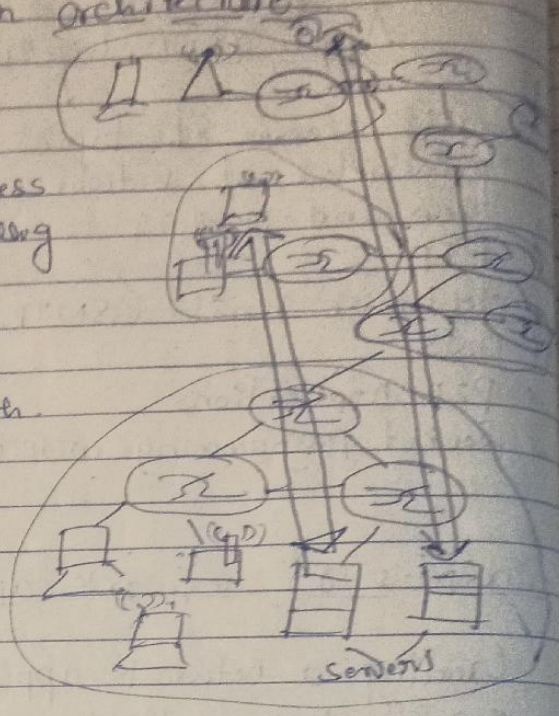
* No - need to write software for network-core
  network-core devices do not run user applications

Application architectures

Possible Structure of applications:-

* Client-Server
* peer-to-peer (P2P)

Client-Server architecture

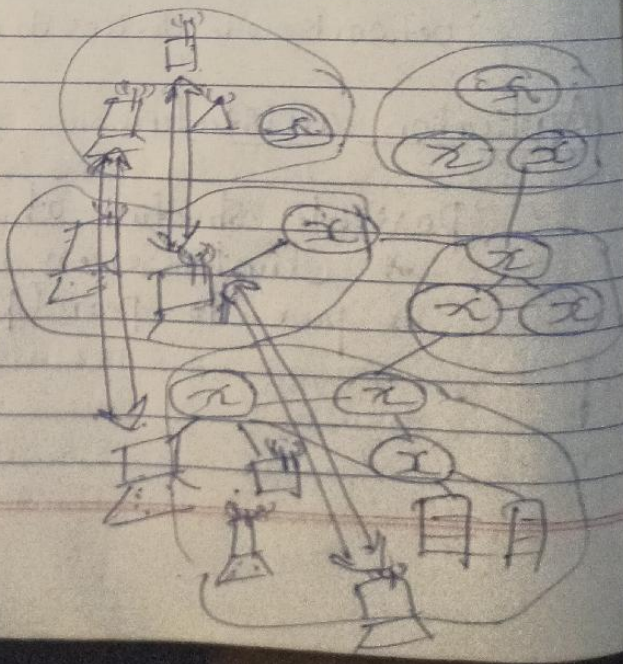


Server
* always-on host
* Permanent IP address
* data centers for scaling

Client
* Communicate with Server
* may have dynamic IP address
* do not communicate with each other

Peer to peer architecture (p2p)

* no-always-on Server
* end systems directly communicates
* peer request service from other peers
* Self scalability
* complex management
* peers intermittently connected and change IP address



Read less these

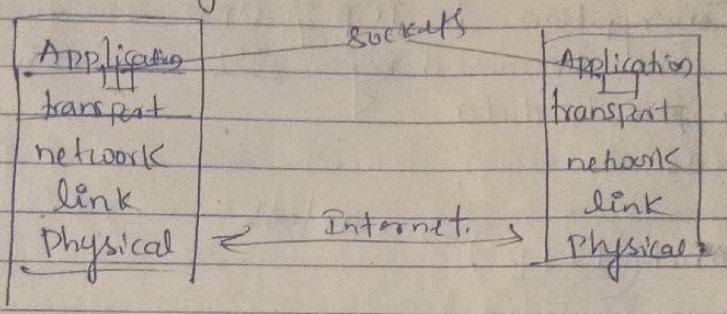
Processes Communicating

Process :- Program running within a host.

within same host, two processes communicate using inter-process communication

Sockets

* Process sends / receives messages to / from its socket
* Socket analogous to door.



Addressing process

* To receive messages process must have identifier
* identifier includes both Ip address and port numbers

What transport service does an app need?

* data integrity :- 100% data transfer.
* timing :- like games response delay time
* throughput :- minimum throughput.
* Security :- encryption, data integrity.

Internet transport protocols services

Tcp services
* reliable Transport
* does not provide timing, minimum throughput guarantee, security

UDP services
* Unreliable transfer
→ same here

Securing Tcp
→ No encryption in Tcp & UDP.
→ This is welcomed by SSL → which encrypts your private data.

Web and HTTP → stateless

Web pages consists of objects

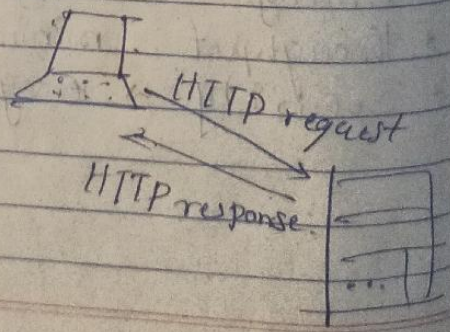
objects like HTML file, JPEG image, Java applet, audio file.

HTTP overview

HTTP: hypertext transfer Protocol.

like client/server model.

* client browse the information needed in request form
* Server gives answer in Response form



Uses TCP in

Client initiates Tcp Connection (Creates Socket) to Server, port 80

Server accepts request from client.

(protocols exchanged from web browser to web server)

gives response to client
Tcp connection closed.

HTTP is stateless in No Server maintains past client requests.

There are two types of HTTP Connections

Persistant HTTP	non-persistant HTTP
→ multiple object can be sent at a time.	→ at most one object sent over TCP connection
→ less time Taking	→ Time Taking
→ single connection can download anything.	→ multiple download needs multiple connection

↓

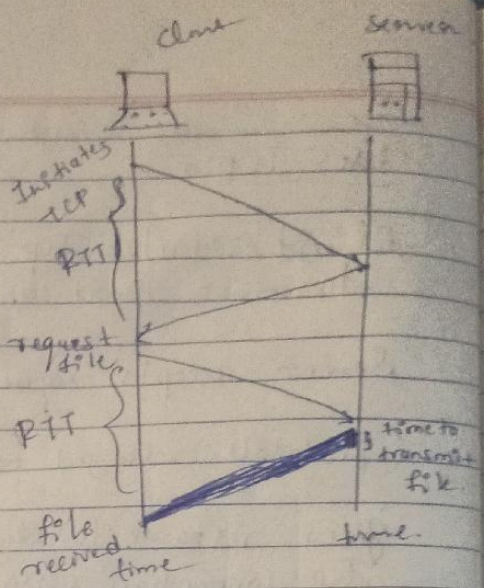
RTT (Response Time)

response Time = 2RTT + file transmission time required.

→ time for a small packet to travel from client to server and back.

* Server leaves connection open after sending response.
→ Same client/server can interact without any restrictions.

response time = RTT



For HTTP message format see 29th page of ~~Web~~ Chapter.2.pdf in github.com/gitout24

HTTP method Types

* HTTP/1.0 :- get, post, Head
* HTTP/1.1 :- Get, post, head, put, Delete.

HTTP response Status Code.

→ Code appears in 1st line of response.

200 OK
301 moved permanently
400 Bad Request
404 NoT Found
505 HTTP version NoT Supported.

User-Server State: Cookies

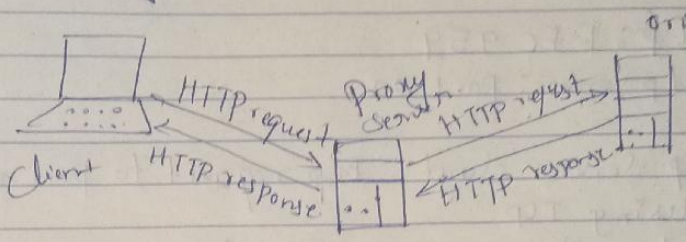
* Used for staying in the state and backup database also Tracing the request & responses generated through TCP.

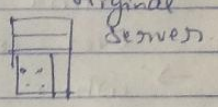
Cookies Creates unique ID whenever you revisit a site and upload it to backend database.

Cookies uses: recommendations
web e-mail
authorization
shopping carts

Web caches
Satisfying Client request without using original Server.

Using Proxy Server



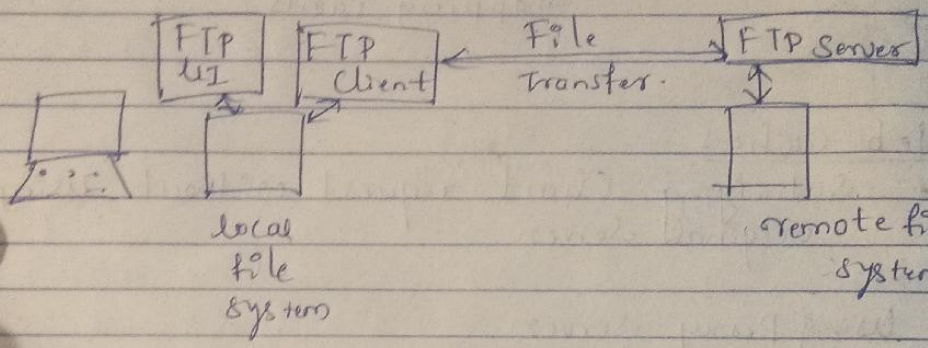


Uses of web caches
* reduce response time
* Prevent original Server from attacking
* acts as both client & Server.
  * Client for original Server.
  * Server for original requesting Client.
* like p2p architecture it maintain real time property in the responsing to request.
* Caching increase RTT & file transfer rate 100x

→ state

FTP :- File Transfer Protocol

→ File transfered from ftp server to ftp client that is remote host
↓ ↓
local host server



ftp: RFC 959
ftp server: port 21

→ FTP Server sends ~~Ftp~~ request to port no 21 using TCP.
→ Client authorize over Connection.
→ Client browses remote directory, sends commands over Control connection.
→ when Server recieves file transfer request and. Server opens 2nd TCP data Connection to Client.
→ after transfering one file Connection closed.

→ Control Connection :- out of band.
→ FTP servers maintain (state)
↓ ↓
earlier current directory
authentication.

Imp

FTP Commands, responses

Sample Commands * Text is sent as (ASCII) Control Channel.	Sample return codes * Status code and phrase.
USER username	331 Username ok, password required
PASS password	125 data connection already open; transfer starting.
LIST return list of file in Current directry	425 Can't open data connection
RETR filename gets file	452 Error writing file.
STOR filename (store file in remote host)	

Imp

FTP Commands, responses

Sample Commands
* Text is sent as (ASCII) control channel.

Sample return codes
* Status code and phrase.

USER username
PASS password
LIST return list of file in current directry

331 Username ok, password required
125 data connection already open: transfer starting.

RETR filename
gets file

425 Can't open data connection

STOR filename
(store file in remote host)

452 Error writing file.

Electronic Mail.

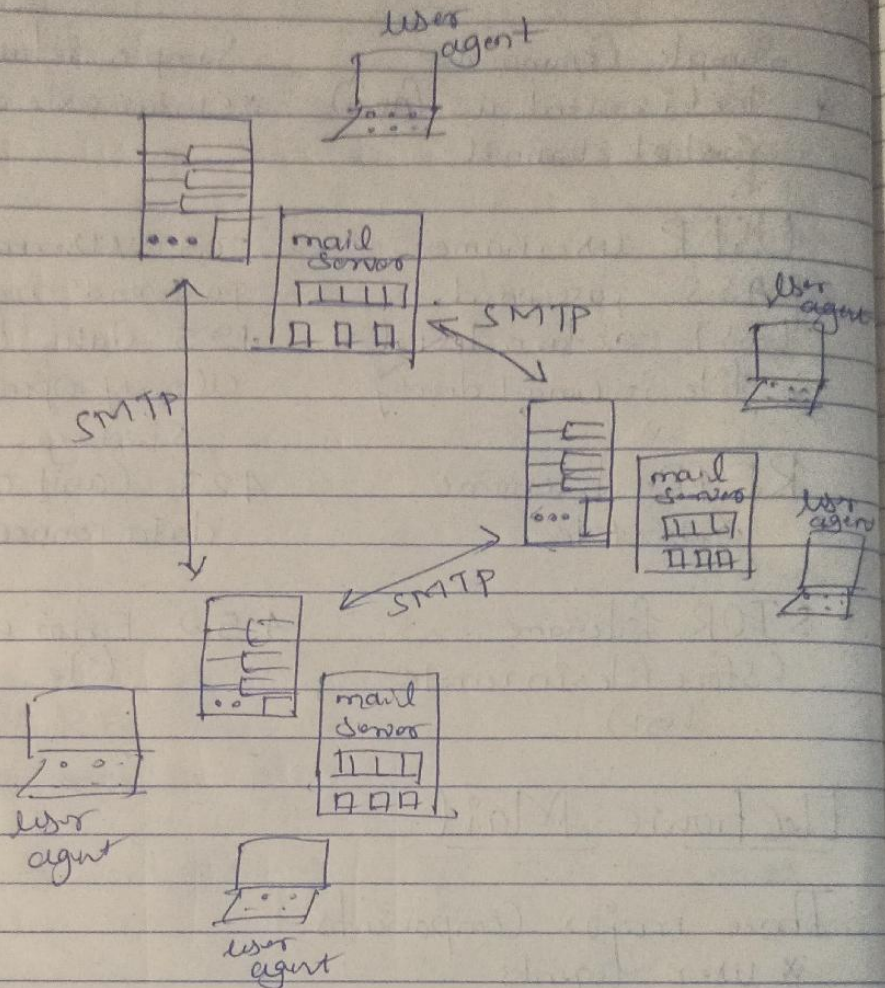
Three major components
* User agents
* mail servers
* Simple mail transfer protocol (SMTP)

User agent (mail reader)

→ Composing editing, reading, mail messages.

eg:- outlook, Thunderbird, iphone mail client

Mail Servers



→ mail box contains incomming messages for user.
→ messages queue that are outgoing
→ SMTP (between mail servers to send email messages)

Client: Sending mail server
Server: receiving mail server

Imp

SMTP is Simple mail Transfer protocol. [RFC 2821]

→ uses Tcp to transfer reliably transfer email message from client to Server, port 25

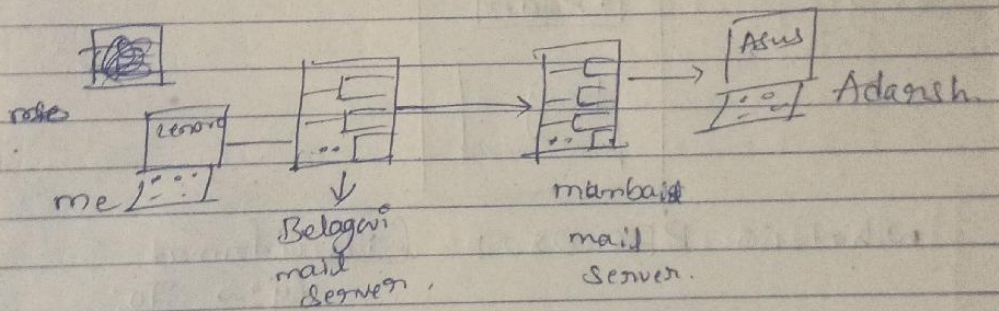
→ direct transfer → Sending Server to receiving Server.

→ Three phases
- * handShaking → connects
- * transfer message → sends
- * Closure → closes.

→ like HTTP & FTP uses Commands as ASCII and response code as status code

→ message must be 7-bit ASCII

Write example like u use email in your mobile



→ Interaction

S: 220 hamburger.edu

C: Hello, Danesh.

S: 250 Hello Danesh, pleased to meet you.

C: mail From: <naikdanesh@gmail.com>

S: 250 Danesh naikdanesh@gmail.com Sender OK

C: RCPT To: Adarsh@gmail.com

S: Adarsh@gmail.com ... Recipient OK

C: DATA

S: 354 Enter mail end with "."

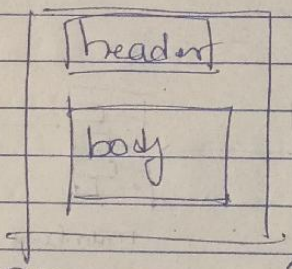
C: my message

S: 250 message accepted for delivery
C: quit
S: 221 hamburger.edu closing connection

SMTP uses Persistant Connections.

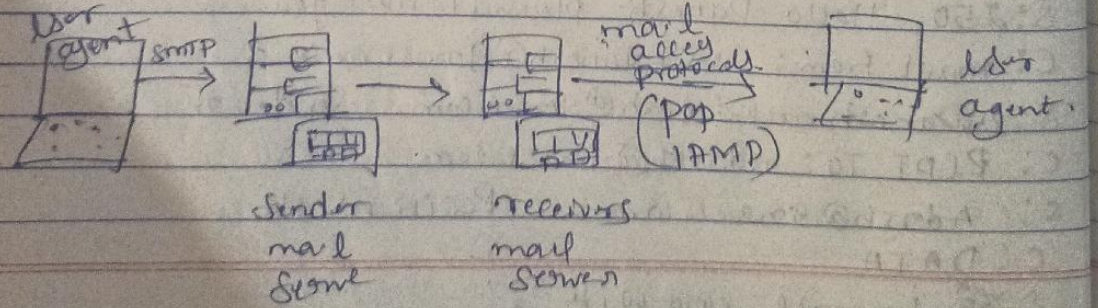
HTTP	SMTP
pull (GET)	push
→ Each object encapsulated in its own response msg	multiple objects sent in multiple Part msg

Format of mail message



what is RFC 2821 (its a format
eg: To:
From:
Subject:

Mail access protocols



[RFC 1939]

Pop3 :- Post office Protocal (authorization, download)

IMap :- Internet mail Access protocal.
[RFC 1730]
manipulation of stored msg on Server.

HTTp :- Gmail, Hotmail, Yahoo! mail etc.

POP3 Protocol

* Authorization Phase
  - Client Commands
    - User : declare username
    - pass : password
  - Server response
    - +OK
    - -Error.
* transaction phase
  - list : list message numbers
  - retr : retrieve msg by number
  - dele : delete
  - quit

Example for pop3 interaction (See page 58)

-> In pop3 we cannot re-read the messages.
-> pop3 is stateless.

IMAP :- maintain sessions and keep all message in one place : at server.
-> user can organize messages in folders.

→ (amazon.com
git.edu)

DNS (Domain name system)

Distributed database
* implemented in hierarchy of many name servers
Application-layer protocol: hosts, name servers communicate to resolve names

→ Complexity solved by hirarchical dividing

DNS Services, Structure

hostname to Ipaddress translation
host aliasing, alias names.
mail server aliasing
load distribution

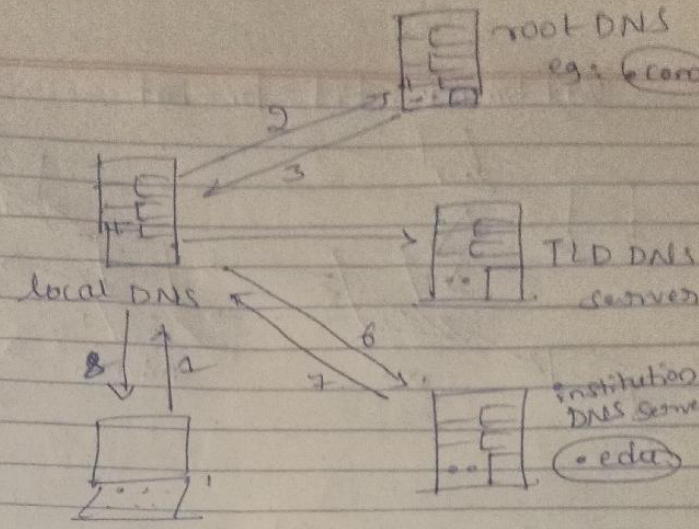
* Single point of failure
* traffic volume
* distinct Centralized database
* maintenance

lets understand DNS (Amazon)

In India we use (Amazon.in)

In world wide server (amazon.com)

It is only hirarchy for example
See your emails.
for collage email its (git.edu)
for personel email its (@gmail.com)



* caching in DNS will go out of date
(please read page 70)) - 74)

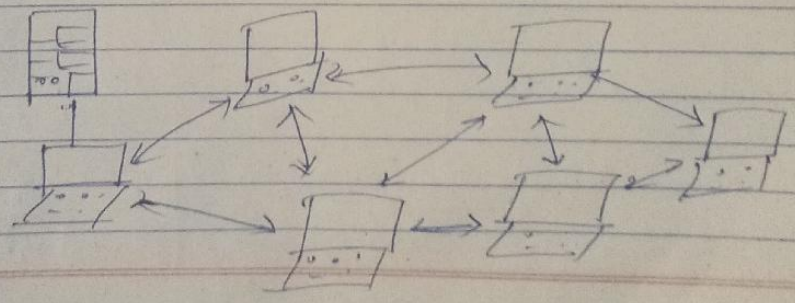
P2P architecture (already did it)

lets study about Bit-torrent know

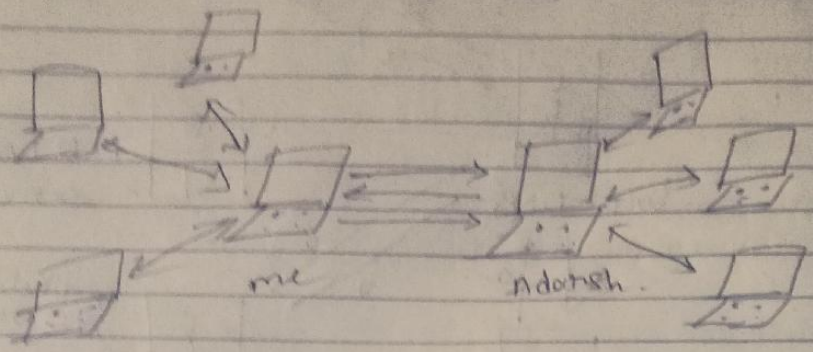
* most effective use is Streaming is done by P2P model

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



BitTorrent : Bit-for-bat



Requesting, Sending file Chunks.
(~~write~~ read ex page its Similar to Protocal transfer. (FTP))