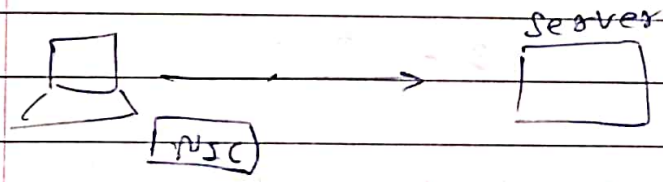
CN

Book : Computer Networking
By Heith w Ross, James kusose

asg:
what is multiplexing ?
who gave first web mailing
→ satbeer Bhatia, Hotmail.

Circuit switching : reserve the path from source to destination
Packet switching :

Queing time + propagation time + transmition delay



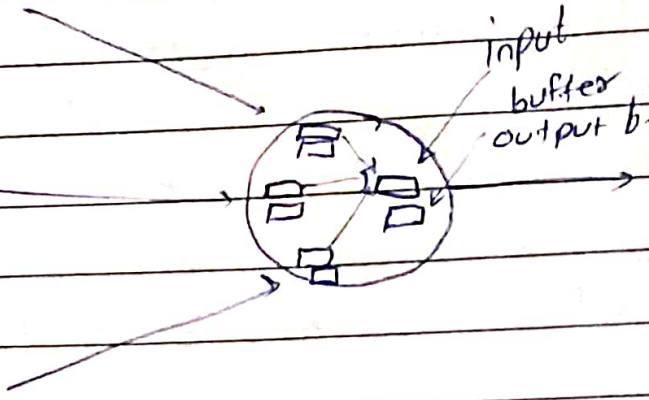
$t_{propogation}$: to carry packet from PC to server.
$t_{Dumping}$: to dump it to the network wire or t_{frame}
$T = t_{prop} + t_{dump}$ for a packet

file → n packets → each packet is dumped to wire → transmitted to server → qued for further foowarding if server speed < sender speed

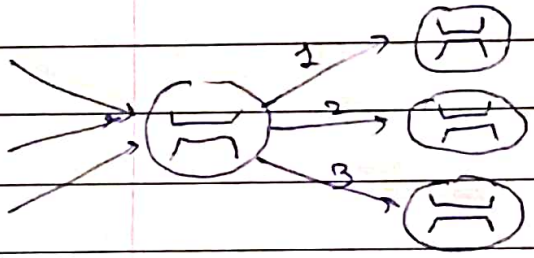
T_{dump} = Data size of paket / speed of NIC card
T_{prop} = Distance / speed of em wave
↑ reflected

CN

in packet switching packet knows its destination adress



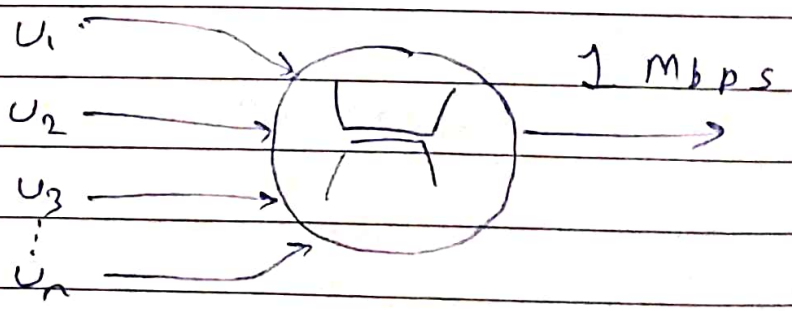
local forwarding table



1	
2	
3	

statistical multiplexing : packets are queued for dispatching in order they arive. when que is full, rest incoming packets are dropped.

Q Each user when active, is given 100 kbps 10 % of time a user is active.



→ in ciocuit switching :

no. of users supported = $\frac{1 \text{ Mbps}}{100 \text{ kbps}} = 10$

asume 35 users with p, $(1-p)$ being prob of each being active

So,

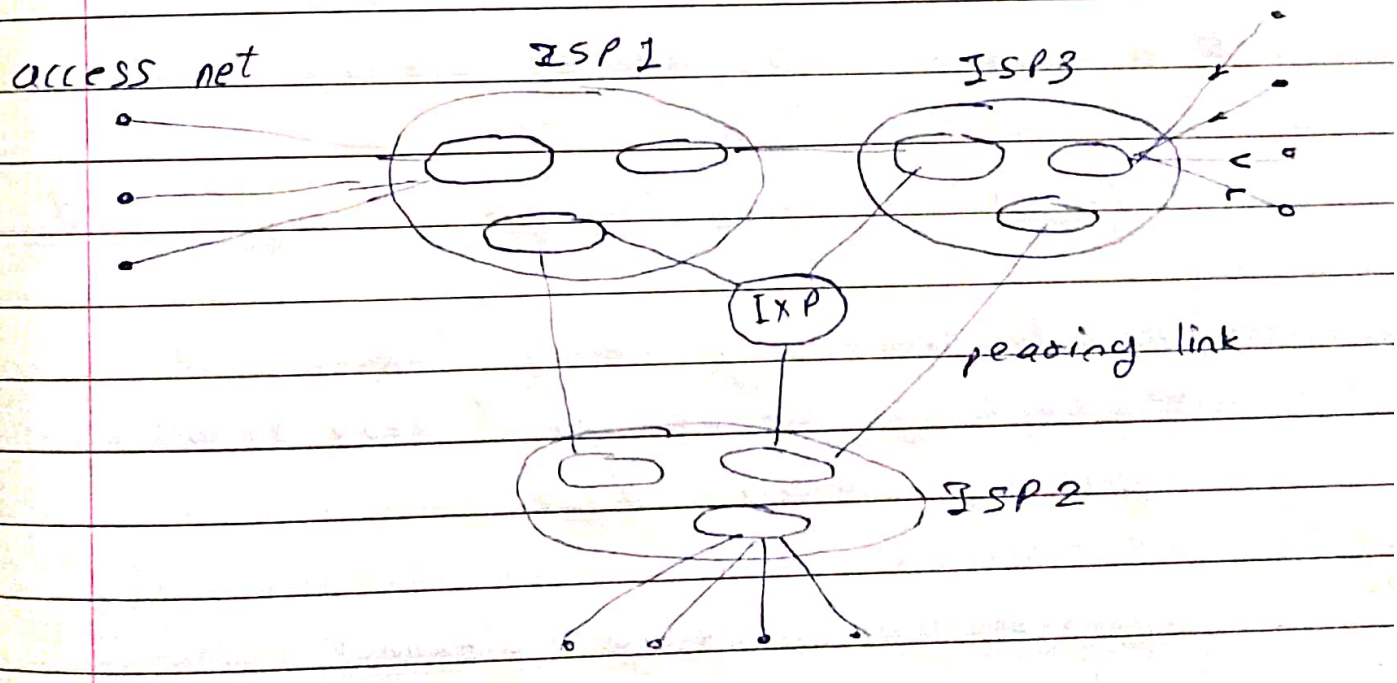
$$p(\text{2 are active}) = p^2(1-p)^{33}$$

$$p(\text{10+ users active}) = 1 - p_0(0) - p_0(1)p_0(34) - p_2(2)p_0(33) \cdots p_0(0)$$

$$= 0.0004$$

So, its safe to make queue of size less than 10, only 4 of 1000 times packets will be dropped

Internet Structure:



IxP: internet

ISP network:

tier 1 ISP tier 1 ISP google

IXP

Regional ISP Regional ISP

access ISP access ISP . . . . .

peer to peer connection: no payment they handle each other's ~~traffic~~ ~~consum~~

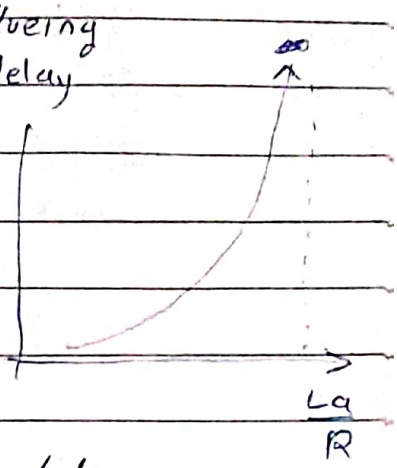
* 4 Sources of packet delay

$$d_{nodal} = d_{proc} + d_{que} + d_{trans} + d_{prop}$$

nodal processing delay (d_{proc})
→ checking bit errors (very small)
queuing delay (d_{que})
→ time to wait
transmition delay (d_{trans})
→ L: packet length, R: link bandwidth, $d_{trans} = \frac{L}{R}$
Propagation delay
→ for travelling the distance

d_{trans} is time taken by NIC to dump packet to ~~mte~~ conneting wire

traffic intensity

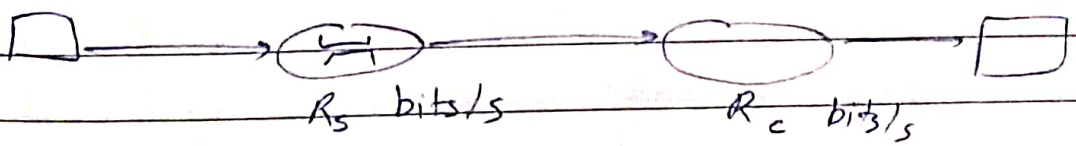


R : link bandwidth (bps)
L : packet length (bits)
a : avr. packet arrival rate (packets/s)

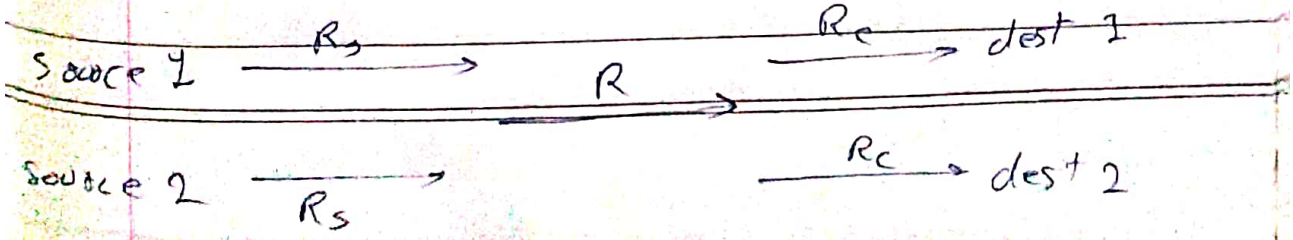
La/R → ≥0 : small queing delay
La/R → ≤ 1 : big queing delay
La/R > 1 : infinite queing delay as incomeing rate is more

TTL value :
each packet has 8 bit value, which is set by sender & each ~~serve~~ router in path reduces it's value by 1, if TTL = 0, then router drops the packet.
Why? → so that packets dont loop into the router network.

Throughput : rate at which bits are transfered from sender to receiver



throughput is smaller of R_s & R_c i.e. bottleneck.



asuming R chanel gives equal chance to both clients.
thrpt : min ($R_c \rightarrow R/2$, R_s) for each.

Internet protocol stack

layers:
- application
- presentation
- session

L5

- transport — L4 — (TCP)
- network — layer 3 — ~~IP~~
- link — layer 2 — (NIC)
- physical — layer 1 — (fiber optic

(ISO/OSI reference model)

lower layer has to ~~give~~ add functionality to layer below it

application layer gives message, forward to ~~pre~~ transport layer, which adds its header, so it becomes segment its fwded to network layer, that adds its header, transforming it to datagram or packet, after link layer header: frame

firewall is L5 device (can read all 5 layers
Data link is L2
router is L3
i.e. these devices implement only till certain layer, not all 5

Q How skype works

Q ISO/OSI model

Chapter 2 : application layer

→ P2P or server client

process communication :
→ same host : inter process handled by os

Socket : memory area asigned to each application, that gets where app. can read / wright, and TCP layer reads from here.
Port number : socket is bound to port no to know which process incomming paket is made for.
- for client, port no. may be asigned by os randomly, but for server, port no. is fixed & pre defined for client convenience

App layer protocal : HTTP, FTP, DNS,
transport layer protocol : TCP, UDP
- data integrity / loss tollerant
- data speed (throuput) (byts/sec)
- time sensetive (frames/sec)

TCP : mostly used
UDP : latency tollera sencitive apps may go for UDP, UDP → no delay

HTTP → TCP
DNS → UDP
FTP →

HTTP

- stateless (at one no history is stored) so we need cookies
- www.google.com/file

Domain name Relative path

- client-server structure

Non persistant HTTP (HTTP 1.0)

1. client initiats TCP
2. server acknowleges TCP request
3. client gives GET request to server
4. server gives back html or JSON... responce
5. server closes TCP connection

→ Now, if returned html page has reference to 10 other objects (image/ vdo/ audio/ html the whole cycle repeats

RTT: time for one object (html/JSS/... to travell to
time for small packet to go from client to server & back from server to client

Time to Transfer (t): for give object (html/css to go from server to client

So, for nonpersistant HTTP total time for loading index.html:

$2RTT + t + 10(2RT + t)$

CN / 25 aug.
Lab

Cross Over Cable

transmission side	receiver side
White orange	White green
orange	green
white green	White orange
Blue	Blue
White blue	White blue
Green	orange
White Brown	White brown
Brown	brown.

RJ 45 connector : socket

Drawback of non persistant:
→ OS overhead for each TCP
→ browsers have to make parallel TCP connections
→ 2RT time

~~HTTP request~~

3 Q What is cookie, where it is stored on machi
1 Q HTTP 2, SPYDY
2 Q Difference in URI, URN, URL

Persistant HTTP (HTTP 1.1)
→ connection is kept open

HTTP message structure:

request:

request line : ~~GET~~ ~~/index.html~~

GET /index.html HTTP/1.1 \r\n

headers { header-name : header-value \r\n
host name : \r\n
~~last-mod~~ \r\n
keep-alive : 115 \r\n
\r\n

(\r\n: carriage, new line)

body of request { body (JSON/url-encoded/...)

HTTP/1.0 : GET	HTTP/1.1 :

response:

{ HTTP/1.1 200 OK \r\n

header of response { Date:
last-modified:
:
set-cookie:

User-Server Cookie State

Web cache

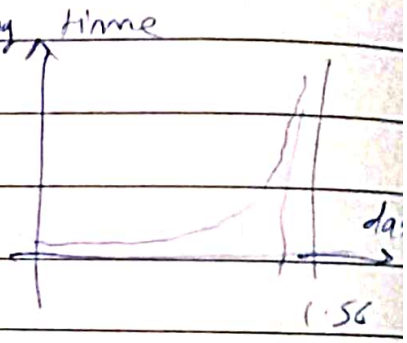
* typically ISP or network providers provide the proxy server / institutes
cache server caches static resources for
- reduce waiting time
- origin servers are of lower capacity. then proxy can enhance it's performance

Internet: Bursty traffic:
access link rate even if is near the average capacity, there will be huge delay.
→ Because in practical, requests are comming in bulk for some time & nothing for rest of time.

* No data link should be operating at more than 60% of its max. capacity

example: from an institution, if ave. 1.5 Mbps of traffic is going for some server, via link having capacity 1.56 Mbps

→ then high latency as 1.5 is bursty traffic



So, if 40% of data is cashed i.e. hit ratio is 40%.

So, if time for responce by cache: t_c

by server: t_s

so total time:

$$1.5 \times \left[\frac{40}{100} r_1 + \frac{60}{100} r_2 \right]$$

also, Now access link will be utilised in lower capacity, r_2 will be lower

$$\text{avrage time for 1 packet} : = \frac{40}{100} r_1 + \frac{60}{100} r_2$$

headers: Date, server, last modified, ETag,

Conditional GET

request header : if-modified-since : 5/1/22
OR ETag : ....

responce : headers : Date : current date
ETag : hash of page requested

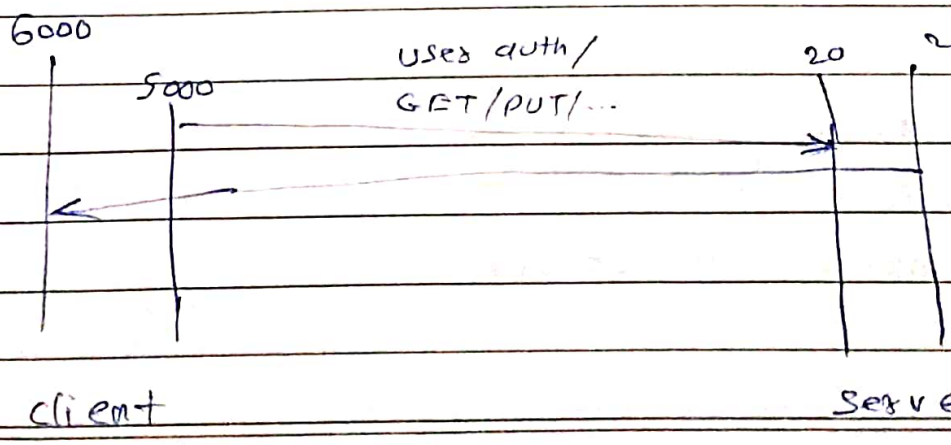
if not modified only responce has code 304 (Not modified)

ETag : hash of file requested, to check if requested has changed.

If ETag or if-modified-since headers are present, then it is Conditional GET request.

FTP

- 2 TCP connections initiated by client & server each
- on one ~~serve~~ channel, commands are sent
- on other : Data response

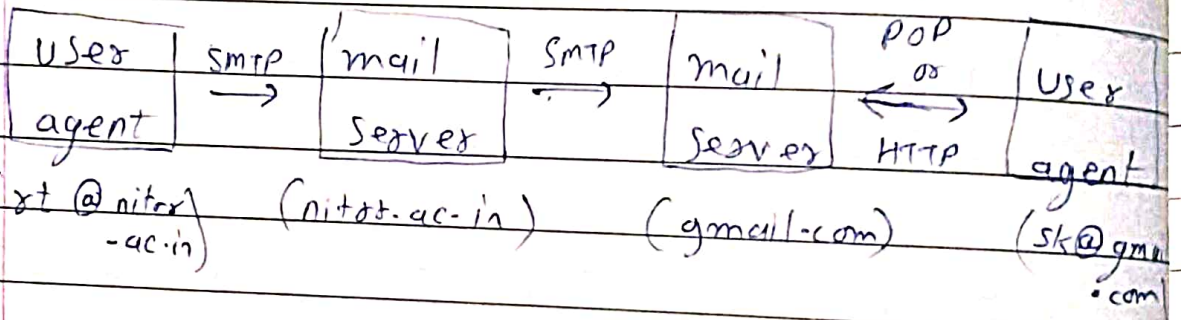


passive : client starts TCP for command, authenticate, informs server, server start TCP for data

(Used because ~~for~~ circuit switching, ages ago)

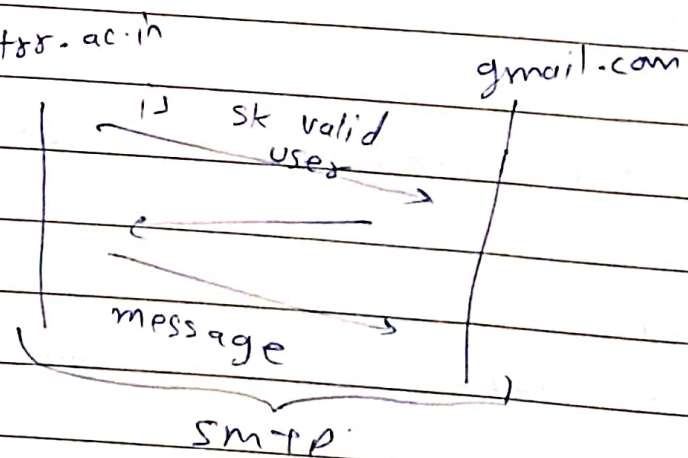
* Electronic Mail

mail's journy :



SMTP : (port 25)

push message protocol



rt @ nitsr.ac.in

↑

user name mail server

POP : when client fetches, server deletes mail

IMTP : mail is kept on servers till deleted

* DNS

old model : centralised DNS

→ only one chart of ip vs name

Drawbacks :
- high maintanance
- singhe point : high failure chance
- traffic value
- distance bias / location : distant req. not happy

DNS function :
- hostname resolution
- hostname aliasing
- mail server aliasing
- load distribution

CN_LAB

IP : 32 bits , 128 bit (V6)

Mac : 48 bits

IP classes:

A : 0 - 128

B : 129 - 191

C : 192 - 223

D : 224 - 239

E : 240 255

Commands: 1) ping

2) ping google.com

3) ipconfig

→ display all

4) arp -a

→ mapping betwin physical and actual adress

5) getmac

→ get mac adress

6) hostname

→ get host name

7) nslookup

8) tracert

9) netstat

9) ~~ark~~

10) ~~rark~~

11) pathping google.com

12) systeminfo

Subnet mask

A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

* P2P file distribution (ex. torrent)

k →	2^{10}
K →	10^3
b →	bit
B →	Byte

elements: server, clients (who can upload/download)

file divided in chunks

- new user joins network
- server gives tracker file having chunks vs clients
- download → rarest first
- Tit for tat :

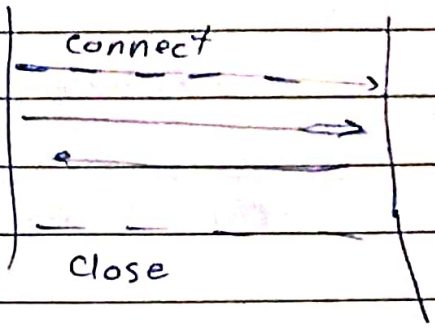
- here we dont have hostname of seeder while requesting

* hashing gives unique hash because:
file: 2^{1000} bit → hash 2^{512} bit
so, theoritically $2^{1000}/2^{512}$ files will have same hash
But practically it is not possible to find 2 files with same hash in finite time

DHT

Server 1 or Client 2

Socket



bytes tosend: b
connect & close :
controll request
say $C_1 + C_2$ bytes
more efficient : b more, C_1+C_2 less

controll overhead contains headers from networks

Server	
	Socket() (1)
	bind() (2)
	listen() (3)
Socket() (4)	
connect()* (5)	accept()* (4) * blocking operation
send() (7)	receive()* (6)
receive*() (8)	send() (9)

Blocking operation: the OS puts code execution on hold for the next process or network process.
accept : will wait for client to connecting
connect : will wait for server to make next call
receive : will wait for client to send something