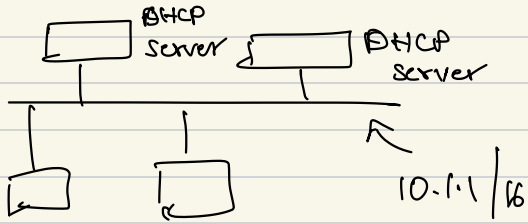
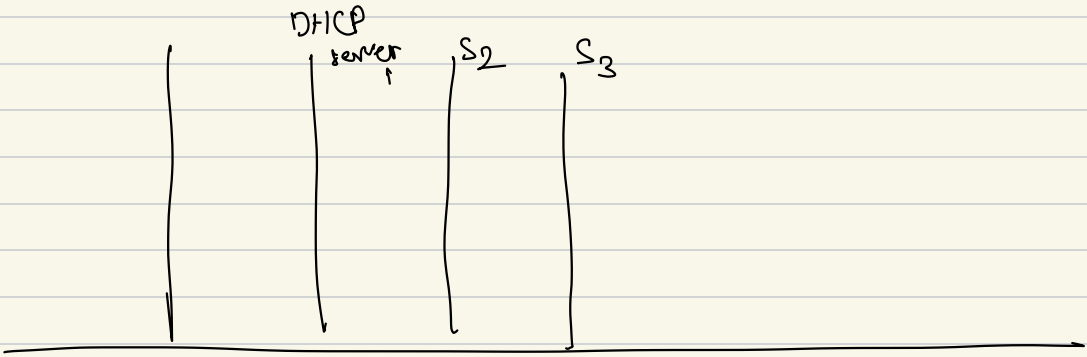
8/10/24 CS348- Computer Networks :-

ARP :-

DHCP :-





UDP TCP

If you care about latency

If you care gaurentee than if it stays for 10sec/a minute.

APPL layer

DHCP (Port = 68 for client A) (port = 67 for DHCP server)

68

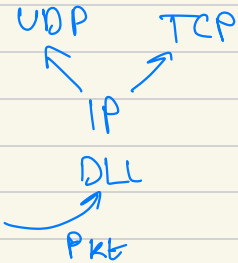
UDP TCP

IP

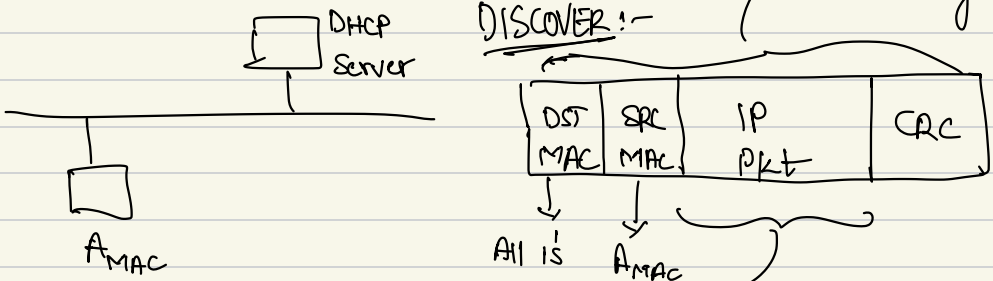
DLL

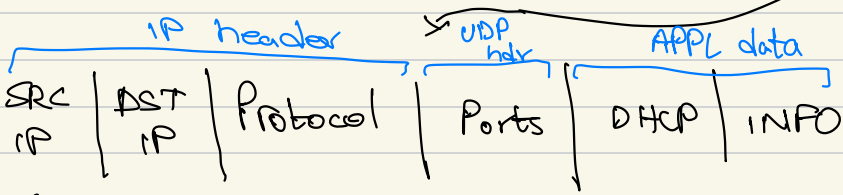
PHY

When a pkt received, go up from DLL, IP. In IP field will have UDP / TCP



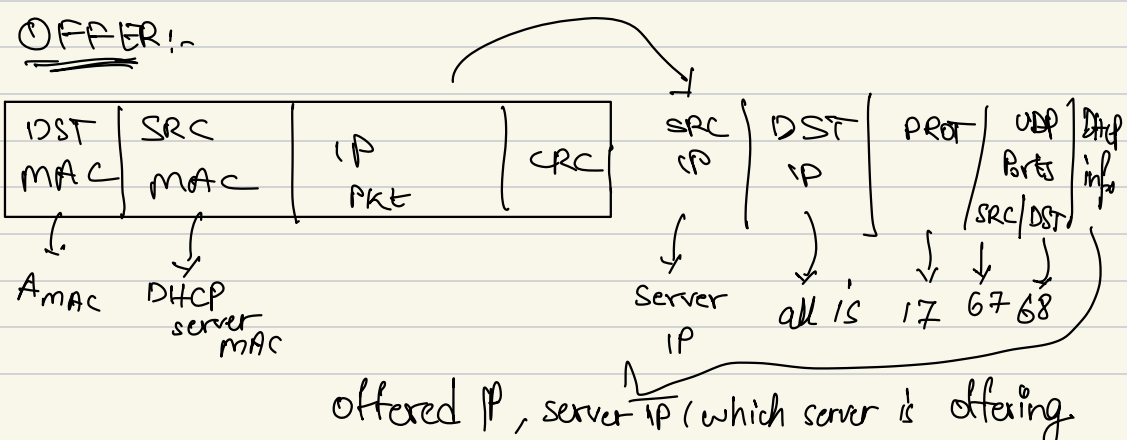
* Main things





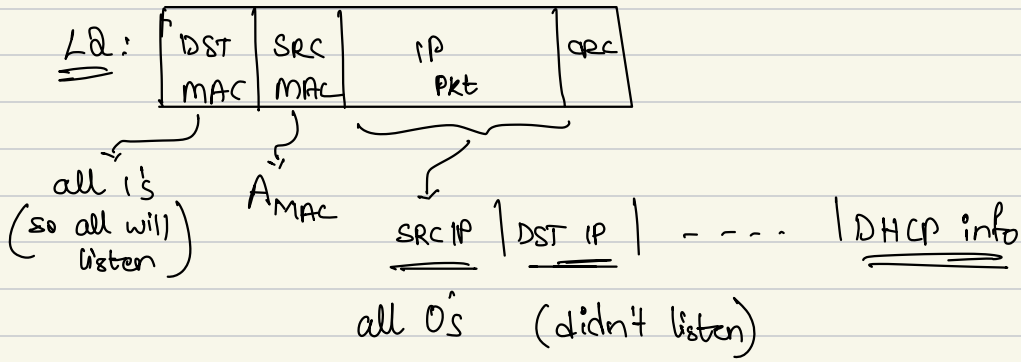
(didn't listen explanation / default values of fields)

* OFFER:-

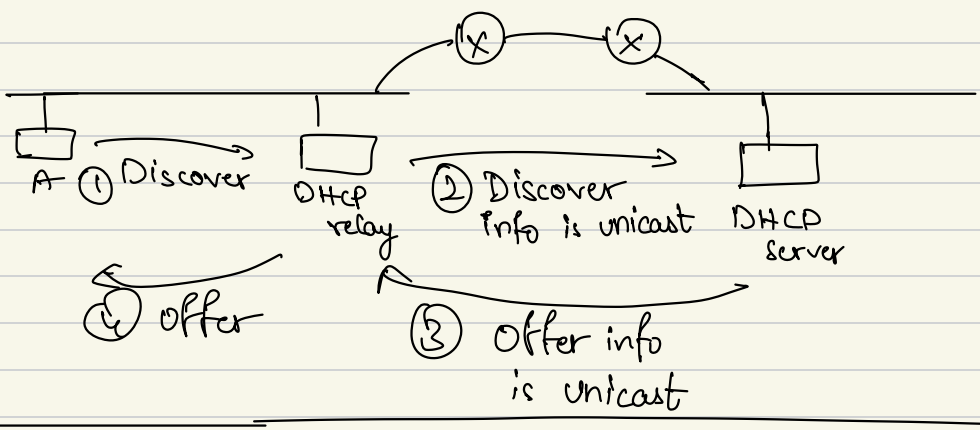


offered IP, server IP (which server is offering

* Request (A → sent to all)



* Relay: sort of a proxy



10/10/24 CS 348 :- (didn't listen 20 min)

* App1 App2 App3 ....

UDP TCP → retransmits for guarenteed delivery
→ inorder delivery to appl at dest
↳ flow?

[1] [3] received, but 2 not received
TCP gives application only "1"
After when 2 received, gives "123"
→ Also does congestion control, flow control

UDT in contrast to TCP, doesn't do all above. Only sends out 1 datagram to given DST IP, port #.
How to get port #? (didn't listen) NET → get here
MAC
PHY

* Recall DHCP uses 67, 68 → client → server. Web server uses port 80
* Why do we use UDP in DHCP instead of TCP?

When to use UDP? :-

(1) if msg fits in 1 pkt

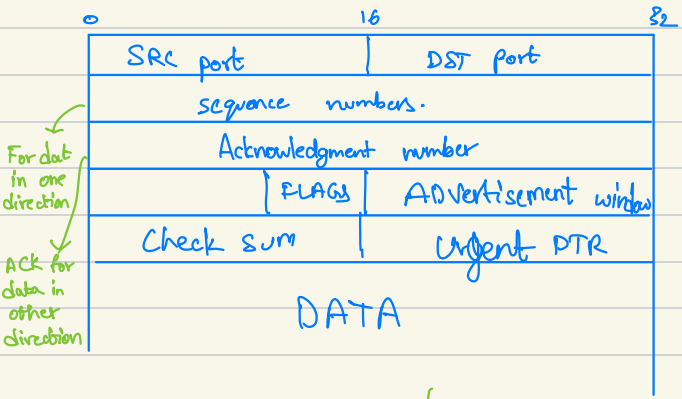
(2) App1: 1 2 3 (2 arrives late) App2: 1 2 3

*

→ Like TCP in your App written above UDP
(But TCP is a tricky & large code. You'll not want to write it in your appli -)

* UDP header :- (didn't listen)

* TCP header :-



How to decide 1, 3 came but 2 didn't

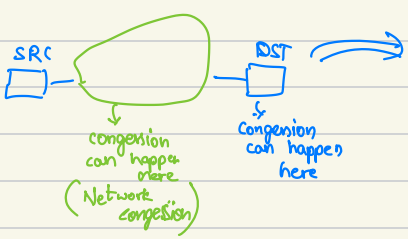
* TCP is 2 way communication

* FLAG bits: SYN BIT | FIN BIT | RESET | PUSH | URG | ACK BIT

SYN BIT → Sent to setup connection

FIN BIT → Used to close connection

*



here buffer may get full

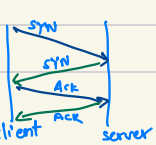
Then DST can send explicit menage that I'm running out of buffer

⇒ Advertisement window used

14/10/24 Class :- (didn't listen 30min)

* 4 way - handshake :- (little bit diff from 3-way handshake)

How to identify a particular TCP connection?

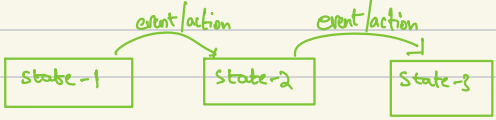


Ans:- IP : SRC IP, DST IP
Ports : SRC Port, DST ports
} (though TCP is a 2-way connection, for now let us call 1 SRC & other DST) } These are in UDP also.

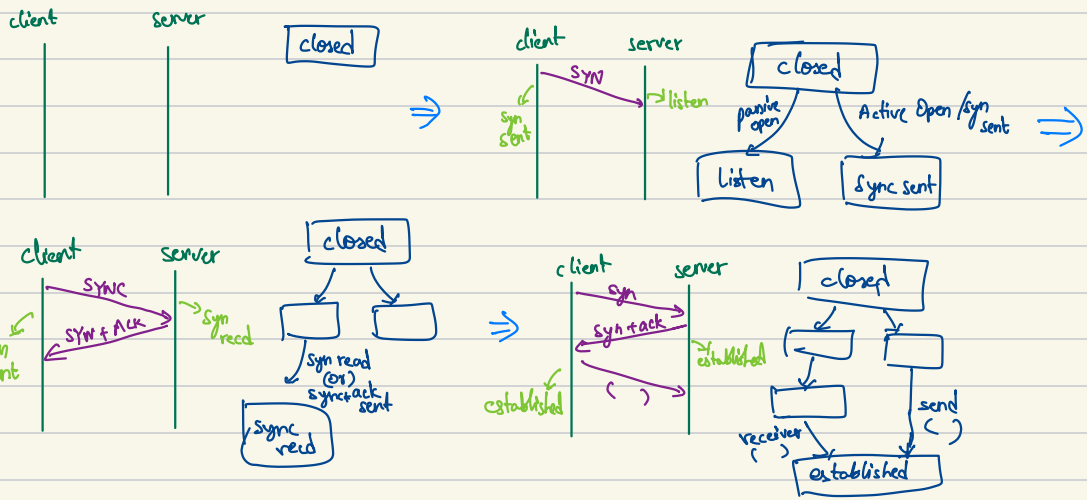
didn't listen. ( ) } These aren't

* (said why we need state diagram)

State diagram :-

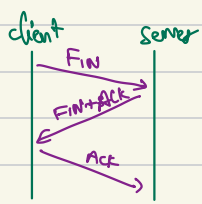


→ Now consider state diagram for establishing a 3-way handshake

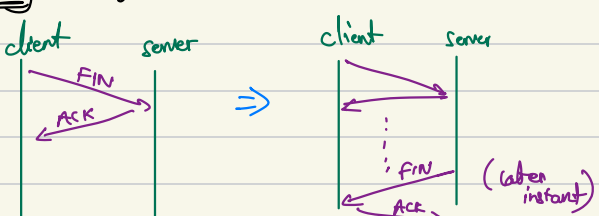


* Connection termination :-

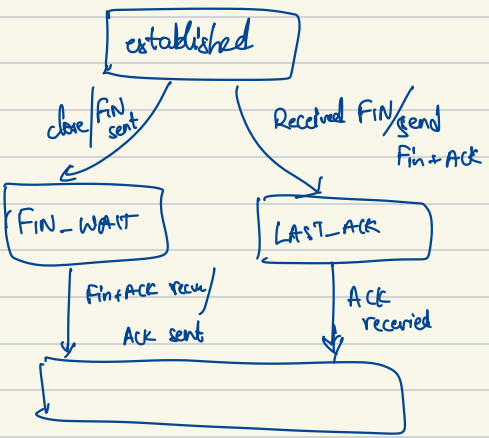
1st way :- 3-way handshake



2nd way :- only 1 terminate at a time



State diagram :- (1st way)



Problems with this :-

* What if (Fin + ACK) not received (i.e., lost) by client?
  client stays in FIN_LAST
* What if Ack not received by server?
  server stays in LAST_ACK

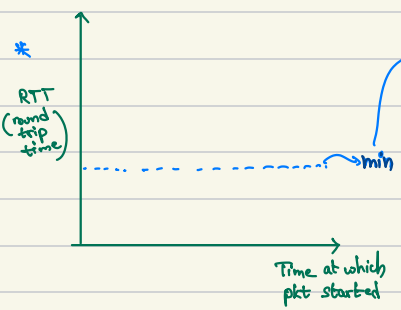
Soln :-

* Time wait (didn't listen)

Nxt class congenion control.

15/10/24 class - :- (missed 20 min)

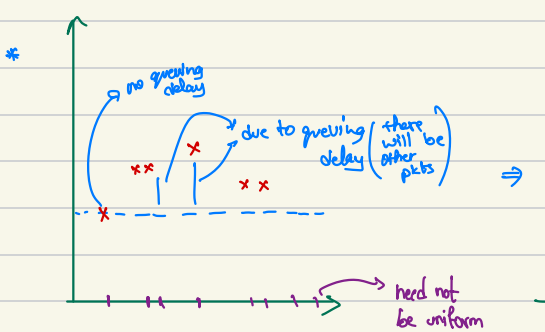
*



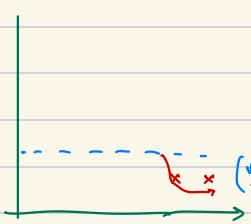
speed of light + transmission (P/C) → We wait till each pkt completely come (store & forward)

(if queing delay = 0)

*



⇒



(may be route changed & speed of light delay lower)

* Plot histogram of . . . (didn't listen)
* Random numbers : $x_1, x_2 \ldots x_N$, $M = \frac{1}{N}\sum_{i=1}^{N} x_i$, Estimate of std deviation $= \sqrt{\frac{1}{N}\left(\sum_{i=1}^{N}(x_i - M)^2\right)}$

But there are 1000's of pkts every sec & many TCP ports data going. Everytime you receive a pkt, if you want to square, it is a overhead.

* Used Mean deviation instead of Standard deviation

$$MD = \frac{1}{N}\sum_{i=1}^{N}|x_i - M|$$

* Algo used by TCP :- (for RTO)
  - Latest RTT estimate : Sample RTT
  - Current estimate of mean estimate : Estimate RTT
  - Update step :-

    Estimate RTT = $(1-\alpha)$ Estimate RTT + α Sample RTT (Exponentially moving average)

    → what happened in past ($(1-\alpha)^T$ if T units ago) will be given lesser weightage w.r.t time

    $\alpha \in (0,1)$
  - difference : Sample RTT - Estimate RTT

    Like $x_i - M$
  - deviation = $(1-\beta)$ deviation + β |difference|
  - Timeout = μ + ϕ

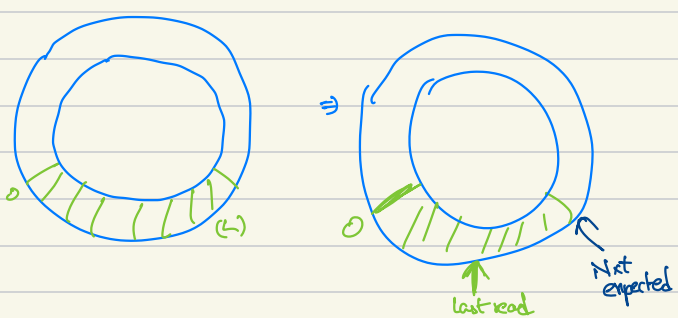
    $\mu = 1$ $\phi = 4$ $\alpha = \frac{1}{8}$ $\beta =$

    (TCP vegas didn't give values. Hence wasn't accepted)

Congestion & flow control :-

- TCP have to make a guess whether there is a congestion in network or not.

* (didn't listen)



* Nxt seen

Window = min (Congestion window, Adv window)

Window → max bytes an acked src can send out

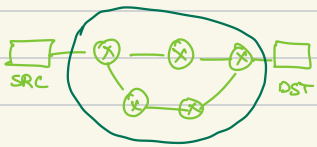
Adv window → depends on router

Congestion window → Depends on network congestion

17/10/24 class-32 :-

* We've learnt RTT estimation

TCP congestion control :



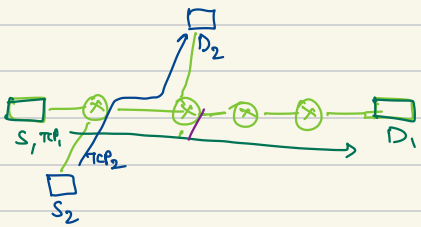
The middle nodes do not have power. They just forward pkt (Anyone people wanted that) (More efficient possible if this is not case)

Power at Ends → TCP

(didn't listen 5 min)

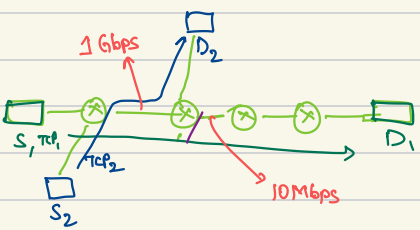
End hosts running TCP:-

- They don't know link speeds
- They don't know link utilizations
- They don't ... (didn't listen)



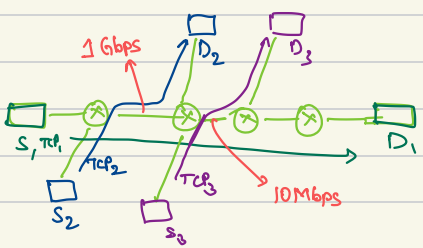
Here it is not true that Both should get same

$\begin{cases} TCP_1 \rightarrow 10\,Mbps \\ TCP_2 \rightarrow 990\,Mbps \end{cases}$



Here to be fair, we divide equally

$\begin{cases} TCP_1 \rightarrow 5\,Mbps \\ TCP_3 \rightarrow 5\,Mbps \end{cases}$

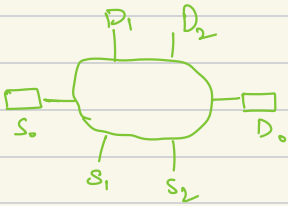


Congestion control issues:-

TCP connection

- wants to use bandwidth resources efficiently
- do not want to cause congestion → (pkt loss, Queues filling up)
- fairness
  - one TCP connection should not have most of bandwidth at expense of other connection (A very high level idea. Actual fairness is much complicated to define)

These are our requirements. How do we design such protocol?
Its non-trivial, one see soln feels .....



Q) How to set data rate of TCP?

Idea 1:-

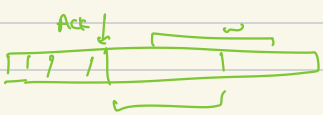
- Each sec, send 10^3 pkts, each of size 10^4 bits
  10 Mbps

↳ What if in 1 sec, we send everything in first 0.01 sec ⇒ we get 1 Gbps & be calm for 0.99 sec. Congestion still takes place bcz of sudden burst into network.

* so may be 10ms is better window than 1sec

Idea 2:-

window based datarate control: W = max amount of un-acked data in flight of ....



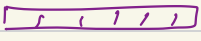
(didn't listen)

how does this do "data rate control". There is no kind of timer as there before.

⇒

$$\text{Datarate} = \frac{W}{RTT}$$ → can be ↑ or ↓ to vary data rate

This is what TCP does.



W can also affect congestion at queues also. If W↓, it is expected to have less traffic at each queue

* **Algo to decide W:-**

$W = \min(\text{Congestion Window (cw)}, \text{Adv window})$ → left out ----, can be sent to SRC via ACK

This is the one which require algo

*

Initial value of W = ?

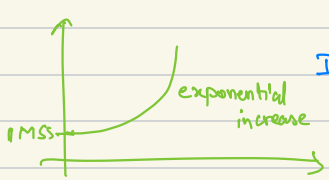
Let 10^3 pkts = 10^7 bits (1 pkt = 10^4 bits)
& RTT be 1 ms

$$\text{data rate} = \frac{10^7 \text{ bits}}{10^{-3} \text{ sec}} = 10 \text{ Gbps}$$ → Will definitely congest the network

* We think conservatively & initially send
W = 1 MSS (maximum segment size)
i.e., 1 pkt ~ 10^4 bits

} sending 1 pkt will definitely not congest network

Slow start: Start with W = 1 MSS (since initially not utilising resource)



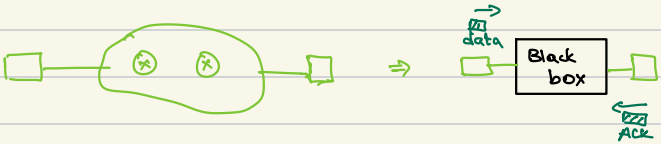
If cause congestion then slowdown

Q_1: How to know congestion happened
Q_2: How to slow down

21/10/20

Class-33 :-

TCP Congestion Control :-



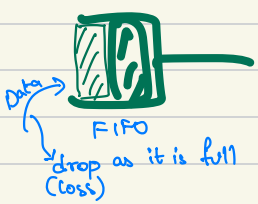
Assume a black box, and don't know any info

*

How do you infer about congestion at this queue?

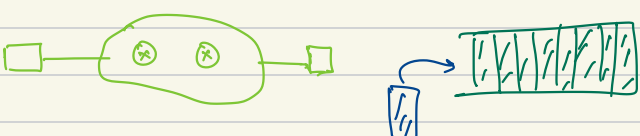
Idea 1 :- See ↑ or ↓ in RTT

Idea 2 :- Packet loss



Q) How do infer that pkt loss occured?
(said one way . didn't listen)

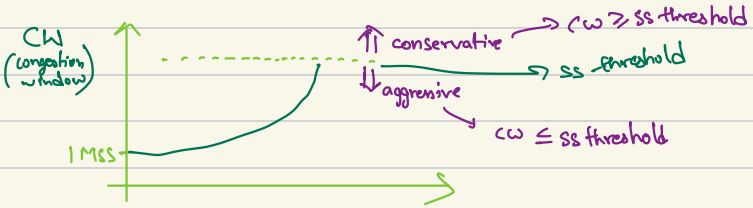
Idea 3 :- ECN: Explicit Congestion Notification
(was research interest in 80's)



Before congestion happens it-self, we inform DST/RCV by setting a bit in TCP hdr of packet

(Sort of a hack. In btn router take part in TCP)

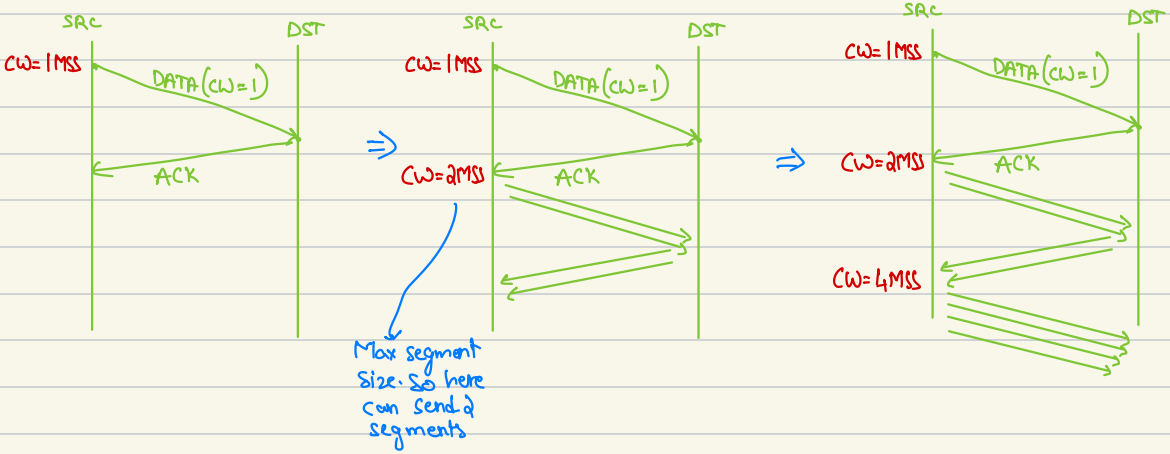
* Another issue discussed: Range of available bandwidth is very high (10 kbps - 10 Gbps)



both are needed,
- if we only do linear, it might take forever to find true bandwidth
-

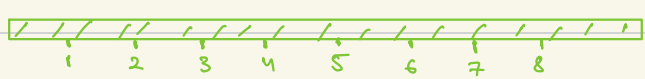
* How to practically increase CW:-

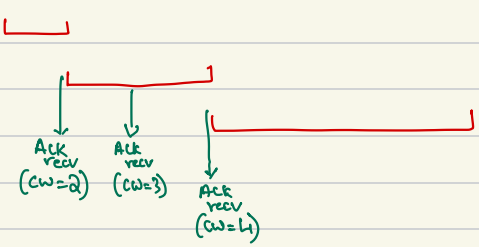
① Slow start: Double CW for every ACK



* On receiving an ACK,

CW += 1MSS (1+1+2+4+8 .... exponential ↑)

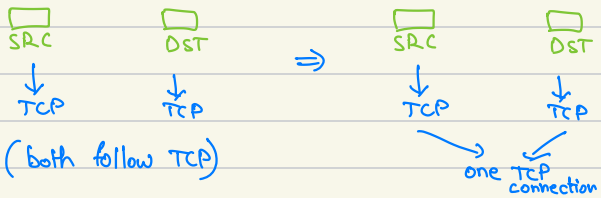




- Moving the window to right
- Increment CW by 1

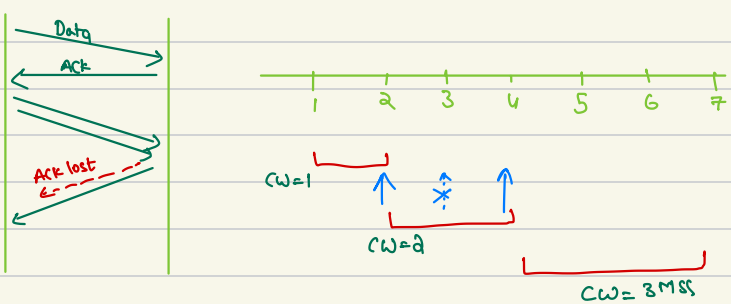
→ Two thing happening here

Note:- TCP isn't a Network level thing all have same parameters



RFC 5681: (A TCP protocol)

*



② Additive Increase:-

* cw += 1 per RTT (in additive inverse)

⇓

cw += x per ACK

How much should x be?

In cw, how many segments are in each RTT? $n = \frac{cw}{MSS}$

$n \cdot x = 1 \cdot MSS$

$x = \frac{MSS}{\left(\frac{cw}{MSS}\right)} \Rightarrow \boxed{x = \frac{(MSS)^2}{cw}}$

Note :-
Whenever 1MSS 2MSS is written, it is like 1 * MSS , 2 * MSS

RFC 5681 :-

22/10/24 Class-34 :-

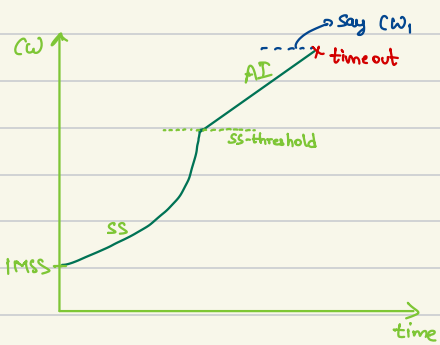
TCP congestion control :-

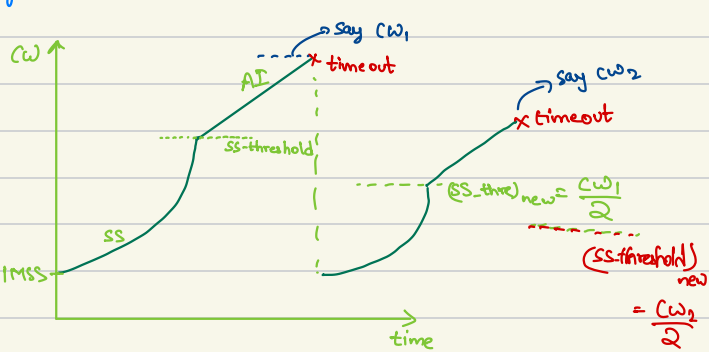
* (Recap) Slow start : cw += MSS per Ack , Additive increase : cw += $\frac{(MSS)^2}{cw}$ per Ack
* What should we do when pkts dropped?

* Congestion avoidance

AI-MD :- Additive ↑ & when a pkt drops, exponential ↓
(drastic change when even little bad occurs)

* TCP TAHOE :- (one early version)
(very aggressive in ↓ window size) (congestion avoidance)



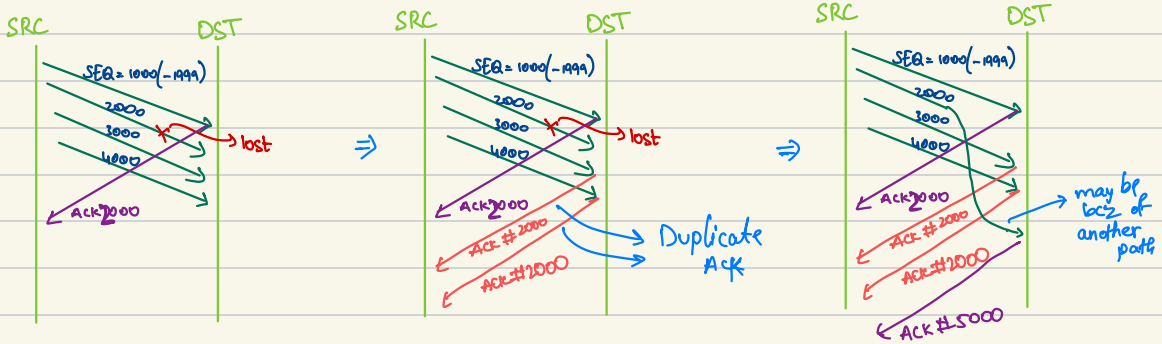


timeout : (if Ack not received in ~~RTT~~ RTO time after sending a pkt)

* This was not default. It is drastic to wait until timeout
Default is TCP Reno

*Before TCP reno, we'll see what & why & how are ideas diff from TCP Tahoe

Can we assume pkt lost when Ack#2000 received multiple times?



* Triple duplicate loss :-

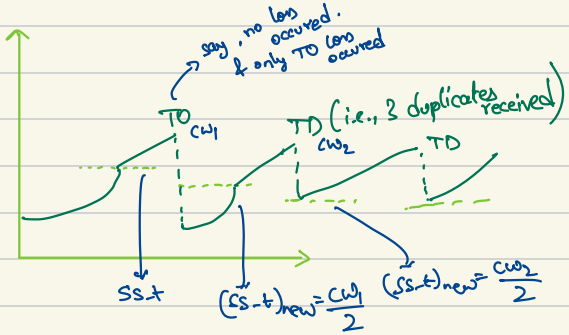
If 3 duplicates received ⇒ assume pkt loss

* Should we bring CW to 1 MSS after loss?

No

→ TO (time out) loss is a drastic congestion
→ TD (triple dup) loss is not that drastic

TCP Reno :-



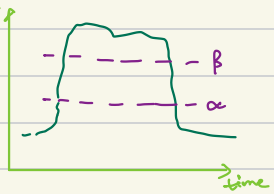
fast recovery
fast retransmission

TCP Vegas :- Parameters, not known how to set. so researchers didn't feel good

26/10/24 class-35 :- (missed 10 min)

TCP Vegas :-

- If queue getting full & if congestion is increasing, Reno waits until pkt loss occurs. But Vegas doesn't.
- Slow start, TO loss, TD loss ⇒ same as Reno
  But when cw > ss_threshold, it does "Congestion Avoidance"
- RTT



(didn't listen exactly)

- stop at α, cw is fixed
- stop at β, cw ↓ by additive decrease

Vegas Rules :- (For congestion avoidance)

* Base RTT - min observed RTT in some recent time window (say W)
  RTT - current (smoothed) estimate of RTT

* Suppose no congestion, then RTT = BaseRTT

* But if they are different,

$$\text{Expected rate} = \frac{W}{\text{BaseRTT}}, \quad \text{Actual rate} = \frac{W}{RTT} \Rightarrow \boxed{\text{diff} = W\left[\frac{1}{\text{BaseRTT}} - \frac{1}{RTT}\right]}$$

* If Diff < α ⇒ A.I (as in Reno)
  α ≤ Diff < β ⇒ Freeze window
  β ≤ Diff ⇒ Decrease W by 1 MSS per RTT
* Suggested α = 30 kbps, β = 60 kbps

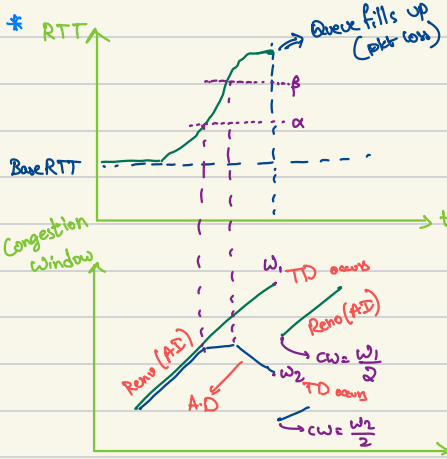
* Other issue than α, β parameters, Question was can VEGAS be used when some other in Network uses other TCP's ?

plan to change window as

Vegas ⇒

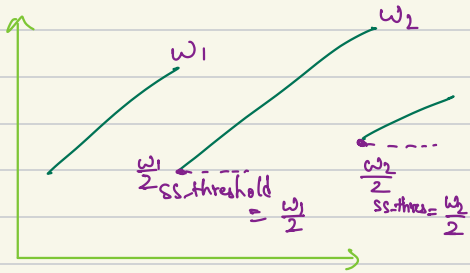
freeze window if diff ≥ α

Even decrease it if diff ≥ β

* 

Note:-

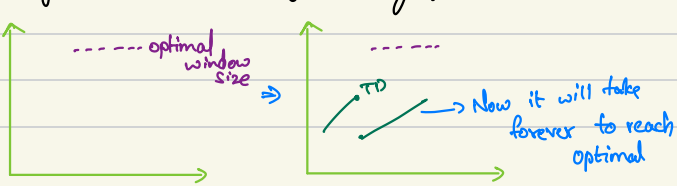
It is not that ss_threshold always decreases



- so overall we see, when both are running VEGAS get unfair

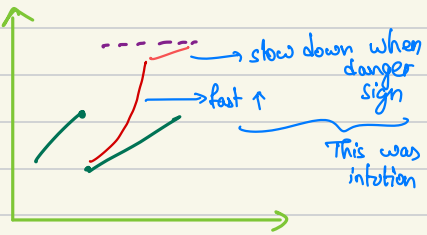
- TCP Reno was default for long time (1988-2012) by many OS
- TCP Cubic ⇒ now default (Linux, MAC OS)

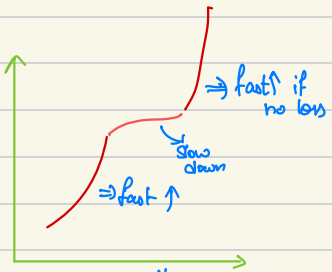
Q) Why TCP Reno was good enough? We have much speed now. Why?



What if +5 MSS instead of 1 MSS ? (didn't listen)

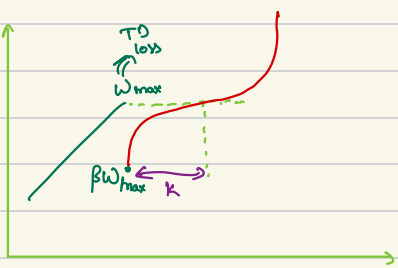
*





Looks like "CUBIC"

* So researchers actually tried CUBIC



$cwnd = C(t-K)^3 + W_{max}$

at time T after loss event

$\left(K = \left[\frac{W_{max}(1-\beta)}{C}\right]^{1/3}\right)$

At T=0

$cwnd = -W_{max}(1-\beta) + W_{max}$

$= \beta W_{max}$

$(\beta = 0.7)\ (C = 0.4)$

* Next 2 weeks ~~Application~~ Layer

28/10/24

class-36 :-

* APPL
  Transport
  NET
  DLL
  PHY

  - DLL, PHY many companies. But some survived. 4G < WiMAX WiMax was ahead of 4G. But due to economics (companies said upgrading to 4G easy
  - NET is study with IP
  - Transport has UDP, TCP → does nothing → does more than required

If you want diff from transport, can build in App layer. Ex:- QUIC, google's. uses TCP cubic

Others no big innovation. But App layer have huge innovation

Ex:- (Have to be in right place in right time)

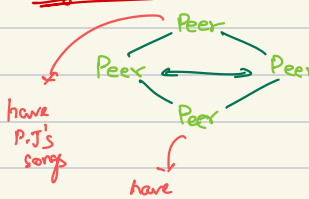
I) Client-Server:-

- Smartpnx (10 yrs ago) (IITD sir student made)
- OLA (IITB alum)
- HTTP (web) → by high energy physicists (CERN)
- Facebook, ...

Another technology,

II) P2P :- (peer to peer)

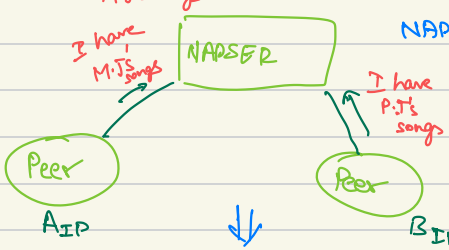
1st generation:- NAPSTER



Q) How do we do search here?

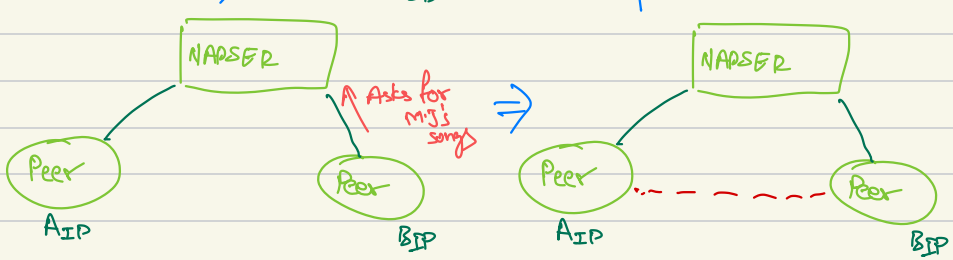
Ans given NAPSTER

(NAPSTER was before google)



NAPSTER maintains list of who have what

Peer ID	Songs (title)
A_{IP}	M.J. ..
B_{ID}	P.F ...



* Bcz of some legal issues, NAPSTER closed
* It has some issues as centralised server

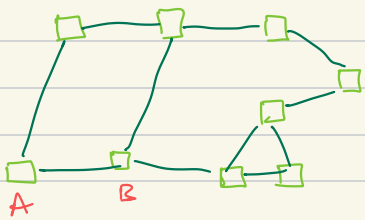
Problems with Centralised server:-

1. Single Point of failure:-

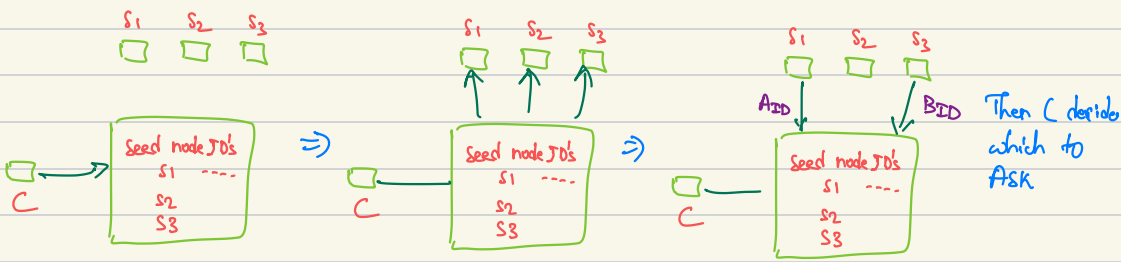
Two aspects · Technical · Legal

- Technical : i) Power cut ii) DDos attack (Just sending many requests to single server to get down
- Legal :- If it had to shut down, just go to 1 place

2nd generation :- GNUTELLA



- There are some seed nodes S_1 S_2 S_3
  They'll keep track of recent transfers
  Also keeps contacting periodically, to know if they're still in network (hello messages)
- In the application software of every node,
  Now C requests S_1, S_2, S_3

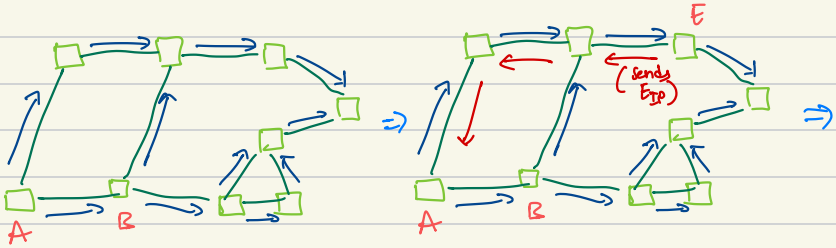


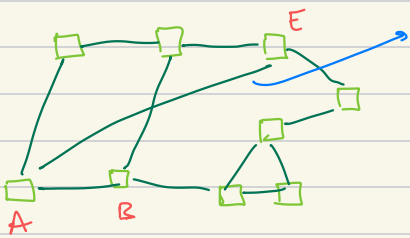
* Seed doesn't have much info, it doesn't want to get more involvement.
  This is where innovation came in
  Bittorent.

Idea 1 :- When a file comes, I'll broadcast that I have. But then each node become knapster server.
Good for small networks. Not scalable

Idea 2 :- (Flip Idea 1) Broadcast the request

Note:- Assume everyone honest





→ Edge added bth them in new graph

* Gnutella did more advanced.

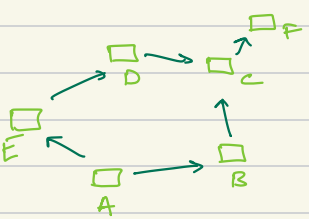
Expanded ring broadcast :-

29/10/24 class-37 :-

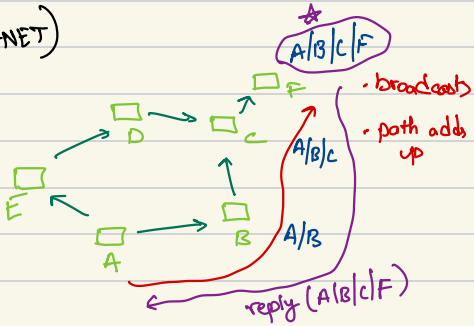
* 15 min Recap (didn't listen)
* For file request, A can local broadcast (i.e., limited depth) (=7 initially) (global can make bandwidth full at all instants)

→ Digression

Similar to P2P, we have "Mobile ad hoc Network" (MANET)



- Tanks - Moving
- A want to search for F



- broadcasts
- path adds up

* This is called "Source routing".
  - broadcasts for path
  - DST replies with 1 or 2 possible path
  - Source decides which path data goes

Note:- Why not use D.V? All moving so weights change. Q) That risk is there normally also. After A knowing that path A/B/C/F, If C moves out? Ans:- Have to do some hacks Ex:- B does source routing

* Coming back, P2P wanted a ~~better~~ efficient way of data ...

3rd Generation :- Distributed hash table (DHT)

=>

Since diff parts world connected via P2P, can assume uniformly distributed IP .

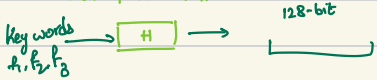
71.32.10.10

If A have to sent to, A want to know which of nodes in network has (closest IP to 71.32.10.10

- First A sends "search" to C i.e., send to closest IP neighbour
- Then C sends "search" to F

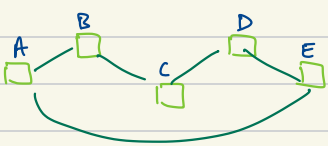
* Having above algo as alternate for broadcast, we want to bring a mapping from file name / key words → IP

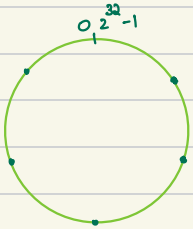
Hashmap function H



(We want to do this deterministic)

* Consider a network as below





(didn't listen)

* High Level :-

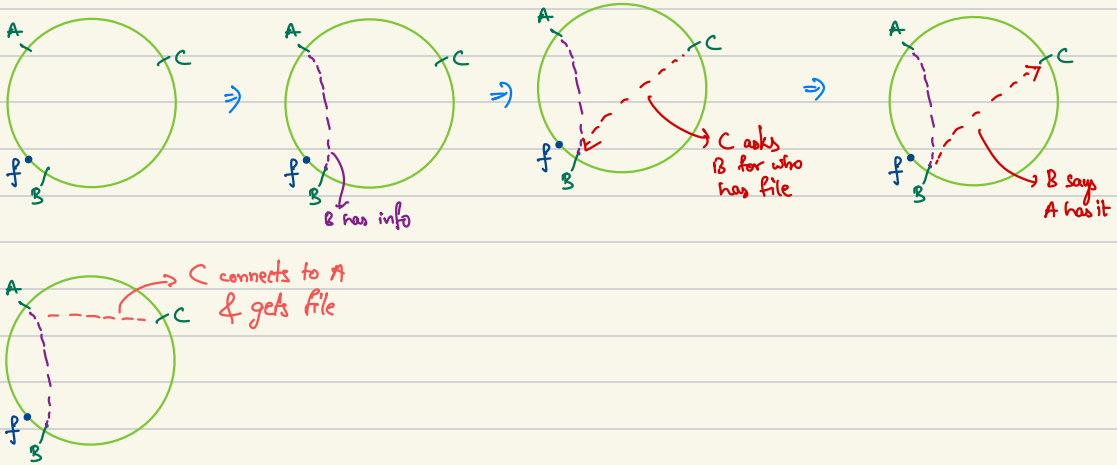
Suppose node A has a file 'f'. A finds node (say B) where IP is closest to f in common space i.e.,

$$dist\left(H(f), H(B_{IP})\right) \leq dist\left(H(f), H(X_{IP})\right) \quad \forall \text{ nodes } X \text{ in network}$$

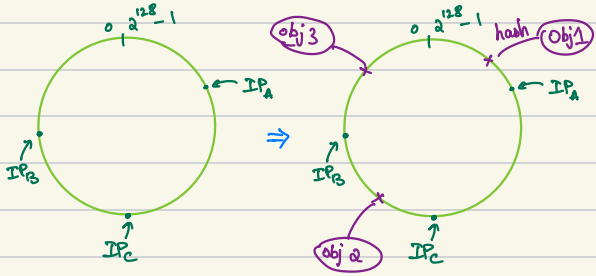
Then B maintains in a table that

file	node
f_1	A_{IP}

- C finds node closest to 'f' in common space
- C asks B: "Who is f?"
- B gives A_{IP} to C
- C connects to A and downloads file

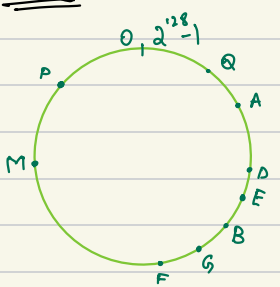


4/11/24 Class-38:-



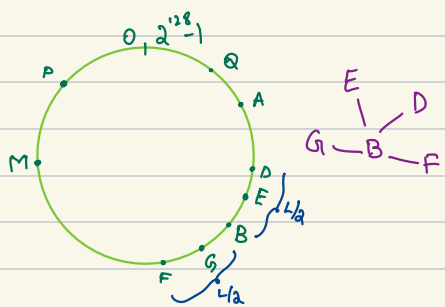
* Suppose Node C has Obj 1
  It doesn't want to say everyone that it has Obj 1
* Finds that A has IP closest to Obj 1
* Tells A that, I have Obj 1
* When B wants Obj 1, asks A (closest to Obj 1)
  Then A says C has it.
* B connects to C & gets Obj 1

Pastry :-

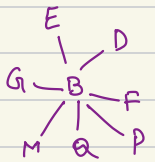


* Leaf Set: Neighbors in P2P network of any node

* if L=4, 2 on right, 2 on left (according to IP)
  L ⇒ set by administrator of network



* Apart from $L/2, L/2$ B could be connected to other nodes in network



Leaf set: Size = $L + O(\log N)$ → # nodes in Network

* If M wants an Obj 1 which is in $(ID_{B-L/2}, ID_{B+L/2})$

* For M, closest neighbour (Not sure) to Obj 1 is B,
  M asks B for Obj 1

* Then B knows D is closest to Obj 1, so asks D to see about it ⇒ Not sure
* D knows $-L/2, +L/2$ neighbours to itself &
  so knows that it is closest to Obj 1. Hence replies to M about who has obj 1.

* Node X wants to join
  * clear on board
  * bcz of longest prefix match, in $O(\log N)$ steps we reach closest

* Node A gets fails
  * Say A,B on right of X, C,D on left of X * Maintain some heart beat messages so X know A failed
  * Now X knows A failed. But want to connect to another on right
  * It knows B is rightmost & asks for info of its immediate right

* What about info in A? Is it lost?
  The copy of info at A is always sent to its $\frac{L}{2}, \frac{L}{2}$ neighbours.
  This is how node failure handled

Now-a-days BITTORENT : simultaneous download. (May be "K-" has has bandwidth) more complex. Seed & how to know which all nodes have file

Domain Name System :- (DNS)

* We don't prefer 32.75.5.9 IP. We want google.com, iitb.ac.in (URL)
  But these can't be understood by lower L4 layers. They understand IP, MAC (or) port number for Transport layer
  So Appl layer should take care

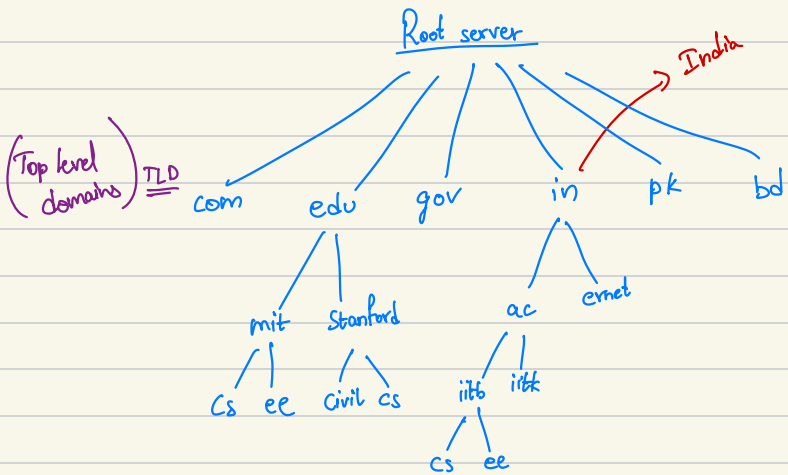
* Another adv:
  Today google may have 8.8.7.5, if it want to change 8.8.7.3. It has to be changed by DNS. & one can still use "google.com"

* Flexibility similar to IP (machine's IP can change anytime)

* Has to be robust.

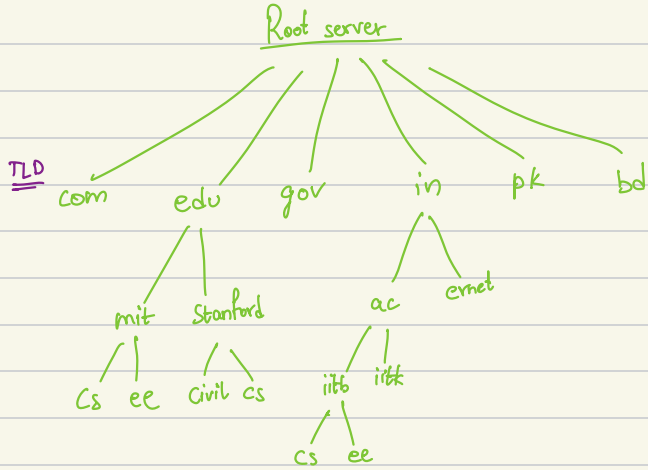
Naive :-
Maintain single server, which has URL ↔ IP mapping. One queries, What is IP of a URL.
Problems: Single point failure, Not scallable

Ideal :-
We have heirarchy



Root server, in, ac, iitb, cs all have a "DNS server"
↳ Has mapping

5/11/24 class-39 :- (didn't listen 10min)



* DNS server of parent domain should know the DNS servers of immediate children
  - i) Name
  - ii) IP address

  } necessary info

(But extraly, in can remember cs
But not necessary)

* Why this is scallable?

Say we want to add surya.cse.iitb.ac.in . We only need to change at cse DNS server.
If centralized server ⇒ changing difficult

*(didn't listen 5min) 13 servers there. What if all made down?

* A. root-server. net = 201.32.5.23

We keep multiple physical machines with same name. We don't care where exactly it goes
↳ called ANYCAST

(advertisiments --- BGP rules ----)

- UNICAST = single destination
- BROADCAST = All nodes are destination
- MULTICAST = subset of nodes are destination
- ANYCAST = Any node of a subset can be destination

↳ Had economic issues. Why not broadcast have that?
broadcast done locally. They planned to do multicast over all network.
(like cricket live, send same pkt to many at a time)

Resource records:-

<NAME, VALUE, TYPE, CLASS, TTL>
CLASS: IN (internet)
TTL: how long...

TYPE:

- CNAME → Canonical name (alias)
- MX → Name of email server of domain specified in NAME field (cs.mit.edu, ee.mit.edu)

7/11/24 class-40 :- (missed 20 min)

* HTTP :-

(missed format)

Requests :-

For example, GET <url>

→ Request operations :-

GET → Retrive document

HEAD → Retrive Meta Info (eg:- last modified, length..)

OPTIONS → Available options (HTTP version at server)

POST → Give new information to server

PUT → Store/modify info at specified URL on server

DELETE → Delete specified URL

giving info to server

Responses :-

For ex, (startline) HTTP/1.1 202 ACCEPTED <CRLF>

(HDR) Content length:

Expiry time:

<CRLF>

(Body)

Request codes :

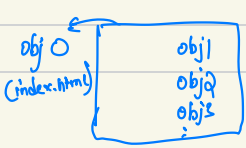
1xx = informational

2xx = succes

3xx =

HTTP & TCP :-

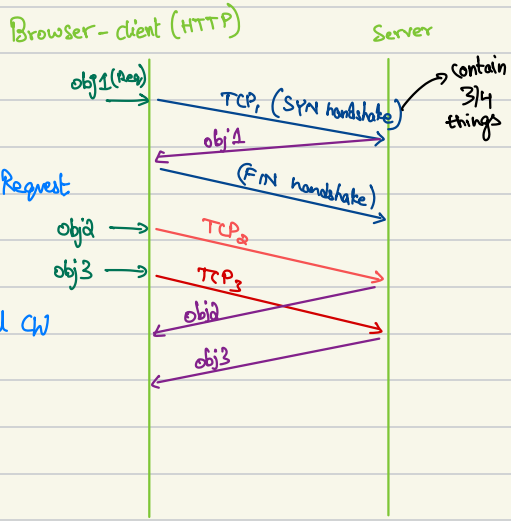
Say a webpage (this itself is an object) contain some objects



Browser → requests → HTTP → TCP

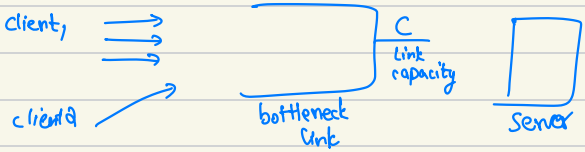
objects one after another is not a good idea

Q) Send all requests at once / sequentially?
How many TCP connections?



HTTP does a naive approach
* New TCP connection for every GET Request

Issues with this :-
1) Each TCP connection takes time to learn optimal CW
2) Overheads (hand shake)
3) Server state is large due to multiple TCP connections being open
4) Fairness
(Say client1 has m TCP connects
client2 has 1 TCP connection



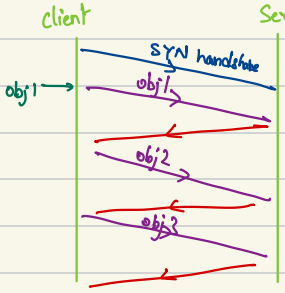
(m+1) TCP connections

Each gets $\frac{C}{m+1}$ (assuming same TCP version, RTT etc--)

$\frac{Cm}{m+1}$ → client1

$\frac{C}{m+1}$ → client2

* So they invented HTTP 1.1 :- (around year 2000)
All requests in single TCP connection, called "Persistant TCP connection"



We have a queue & all are done sequentially

* HTTP 2 :- (around 2015) issues in 1.1 :- What if obj 1 takes more time but 2 takes less time

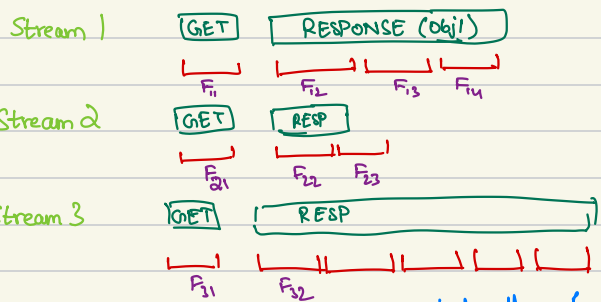
(best of both 1.0, 1.1)

↳ Head of line blocking :-

obj n' gets delayed ....

- Single TCP
- divide requests into streams

Say we have 3 objects with 3 streams



but allows frames from diff streams in the TCP

Same TCP

(may endup seeing obj2 before obj 1)

* HTTP 3 more optimization