TCP Header (Day-9)

Source Port (16)	Destination Port (16)
Sequence Number (32)	
(32) Acknowledgement Number	

Header length (4)	6 bits reserved	URG	ACK	PSH	RST	SYN	FIN	Windowsize adv. or window
Checksum (16)								Urgent pointer (16)
Options (0-40 Bytes)								
Data								

adv. → advertisement window (16)

Only option feild is in bytes.

Top 5 rows are of 32 bits and definitely acquired.

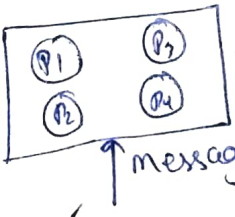
So,

Min. Header size = 4 x 5 = 20 Bytes

Max. ———— = 20 + 40 = 60 Byte

Source, Destination Port and Socket :-

with 16 bit 65,536 numbers are possible (0 - 65,535).



Each process have a port no.

TCP header contains destination port no. Message reaches destination and host transfer the message to process or port using multiplexing or de-multiplexing.

→ TCP is end-to end protocol

→ TCP can do multiplexing and demultiplexing.

Well known port no.

HTTP → 80 FTP → 21
Telnet (Remote login) → 23
SMTP (Email) → 25

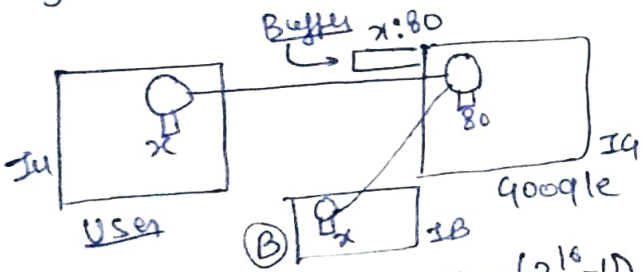
well known services have port no in range (0–1023)

Reserved port no. (1024–49,151)

IANA → Internet assigned Number authority.
↳ It allot port no. to services.

(49,152 – 65,535) → used by general public.

→ 'netstat' command use to get all open port no.



x lies between (49,152 → $(2^{16}-1)$)

(1) TCP is connection oriented.
Connection oriented means some buffer are reserved at both side before sending data.

(2) TCP apply at 'transport layer'.
→ Packet first gets in buffer and then process.

→ If (B) is another host chose the same socket as user choose then it will create a problem.

So to identify the connection uniquely only port no is not sufficient. We can solve it using IP address as
IA : IB instead of x : 80.
But if another process from same host generate same request then it will again create problem.

So, we will use combination of both.

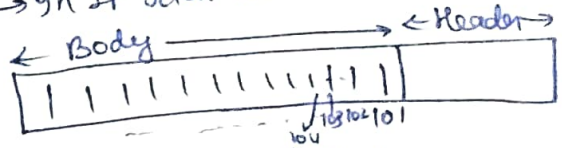
IP (32) + Port (16) → Socket (48)

→ Socket is definitely an unique number.

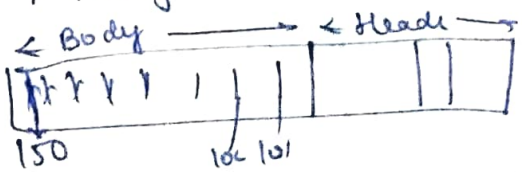
Sequence No., Acknowledge No., and Random initial sequence no.

Sequence no. (32 bit)

→ TCP is a byte stream protocol. (Every byte count)
→ IP is a packet stream.
→ HDLC is a bit stream protocol
→ Java is Byte stream.
↳ In IP Iden. no. used for identification



Packet (general name)
→ segment at TL
→ datagram at NL



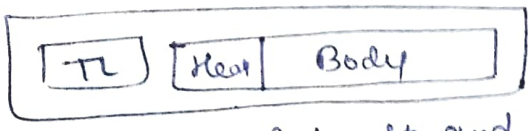
Given a number to each byte in body and put the value of first byte in seq. number field of header.

Acknowledgement number is (last byte [seq. no + 1])

So, Ackno. number = 151

So, task is how to get last bit number?

Soln Every TCP segment travel in IP datagram.



It has Total length and header length. So we can find using TL-HL

HL → header length

TL-HL gives length of body

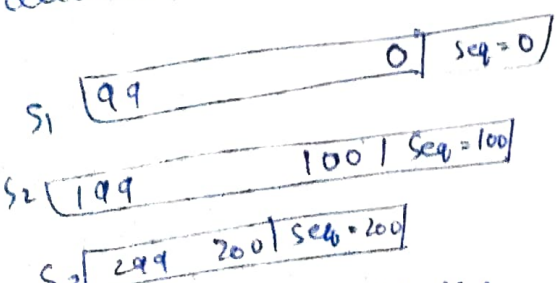
So last byte number you will get

(size of body) + sequence number.

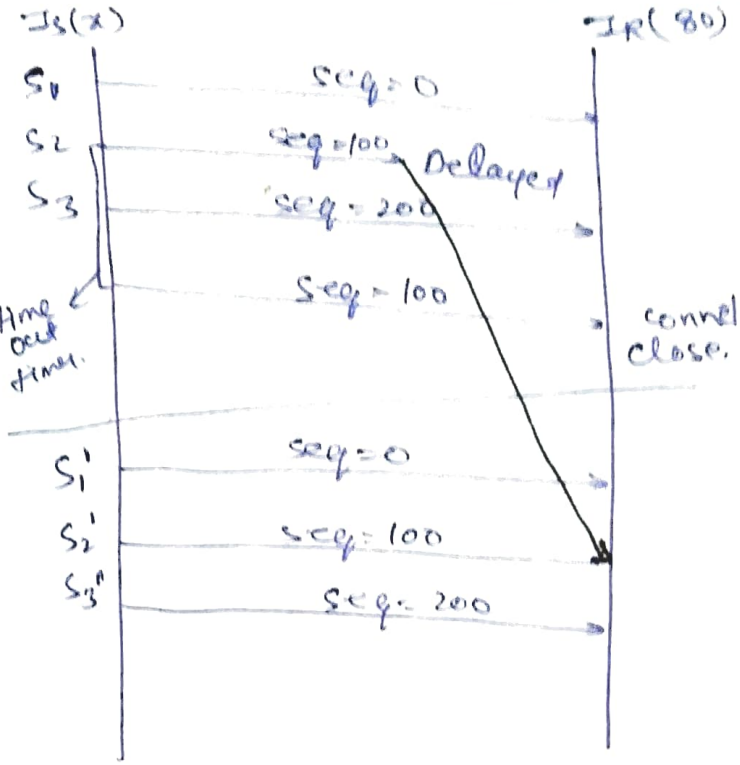
With 32 bit no. of seq. no possible = $2^{31}-1$

Random Initial seq. no

Assume that you want to send 3 byte stream segments to receiver and seq. no. allotted to each segment is



and you transmit the packet from sender end (IP add Is and port no x)



→ Packet doesn't received at another end with seq=100 in first attempt so we send it again after time-out.

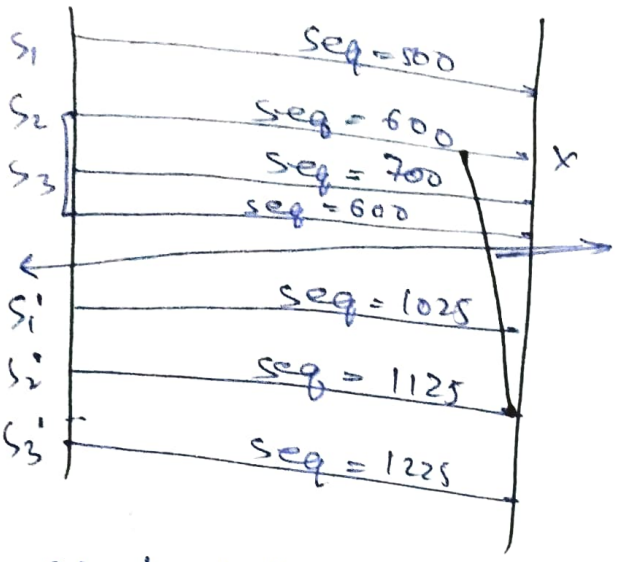
And just after few microseconds of closing connection, we again send 3 segment with same seq no and different byte streams. at that time if packet delayed with seq no 200 get arrive at the same time when packet with seq no 200 get arrived in IInd connection.

(3 पैकेट दोनो एक साथ आ गए तो Collision हो जाएगा।)

To resolve this scenario we use random initial seq. no.

→ Probability of getting same seq. no. in random initial seq. no. is equals to $1/2^{32}$

$= 0.000\ldots025$ which is very less.



If packet with seq = 600 collide with seq = 1125 then receiver can identify that seq no 600 packet is from closed connection and discard it.

→ TCP uses random initial sequence number.

Good for Academical Purpose

Wrap around time: —

TCP uses 2^{32} seq no.

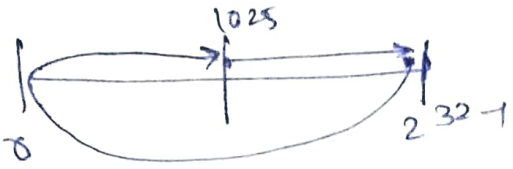
So $2^{32} = 4 \times 2^{30} = 4G$

Total 4G sequence number are possible

If each stream is byte stream then 4GB data atmax can be transmit.

So, to send more data we have to wrap around same data.

In case of random initial seq. no you will also get $2^{32}-1$ seq. no.



WAT → Time taken to wrap around.
(After how much time you will take to start from same seq. no).

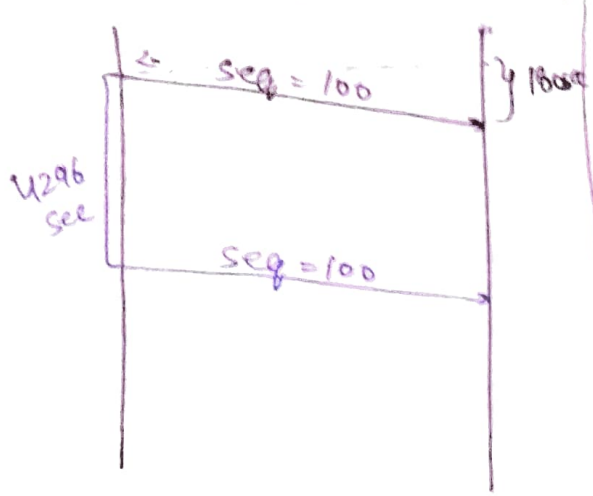
WAT → depends on range of sequence number.
→ depends on Bandwidth
More the Bw more the seq. number exhausted in certain time.

Bw → 1 MBPS
10^6 B → 1 sec
10^6 seq num. → 1 sec
1 seq num = 1 μsec.

2^{32} → 2^{32} μsec

WAT → 4294.96 sec
↳ Just remember this not to byheart

Life Time → In todays Internet Packet can alive for 3m (180 sec) in path between Sender → Receiver.



You can generate same seq no. after 4296 sec but upto this time first packet with seq no = 100 will get dead.

So seq. no. will not create any collision problem till the condition satisfied as

WAT > Life time

But problem start if

LT ≥ WAT

Assume Bw = 1GBPS

then,

10^9 B — 1 sec

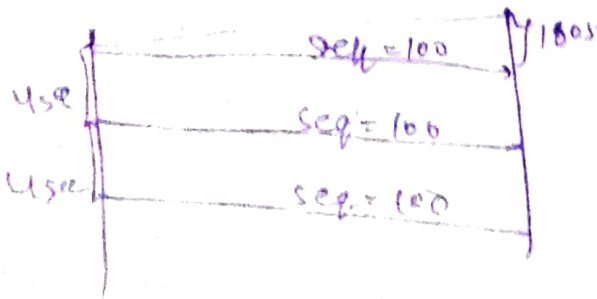
10^9 seq → 1 sec

1 seq → $\frac{1}{10^9}$

2^{32} seq no → $\frac{2^{32}}{10^9}$ sec

WTA → 4.3 sec

WAT < 180 sec



There are more streams with same seq. no.

So, solution can be reduce the Bw value but we want to increase it bcoz we want to send more and more packets in seconds.

Next solution can be increase the size of seq. no. feild.

How many extra seq. number we needed?

Bw = 1GBPS

LT = 180 sec (Todays)

we dont want wrap around before 180 sec.

→ 1 sec = 1GB

1 sec → 1G seq. no.

180 sec → 180 × 1G seq no.

we need min → 180G seq no. to avoid wrap around within life time.

no. of bits required

$\lceil \log_2(10^9 \times 180) \rceil$

≈ 38 bits

Extra bits needed

= 38 − 32 = 6 bits

You will get these 6 bits from option feild in TCP header as time stamp

Min size of seq no

= Bw × LT

No. of bits required (N)

$= \lceil \log_2(LT \times Bw) \rceil$

Extra bits = N − 32

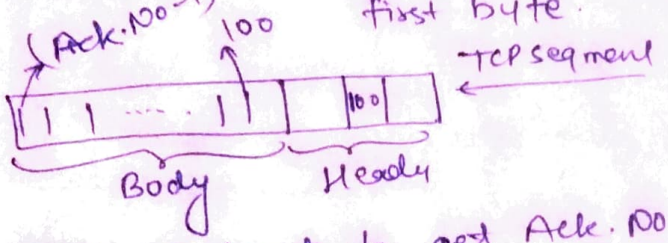
Header length (4 bits)

But Header size is 20-60B so from 4 bit we can form 24 (16) combination. So task is to represent 60 B with 16 combination

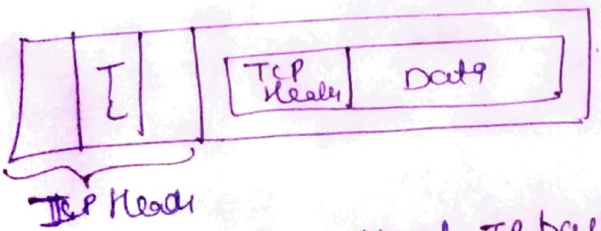
→ Already discussin IPv4 Header.

Seq. number: seq. no. of first byte

Ack. Number: seq no. expected in next packet first byte.



It is a task to get Ack. no. we know that TCP packet travels in IP datagram



TL → Total length of IP packet

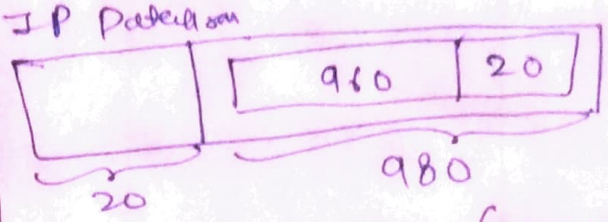
Ack no - [TL - (IP Header) size - (TCP Header)]

Q TL and HL of IP datagram is 1000 and 5 respectively. Inside IP, TCP segment header length is 5 and seq no is 100. find ack. no.

Soln HL=5 means it is scaled down so multiply with 5.

HL = 4 x 5 = 20

IP Packet



Last byte seq. no = (960 + 100 - 1)
= 1059

960 no. in range (100 — 1059)

Ack. no = 1059 + 1
= 1060

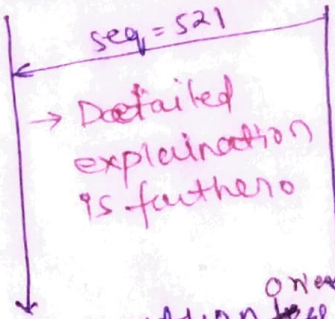
TCP connection establishment

Flag → 1 bit information

SYN flag → Synchronization flag.

Ack flag → Acknowledgement flag.

(80, Is) Server (x, Ic) Client



TCP is connection oriented so,

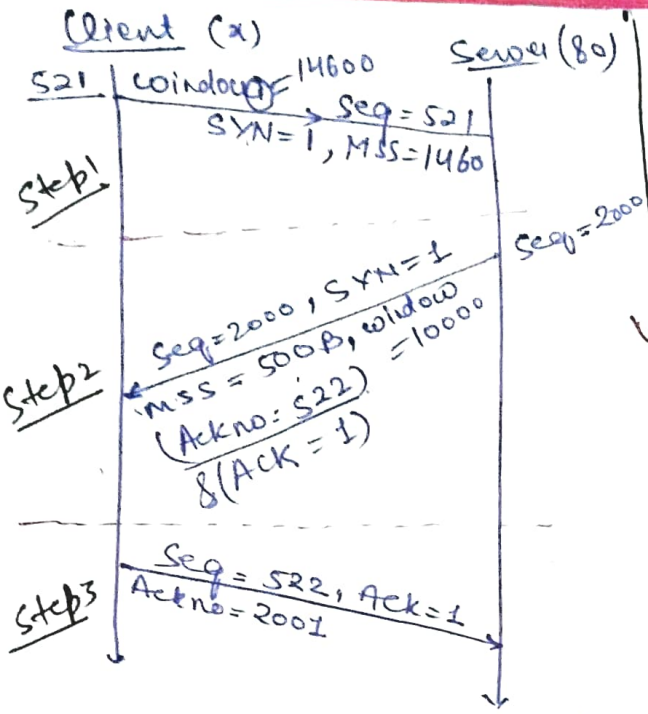
→ Connection establishment

→ Data Transmission

→ Connection termination

'x' → port no. choose by OS at client side

Explanation of fig. → Cont.



① packet will have only header so that sever can take relevant info. It is use to establish connection.

→ MSS → Maximum segment size
It can be vary from N/w to N/w

→ client send MSS to avoid packet that required fragmentation or segmentation; when receive to client.

→ MSS kept in option feild of header.

→ TCP is full duplex.

→ window size is receiving buffer from sever.
means sever can send $\left(\frac{14600}{1460}\right) = 10$ packets without acknowledgement
It is used for flow control and denotes the capacity of host as a receiver.

→ SYN = 1 means seq = 521 is first seq. no.

Since MSS of sever is 500.
So both client and sever work at MSS = min(1460, 500)
⇒ MSS = 500B.

→ SYN = 1 means byte with seq. no 521 get used) so sever send ack no = 522 means, sever is waiting for data with seq no - 522.

→ TCP uses 'piggback acknowledgement' and 'pure acknowledgement' both.

So, 3 steps used for connection establishment (in fig.)

① → Each SYN = 1 consume one seq. no. (step 1)

② → Pure Acknowledgement (Ack = 1) doesn't consume any seq. no (step-3)

③ → FIN = 1, consumes 1 seq. no.

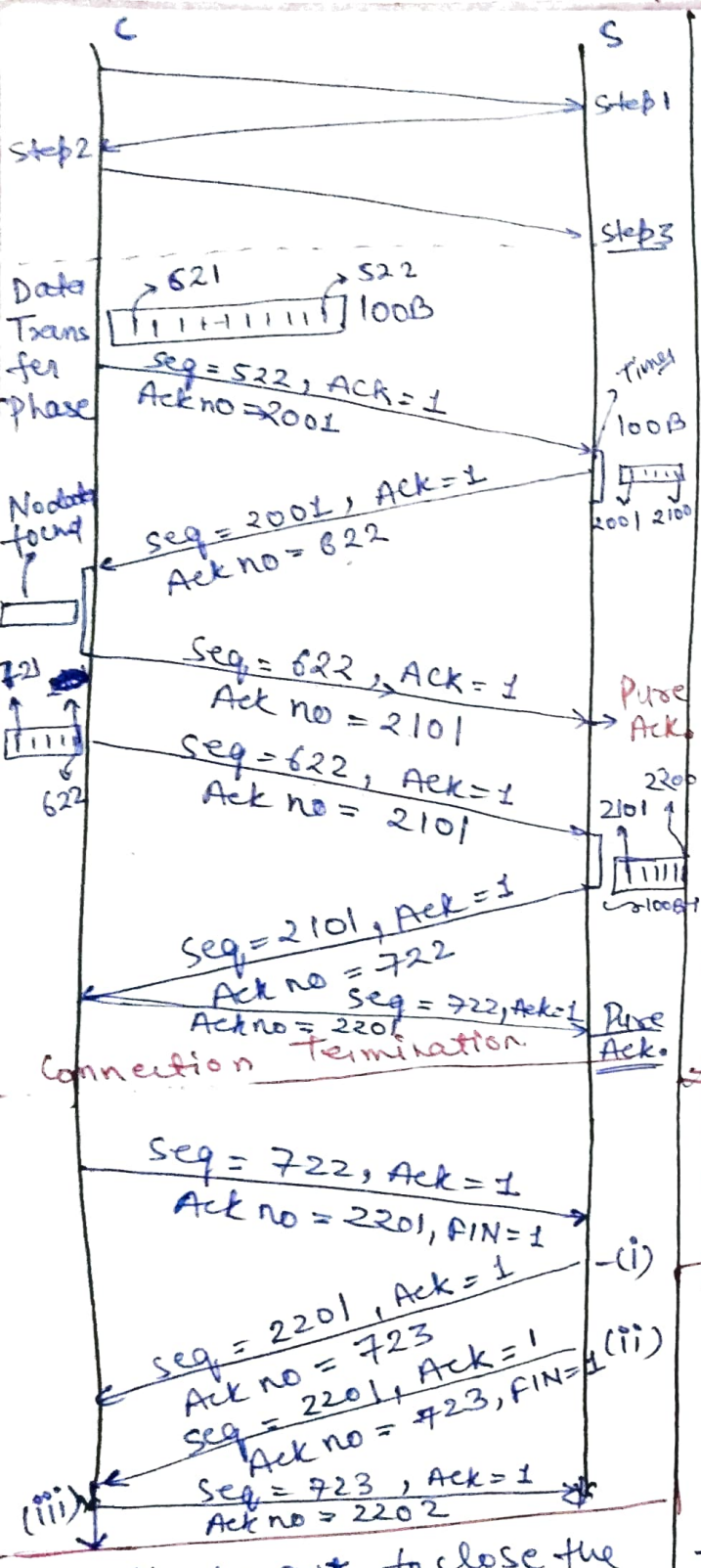
④ → 1 DataByte take 1 seq. no.

1 Step → Request packet
2 Step → Reply packet
3 Step → Acknow. packet

So, this way / method also called 3-way hand-shaking.

TCP data transfer after Connection establisment

→ Same time line diagram
Cont...



→ client wants to close the connection

→ FIN → Finish flag

client said to 'S' that free up the resources, I have no data to send.

→ Ack no means acknowledgement number assume in return messages

→ At sender side coe have time out timer and receiver side coe have acknowtimers

→ Timer denotes the time taken by sender to form a packet with valid seq. no.

→ If packet is not formed in given time it will send the pure acknowledgement

→ coe assume that both got agreed at MSS = 100B

SYN	Ack	
1	0	→ Req. segment (I)
0	1	→ Ackno. is present in header
1	1	→ Reply segment (II)
0	0	→ Not possible.

TCP connection Termination

full duplex

→ FIN flag take 1 sequence no, as SYN flag.

→ Client close the connection after sending FIN. After closing connection.

Data from C → S (X) not possible
Data from S → C (L)
Ack. from C → S (Pure) L
Ack. from S → C (Piggy back) L

→ 3 packet are required for connection establishment and 4 packet are required for connection termination in worst case but you can also terminate in 3 packets

→ **ACK**: Indicate whether the Ack. no. is valid or not

→ **FIN**: Request for connection termination.

PSH Flag: Push flag

On interactive applications this flag is being used.

AL ↓ Data flow in bytes

TL [ ] segment form

If we get only single byte data from AL to TL then to send that 1 bit we have to add the header of 20 bit at both TL and NL and this way efficiency get reduces.

So, if data size << 1460 B at TL then we push it segment and send it using PSH = 1.

Ex → Chat, Telnet, putty, SSH, Rlogin etc.

If AL send 1 bit then TL save that bit in buffer and when data of size ≈ 1460 come it include all data in segment and send it. but if,

AL ↓ (1 bit, PSH=1)

TL [ ]

then TL send the data at the same time without saving in buffers.

URG Flag and Urgent Pointer

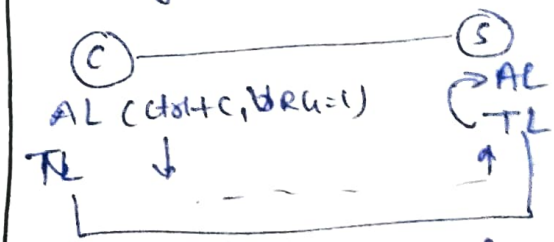
URG → Urgent flag.

→

→ We are working using remote login

→ Assume that we are using Telnet service in which (Ctrl+c) used for logout.

Suppose that you send three files to S and suddely you got that there is an error in any of first 3 files and you want to logout so you send (ctrl+c) but this logout command is at end. So to execute it first before all the other 3 files we do some modification on AL of Client as

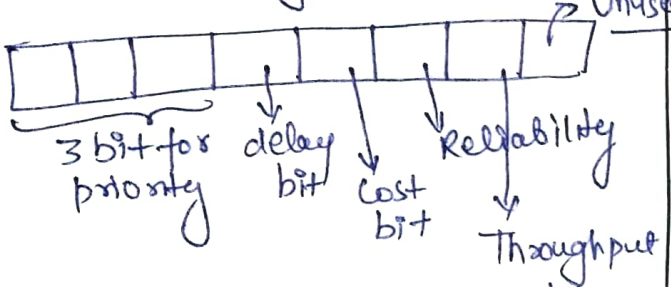


So TL at S'end perform command (Ctrl+c) prior to all three files.

→ URG = 1 get identified at TL but routers doesn't have properties to work on TL.

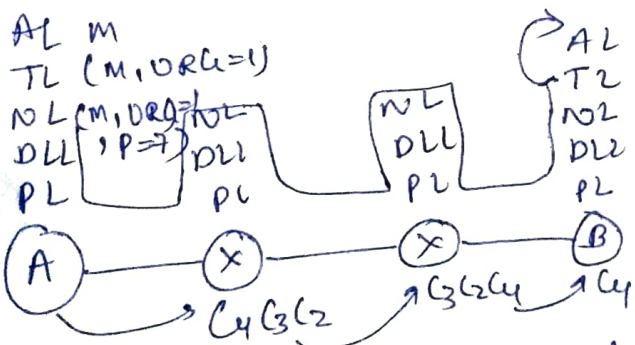
So if there are routers in between client and server then how we can identify the which command have URG=1. for this purpose use use priority.

→ IP datagram contains TCP packet have a field as Type of service (8 bit)



with 3 bit max. no. possible is $(111)_2 = (7)_{10}$

So if priority = 7 means max. priority.



Assume that C4 is command with URG=1 and it has to execute 1st then set its priority as 7 at NL. At NL of each router it will get verify that C4 has max priority and get execute first or prior to C3 & C2.

Delay=1 → means send packet to path where delay is least.

Cost=1 ! for low cost

Reliability: for High reliability

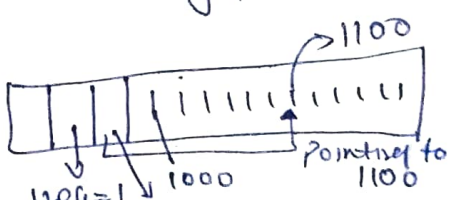
Throughput: path for High throughput

Last bit is unused in type of service.

If URG=1 and URG pointer is pointing to some bit of data.

URG pointer is used to indicate that some part of data upto which pointer is indicate is urgent to send not complete data.

Ex → Seq no. = 1000
URG = 1
Urg pointer = 100

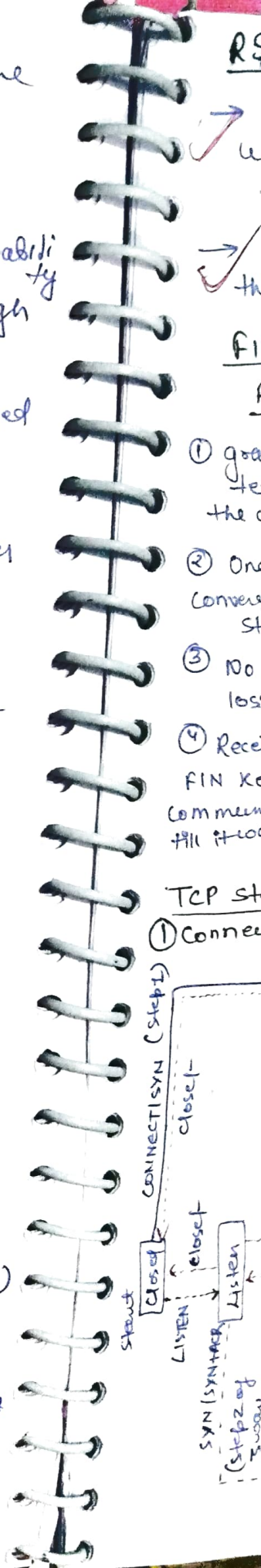


Urg pointer = 100 byte

So, 1000 — 1100 (101 byte) are important.

So if URG pointer = n then (n+1) byte are important

→ URG flag take care of out of order data.



RST flag

→ Use when something unexpected happen in the network.

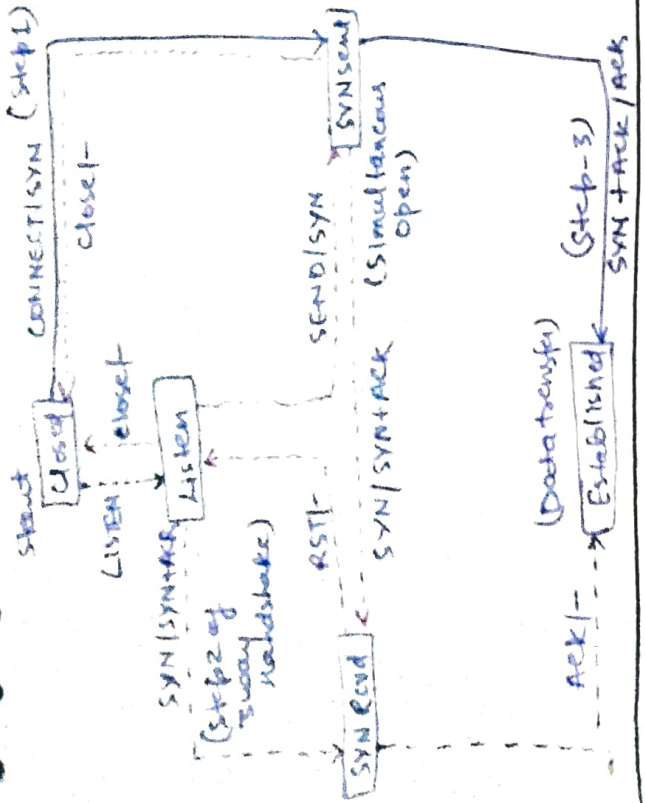
→ Reset is used to terminate the connection.

FIN vs RST

FIN	RST
① gracefully terminates the connection	→ abruptly tells the other side to stop commun^{n}.
② One side conversation is stopped	→ whole conversation stopped
③ No data loss	→ Data discarded
④ Receiver of FIN keeps communicating till it want to	→ Receiver has to stop the communication.

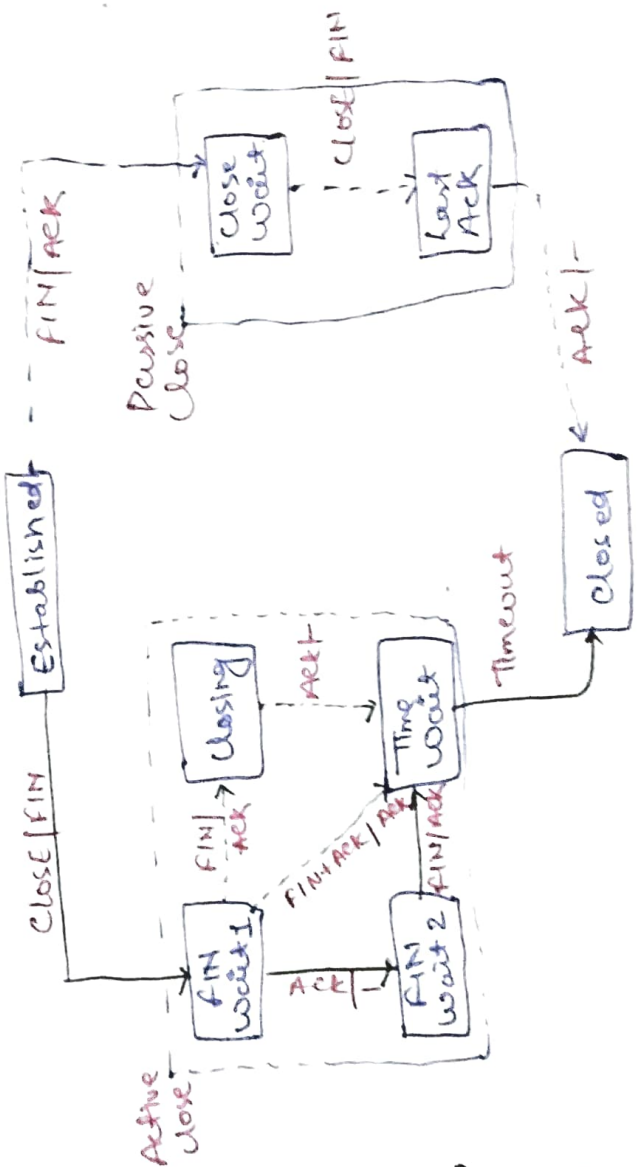
TCP state transition diagram-

① Connection establishment



- - - - - → Server
——— → Client
..... Unusual events (Not Required)
(Event / Action Pair)

② Connection release

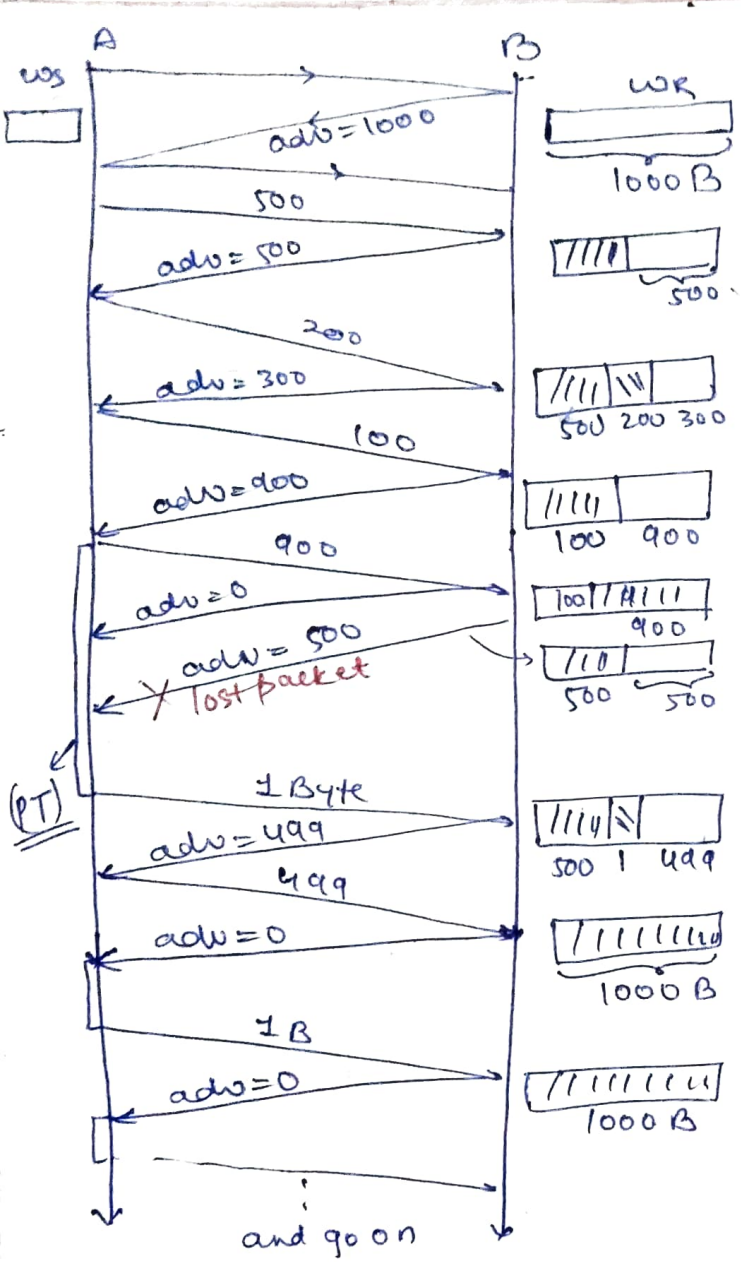


TCP flow control using advertisement window.

Window size or Adv. window (16)

→ use to manage flow control

→ we assume the half duplex connection but it will apply to full duplex too.



→ first 3 flow for connection establishment.

→ when 100 B data comes from A upto that time the data of window (700B) sent to Appn layer so it was empty.

→ when A sends 900 B then buffer get full after sometime when 500 B space get empty B sends an ack to A but it get lost.

→ After persistent time A send 1 Byte again to check that B has space or not and B will send ack again.

This way complete flow control manages.

PT → Persistent Timer.

Disadvantage –

(1) Max. no. representable for buffer is (65,535) if receiver have more than 65,535 space then it will only represent 65,535 but we can take more bits from option field.

Assume you require 25 bits to represent data then 16 from adv. window + 9 from option field.

→ Adv. window used to manage flow control dynamically.

TCP Checksum (16 bit)

TCP checksum consist of 3 fields

(1) IP header
(2) TCP header
(3) TCP data

But IP header have lot of sections and any one of section might get change in between the path. So we took some part of it.

IP fields to include -

① SIP ② DIP
③ Protocol ④ TCP segment length
⑤ 8 bits of 0's.

All 5 feilds combine to form a pseudo IP header.

IP header have all the feilds but only 5 feilds use to compute checksum.

→ We use IP checksum at NL and pseudo header at TL. This way we do double checking.

Options in TCP Header

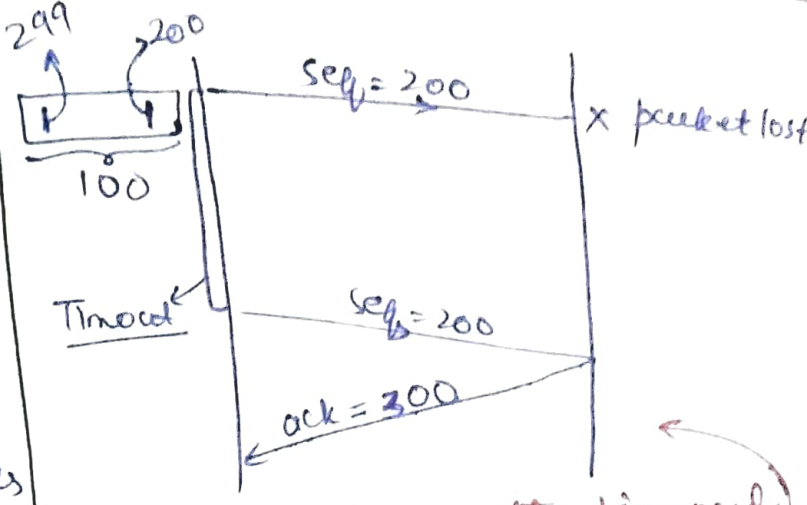
① Time stamp - useful if (WAT<LT)
② window size extension
③ Parameter Negotiation.
④ Padding size
↳ used to make header size meetiple of 40.

Retransmission in TCP

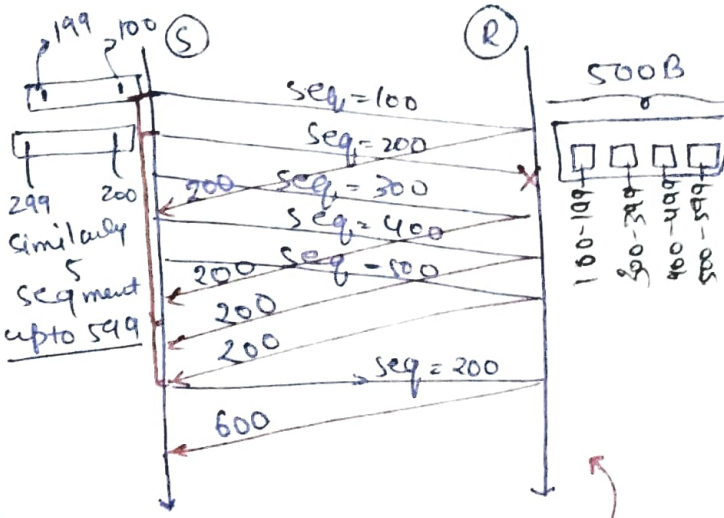
→ TCP uses Combination of Go Back N and SR.
→ $W_S = W_R$ and out of order packet are not accepted.

SR (75%)
→ $W_S = W_R$
→ out of order packet

GBN (25%)
→ Acknowledgement are cummulative



→ Retransmission after timeout



→ Early Retransmission and Retransmission after 3 duplicate Acknowledgement

→ for better performance we use 3 duplicate acks.
→ we assume that receivers sends it W_R as 500 B.
→ If 2nd packet with seq=200 has lost even the packet sent after it, all are accepted because it is (SR) and in SR out of order packet get accepted
→ But ack. send by all packets are same (duplicate) and fold that we have lost packet with 200, so resend it to

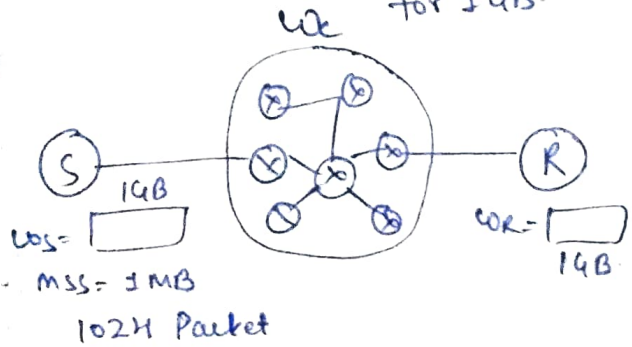
→ So we have no need to wait upto timeout, we can resend it as the 3rd ack. received and when 2nd packet send, it sends the ack = 600 because all the other packets are already in queue.

→ So in 3 dup ack. one packet is lost and some packet get received after it. It is not congestion but it might be possible.

Introduction to TCP Congestion Control

W_C → N/w Capacity.

→ Assume if n/w have no buffer for 1GB.



→ If N/w have no buffer for 1GB then even receiver can accept complete 1GB but you can't send it through N/w.

→ You can send data min(W_C, W_R).

→ To protect receiver to discard data we have adv. window (flow control).

→ To protect N/w we have end to end congestion control only.

→ Assume that adv. window = 8KB and MSS = 1KB

So, W_R = 8KB

W_C = 1 Segment

2

Threshold → 4

5

6

7

8

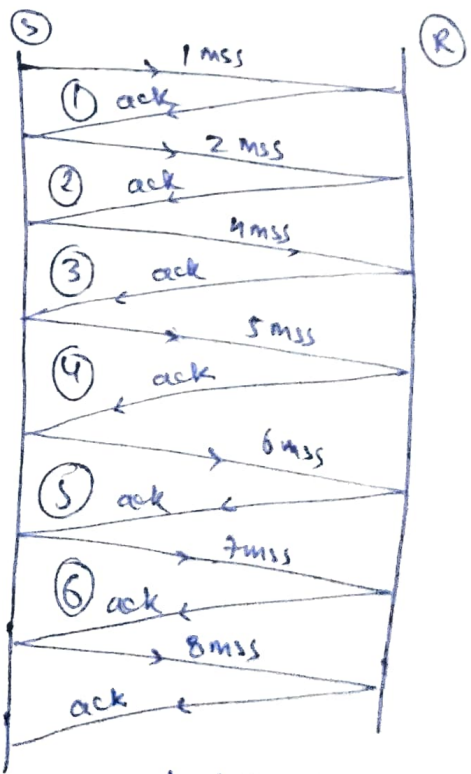
8

8

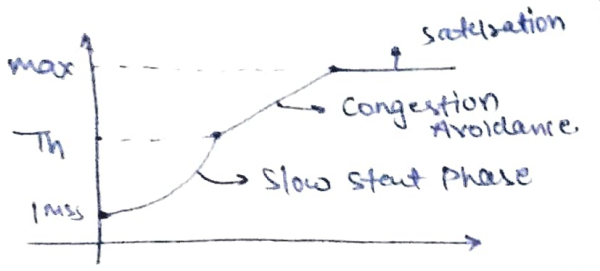
1

Upto threshold W_C increases in power of 2 and after that linearly.

→ Threshold = $\frac{W_R}{2}$ = 8/2 = 4



→ After 6 round trip time sender reaches the max. window size or first full window.



→ Name is slow but process is not slow.

Imp → Sometimes instead of 1 MSS we start with 2 MSS

Imp TCP Congestion Algo. with example.

(1) Slow start Phase
(2) Congestion Avoidance Phase
(3) Congestion Detection Phase
   - → Timeout
   - → 3 duplicate Ack.
   - → It can occur anywhere.

→ Timeout indicate congestion is Severe.
→ 3DA indicate congestion is mild.
→ ICMP Source Quench also can be use as congestion detect but (it is not taking in consideration.)

Ex- WR = 64 KB
MSS = 1 KB
Th = 32 MSS
WR = 64 MSS

1 2 8 16 32
Slow start Phase
33 34 35 ---- 64 64 64 ...
Congestion Avoidance phase

→ If Congestion detection due to Timeout then $(Th)_{new} = \frac{1}{2}(cwc)$ and enters in 'Slow start phase'

→ If Congestion detection is due to 3DA then $(Th)_{new} = \frac{1}{2}(cwc)$ and enters in 'Congestion Avoidance Phase'

cwc = Current window size.

Ex → WR = 64 KB, MSS = 1 KB
WR = 64 MSS, Th = 32 MSS

1 2 4 8 16 32 33

34 → 34/2 = 17 → You can jump upto 17 only
34 1 2 4 8 16 17 18
Timeout — Slow start phase

19 20 → 20/2 = 10 10 11 12 ... 64
3DA — Congestion Avoidance

64 64 → 64/2 = 32 1 2 4 8 16 32 → 32/2 = 16 17
Timeout — SSP — 3DA

18 17 → 17/2 = 8 1 2 4 8 9 10 ...
Timeout

→ It is the responsible of ISP to manage congestion control.
→ OS is designed to follow these rules.

TCP Timer Management (Day 10)

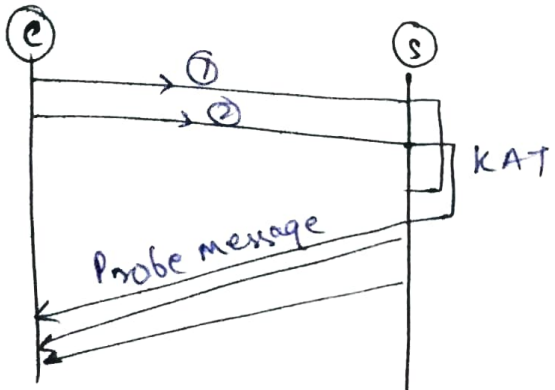
① Time-wait Timer.

→ Don't release the resources (port) assign for process-process Commn. If requst to terminate the connection come then even wait for atleast (2 * Life time) because it might possible that some packet delayed. So don't release the resources at the instants

② Keep-alive Timer (KAT)

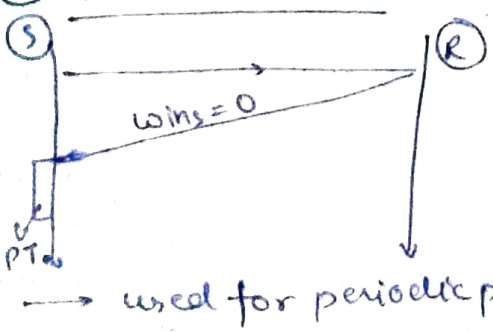
→ Close the idle connection at server side.

Idle connection are those systems which are in connection with server but doesn't sharing any data.



Server create a KAT for each connection. KAT windows updated if msg comes on regular basis. If no. msg. coming from client then server wait upto KAT and then send some probe message. If client doesn't reply then server close the connection and provide resources to another client.

③ Persistent timer

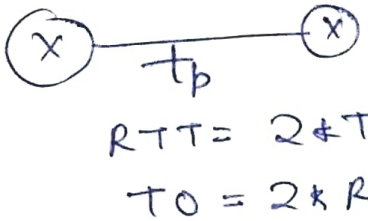


→ used for periodic probing

④ Acknowledgement Timer

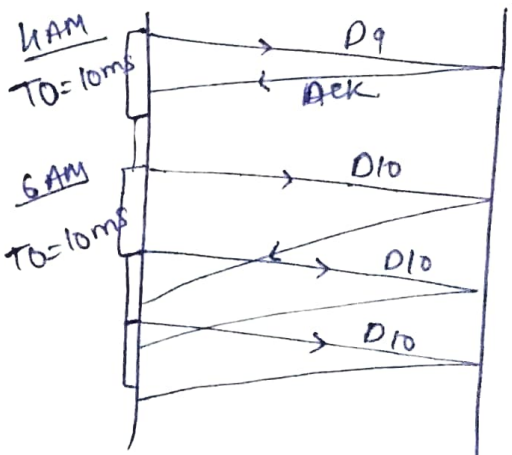
⑤ Time-out Timer

↳ Time-out Timer management is difficult at TL.



But at TL there can be any no. of HOPS in between.

→ Static Timeout Timer doesn't use at TCP.



Suppose at 4AM due to very less N/w traffic you get ack very soon. So TO = 10ms was enough but at 6AM traffic Increases and ack. time get Increases due to this you are unnecessarily sending the same packet again after the TO.

so its betty to change TO according to N/w traffic, to avoid packet retransmission.

Algo. used to manage TO Timer –

① Basic Algo.

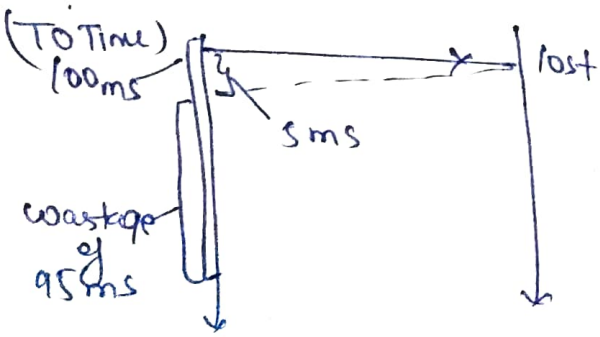
② Jacobson's Algo.

Basic Algo. for Timeout timer computation

→ This Algo. works to implement dynamic Timeout timer.

→ Previously we seen that if TO Timer is very less, it leads to the problem of unnecessarily retransmission bcoz acknowledgement doesn't reach within time.

→ Instead of this if we want to make timeout timer then we can choose a very large timeout timer but in this case if packet get lost then we have to wait for long time to retransmit the packet.



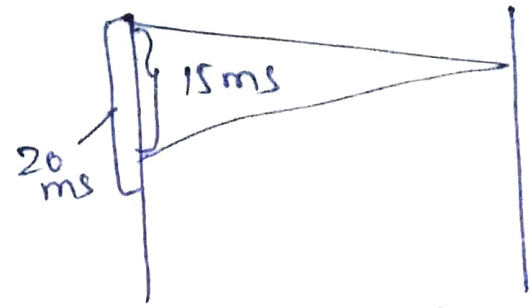
So, that is why we prefer dynamic TOT.

To compute it we just guess the Round trip time for first packet say

IRTT = 10ms

then (TimeOut) = 2 * IRTT
= 20ms

But assume first packet take only 15 ms



So, Actual RTT, (ARTT) = 15ms

then, (Next RTT) = (α) IRTT + (1-α) ARTT

here, α → Smoothing factor, used to calculate ratio.

If α=0 means,

(NRTT) = (ARTT)

or α=1

NRTT = IRTT

in both the cases NRTT becomes static but we want a dynamic Timeout Timer.

So, $\boxed{\alpha \in [0,1]}$ → Given in exam.

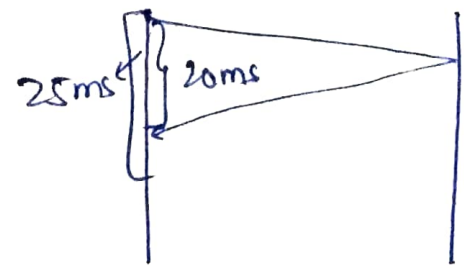
we assume α = 0.5 (here)

(NRTT) = 12.5 ms.

So for next packet

IRTT = 12.5ms

TO Time = 2 * 12.5 = 25ms



So, ARTT = 20ms

then (NRTT) = $\frac{1}{2}$ x 12.5 + $\frac{1}{2}$ 20

= 10 + 6.25 = 16.25ms

So (IRTT) for 3rd packet
= 16.25 ms
TO = 32.5 ms
(Say) ARTT = 10 ms
NRTT = 5 + 8.125
= 13.125.

→ We can continue this packet for more packets too, and this way we provided dynamic timeout timers.

→ If packet comes earlier as assume then timout time decreases and it means traffic decreases or vice-versa.

Disadvantage.

Always TO = (2) * IRTT
↓
Why 2 only?

→ 2 was used conventionally but Jacobson's rectify it more precisely

Jacobson's Algorithm & Karn's Modification

IRTT = 10ms
ID = 5 ms (Initial Deviation)
Deviation 5 ms means RTT may be (5, 15) ms
And both the IRTT and ID are being assumed for first packet transfered.

Then,
Time Out = 4 * D + RTT
= 4 * 5 + 10
= 30 ms
Assume ARTT = 20 ms
and AD = 10 ms ←
↳ |IRTT − ARTT|
| 10 − 20 |

for 2nd packet
IRTT = 15 ms
ID = 7.5 ms

NRTT = α(IRTT) + (1−α) ARTT
= 15 ms (α = 0.5)
ND = α(ID) + (1−α) AD
= 7.5 ms

TO = 4 * D + RTT
= 4 × 7.5 + 15
= 45 ms
ARTT = 30 ms
AD = 15 ms
NRTT = 15 (0.5) + (0.5) 30
= 22.5
ND = (0.5) (7.5) + (0.5) 15
= 11.25

for 3rd packet
IRTT = 22.5
ID = 11.25
TO = 67.5 ms
ARTT = 10 ms
AD = 12.5

NRTT = 16.25
ND = 11.875

And similarly move on upto required packets

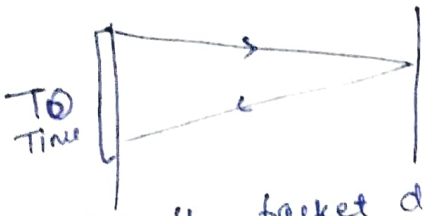
→ So Jacobson's gives a different formula for TO that proves better than basic algorithm.

Q IRTT = 10ms, TD = 5ms
$\alpha = 0.5$. Assume Acknowledgement Successively ~~time out~~ for 3 packets are 20ms, 30ms, 10ms then find TO for 4th packet?

→ Ask like this Gate.

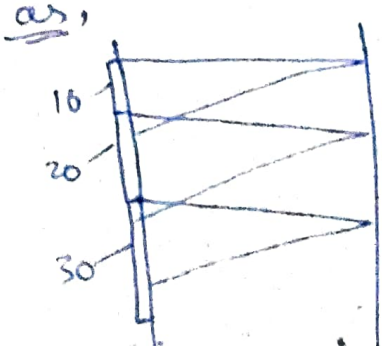
Disadvantage of both basic & Jacobson's :-

we assume that first packet is successfully sending the acknowledgement within TO



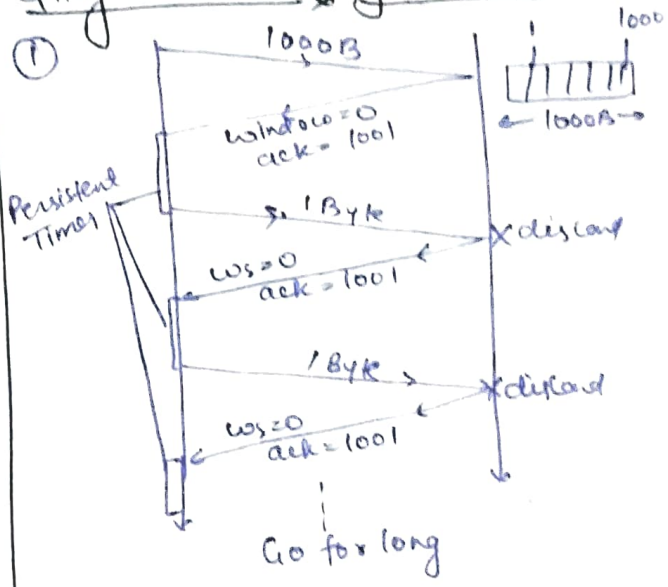
but if the packet doesn't come within TO time then we have to retransmit the packet.

So, Karn's says that just assume the TO for first packet and if ack. doesn't comes in time then just do double the 'TO time' as,

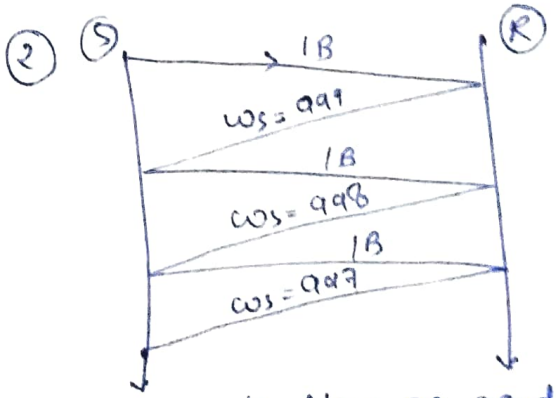


And wait upto which TO time packet received within time. No need to apply any Algo. In this case.

Silly Window Syndrome

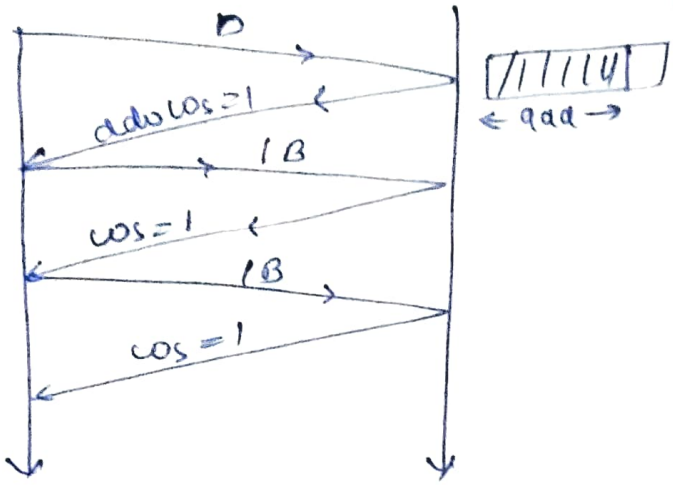


So, receiver window is full and it is not receiving packet for long. This is not a persistent problem because the mechanism of sending packet to TL is slow down but not have any problem in sender and receiver. So this is a Sanctent problem.



This is a problem on sender side so it will persist for long because sender is wasting time by sending small packets.

(3)



Receiver is not receiving more than 1 byte. So this is also a persistent problem.

Nagles Algo :- Solution for (2) problem

If sender is slow then TL doesn't send 1B of data, TL needs to wait for 1 RTT and store all the data in buffer then send it in one segment.

→ If packet prepaid before 1 RTT then you can send it before.

Ex Rate = 1B/ms from AL → TL

RTT = 100ms

MSS = 200B

then, 1 RTT = 100B.

Send 100B in one packet.

Clarck → He gives soln for 3rd problem

Receiver should not advertise 1. It have to wait upto

$$WS > \frac{1}{2} \text{ buffer}$$

→ for (1) st problem solution is use RST (Reset flag) and flush the connection to restart it.

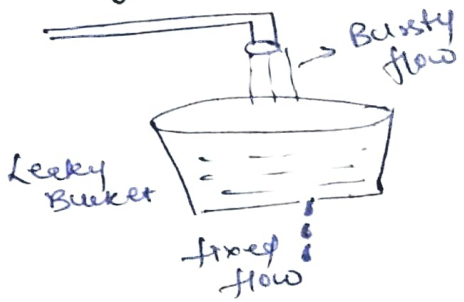
Traffic Shaping

→ Another method of congestion control is to "shape" the traffic before it enters the network

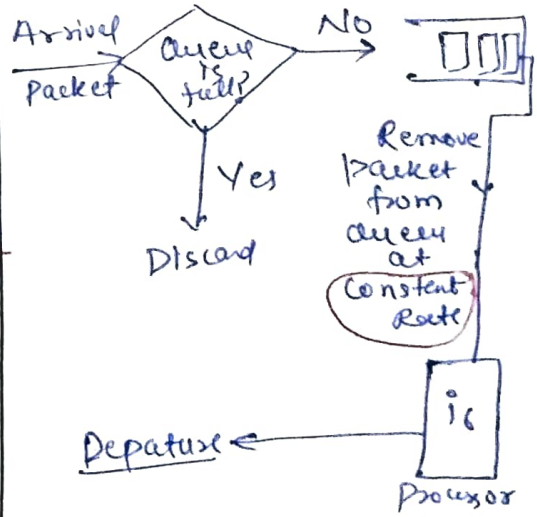
→ Traffic shaping controls the "rate" at which packets are sent.

→ During connection establishment, the sender and the carrier negotiate a traffic pattern (shaping).

Leaky Bucket



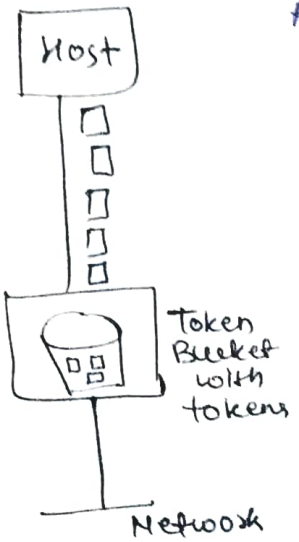
what ever the rate of input, output rate is fixed.



implementation of leaky bucket

Assume constant rate is 100 B/s and packet is of size 300 B then you have to wait for 3 sec and after 3 sec you can send complete 3B data

Token Bucket



Assume that rate is 10 packet/s and you didn't send any packet for 10 sec then in 11th sec you are allowed to send 100 packet in token bucket.

Terminologies

- Let the capacity of token bucket = 'C' tokens
- Rate of tokens entry in bucket = r
- The max. number of packets that can enter the network during a time interval 't' is -

Max. no. of packets = $C + rt$

Max. avg rate = $\frac{(C+rt)}{t}$ /sec

Ex→ Consider a token bucket of capacity 250 KB and the tokens arrive at a rate of 2MB/sec. If the max. out rate is 25MB/sec. what is the burst time?

Soln token rate (r) = 2 MB/sec

Capacity = 250 KB

So, $\left(\frac{C + rT}{T}\right) = M$

$\Rightarrow 25\,MB = \left(\frac{C + rT}{T}\right)$

$\Rightarrow T = \frac{C}{M - r}$

$\Rightarrow T = \frac{250\,KB}{25\,MB - 2\,MB}$

$\Rightarrow T = \frac{250\,KB}{23\,MB} = 10.86\,ms$

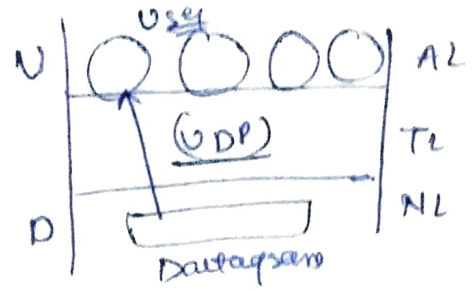
C = 250 KB means 250k packets bcoz 1 packet of 1B.

UDP

Need of UDP → TCP is overhead for some application, so we need UDP

1. 1 req / 1 reply applications
   - i) DNS
   - ii) BOOTP & DHCP
   - iii) Network Time Protocol
   - iv) Network News
   - v) Quote of the day protocol
2. Broadcasting or multicasting application.
3. Some app^n requires speed rather than reliability and constant data rate
   - i) Multimedia application
   - ii) Games.

HD videos buffer in TCP and all other in UDP.

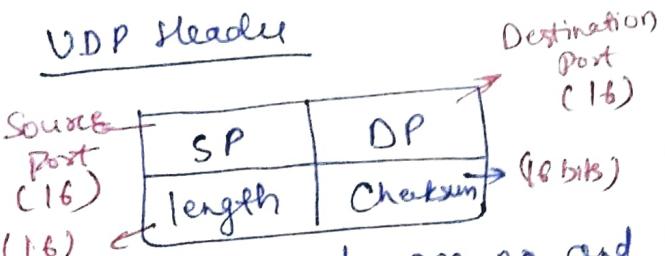


TL takes the datagram and send it to user. So it is called as User Datagram Protocol.

UDP doesn't require to provide all the services as TCP.

UDP uses when TCP's all facilities are not required.

UDP Header



→ We don't need seq no and ack no. bcoz we are sending packets not byte streams

→ We have no need of window or flag bcoz there is no flow control, conn establishment and connection termination

→ Length (16 bits) → UDP Header + UDP Data

→ UDP Header = 8 Bytes

Checksum = UDP Header + UDP Data + Pseudo header IP

Sometimes inside same LAN checksum doesn't use then put 16 0's in field. Or you can put all 1's.

→ You can set traceroute as option in TCP using UDP, and similarly record route, Timestamp can be set.

→ UDP manages ICMP error as destination unreachable, source quinch

Basically UDP is Intermediate between IP and Appn layers.

→ UDP can be assume as NULL protocol bcoz it does nothing just take packet from NL and forward to AL.

→ RIP, OSPF, FTP also uses UDP ↓ TFTP (Trivial FTP)

Hardware and various devices used in Network (Not for Gate)

① Cables, Repeaters and Hubs

Computer Network is Network of computers.

Types of Ethernet Cables

10 Base I → 100m range
↓ 10MBps
↳ No multiplexing means one 1 host can send signal only.

10 Base 2 (10 Mbps, NoMUX, 200m)

10 Base 5 (10 Mbps, NO MUX, 500m)

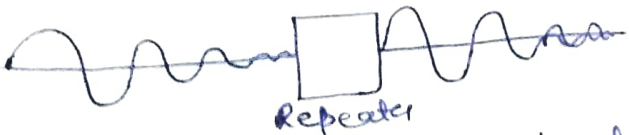
100 Base T and many more.

→ All cables operates on physical layer and having problem of attenuation.

→ Collision is a problem in these cables.

→ Length of LAN decided by length of cables.

→ To increase length of LAN we have to join two cables using repeaters.



→ Repeater generates same signal again. Don't consider it as amplifier.

LAN < 10 KM MAN < 100 KM

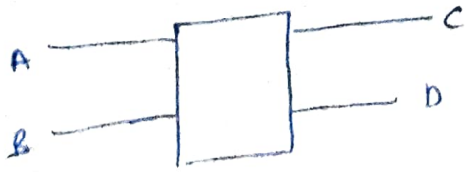
WAN > 100 KM

→ Repeater is used to connect to LAN of same type and works on PL.

→ Collisions are possible inside repeater also.

→ Range of LAN increases.

HUB :- It is a multiport repeater.



① Traffic is very high.

② Works on PL

③ Collisions are possible.

④ HUB is cheaper device.

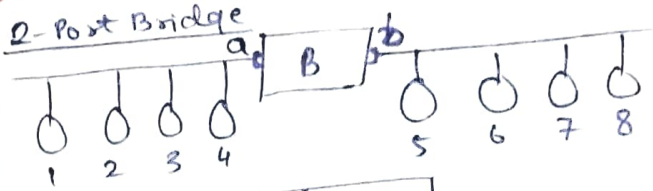
⑤ In HUB message from single sender get broadcast to each node.

Bridges :-

It is use to connect two LAN's Both LAN can be different.

→ Works at both PL & DLL.

2-Port Bridge



Mac	Port
1	a
2	a
3	a
4	a
5	b
6	b
7	b
8	b

① Static Bridge → Changes done manually in tables.

② Dynamic / Learning / Transparent Bridges :-

→ Message get filtered to particular node doesn't broadcast. (filtering)

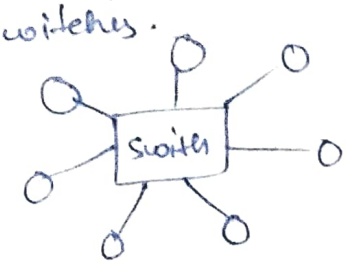
→ Bridges can do forwarding too.

→ If entry is not in table then they do flooding.

→ It is able to store & forward a packet.

→ No collision inside a bridge.

Switch : Unlike bridges we directly connect host with switches.



→ It has both NL and DLL.
→ It has m/m and processor.
→ More than 1 communication can establish.
→ Full duplex cables.
→ No. of ports can be 8, 16, 32, 64, 128....
→ No collisions in a switch.
→ Collision domain reduced to 0.
→ Very less traffic.
→ It is costly.

Collision Domain and Broadcast Domain of all devices.

(Wire + Hub + Repeater + Bridges + Switch) → LAN Components

	BD	CD
Repeater	Same	Same
HUB	Same	Same
Bridge	Same	Reduces
Switch	Same	Reduces to 0

BD → Broadcast Domain
CD → Collision Domain

Internet Protocols :- (AL) (In Gate)

Introduction to application Layer.

① DNS ② HTTP, ③ FTP, ④ SMTP ⑤ POP ⑥ ICMP

→ ICMP works at NL not at AL. We will not consider it here.

HTTP → web pages
FTP → file transfer
SMTP → email
POP → emails

DNS :- (Domain Name Service)

→ Domain names are easy to remember and IP address are not static.

DNS : Converts DN → IP
(Domain Name)

Types of Domain

① Generic Domain
.com, .edu, .org, .net, .mil and many more.

② Country Domain
.in, .us, .uk, ...

③ Inverse Domain
(IP → Domain mapping)

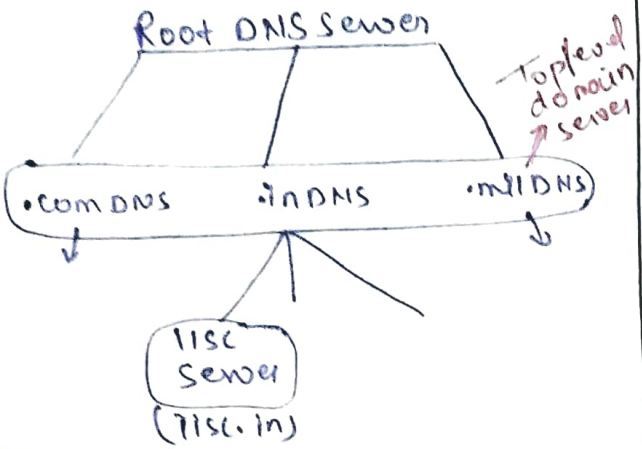
NSlookup www.google.com
↳ Command use to get IP address.

Good companies make several copies of same data and upload them on different servers. So if 1 server gets down. You can access data from different servers.

→ for load balancing DNS will give you different add. each time.

→ DNS have record for each server service arrange in form of tree.

→ Arrange of DNS server is like the distributed database-



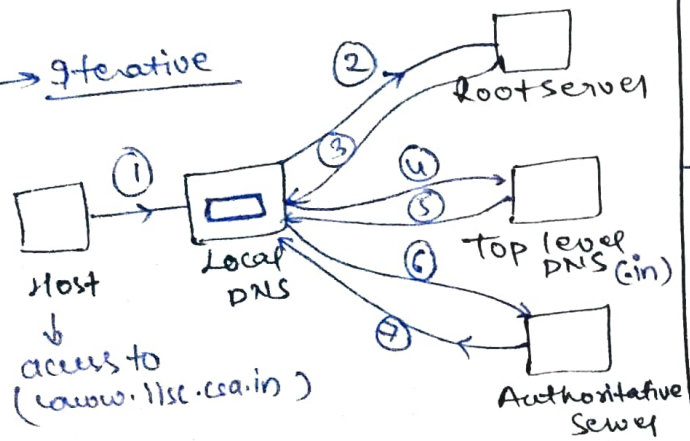
→ We have lot of root sewers so, that if any of root sewers gets down, then we can get access from another one.

→ India's nearest root sever in Tokyo (Japan).

→ Each ISP provide local DNS Sewer to use.

→ If entry is not in DNS then DNS will get the data from above sewer called as DNS overhead.

→ Iterative

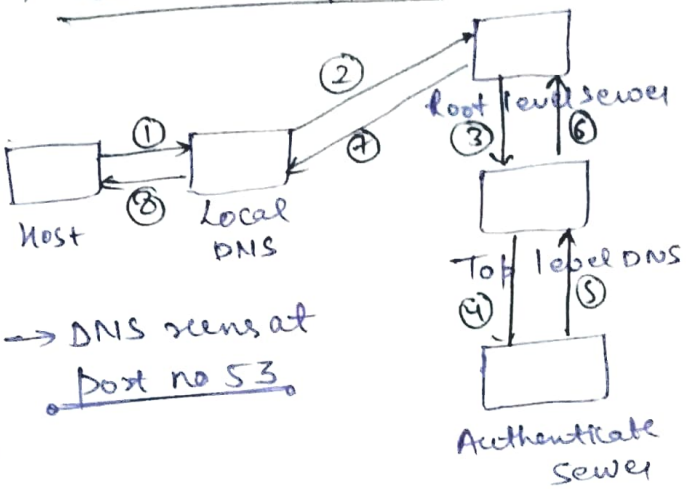


→ If entry is not is table and Local DNS ask to Root sever it replies that entry is not in root sever contact to top level DNS.

Similarly if Top level DNS doesn't have any info it will responde and say to contact with authoritative sewer.

And finally it will get the IP Address of required domain name.

→ Recursive Approach



→ DNS runs at port no 53

→ Increase local DNS cache at max as possible to reduce traffic.

→ DNS uses UDP at TL.

HTTP : Mainly used for getting web pages.

→ Run at port no. 80.

→ Always relies on reliability from the layers below AL.

→ It uses TCP at transport layer.

→ It is 'In band' protocol means both command and data sends in one connection.

→ HTTP is stateless (not going to maintain info.)