IPV6 addressing:-

Initially 32-bit Ip address 2^{32} addresses are possible.

→ IPV6 address = 128 bit

→ 2001 : 0db8 : 0000 : 0000 : 8a2e : 0370 : 7334.

16-bit 16-bit

Each digit is in hexadecimal form.

Address length = 16×8 = 128-bit.

→ 2001 : db8 : : 8a2e : 370 : 7334 {compressed format}

You can also remove zero prefix

If you have this double colon representation this means in between everything is zero

Note → You can use double colon only one time in the address.

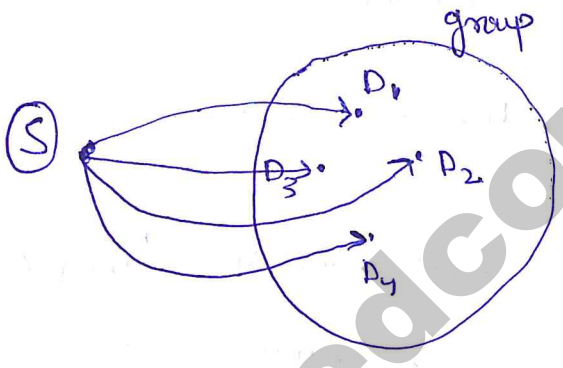
→ IPV4 and IPV6 are both used today.

UNICASTING:

Unicasting means single node to single node.

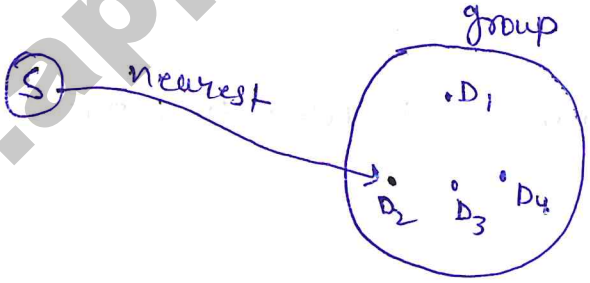
MULTCASTING:

Multicasting means if you want to send from single node to group of nodes.



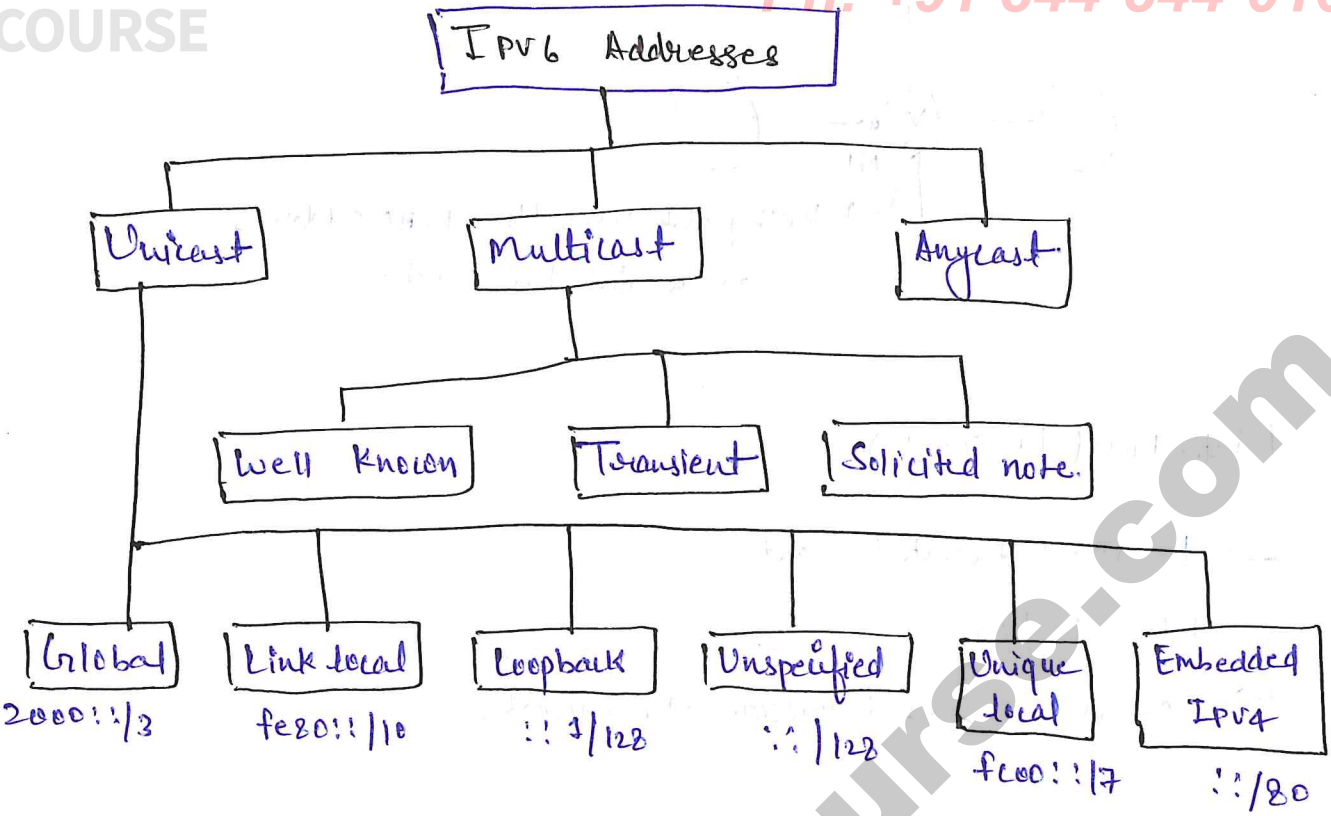
{ Broadcasting is a special case of Multicasting }

Anycasting:



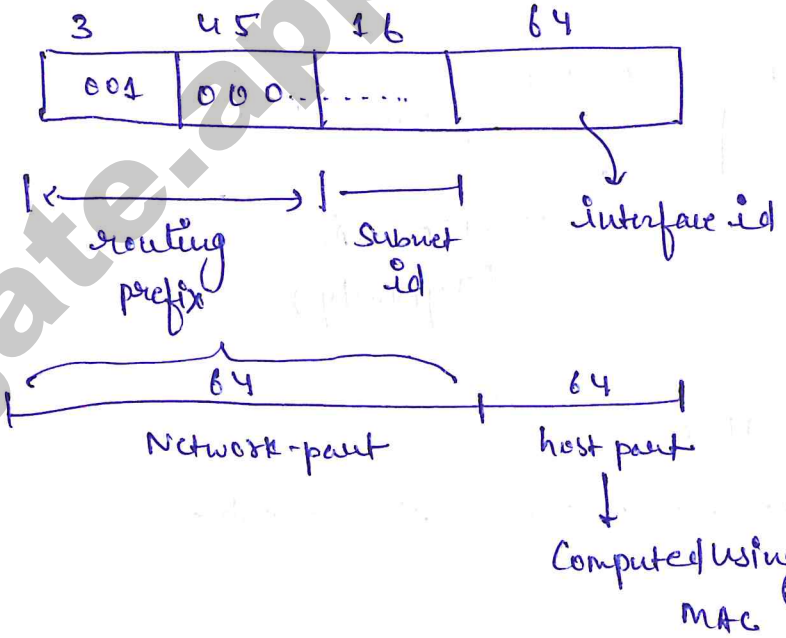
In Anycasting the objective is to send the packet from source to one of the node of the group.

* It is typically sent to the nearest host.

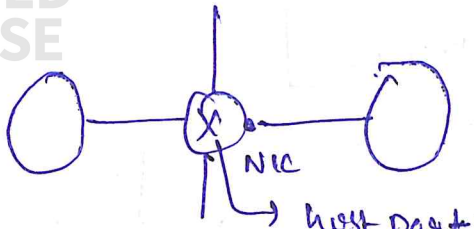


Global-unicast → Global unicast is just like public Ip addresses in IPV6.

Node can be host / router.



* In Global unicast address first three bits are 001 fixed.
* It is globally routable any router gets this will successfully routed to destinations.



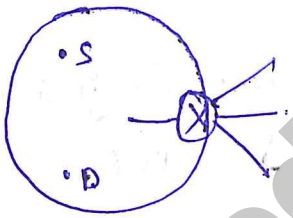
→ host part uses the MAC address as an interface id.

Link-Local (Unicast)

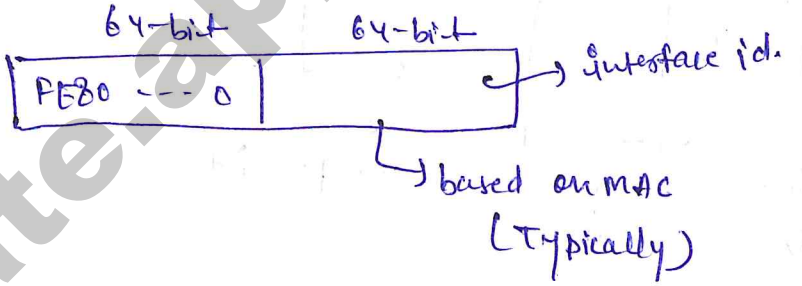
→ Local-network unicast

→ FE80::/64

→ Can't be routed outside the network.



If source and destination are in same network we use link local address.



* First bits are = FE80
* This packet will not route outside the network.

Loopback / Local host

→ ::1/128

* Loopback Ip address for testing / debugging.

Unspecified → ::/128 {all zeros}

→ not assigned to any interface

This is an unspecified Ip address

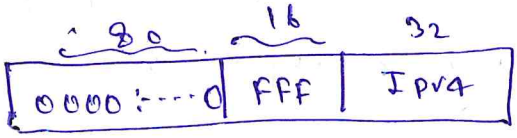
Unique Local → FC00::/7

→ Private Ip address just like IPV4

Embedded IPV4: → Transition period

IPV4
IPV6

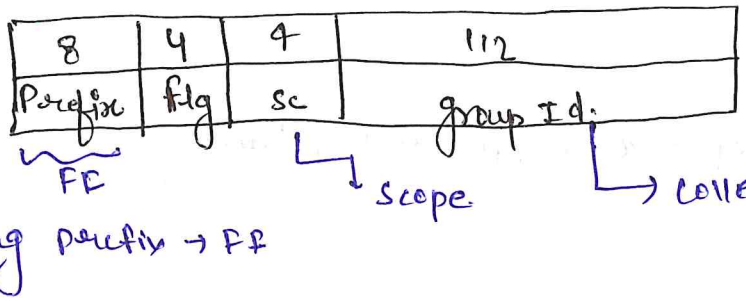
for converting IPV4 into the IPV6 address



→ 80 bits are zeroes, 16 bits are ffff, 32-bit IPV4 address

to convert IPV4 to IPV6.

Multicast:



Starting prefix → FF

* Multicasting is used for sending single node to group of nodes.

* Group Id is the collection of hosts, routers map group Id to hosts.

flag: 0000 → Permanent grouping

* Permant grouping can be used for fixed hosts like group of routers, gateways which are used by country.

flag: 0001 : Temporary / Transient

* We can create a temporary grouping for a meeting or company for some time.

Scope:

0000: reserved
0001: Node local
0010: Link local
0101: Site local
1110: Global
1111: reserved.

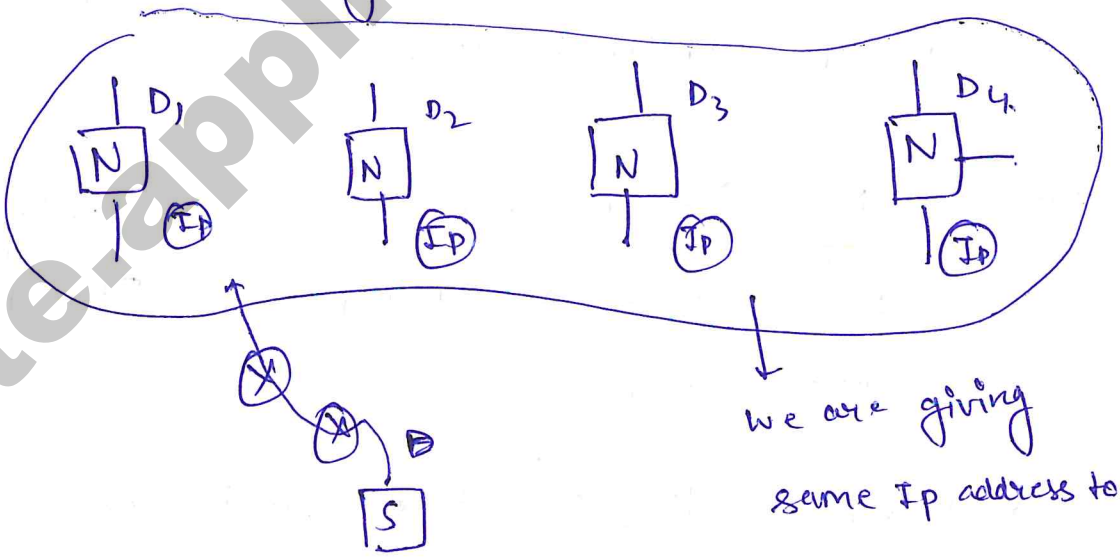
→ All the interface are local to the network

→ There are on the same geographical location

Anycasting:

→ No specific IPV6 range.

→ All Unicast address can be used for multicasting



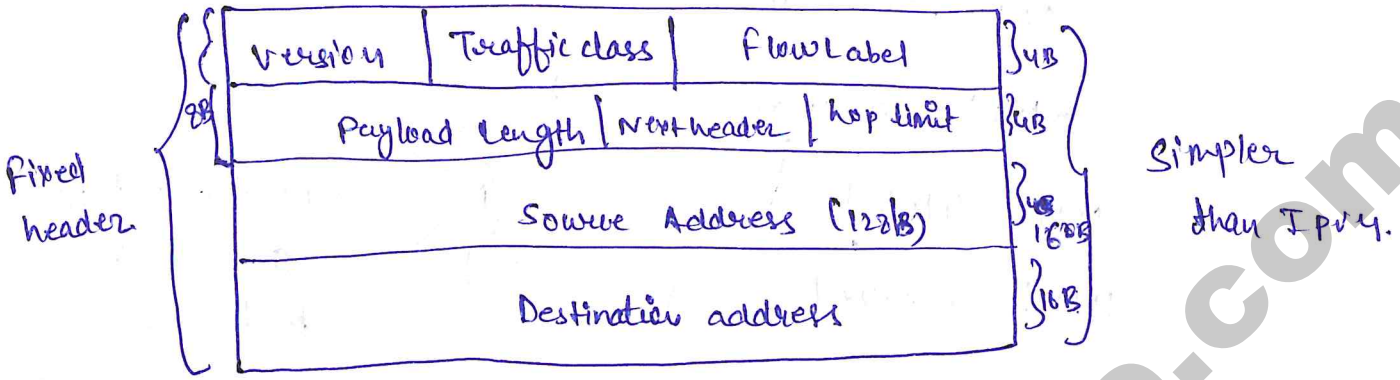
we are giving same Ip address to all

Anycasting

In IPV6 multiple interface has same Ip address.

* Source can send any of the interface using that Ip.

IPv-6 header :-



Extension - header - 1

Extension header - 2.

⋮

Data

* In IPV6 header there is fixed header followed by multiple extention header.

Version :- By seeing the version bits we can easily recognize whether it is an IPv4 packet or IPV_6 packet.

0110 → IPV_6.

* 40 Bytes is the size of the fixed header.

Traffic class:

- Same as (DSCP + ECN) fields (TOS) in IPV4.
- Priority, congestion-notification

Flow-label: Its a 20-bit field.

The default value is 0.

* From source to destination there is sequence of packets that we are sending from source to destination.
- TCP packets of same flow/stream.

* Groups of packets are part of the same flow.

* routers will give the special service to these packets.

Payload length:— Extention-headers + Data (from layers above)
(16 bit)

If length of payload ≥ 65535

* Then, Set the payload length to zero and use hop by hop extention header using Jumbo payload option.

Next header → TCP | UDP | ICMPv6 | extention-header type.

It stores the protocol id of the data header.

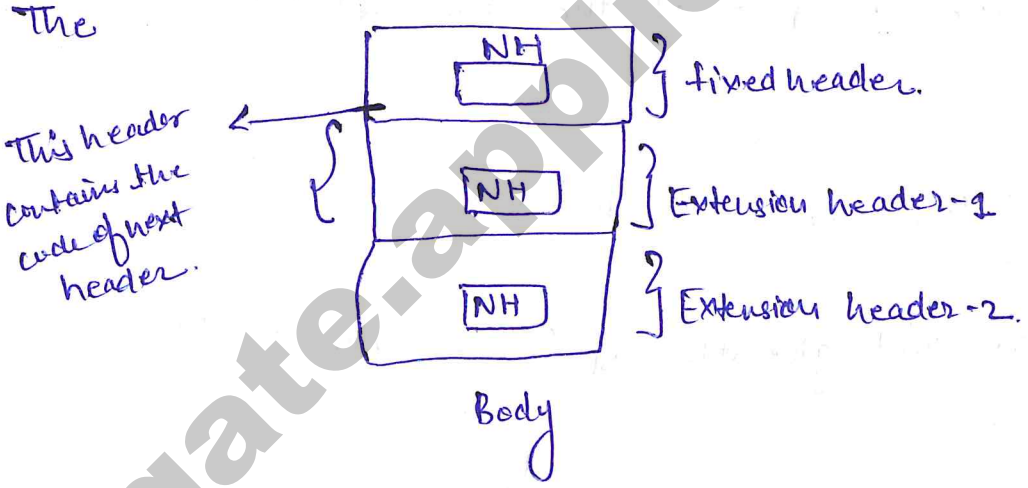
* If data contains TCP packet then next header contains id of the TCP.

Hop limit → Exactly like TTL in IPv4

Extention header: Initially we have a fixed header and afterwards will have extension header :1.

The extension header contains the extra functioning that you want in packet.

The



The order of extension header is

Recommended order

order	Header Type	Next Header code
1.	Basic IPv6 Header	-
2.	Hop by hop options	0
3.	Destination options	60

} Like that we follow heads in extension header

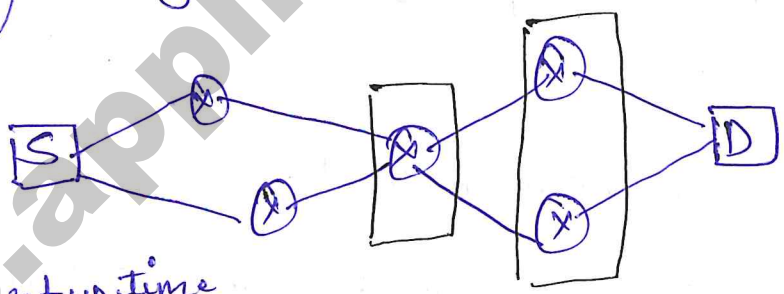
Switching:

- Circuit Switching (Telephone)
- Packet switching
  - Datagram (IP)
  - Virtual circuits (ATM) Async. Transfer mode
- Message Switching (Military applications)

Circuit Switching:-

→ Telephone - exchange (1960s)

VOLTE



→ connection set up time

→ No headers, no store and forward

→ data arrives in order

→ Message size: m.

↳ Here we call exchange to establish the connection upto the destination.

→ $\left[\text{Setup time} + \underbrace{\frac{m}{B}}_{T_t} + \underbrace{\frac{\text{dist}}{\text{vel}}}_{T_p} + \text{Teardown time}\right]$

→ Once a dedication connection is set up, we can share the informations from source to destination.

→ data arrives in order in the circuit switch model.

→ In datagram model, packet need not arrive in order.

[

Packet Switching :—

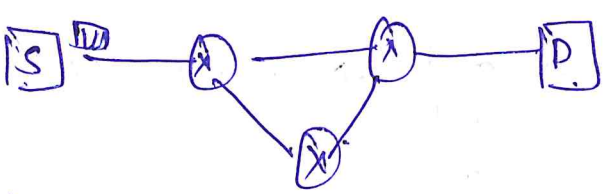
→ Datagrams (IPV4) {routers}

→ Virtual circuits (ATM)
  * Only 1st packet has header
  * Same path by all packets (Resource reservation)
→ INORDER.
→ Reliable and connection-oriented.

* In packet switching we breack the data and form chunk of packet for sending.

* Datagram is the most popular in packet switching.

* Virtual circuits (Asynchronous transfer mode)

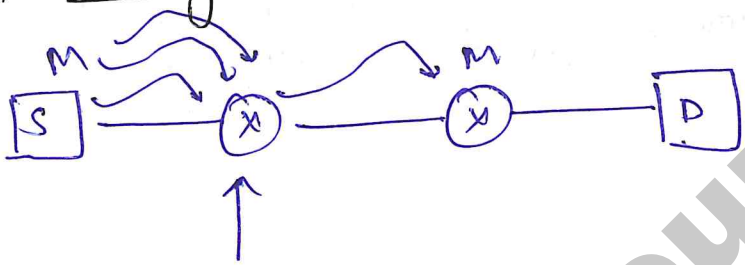


↳ It is using functioning of circuit somewhat

* The first packet in the intermediate routers tells to all the intermediate routers to reserve some space for coming packets.

* Virtual circuits are reliable and connection oriented.

Message Switching :-



{ Complete Message
Store and forward.

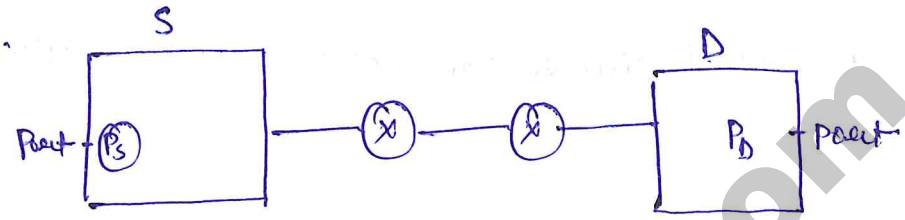
Message completely stored to the next hop and then it is forwarded.

→ It is used in Military applications and satellite communication.

Transport - Layer (TCP, UDP)

→ Process to process communication.

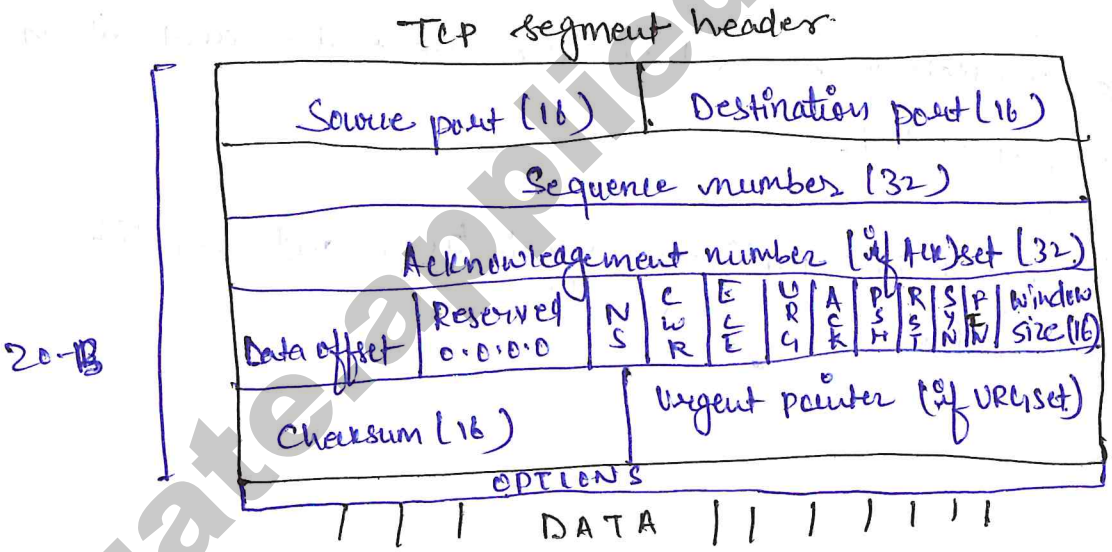
→ Reliable end-end channel.

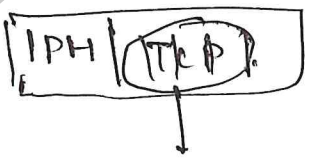


- TCP and UDP
  - TCP → Connection-oriented | reliable
  - UDP → Connectionless

Transport layer gives you process to process delivery.

TCP - segment format





TCP = header + data.

* Segment is the term used for transport layer. for packet.

Source and destination port (16 bit)

→ 0 to $2^{16}-1$ = 65535 [Total ports possible]

→ Pre-assigned ports

↓

[0 - 1023) → Predefined ports, Universal ports, fixed ports

IANA

[HTTP : 80
FTP : 21
SMTP : 20
DNS : 25
SSH : 22]

→ 1024 - 49151 → reserved for future.

→ 49152 - 65535 → Private-ports, Dynamic ports.

↳ These ports used by Network programmers

Network Socket → Java / python [video-chat]

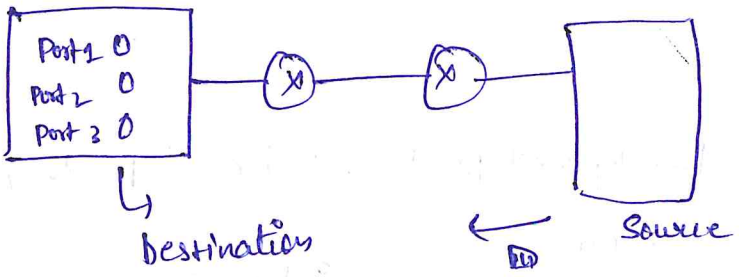
* This is [IP: port] → Socket is the combinations of IP and PORT.

socket

* IP address uniquely identify the computer, port address uniquely identify the process.

Socket → 12.13.14.16 : 4956 2

IP address — Port number



We need a port to reach the exact process because IP address will lead you to the destination after that we need a port to reach the exact process.

Sequence No. and Ack No.

IP: Packet stream [Identification number]

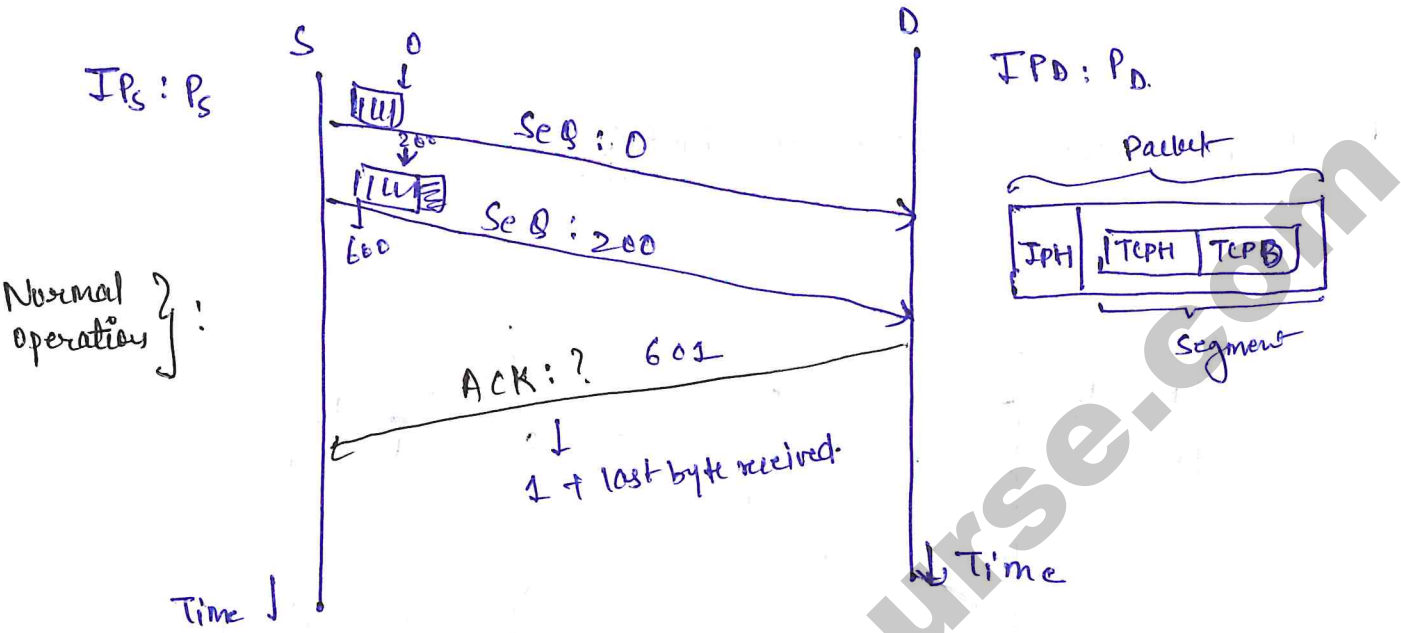
* IP cares about the sequence of packets.

TCP: Byte stream

→ TCP cares about sequence of bytes

Sequence no: = First byte in a segment

* In the transport layer the general term we are using is a segment.



SEQ 0 : first byte of header SEQ 200 : first byte of data.

* TCP is the reliable service and we also have acknowledgement in TCP.

* If last byte of data is 600 then ACK will be 601.

* Computing ACK: TCP packet enclosed under IP packet.

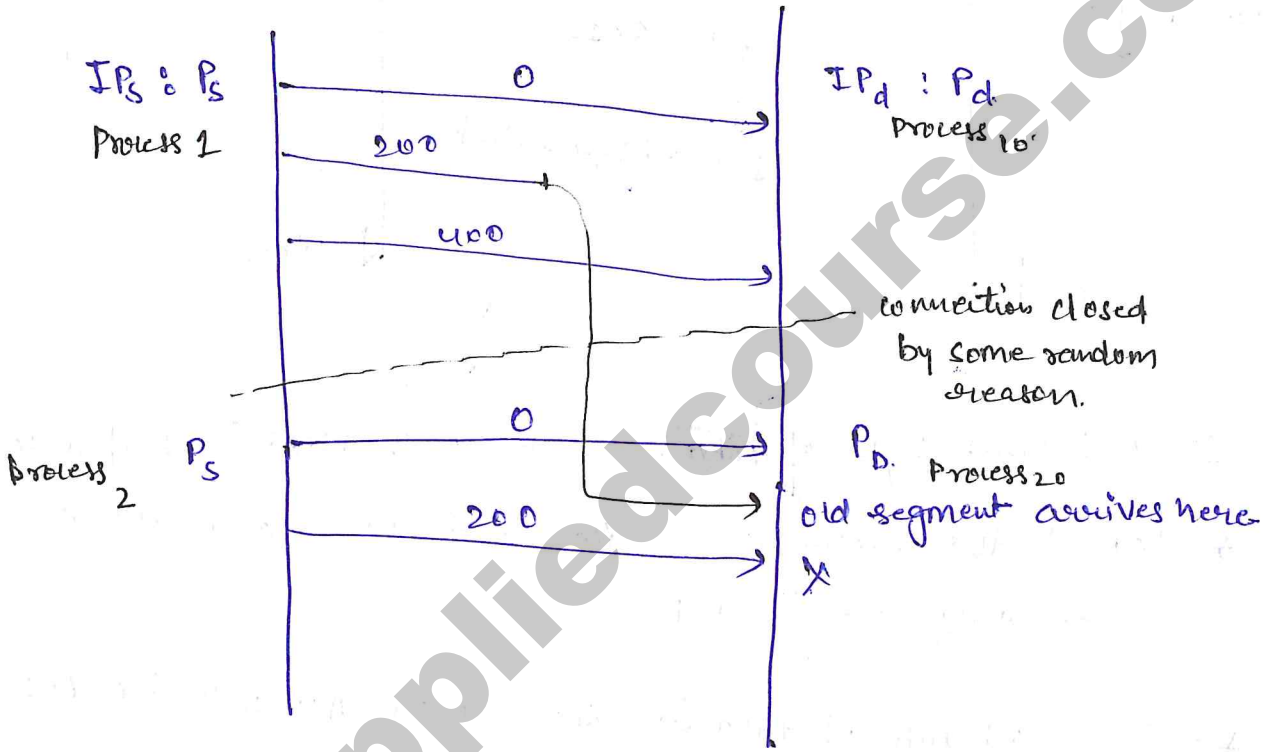
TCP data length = IP packet length − IP header length − TCP header length

- IP packet length: We will get by Total length bits in IP header
- IP header length
- TCP header length: Using data offset field in TCP header

ACK = TCP data length + 1.

* The way we computed the tcp data length by use of IP header.

Random sequence number as start:

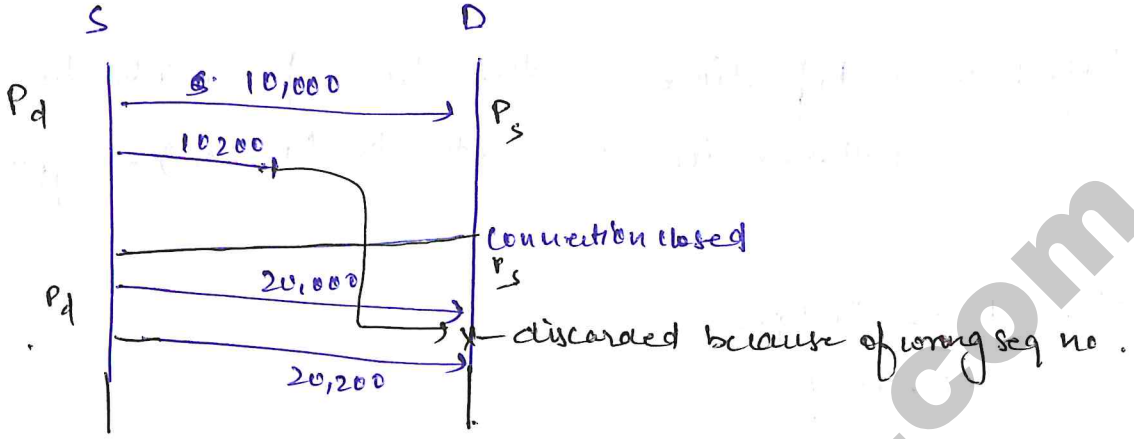


Here our packet with sequence number 200 get delayed and afterwards connection closed because of some reason.

The new connection is established by same port number coincidentally on both the sides.

The delayed packet in the older connection came and accepted as correct packet because the port number of previous connection and new connection are same.

To avoid this problem we use the concept of random number.



* for starting sequence number instead of giving the number zero. we can give the sequence no. 10,000 randomly.

* The probability of choosing the same sequence number by both the connection is very low.

* $0 - 2^{32} - 1$ (Seq number)

WRAP AROUND SEQ NUMBER:

If we are starting without sequence no. 10,000

Seq no = 2^{32} = 4GB data we can send.

So, if our data is more than 4GB. How will we send because we don't have much seq number?

Solution: WAT = wrap around time.

10,000, 10001, 10002 ---- $2^{32}-1$, 0, 1, ---- 10,000

WAT

Again we start with zero.

* WAT is starting from that sequence number again come to that sequence number.

Life time → Life time is the time for which the packet will remain in network (180sec) → Typically.

Que: Bw = 1MBps
WAT = ?

Ans:
1MB → 1sec
2^{20} B → 1 sec
1B → $1/2^{20}$ sec

Total seq no. ← 2^{32} B → $\frac{2^{32}}{2^{20}}$ = 2^{12} sec = 4K sec = WAT
from starting to come again

WAT > LT
4000 > 180 ✓

* If WAT > LT then there would be no confusion of sequence number.

Eg: BW = 1 GBps

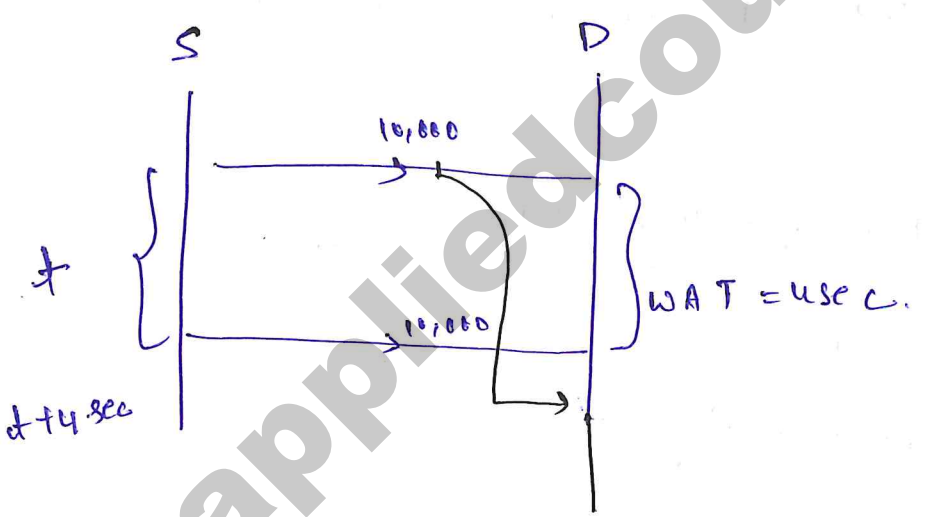
WAT = ?

1 GB = 1 sec

$1\ B = \frac{1}{2^{30}}$ sec

$2^{32}\ B = \frac{2^{32}}{2^{30}} = 4$ sec = WAT

WAT	<	LT
4		180



Problem:

If WAT < LT then after 4 sec that packet again come in the network with same sequence number.

Solution: Increase the number of bits in sequence number.

LT = 180 sec

then WAT > 180 sec

1 sec = 1 GB

180 GB = 180 GB

Hence of 180 × G sequence number

$= 180 \times 2^{30}$

8-bit to represent this

30-bit for this

* Total we need 38 bit

Options	Seq no.
6-bit	32 bit

* So, least significant 32-bit will store in sequence number. and 6-bit will represent in options.

→ for WAT < LT we can use this idea.

Header length / Data-offset (4b) : TCP header

→ Just like IPV4

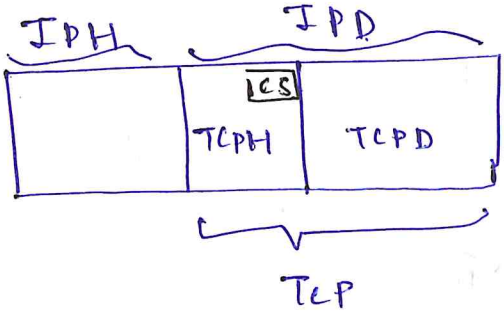
- HL × 4 = header bytes

→ min header = 20B
max header = 60B (max HLf = 1111)

The minimum value of data offset = 5 because min size of header = 5 × 4 = 20B

So, if data offset > 5 then we have options

Checksum (16 bit) :-

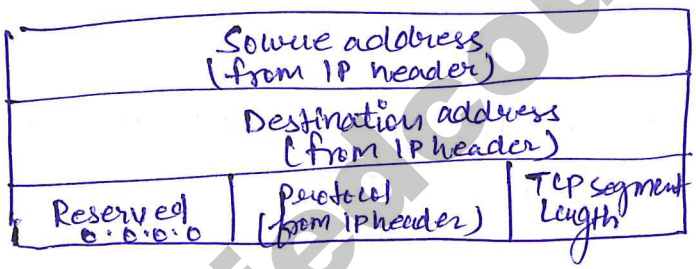


TCP checksum = 16-bit checksum computed using CRC

TCP checksum = It is computed on TCPH + TCPD + pseudo header*

↳ Three parts.

Pseudo header →



"TCP pseudo header" for checksum calculation

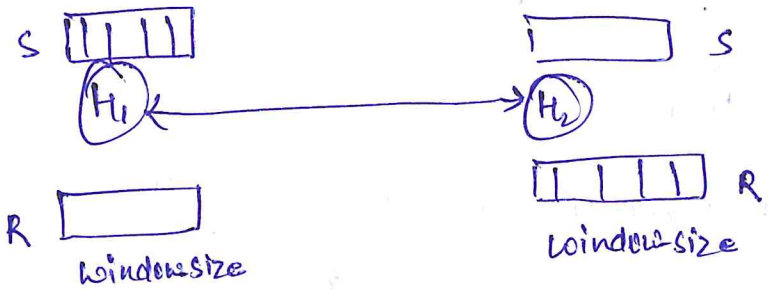
* Here take some important fields in the pseudo header from the Ip header.

→ In IPv4 the checksum is calculated for header not for whole packet.

→ TCP checksum also uses pseudo header for double checking for misdelievery of any packet should not happen.

TCP connection :-

→ connection oriented protocol



TCP enables the full duplex and buffers.

Buffers → some preallocated memory

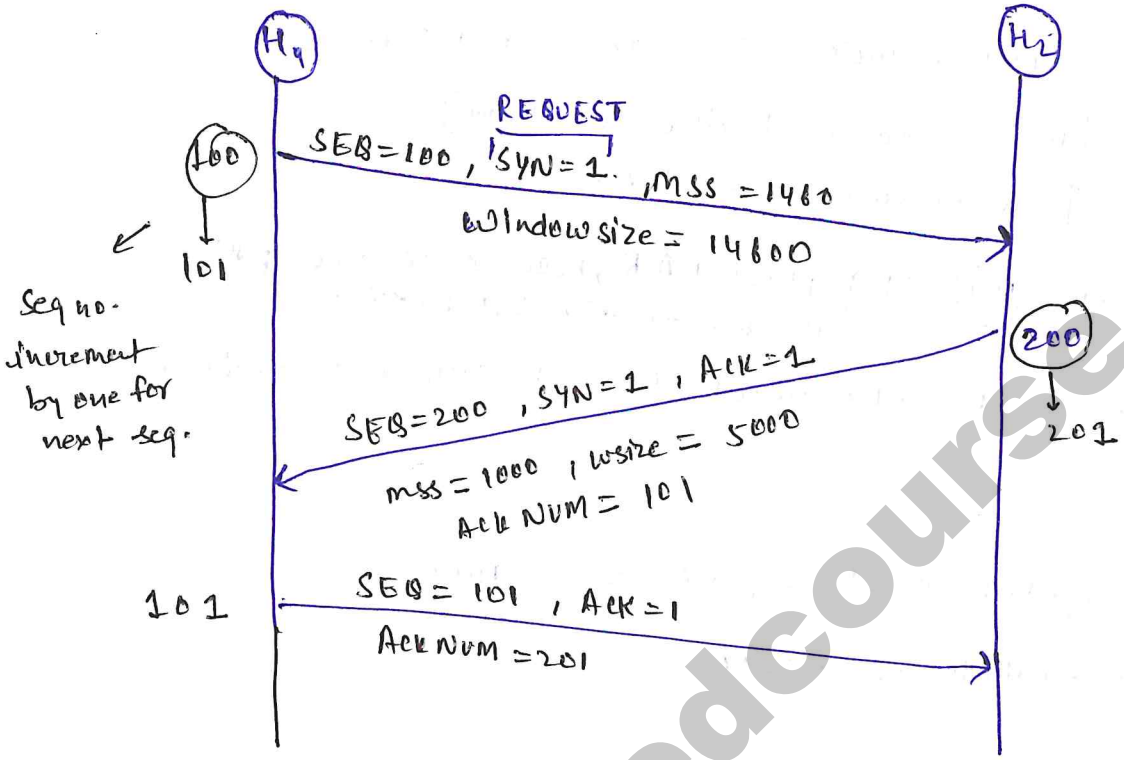
Sender buffer → Stores the bytes which we are transmitting on channel.
or
Receiver buffer

Three stages of TCP :-

- → establishment
- → Transfer
- → Termination

Establishment: 3-way Handshake.

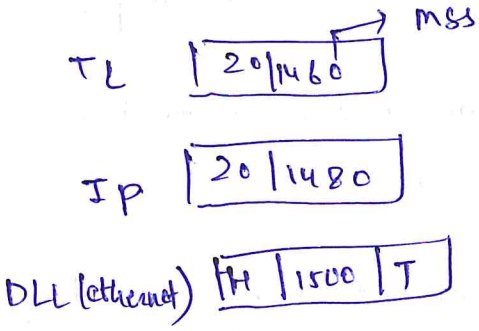
Three way handshake are used for establishing a connection.



* Host 1 sends an packet to H_2 with random seq num = 100 and SYN = 1

MSS (Maximum segment size) → we place mss in options part

→ we are sending from H_1 to H_2, H_1 can be on Ethernet and H_2 can be on another protocol.



* H_1 is informing H_2 that MSS I can handle is 1460 Bytes only.

* By sending window size from H_1 to H_2. H_1 is saying that the maxm window size = 14000 buffers I have to handle.

→ H_2 replies with random SEQ = 200 and SYN = 1

ACK = 1 [means do read the Ack-num]

ACK = 0 [means do not read Ack-num]

Ack NUM = (101) → piggybacked Ack, because we are putting the Ack in the SYN packet.

SYN = 1 means H_2 also want to establish a connection with

→ H_2 also computes maxm Segment size = 1000

and window size = 5000

So, Smaller of [Sw, Rw] will be the size of window established on both sides.

SYN = 1 → we increment the seq no by 1 for the next seq.

ACK = 1 → we increment the seq no by zero for the next seq.

FIN = 1 → we increment the seq no by one for the next seq.

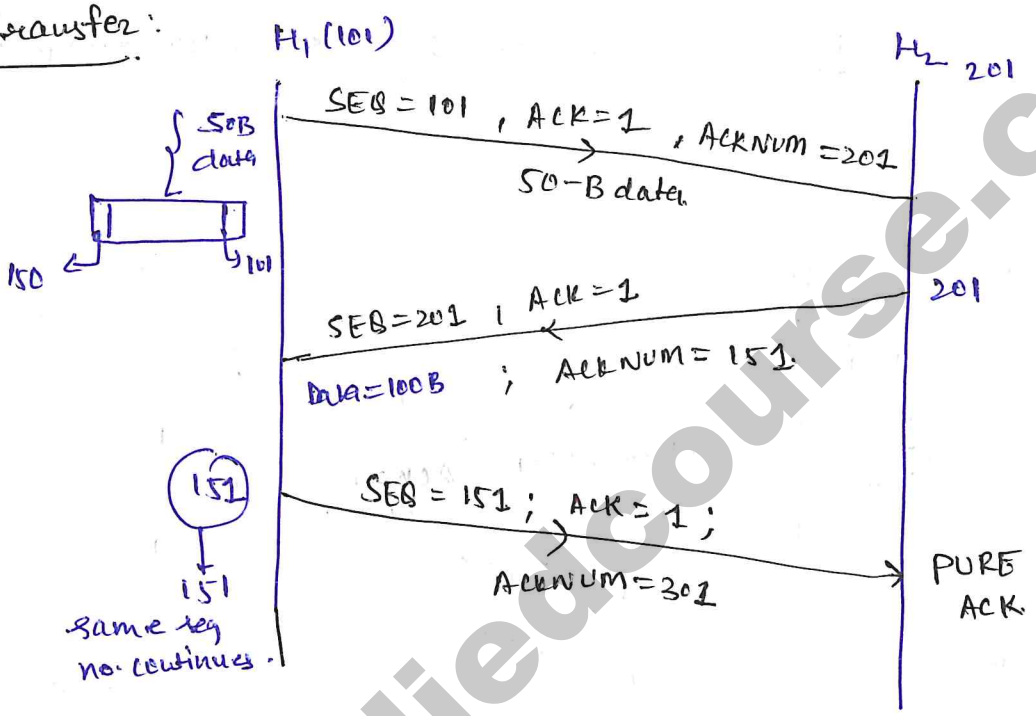
One Data byte → we increment the seq no by one for the next seq.

→ Host 1 acknowledges the Ack from H_2 and replies with SEQ = 101, ACK = 1, Ack-num = 201

↑

The whole process is called the three way handshake.

Data Transfer:



* In data transfer after connection establishment we are using SEQ = 101 further [because last packet sent in the connection establishment is ACK and we do not increment seq no. for ACK]

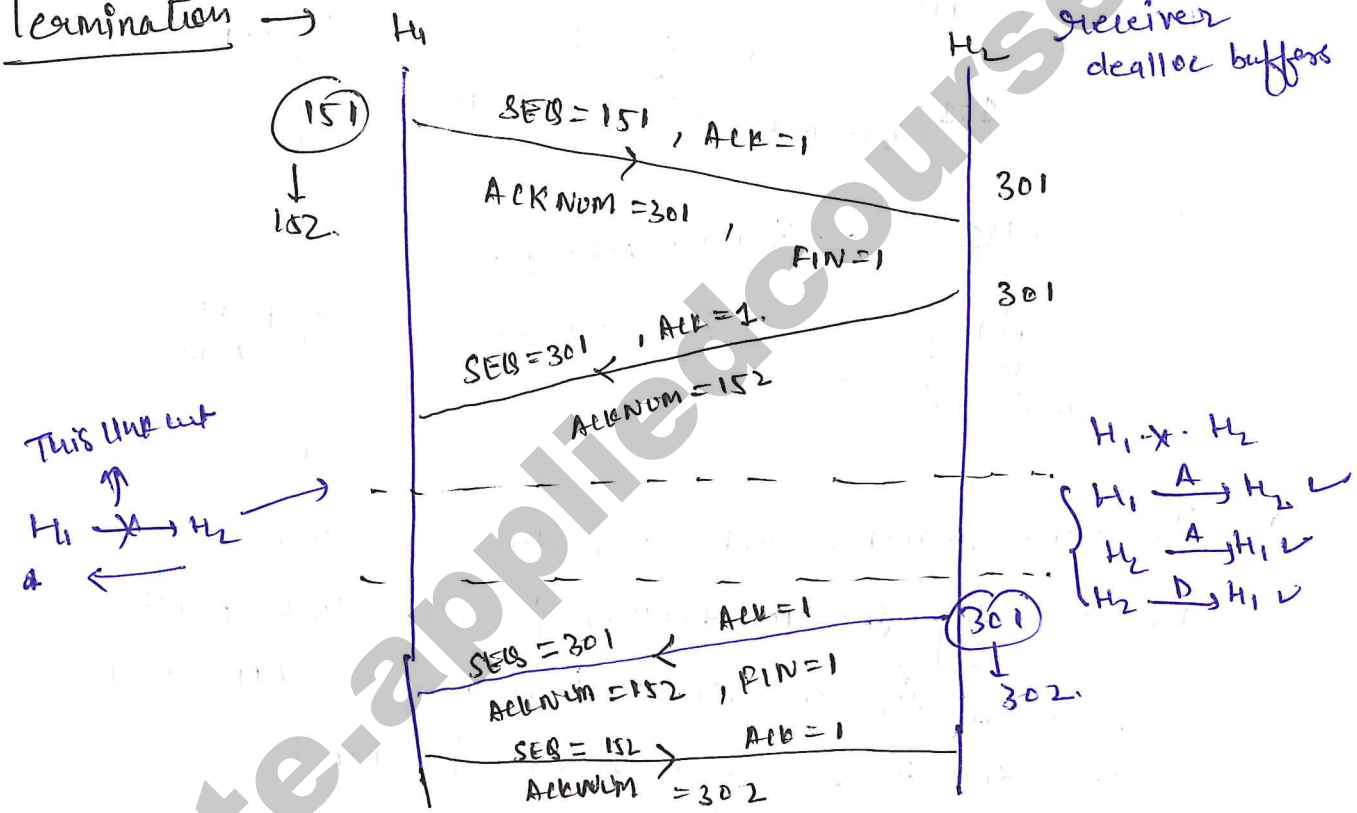
→ we are again sending ACK = 201 for fault tolerance if previous Ack might have lost.

→ The H_2 sends SEQ = 201, ACK = 151 because we have ~~sent~~ received 50 B data

→ Now if we don't have data from H_1 to H_2 we will send Pure ACK with seq no = 151, ACK = 1 Ack Num = 301.

SYN	ACK	
1	0	→ REQUEST (1st)
1	1	→ REPLY (2nd) segment.
0	1	→ Other segments [like data packets]

Termination →



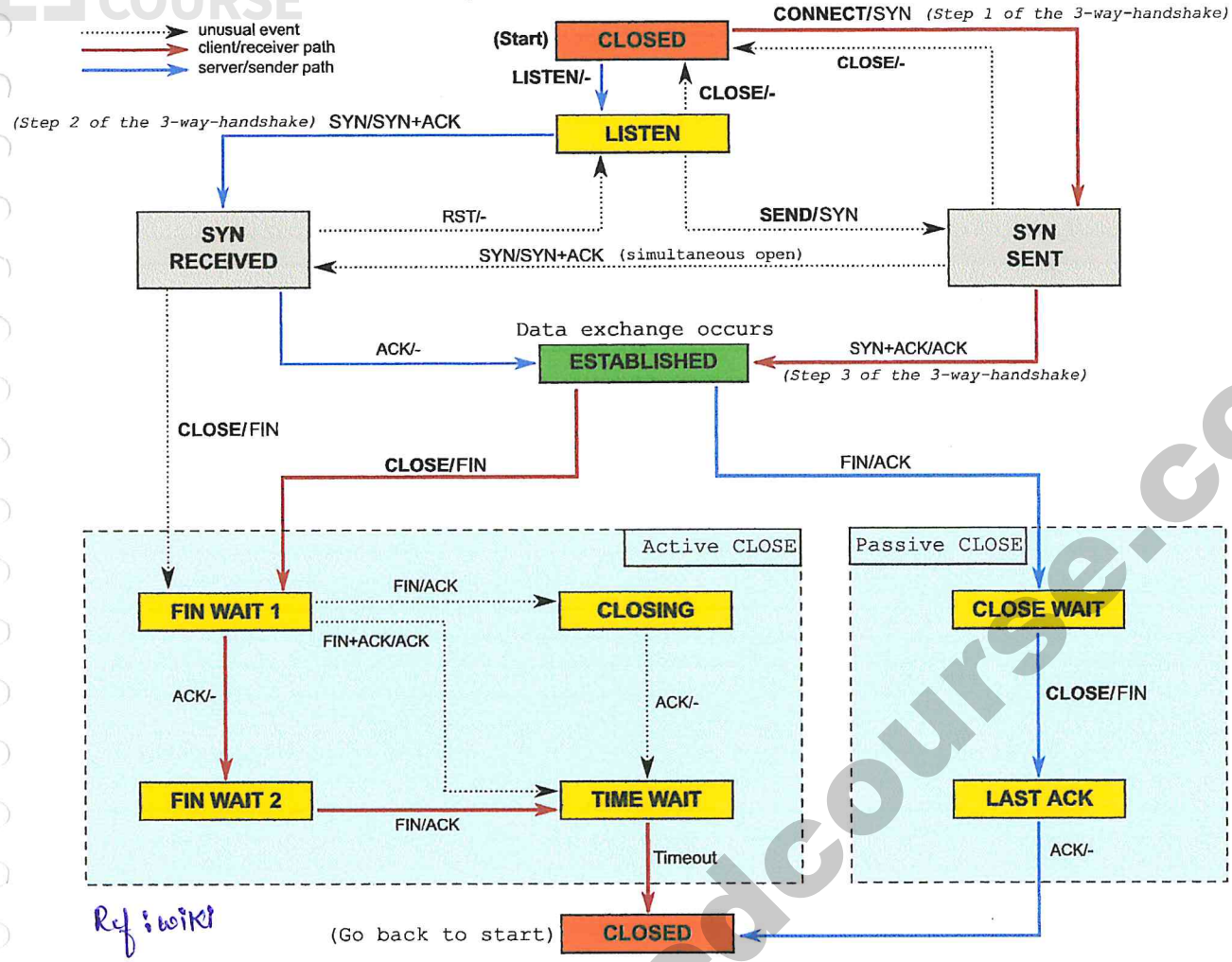
→ By sending FIN = 1 H_1 informing H_2 that it want to terminate the connection.

→ After receiving the Ack from H_2. The link between H_2 to H_1 cut. Now H_1 can not send data.
but H_2 can send data to H_1

→ H_2 also send FIN = 1 for closing the connection

→ Worst case: 4 packets

State Transition diagram:-



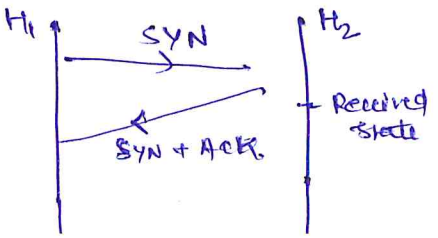
Ref: wiki

* First everything in closed state.
* The brown lines represent the Action corresponding to client, the blue lines correspond the action. corresponds to server, dotted lines are unusual events.

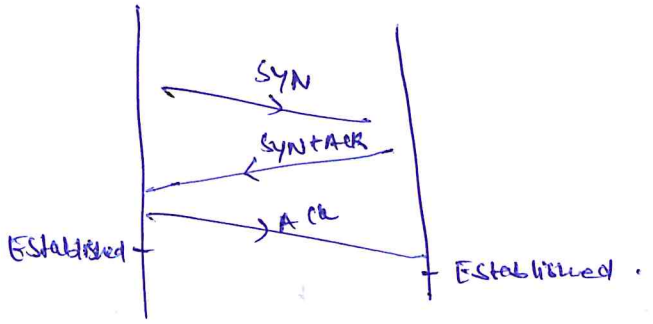
→ Client in closed state and then uses the connect system call and sends a packet SYN=1 request packet.

→ The server is in Listen state and client is in SYN sent state.

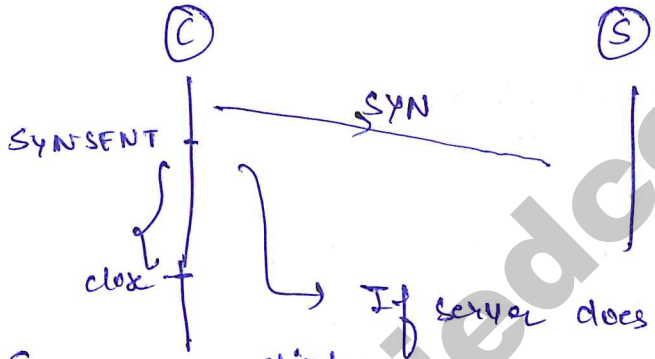
→ The server sent the SYN+ACK and goes to received state.



→ Now sender sent the ACK and both of them went to connection establishment state.



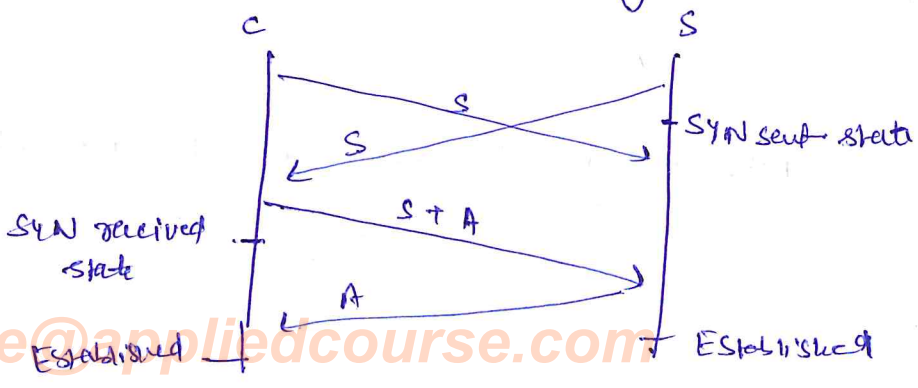
Now understand unusual events: —



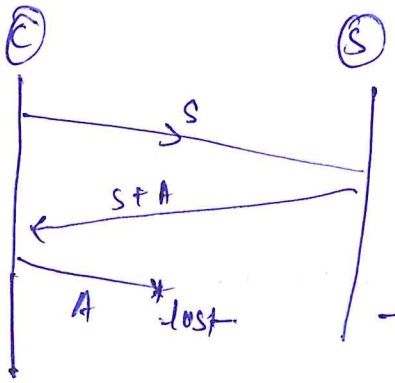
SYN, then client. If server does not respond to the closes the connection and client re-initiates the connection.

→ If server in the listen state and does not get SYN from long time then it also closes the connection.

Another event: — If both server and sender sent the SYN packet together [simultaneous open]



Another unusual events :-



→ Server wait for some time and Reset the whole thing and go to Listen state.

Connection closed mechanism :-

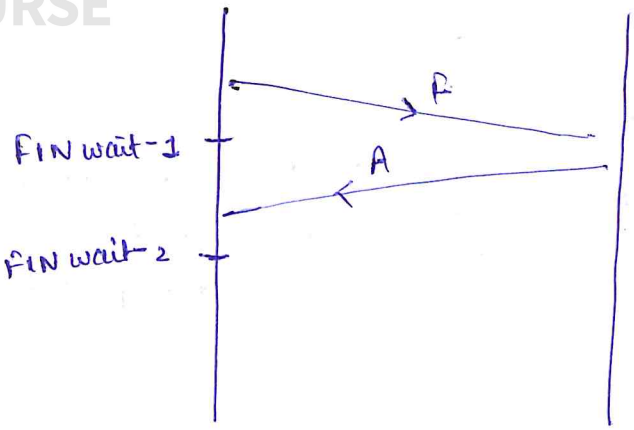
→ Client sends the close signal with FIN packet and goes to FINwait-1 state.

→ Server sends the Ack in response and goes to close wait state.

* There is two types of close : (a) Active close
(b) Passive close.

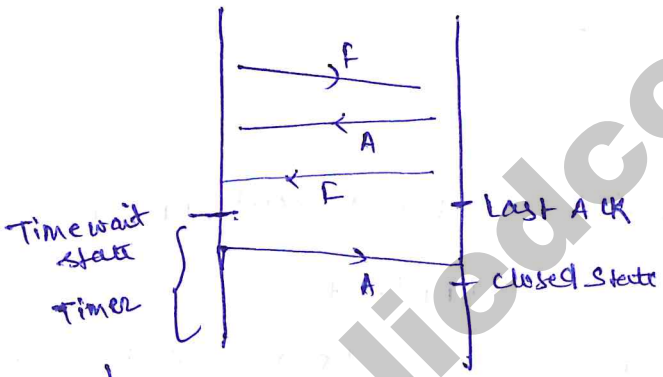
→ Whoever initiates the close Request will go to active close either client or server and other will go to passive close.

→ Client receives an Ack. and goes to client goes to FINwait 2 stat



Now connection from client to server is closed c ⇒ s and client is waiting for FIN state from server.

→ Server sends the FIN packet, client recives the FIN segment and goes to time wait state.

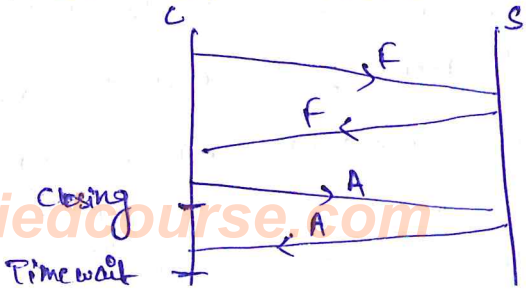


↳ client waits for a few seconds if any packet is coming from server. After timeout it goes to closed state.

I Unusual events:

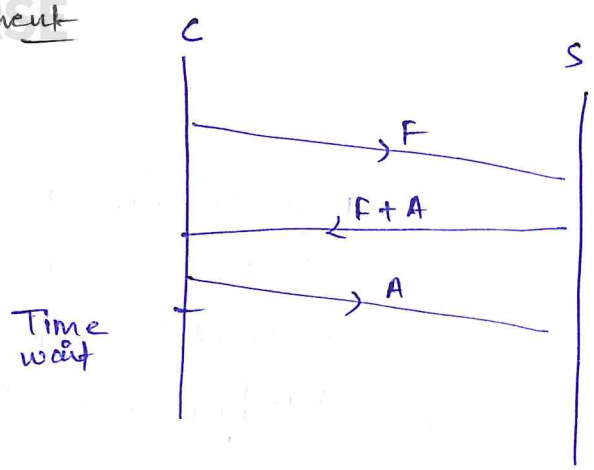
→ Let suppose server is in SYN-received state and wants to close the connection because of some reasons.

II Unusual event → Both client and server sends a FIN



→ Simultaneous closing

T4 Unusual event

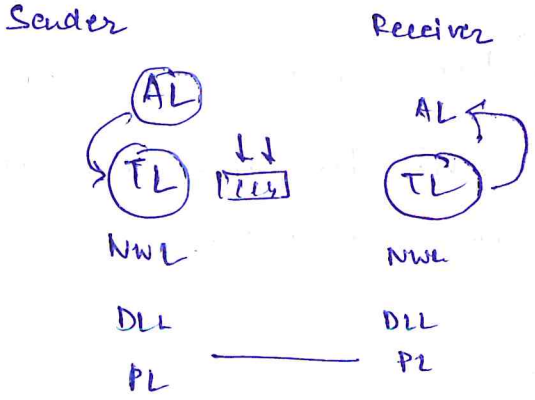


→ If client receives F+A in the same packet

* He can close in less packets also by piggybacking.
* Worst case we need 4 packets to close.

Other fields in TCP header:-

PSH flag: → Max segment size [don't wait upto MSS]
→ Real time [video/audio/RDP/chat]
→ Push data to Network layer (S) → Sender side
→ Push data to AL → Receiver side

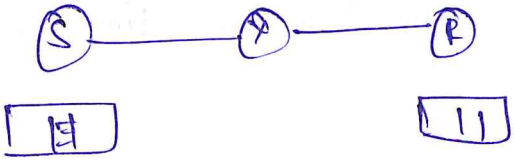


→ The maximum segment size sender and receiver let suppose negotiated with 1000B. If PSH=1 you don't wait 1000B to come from AL to TL to package a segment. whatever data you have [<1000B] also you push down and send to Network Layer.

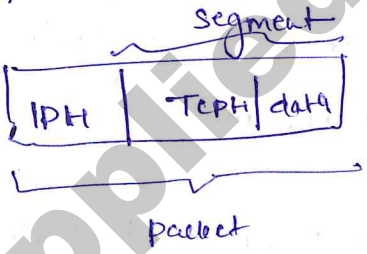
URG and Urgent pointer → (Out of order packet)

→ Send data urgently to the destination.

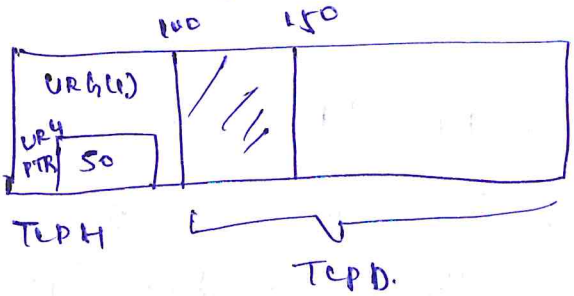
* PSH sends the data in-order while URG can send the data out of order.



→ As routers is the network layer device which can not read the TCP header. So it won't be able to read the URG in TCP header.

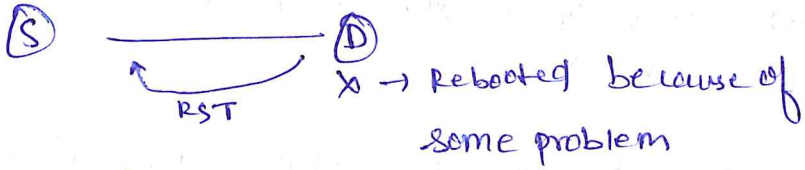


Solution for this problem is In Ipheader we have the priority field where we set priority = 7 (max) for higher priority.



* If URG=1 and URG PTR=50. It means that we have 50 bytes in the whole packet needed to be transfer urgently from SEQ = 100 to SEQ = 150

RST FLAG :- Close the connection due to unexpected errors.



→ In such condition the destination looses the track and close the connection.

→ If Destination gets the unexpected sequence number way higher than expected. also RST the connection.

Options :-

(1) WATCLT ⇒ options field has extra bits (Timestamp).
↳ we have studied earlier.

(2) Max segment size : Parameter negotiation do to fix the mss.

(3) Padding : {header length must be multiple of 4}.
if less fill padding.

(4) **window size :-** (Parameter negotiation.)

16b ⇒ 2^{16} = 64 KB (too small)

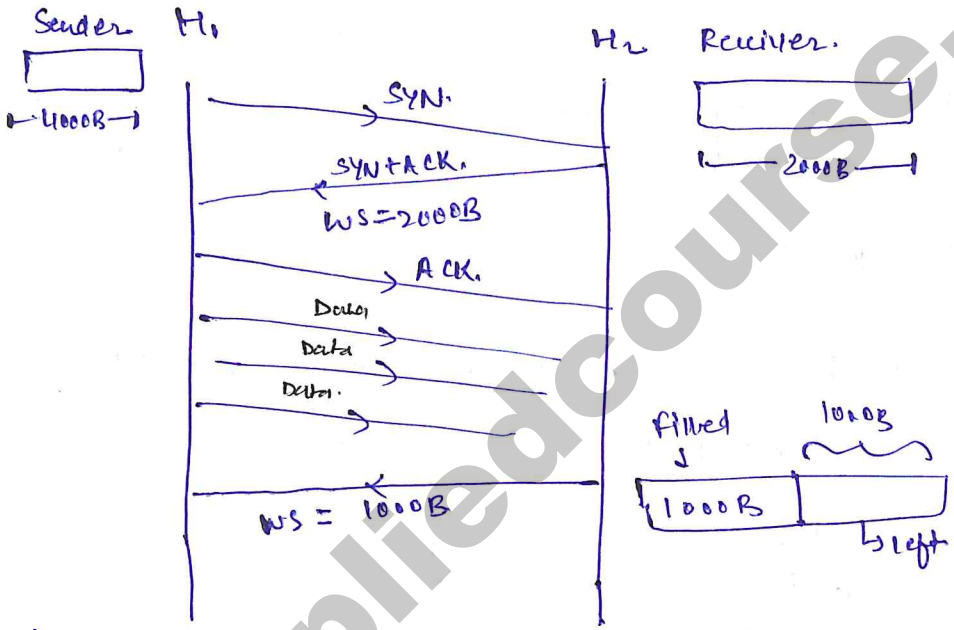
∃ we have 16 bits in window size. If we want bigger window size then we can take some bits from options

30 b ⇒ 1 GB [Good window size]
16b (window size) 14b (Options : Extended window size)

Flow control :-

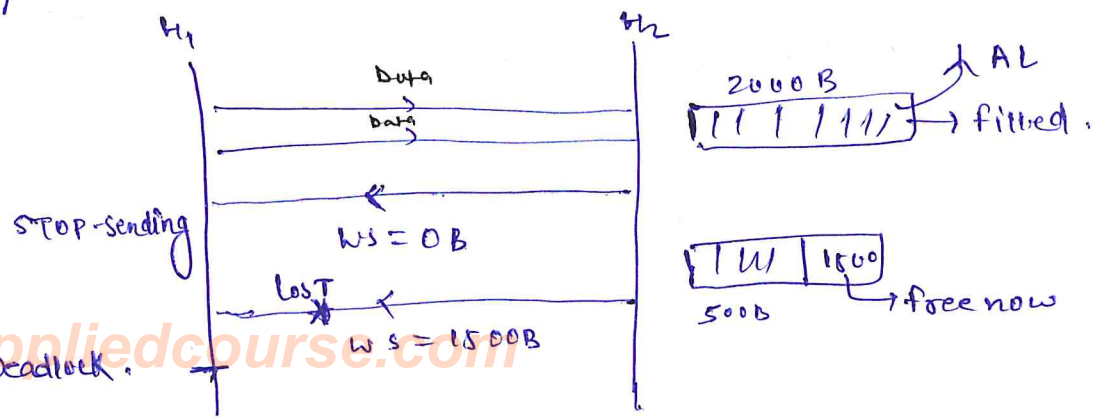
The control the rate and flow of data.

→ Window scaling
→ advertised window size. → This term is similar to window size we read earlier.



* In three way handshake. we are sending the window size. which is also called the advertised window size. because sender and receiver advertising its window size which each other.

* The Receiver again advertised with the window size = 1000B because buffer of receiver is filled and now it is left with only 1000B.



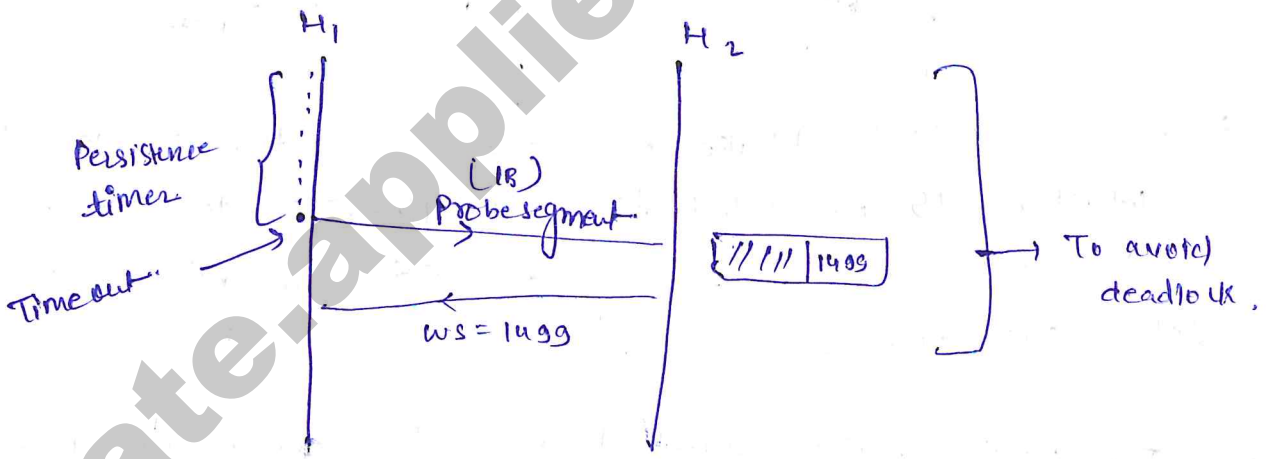
→ When receiver window size is filled. It relpy with WS = 0B to the sender.

→ The sender is waiting for onew window size. After some time when receiver buffered get empty.

→ The Receives again advertised with WS = 1500B and if this packet is lost. This situation is deadlock. because the sender is waiting for window size which receiver has already sent and receiver is waiting for data.

To solve this problem:—

As soon as the sender receives the WS = 0B. The sender will start the persistence timer.



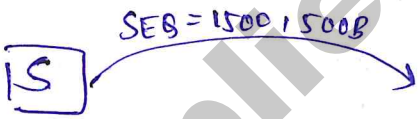
The probe segment of 1B is send by sender and receiver than send the WS = 1499 B because 1B used by buffer.

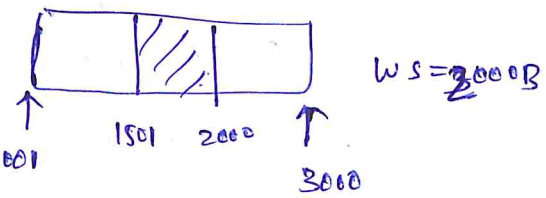
Sliding window and Re-transmission :-

TCP (Hybrid)
- Selective Repeat → Sender ws = Receiver ws, accepts out of order.
- Go-Back-N → Cummulative Ack.

→ TCP uses the hybrid of both.

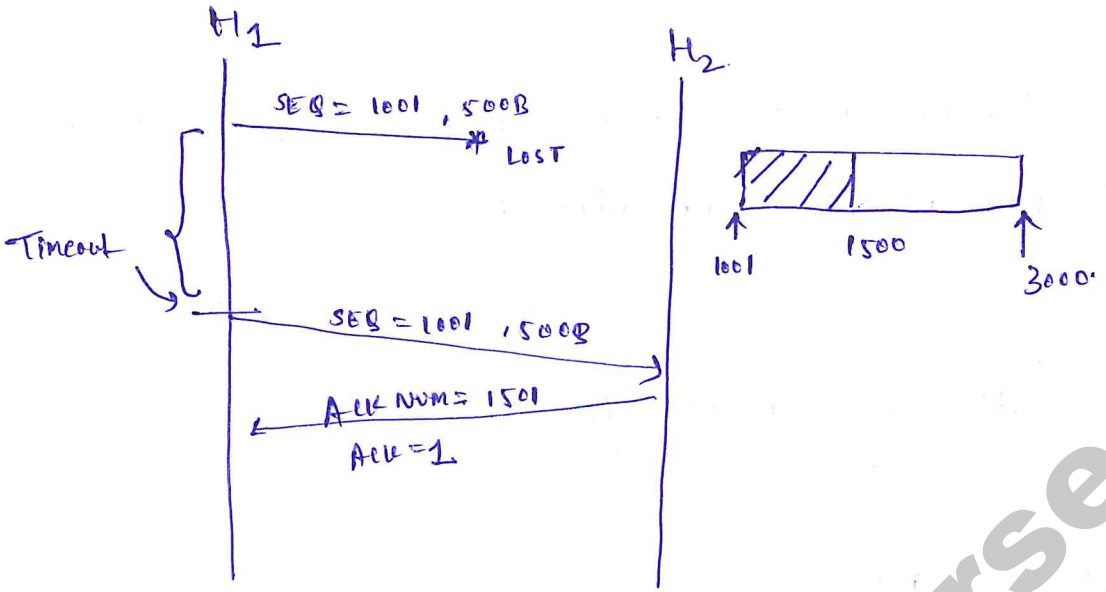
In TCP: * SW = RW
* It also accepts out of order
* It also uses accumulative Ack.



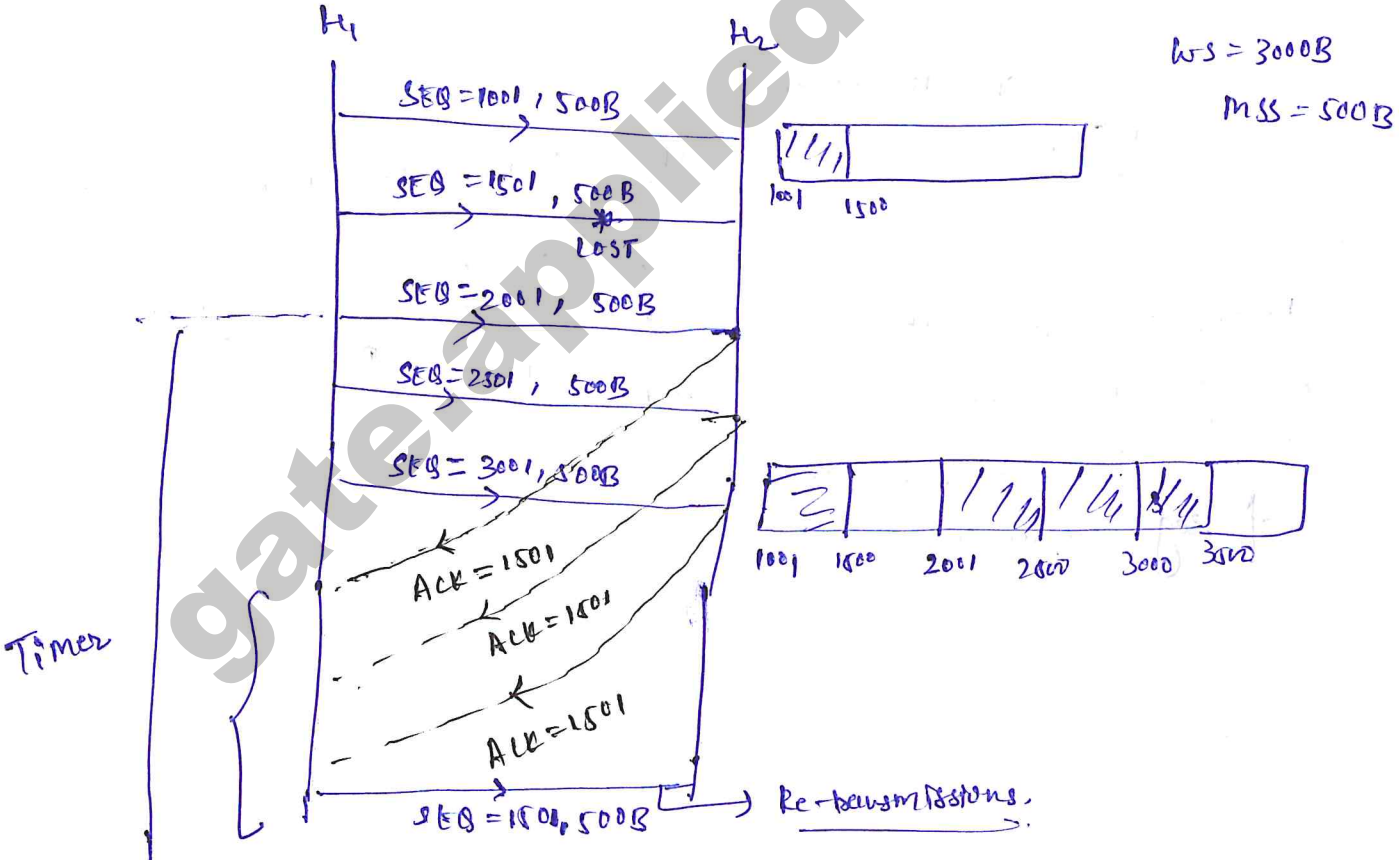


Here, we accept that out of bounds sequence number because belongs to the same window size.

Timeout - based Retransmissions :-



Dup Ack based / Early Retransmissions : (3 duplicate Ack)



→ In TCP we continue sending the segments don't wait for individual Acks.

→ Sender can send out of order frames but receiver always send in-order ACKs in TCP.

→ After receiving three ACK's from sender knows that something has happen and it again sends the lost segment.

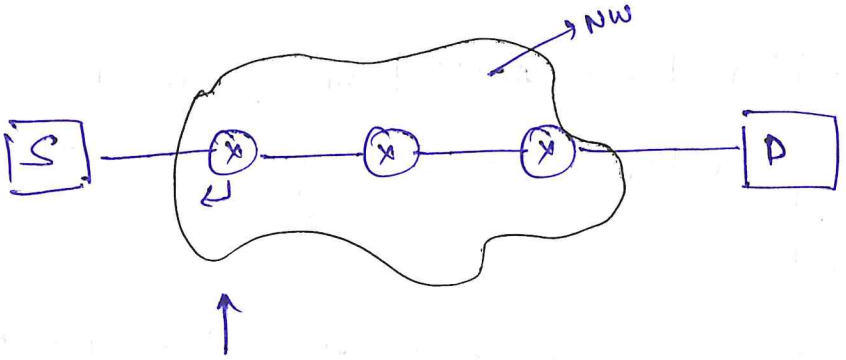
→ Before timeout sender sends the lost packet so called as early re-transmission.

→ TCP doesn't worries about individual ACKs because next ACK after re-transmission will be 3501.
This way TCP knows before that all frame successfully reached.

* If there is too many time outs in the TCP it tells there is strong congestion.

TCP Congestion control :

→ end-end congestion control



ICMP source Quench.
help us to know the congestion on single router

→ In TCP we are concerned about end-end congestion control.

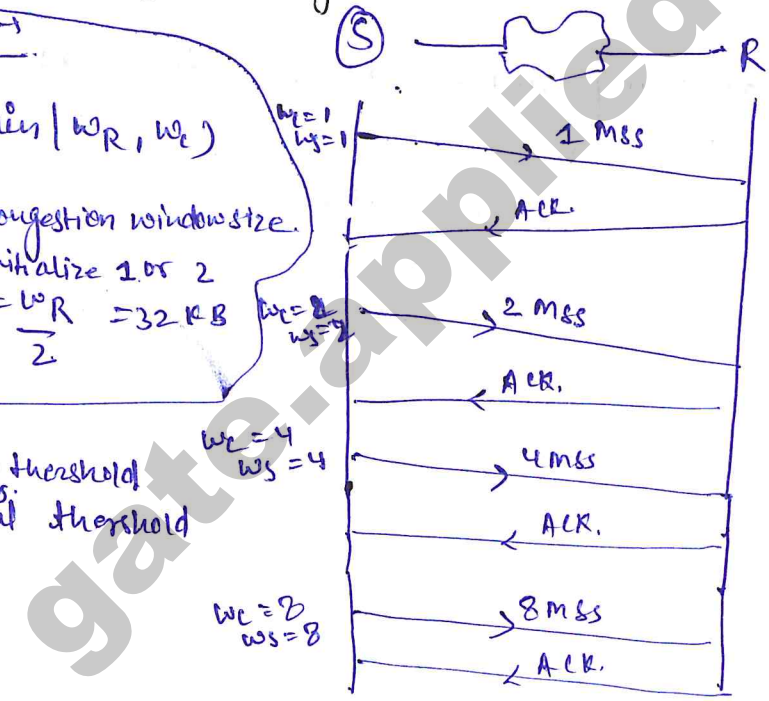
We are assuming receiver window size = fixed

Concept →

$W_S = \min(W_R, W_c)$

W_c = Congestion window size.
W_c = initialize 1 or 2
$W_{th} = \frac{W_R}{2} = 32$ KB

window threshold or initial threshold



$W_R = 64$ KB → 64 segments
MSS = 1 KB

1 2 4 8 16 32 33 34 ---- 64

Slow start phase.
Exponential growth
×2

After threshold go linearly.

Congestion avoidance

We have two phases in congestion control:

(1) Slow start phase [grows exponentially upto threshold]

(2) Congestion avoidance phase [After threshold grows linearly]

→ No. of round trip in previous ex to reach 64.

$= 6 + 31 = 37$

(6) → Slow start; (31) → congestion avoidance.

→ No. of round trip to complete 64 seg.

$= 6 + 32 = 38$

Algorithm:

(1) Slow start phase $w_c = 1$ or 2. (Initial congestion window)
↳ exponential growth.

(2) Congestion avoidance.
↳ linear growth in w_c

(3) Continuously detect congestion.

(a) Timeout ⇒ severe congestion ⇒ $w_{Th} = \frac{w_c}{2}$ & slow start

(b) 3-dup Ack ⇒ mild congestion ⇒ $w_{Th} = \frac{w_c}{2}$ & congesti avoida

Ex→ Initially

$w_R = 64$

So, threshold $= 64/2 = 32$

We perform

Let, Suppose Timeout

$w_c = 1, 2, 4, 8, 16, 32, 33, 34, 35, 36$

Slow start — Congestion avoidance — Sense congestion

So, New $Th = 36/2 = 18$

$w_R = 64$

Now again Slow start :—

1, 2, 4, 8, 16, (18), 19, (20) → 3 duplicate Ack so, mild congestion

18 → 32 not possible so go upto threshold

Slow start — CA

So, new $Th = 20/2 = 10$

$w_R = 64$

Now apply congestion avoidance.

10, 11, 12, 13, 14, 15, 16, 17 ..... (64), 64, 64

Congestion avoidance

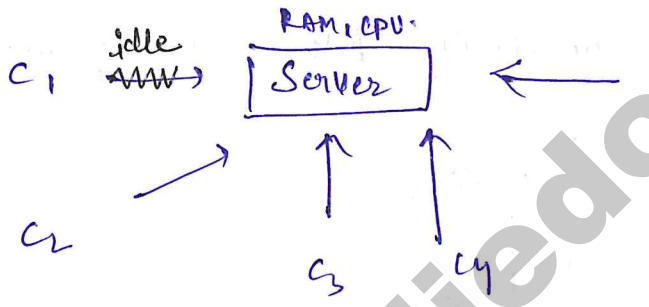
It can not go beyond 64 because receiver window size = 64

Timers:

① Time-wait → This timer we have studied in TCP state transition diagram.

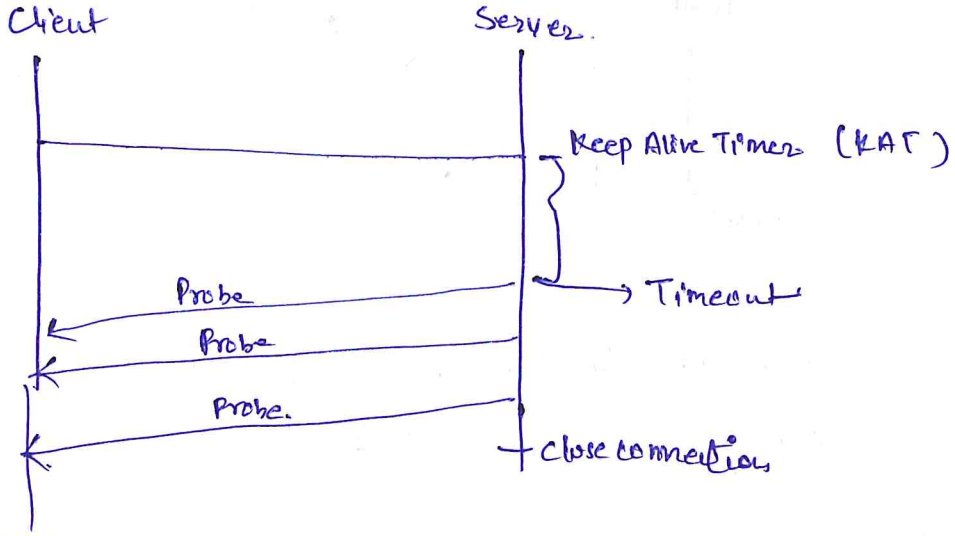
② Persistent timer → This we have studied in the flow control. If ws = 0B advertised by receiver then we start at persistent receiver.

③ Keep alive timer →



Close idle connections.

Server is connected to multiple clients. C_1 is not sending any data such a connection called the idle connection. We want to terminate the idle connections



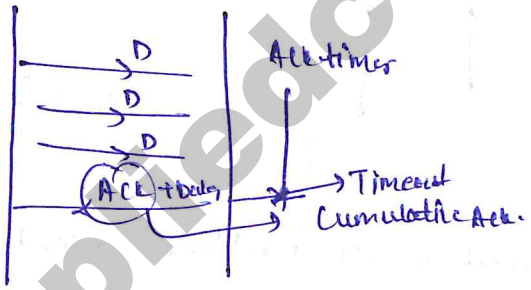
* When server are getting packets from the client it always

update the keep alive timer..

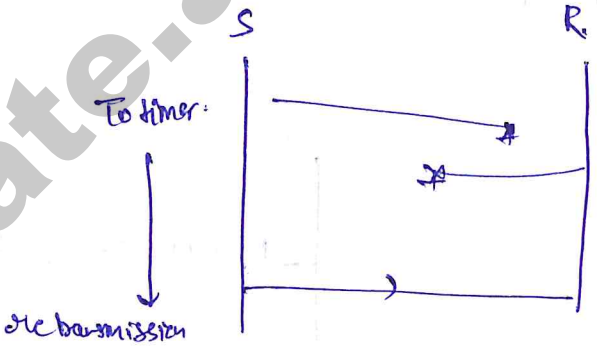
If for some duration of time if there is no packet send by client to server then KAT will timeout

→ Before closing the connection servers sends three probe segment over a period of interval. If there is no response from the sender then client closes the connection.

(4) Acknowledgement Timer → Cumulative Ack and Piggybacked Ack



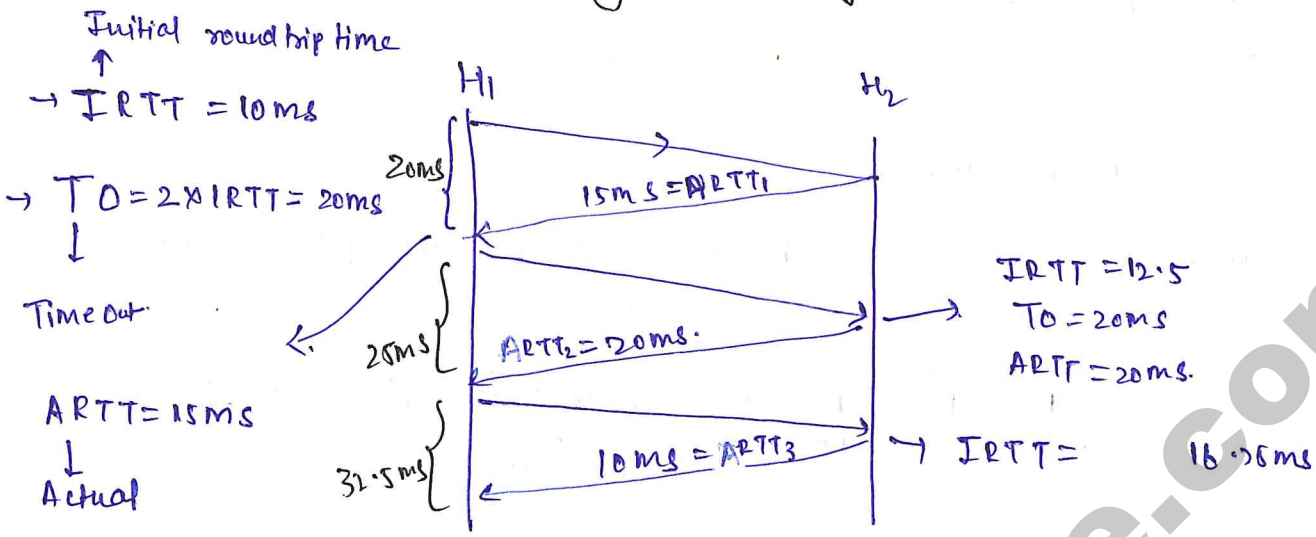
(5) Timeout timer :-



The time-out timer is used for re-transmissions if Ack from receiver is lost

* The timeout timer dynamically changes the value because of change in traffic

Timeout Timer setting (Basic Algo)



NRTT (New round trip time) = α IRTT + $(1-\alpha)$ ARTT

(Initial RTT, Actual RTT)

$0 \leq \alpha \leq 1$

α : Smoothing factor

Let $\alpha = 0.5$

$NRTT = 0.5 \times 10 + (1-0.5) \times 15ms$

$NRTT = 12.5\ ms$

New $TO = 12.5 \times 2 = 25\ ms$

$NRTT_2 = 0.5\ (12.5 + 20) = 16.25ms$

$TO = 32.5\ ms.$

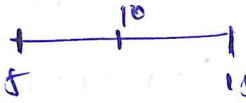
$NRTT_3 = 1/2\ (10 + 16.25) = 13.125ms$

$TO = 26.25\ ms$

Jacobson's Algorithm:

IRTT = 10 ms.

Deviation = ID = 5 ms ⇒ Interval estimation not a point estimation.



⇒ The deviation of 5 ms in the interval.

Timeout = (4 × ID) + IRTT = 30 ms.

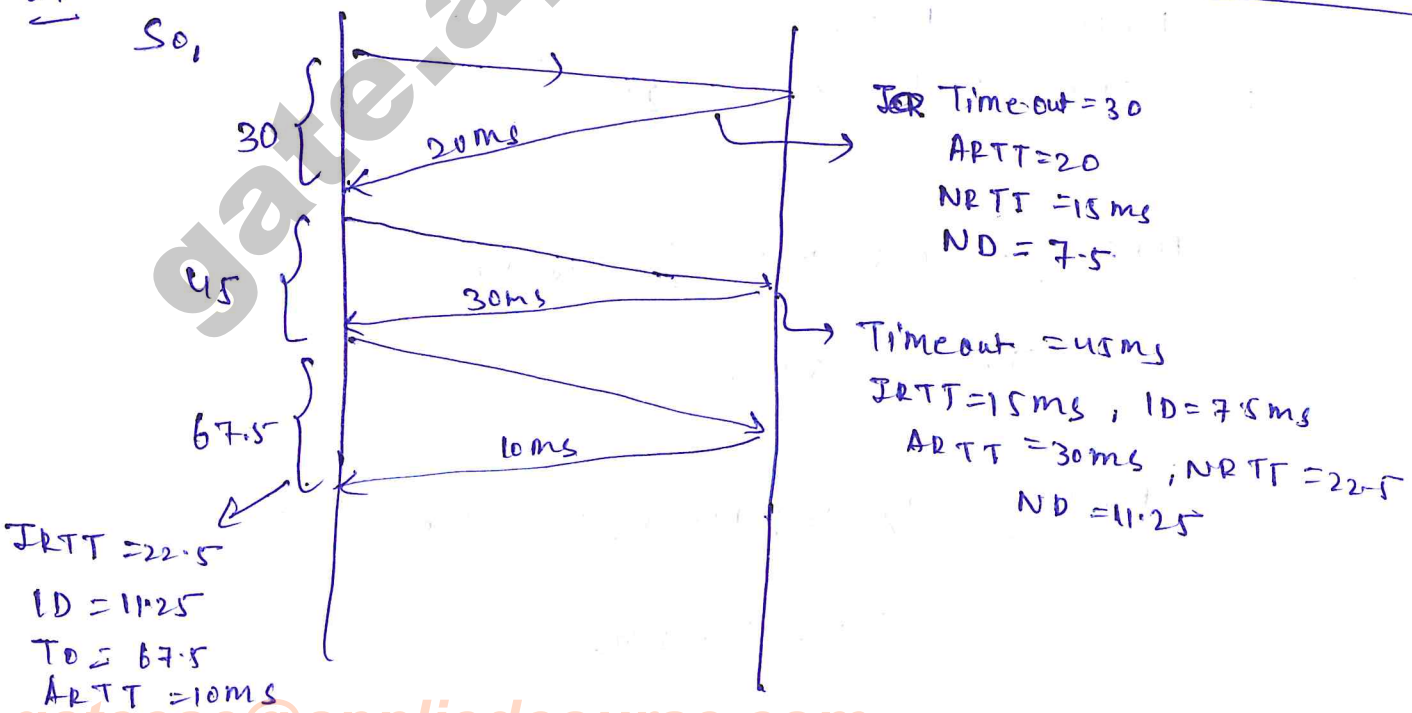
NRTT = α × IRTT + (1−α) ARTT

ND = α × ID + (1−α) AD
(New deviation)

Ad (Actual deviation) = (IRTT − ARTT)

Ex →

So,



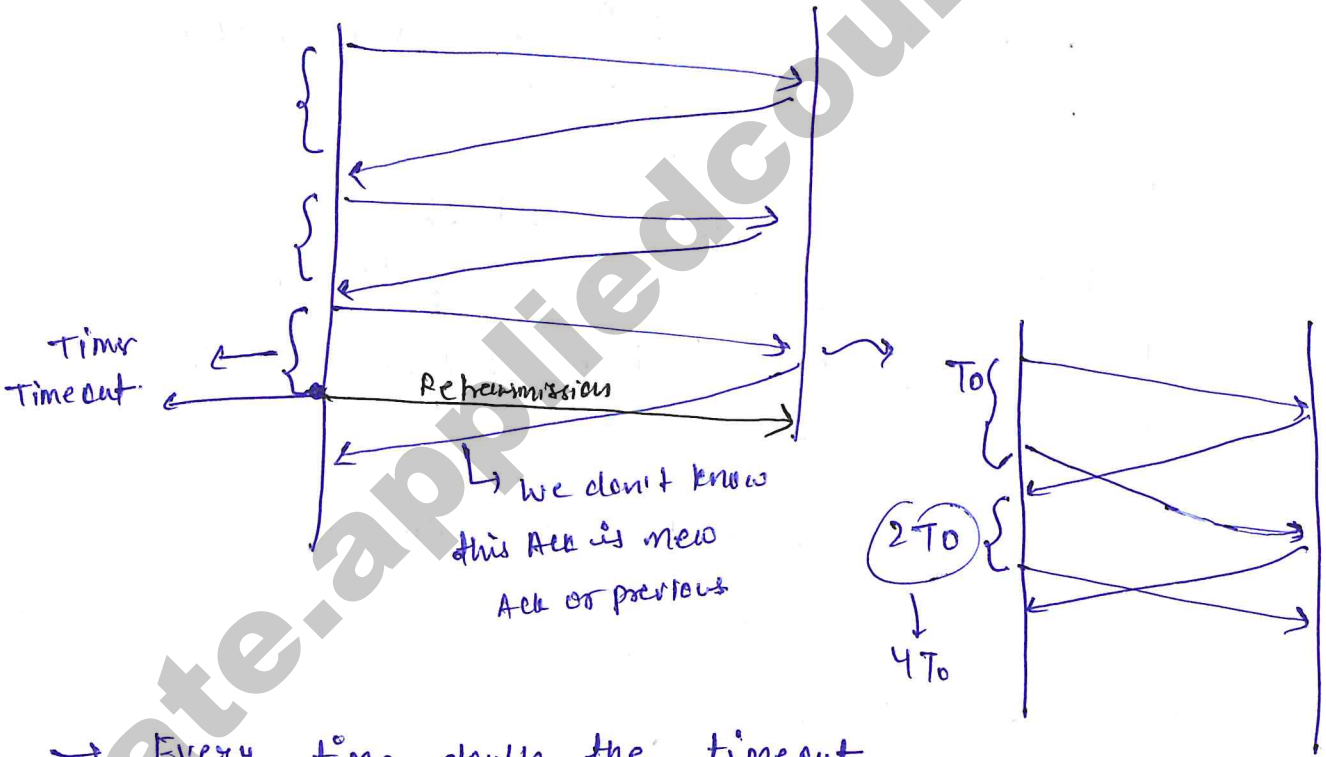
Karn's Modification

→ Whenever a situation arrives

ARTT > TO

* We have to retransmit because timer has already expired.

{ how to calculate NRTT as ARTT is not available.



→ Every time double the timeout when ARTT > TO

↓

Once ARTT < TO then follow the normal concept.

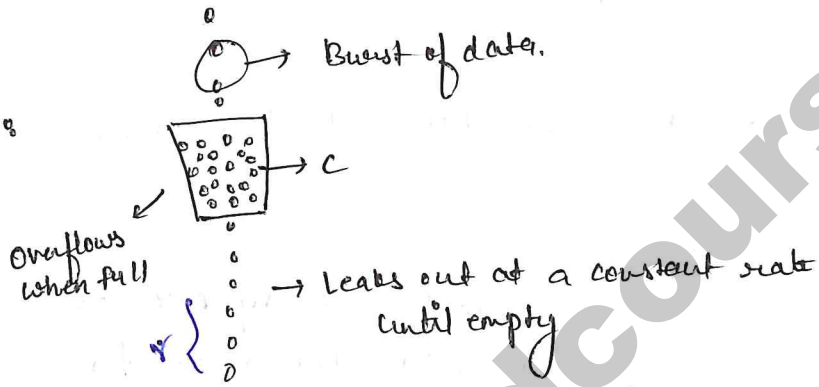
Congestion control : Traffic shaping

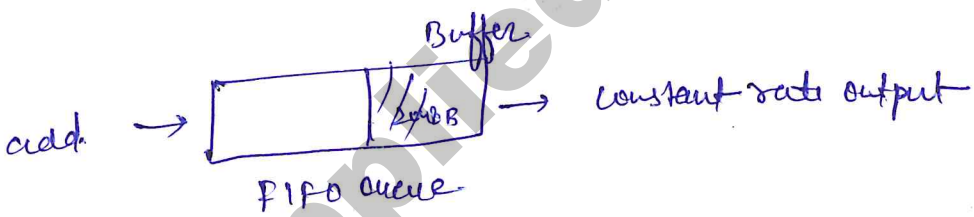
→ Parameter Negotiation : rate of data transfer

* we can negotiate the parameter of rate of data transfer

→ Sender maintains the agreed rate.

Leaky Bucket :

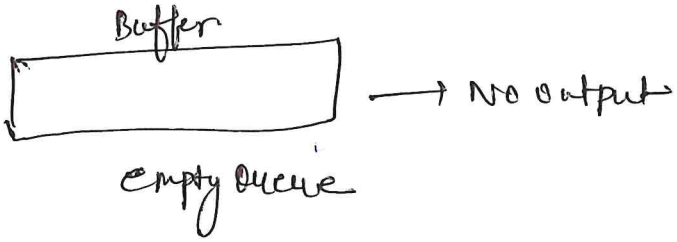




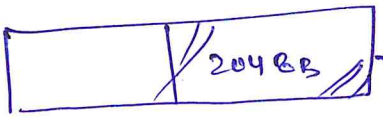
* The output rate is always constant
* If queue is full it will discard the packet / overflow.
* The adding rate might not constant.

Disadvantage :

$t = 0$ to $t = 10$ sec



Initially upto 10 sec there is no data in the Queue.



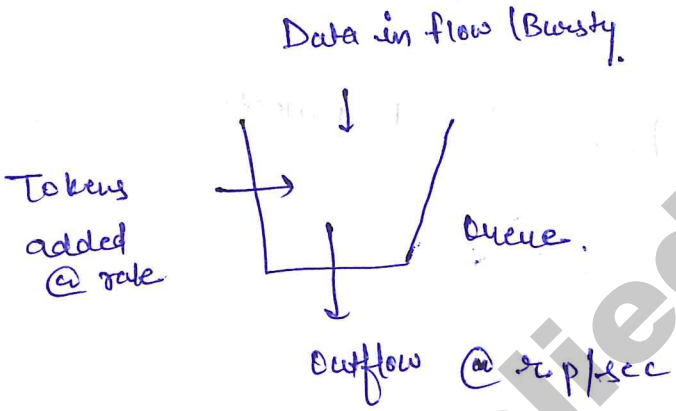
t = 11 sec
(Burst of data)

↳ 4 sec to output this data

T = 11 sec we get burst of data.

Initial 10 sec can not be used because we have no data that time.

Token Bucket :



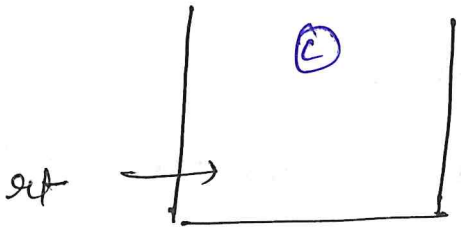
* Let suppose we are adding 5 tokens per second
* So, in 10 sec we have 50 tokens in the token bucket
* At 11 sec we got 4 packets in the bucket.

These 4 packet can outflow suddenly assigned with tokens. We don't have to wait one by one.

Bucket capacity ≥ Tokens in the bucket @ any time

Max no of packet in t sec.

0 ——— t



→ In t sec rt number of tokens will be added in this bucket.

→ Max no. of packet in t sec = C + rt (Initial C packets)

→ Max avg rate = $\frac{C + rt}{t}$ · p/sec → p = packet

UDP (User datagram protocol)

(1) UDP is connection less protocol. It does not uses byte stream like TCP

(2) Application layer protocol need few req/response packets.

e.g. DNS, BOOTP, DHCP, OSPF (Routing)

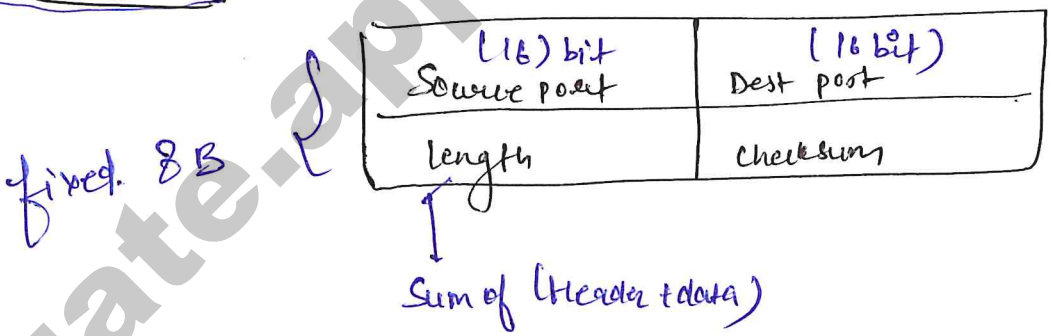
↳ There is no need of connection establishment overhead.

(3) Broadcasting and multicasting (many hosts).

↳ TCP has lot of overhead needed buffer

Use UDP for most of these operations.

UDP header →

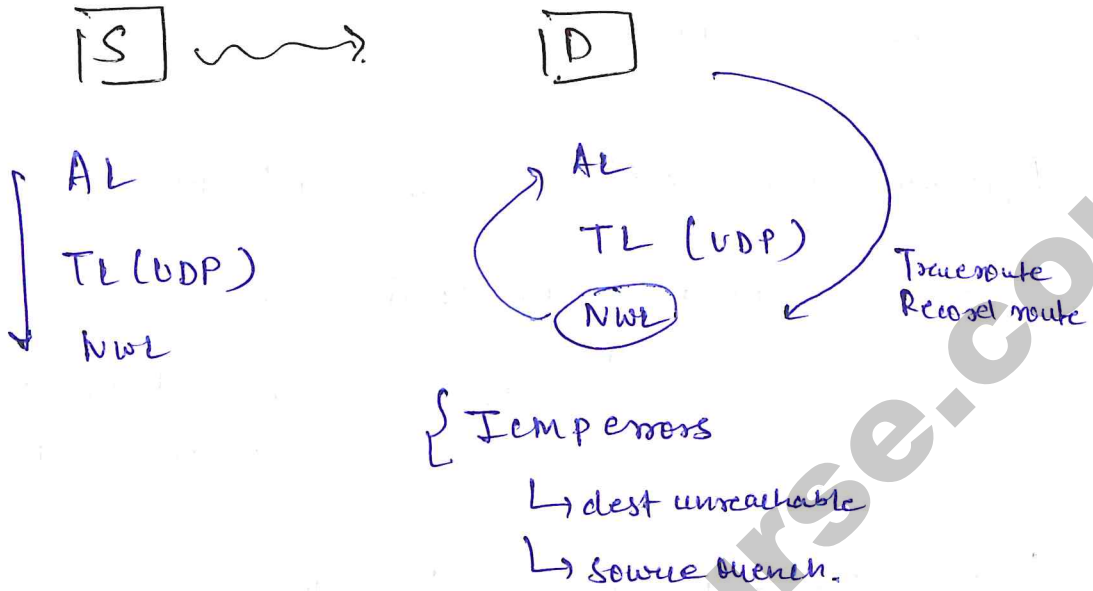


* $2^{16} - 1$ = max size of segment.

Checksum is computed = UDP-H + UDP data + Pseudo header

* For speed we don't compute checksum and often ly. it is disabled and all one's are filled.

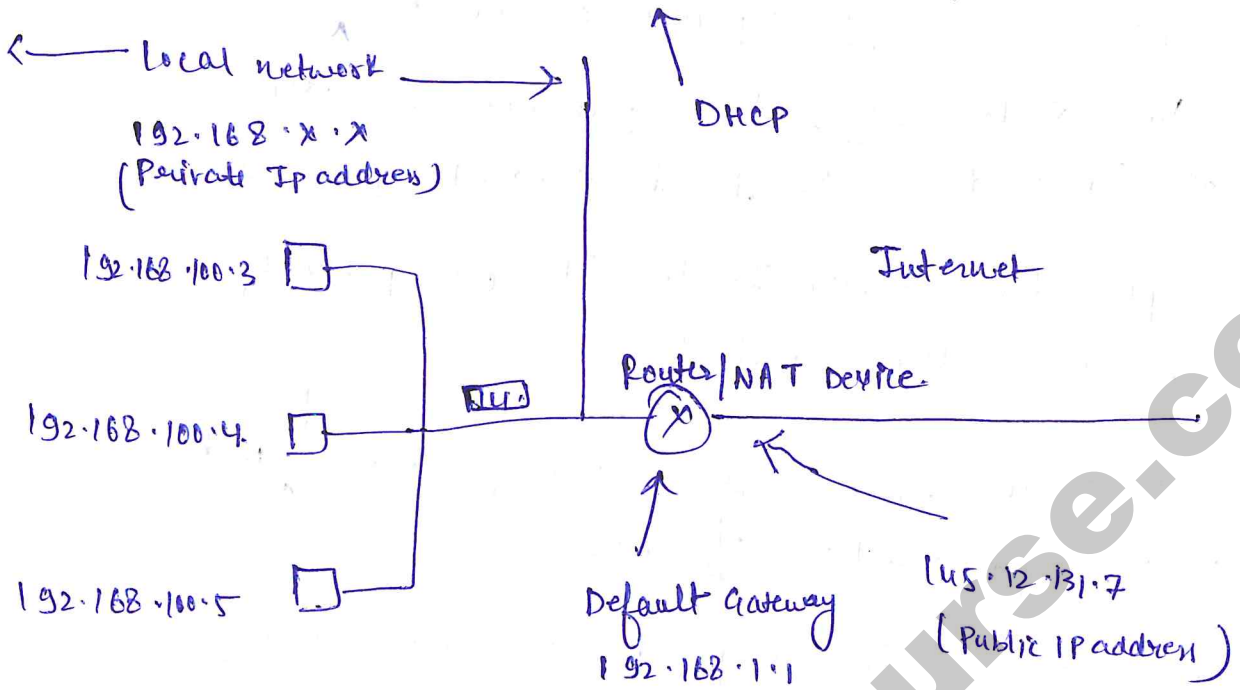
Additional task of UDP:



{ Icmp errors
  ↳ dest unreachable
  ↳ source quench.

* One of the main task of UDP to communicate key information from NWL to AL like Icmp errors.
* It also facilitates additional information from AL to network layer

Network Address Translation :- Task of network layer.



Private Ip range : 10.0.0.0 — 10.255.255.255 /8 (16777216 hosts)
172.16.0.0 — 172.31.255.255 /12 (1048576 hosts)
192.168.0.0 — 192.168.255.255 /16 (65536 hosts)

Translation table of NAT router :

Pedrate Ip	Post
192.168.100.3	5000

* The NAT router converts the private Ip to public Ip while sending packet out in the network.
* The NAT router converts public Ip to private Ip when sends packet in the network.

* The another task of NAT router is whenever the system is sending a packet outside the network the NAT router assigns the port number with private Ip address in the table and after assigning it replace the private Ip to its own public Ip and send to the destination.

* The destination replies with public Ip address with assigned port number of router.

* Router forward the reply packet with changing the public Ip to private Ip with the assigned port number by looking in table.

Solved Problems:

Que: Consider a long lived TCP session with an end-to-end bandwidth of 1 Gbps = (10^9 bits-per second). The session starts with sequence number of 1234. The minimum time (in seconds, rounded to the closest integer) before that sequence number can be used again is? ____

Ans: Bw = 1 Gbps

For seq no = 32 bits

we have = 2^{32} B (data to transfer)

→ sequence bits goes in cyclic [ignore headers]

So, we have to pass 2^{32} B to again come to 1234

$$\text{Total time} = \frac{2^{32} \times 8\ b}{10^9\ bps} = 34.35\ sec = 34\ (Ans)$$

closest integer

Que:

	Field.		length in bits
P.	UDP header Port no.	I.	48
Q.	Ethernet MAC addr	II.	8
R.	IPv6 next header	III.	32
S.	TCP header's sequence number.	IV.	16

Match the following?

(A) P - III, Q - IV, R - II, S - I

(B) P - II, Q - I, R - IV, S - III

(C) P - IV, Q - I, R - II, S - III

(D) P - IV, Q - I, R - III, S - II

Ans → C.

UDP portno = 16bit

Ethernet mac addr = 48bit

IPv6 next header =

TCP header's sequence no = 32

Que: Consider the following statements regarding the slowstart phase of the TCP congestion control algorithm. Note that Cwind stands for the TCP congestion window and MSS denotes the maximum segment size.

1. Cwnd increase by 2MSS on every sucessful acknowledgement.
2. The Cwnd approximately doubles on every sucessful acknowledgement.
3. The Cwnd increase by 1MSS every round trip time.
4. The Cwnd approximately doubles every round trip time.

Which one of the following is correct?

(A) Only (i) and (iii) are true
(B) only (i) and (iv) are true.
(C) only (iv) is true
(D) only (i) and (iv) are true

Ans: MSS = Maximum segment size

Cwnd = Congestion window.

1 → wrong it doubles in slow start phase

2 → It does not necessary for TCP to wait for ACK.

3 → wrong

4 → True, It approximately doubles every round trip time.

Que: Consider a TCP client and a TCP server running on two different machines. After completing data transfer, the TCP client calls close to terminate the connection and a FIN segment is sent to the TCP server. Server-side TCP responds by sending and Ack. which is received by the client-side TCP. As per the TCP connection state diagram (RFC-793), in which state does the client side TCP connection wait for the FIN from the server-side TCP?

1. LAST-ACK
2. TIME-WAIT
3. FIN-WAIT-1
4. FIN-WAIT-2

Ans: Ans: FIN WAIT-2 [Refer State transition diagram TCP]

Que: Consider a socket API on Linux machine that supports UDP socket. A connected UDP socket is a UDP socket on which connect function has already been called. Which of the following statements is/are correct?

I. A connected UDP socket can be used to communicate with multiple peers simultaneously.

II. A process can sucessfully call connect function again for an already connected UDP socket.

(A) I only

(B) II only

(C) Both I and II only

(D) Neither I nor II

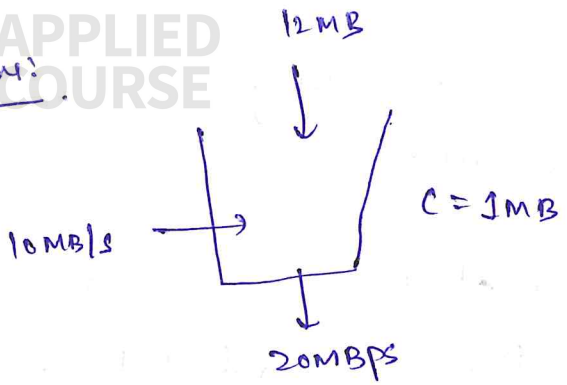
Ans:

I → In a UDP socket when it is connected we can not communicate to multiple peers because every peer have different socket address.

II. Yes, UDP is connectionless service we can call connect function again to serve a different socket. There is no concept of closing or terminating a connection.

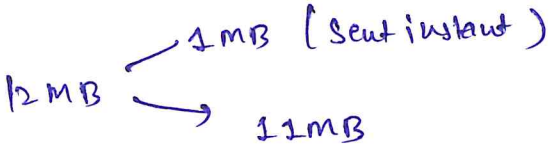
Que: For a host machine that uses the token bucket algorithm for congestion control, the token bucket has a capacity of 1 megabyte and the maximum output rate is 20 megabytes per second. Tokens arrive at a rate to sustain output at a rate of 10 megabytes per second. The token bucket is currently full and machine needs to send 12 megabytes of data. The minimum time required to transmit the data is ____ seconds?

Ans:



In 12MB data because of 1MB token.

1MB data will send instantaneously.



1 sec = 10MB tokens generated.

1.1 sec = 11MB tokens generated

The minimum time to transmit is 1.1 sec.

Que: Suppose two hosts use a TCP connection to transfer a large file. Which of the following statements is/are false with respect to the TCP connection?

1. If sequence number of a segment is m, then the sequence number of the subsequent segment is always m+1.
2. If the estimated round trip time at any given point of time is t sec, the value of the retransmission timeout is always set to greater than or equal to t sec.

3. The size of the advertised window never changes during the course of the TCP connection.

4. The number of unacknowledged bytes at the sender is always less than or equal to the advertised window.

(A) 3 only
(B) 1 and 3 only
(C) 1 and 4 only
(D) 2 and 4 only.

Ans: (1) In TCP we uses the the byte stream the. Sequence number of next segment is [m + how many data bytes.] → false

(2)(3) The size of advertised window do change so this statement false.

Ans → B.

Que: Identify the correct order in which a server process must invoke the function calls accept, bind, listen and recv according to UNIX socket API.

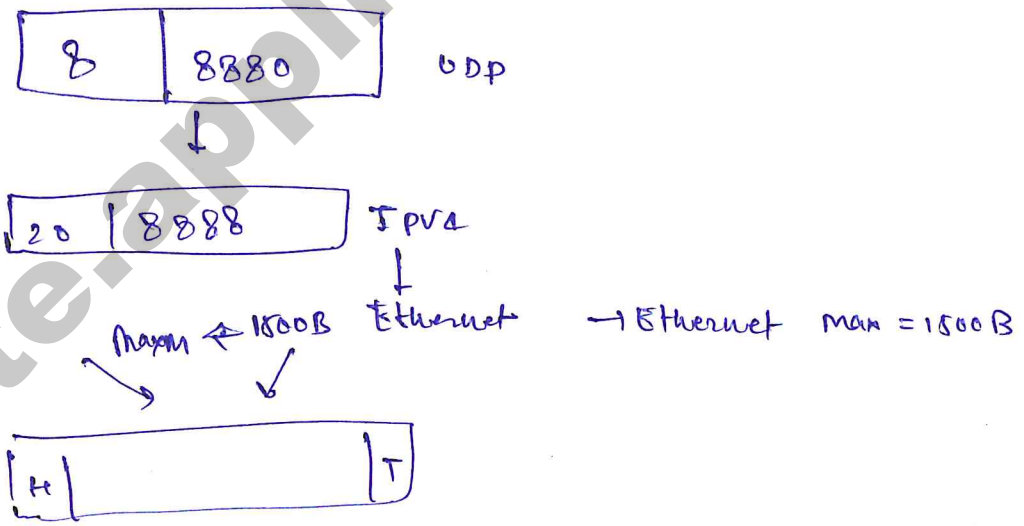
(A) listen, accept, bind, recv
(B) bind, listen, accept, recv
(C) bind, accept, listen, recv
(D) accept, listen, bind, recv

Ans → First bind the port number, listen for connections, accept and then recv.

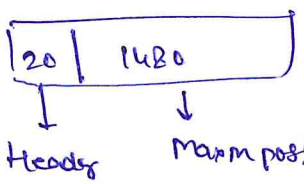
Que: Host A sends a UDP datagram containing 8880 bytes of user data to host B over an Ethernet LAN. Ethernet frames may carry data up to 1500 bytes (i.e MTU = 1500 bytes). Size of UDP header is 8 bytes and size of IP header is 20 bytes. There is no option field in the IP header. How many total number of IP fragments will be transmitted and what will be the contents of offset field in the last fragment?

(A) 6 and 925
(B) 6 and 7400
(C) 7 and 1110
(D) 7 and 8880

Ans:



So, $\frac{8888}{1480} = 7$ fragments.



$$\text{Fn}_{\text{fragments}}^{6} = \frac{1480 \times 6}{8} = 1110$$ bits

Que: Consider the following statements.

✓ I. Tcp connection are full duplex.

✗ II. Tcp has no option for selective acknowledgement.

✗ III. Tcp connection are message streams.

(A) Only I correct

(B) only I and IV are correct.

(C) only II and III are correct.

(D) All I, II, III are correct

II → Using option field we can give selective Ack's.

III → They are byte streams

Application Layer Protocols :-

→ DNS, SMTP, POP, FTP, HTTP

DNS → will see.
SMTP, POP → email
FTP → files
HTTP → web pages.

DNS → Domain name ~~server~~

Google.com → Domain name

→ DNS helps us getting Ip address of the domain name.
→ DNS uses UDP protocol.
→ IP address are dynamic they can change.
→ If you do nslookup command on terminal

nslookup google.com → Uses DNS protocol.

It will give you ip address of google.

DNS Tables:

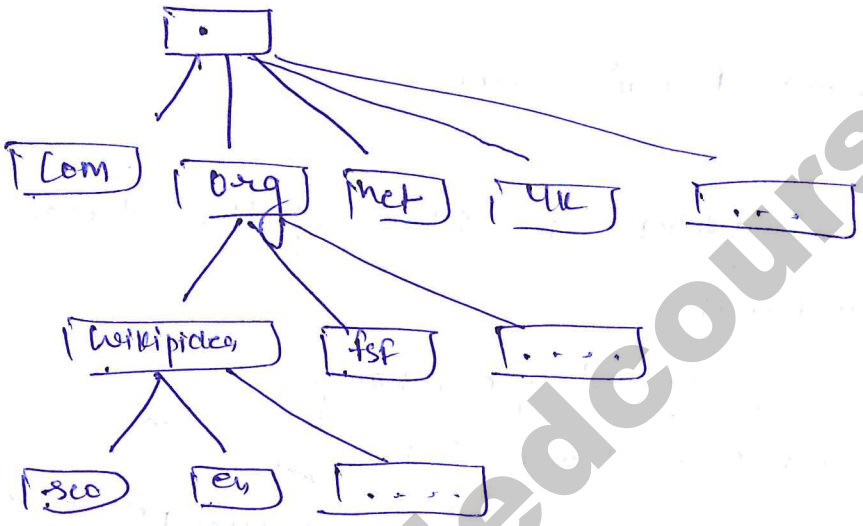
DN	IP	TTL

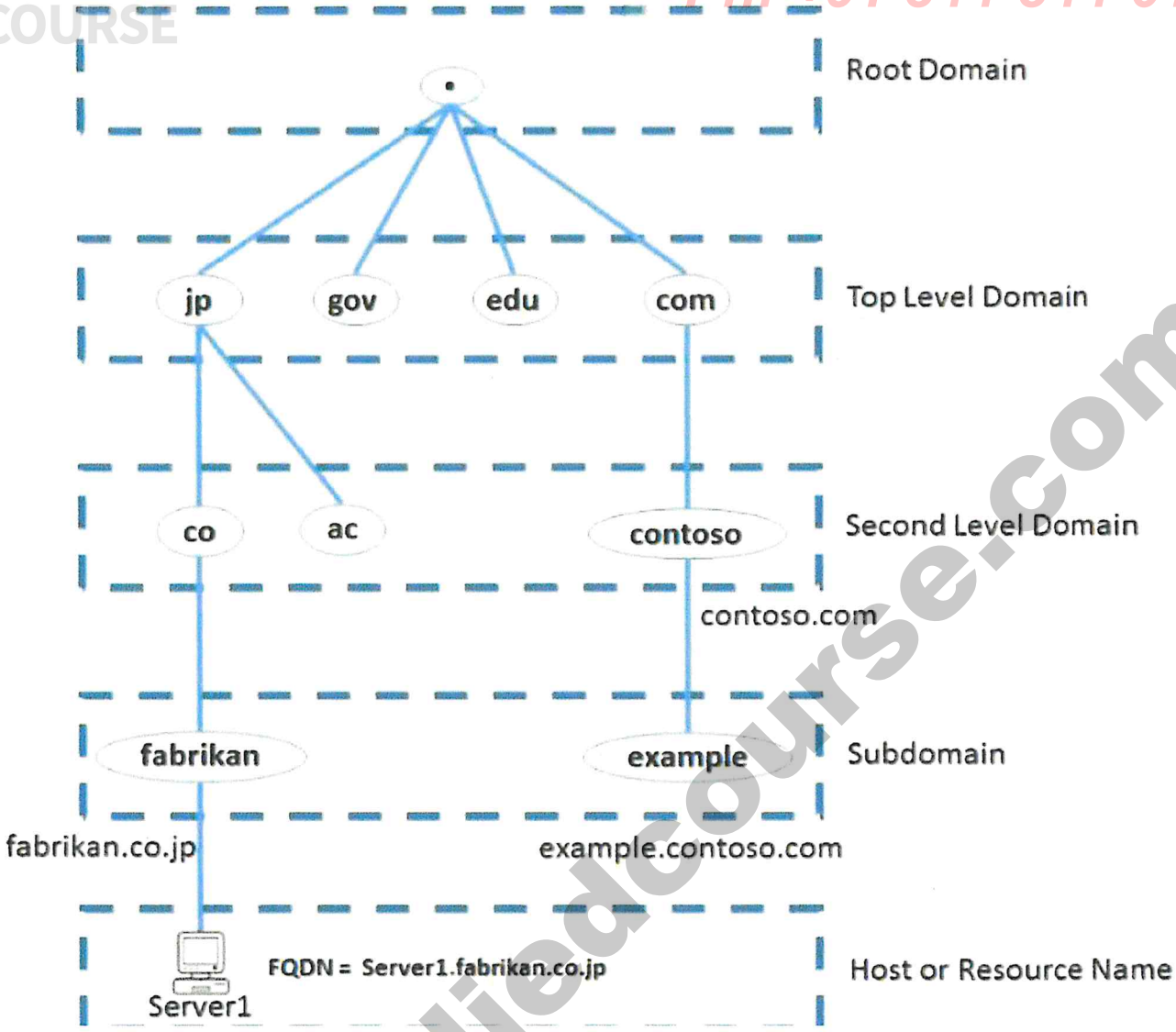
Domain names:

Generic domain: .com, .org, .edu, .net, .mil

Country domain: .in, .jp, .uk, .de

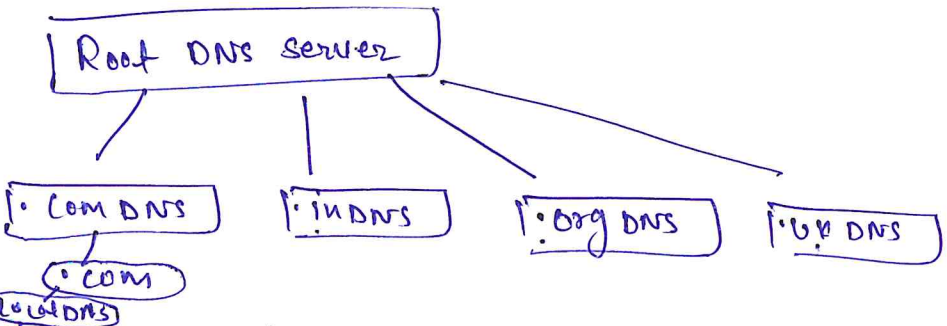
.in → India
.jp → Japan





→ This is how the hierarchy of the DNS servers.

* DNS database organization :-

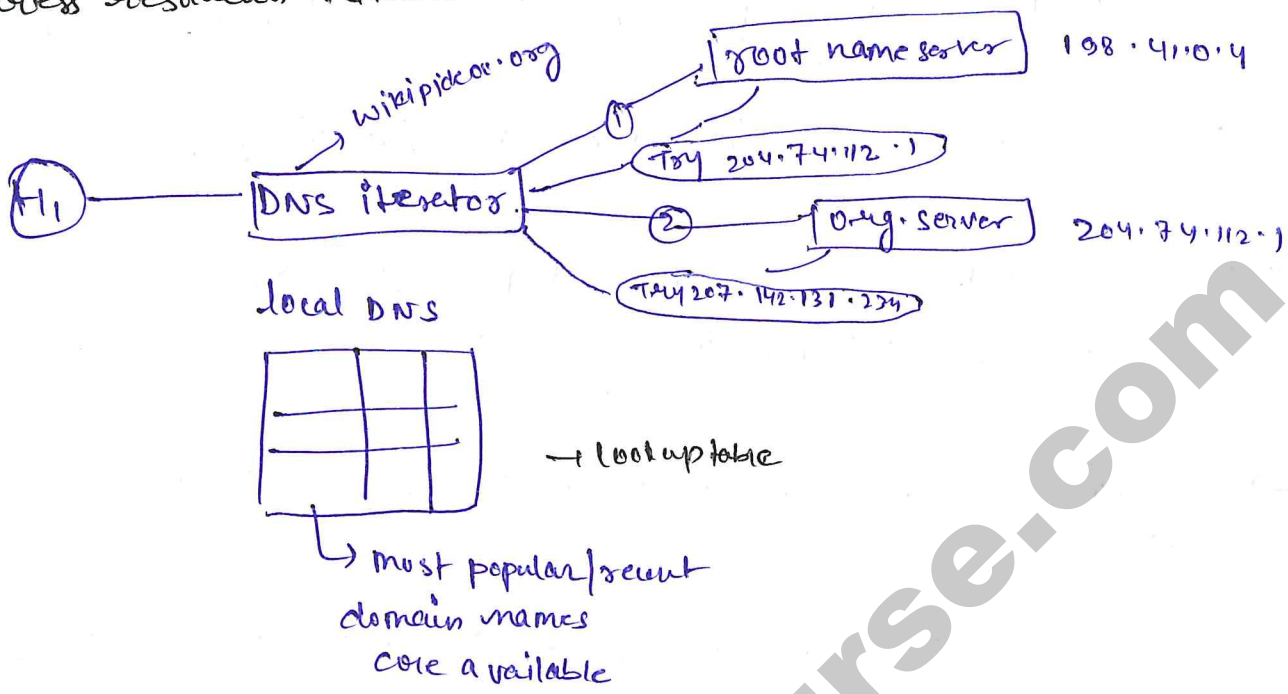


* We have 13 root DNS servers.

* Local DNS is ISP.

→ DNS entry table contains : Domain name, IP, validity

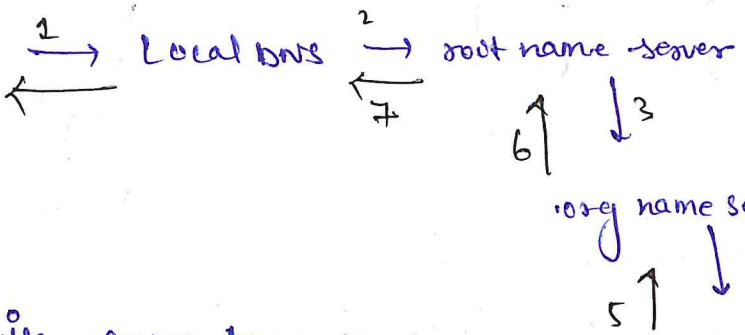
Address resolution : iterative method.



→ Let's take a case we are searching for wikipidea.org Ip address, this is not available in Local DNS, then request goes to root name server, It says I don't know please try 204.74.112.1. Now DNS req goes 204.74.112.1 (org server). It replies the answer.

Here we are using iterative method one by one.

Address resolution : Recursive



→ Here in recursive method the server forwarding the request to another server by itself. Here we are not doing iteratively.

Simple mail Transfer protocol and Post office protocol (POP)

→ not same as gmail (web-based email).

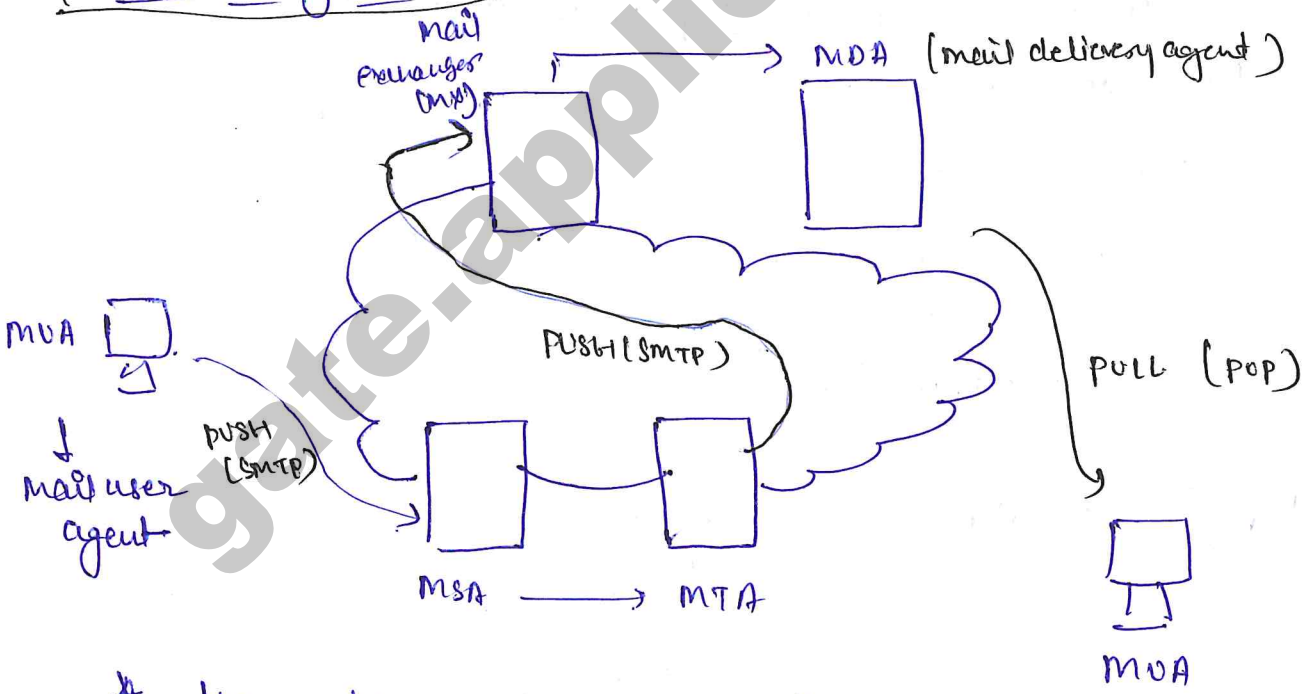
→ Ms outlook uses SMTP and POP for sending and receiving mail.

→ gmail: can use POP and SMTP to pull and push email.

↳ we can connect gmail in ms outlook to pull and push email.

* Inherently Gmail supports SMTP and POP.

Mail processing model:



* Here first we push the mail using SMTP from sender [mail address] to MSA.

* MSA is mail submission agent

→ MSA sends to MTA (mail transfer agent)

→ The mail transfer agent who are in local server of sender institution.

→ The mail transfer agent using DNS [mail exchanger record] gives us the address of email server of receiver cosporation.

→ The MTA pushes to mail exchanger

→ mail exchanger sends to MDA. [mail delivery agent]

→ The job of mail delivery agent saves the mail

→ The receiver MUA comes online, the MDA job is to pull the mail to the MUA.

→ Pop is used to pull the mail (post office protocol).

MIME → It helps us to send non-text data also like an attachment. (like image)

Multimedia
Internet
Mail extension

Commands and data:

→ Port 25 for SMTP between mail servers (TCP or UDP) IANA.

→ In email commands and data on the same connection (in-band).

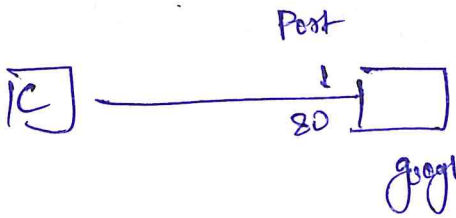
commands → PUSH & POP

data → email

Hypertext Transfer Protocol : (HTTP)

→ HTTP along with some security is HTTPS.

→ needs a reliable transport layer protocol (TCP)

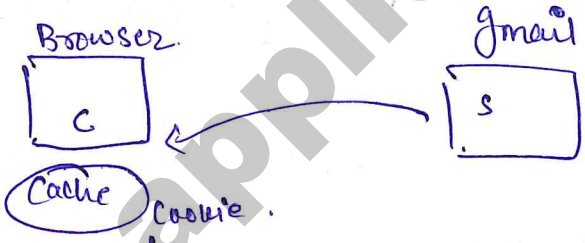


* HTTP uses port = 80

→ It is stateless protocol.

↳ The server doesn't maintain the state information.

→ Client maintain states using cookies. [in HTTP header]

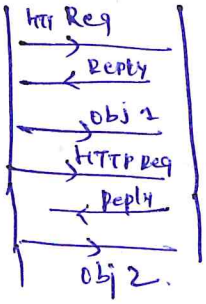


All the information of state is stored in cookie like login information

→ HTTP 1.0 Non-persistent connection

HTTP 1.1. Persistent connection

In Non-persistent → For every object you need to open a connection and close a connection

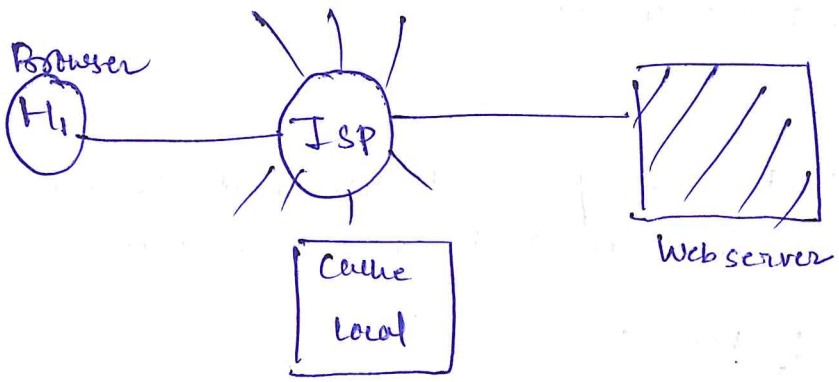


- Persistent - connection → For every object we need only one connection.

HTTP methods:-

GET → Get a webpage (w3c)
HEAD → Same as get but response header only
POST → filling the form [submit]
PUT → Upload an object in google drive.
DELETE → For deleting an object.
CONNECT → For connection of secure HTTPS
OPTIONS → what options are provided by server
TRACE → TRACE servers
PATCH → Partial modifications.

HTTP cache and TRACE



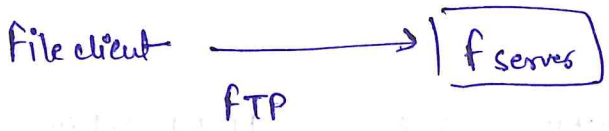
→ HTTP caching helps if some webserver page is locally available

→ In ISP we store some webpage that frequently not changing in cache local.

→ TRACE helps us to understand whether that page is served by ISP or webserver.

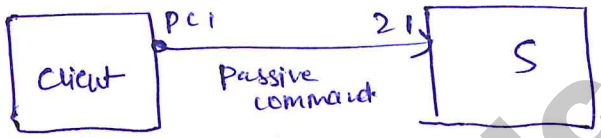
File Transfer Protocol (FTP)

This is used to transfer file from file server to a file client.

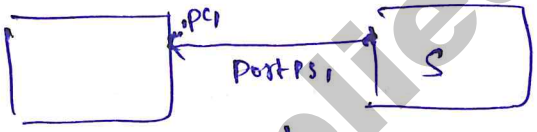


→ Uses reliable TCP

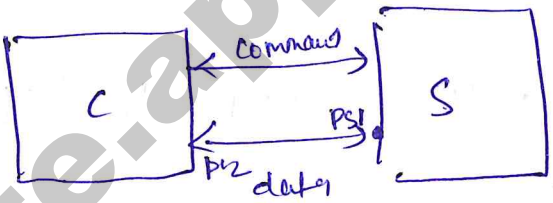
→ FTP is out of band. [different operation of command and data].



Ask for file



→ I will share the using PS1



→ FTP are stateful. They store all information of state in server log.

→ POP also uses stateful mechanism.

Que: Which of the following protocols pairs can be used to send and retrieve emails (in that order)?

(A) IMAP, POP3
(B) SMTP, POP3
(C) SMTP, MIME
(D) IMAP, SMTP

Ans: SEND → SMTP Retrieve → POP3 IMAP → Retrieve
MIME → Extension.

Que: Which of the following protocols is not used to resolve one form of address to another one?

(A) DNS
(B) ARP
(C) DHCP
(D) RARP

Ans: → DHCP are used to get the Ip address but not to resolve one from to another.

Que: Which of the following is/are examples of stateful application layer protocols?

(i) HTTP
(ii) FTP
(iii) TCP
(iv) POP3

Ans: HTTP not
FTP stateful
TCP ✗
POP ✓

(A) (i) and (ii) only
(B) (ii) and (iii) only
(C) (iii) and (iv) only
(D) (iv) only.

Que: Which of the following is not correct about HTTP Cookies?

(A) A cookies is a piece of code that has the potential to compromise the security of an Internet user.

(B) A cookie gains entry to the user work area through an HTTP header.

(C) A cookie has an expiry date and time.

(D) Cookies can be used to track the browsing pattern of a user at a particular site.

Ans:-

B → Cookie shared by httpheader

C → Yes, they have TTL

D → Yes they are used for that.

Que: Identify the correct sequence in which the following packets are transmitted on the network by a host when a browser requests a webpage from a remote server, assuming that the host has just been restarted.

(A) HTTP GET request, DNS Query, TCP SYN
(B) DNS Query, HTTP GET req, TCP SYN
(C) DNS Query, TCP SYN, HTTP GET Request
(D) TCP SYN, DNS Query, HTTP GET request

Ans: First we need → DNS Query to resolve hostname.

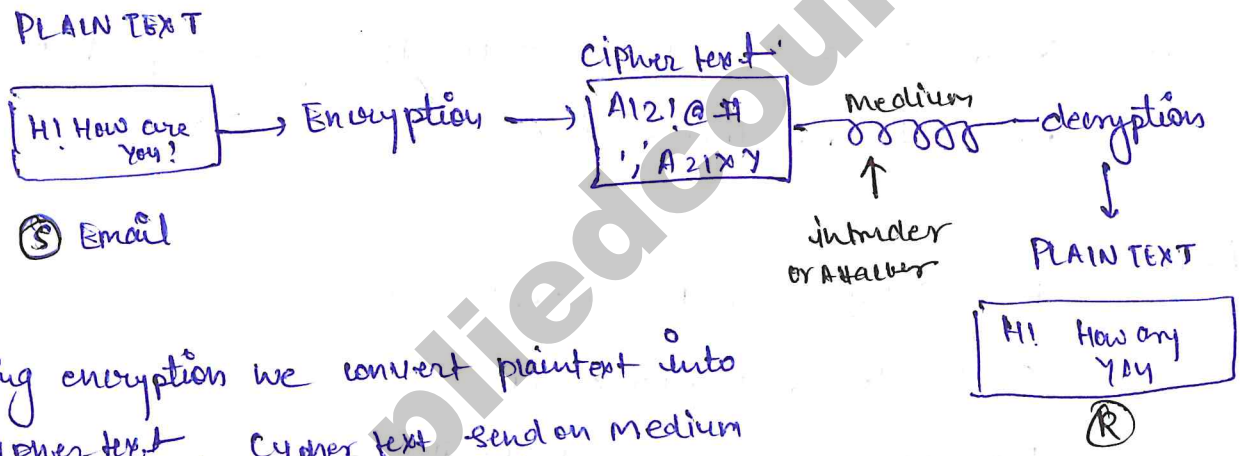
Second → TCP SYN to establish a connection.

GET → To get the page.

Network security and Encryption [Alan Turing during WW-II]

Prime fact | Group theory

Classical encryption:



→ During encryption we convert plaintext into cypher text. Cypher text send on medium

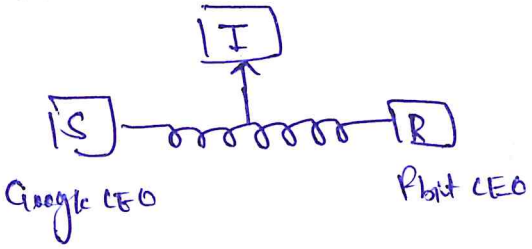
→ If intruder get the message. It won't understand anything.

→ During decryption cypher text converted to plain text which can be used by reciver.

Passive VS Active attacker

→ Attacker can read the message but not modify it is called passive attacker

(2) Traffic pattern Analysis

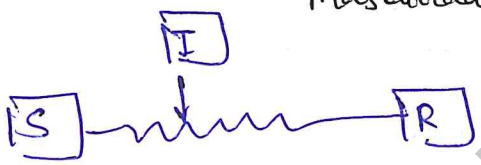


→ Passive Attacker.

* Intruder can track how many E-mails are send from google CEO to Fbit CEO. It will start doing data analysis. Many important information can be done.

Active Attacker :-

(1) Masquerade / pretending

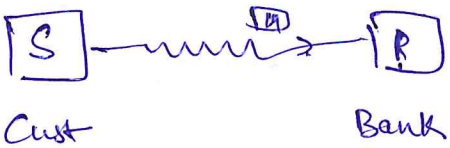


→ If Intruder can be pretend like a sender then receiver will be misguided.

→ In online banking it is dangerous.

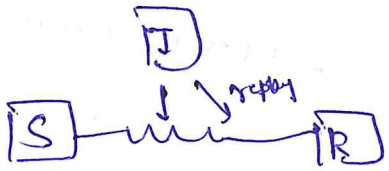
(2) Message modification → Attacker can modify the message of sender.

(3) Repudiation (deny) →



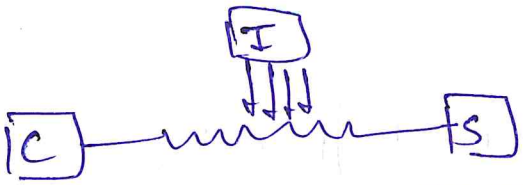
→ If someone deny that this message was not sent by them.

(4) Replay →



Attacker replaying the same packet that is captured from sender to receiver.

(5) Denial of service:



amazon.com

To down the server of amazon, Intruder is pushing so many requests acting as client because of so many requests the server is down.

The authentic user will not able to take the service.

Services expected by a security system:

(1) Authentication → Receiver should know whether packet is coming from authentic sender.

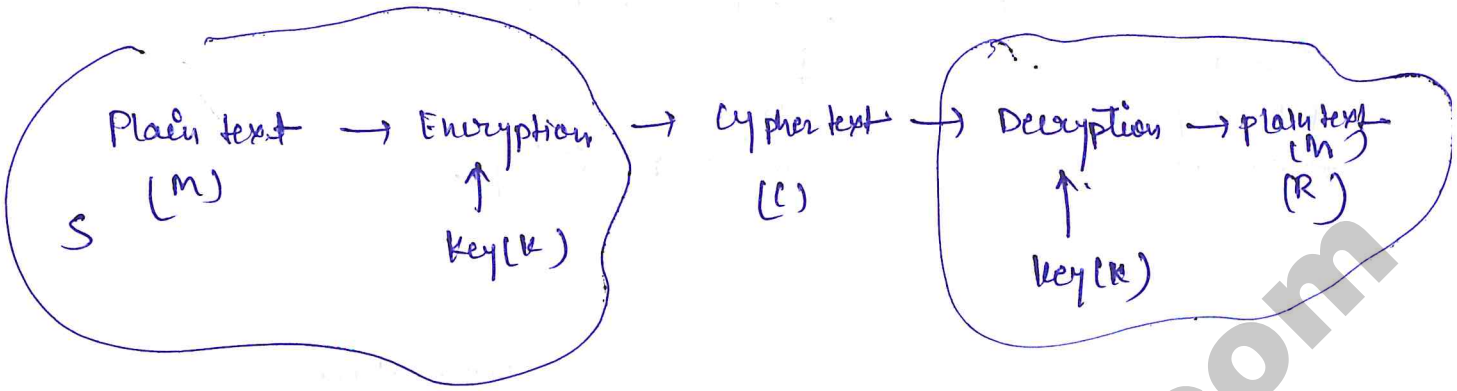
(2) Data integrity → We want data must not tempered between sender and receiver.

(3) Access-control → Gmail login, pass, OTP etc.

(4) Confidentality of data → I don't want anybody to read the information in between.

(5) Non-repudiation → The person can not deny its usage therfore OTP etc are used.

Cryptography: Symmetric



→ We are using same key for encryption and decryption therefore it is called symmetric cryptography.

Eg: DES, AES etc.

Challenge ⇒ How sender and receiver can agree on the same key. (key exchange)

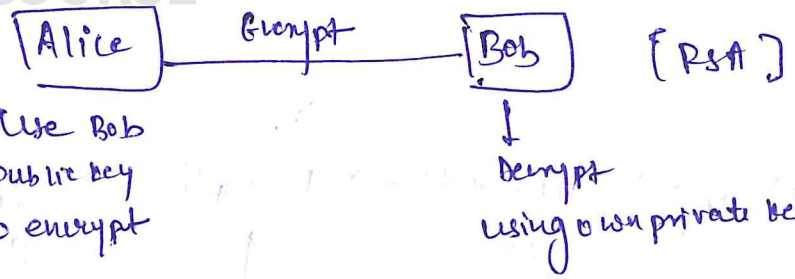
Cryptography: Assymetric key cryptography {RSA algorithm}

→ In Assymetric key cryptography we use two keys: Public key and private key

→ We shared the public key to everyone. Sender and receiver public keys are available and private keys are not known by anyone outside.

Public key

Private key → Not share.



Modulo Arithmetic :—

Number theory

[Mathematics required]

$a \equiv b \pmod{n}$ $\Rightarrow$ a R b under mod n

$\Rightarrow$ a is congruent to b mod n

e.g. $38 \equiv 14 \bmod 12$

$\Downarrow$

This equal to $\boxed{(a-b)}/n \Leftrightarrow ak+n$ (a−b divisible by n)

e.g. $38 \equiv 2 \bmod 12$ ✓ yes

e.g. $-8 \equiv 7 \bmod 5$

$-8-7/5 = -5$ ✓ [yes divisible]

Properties:-

(1) st $a \equiv b \bmod n$ and $c \equiv d \bmod n$.

$\Rightarrow (a+c) \equiv (b+d) \bmod n$

Proof: $a = nK + b$

$c = nl + d$

$(a+c) = n(K+l) + (b+d)$

(2) nd $a_1 \equiv b_1 \bmod n$ and $a_2 \equiv b_2 \bmod n$

$\Rightarrow a_1 \cdot a_2 = b_1 \cdot b_2 \bmod n$

$\Rightarrow a_1^K \equiv b_1^K \bmod n$ $\forall$ non-negative integers K.

(3) rd $p(x)$: polynomial function

$\Rightarrow p(a) \equiv p(b) \bmod n$

* $(a \equiv b \bmod n) \not\Rightarrow (K^a \equiv K^b \bmod n)$

Not always

(4) $\{a \equiv (b+c) \bmod n\} \Rightarrow \{a \equiv (b \bmod n + c \bmod n) \bmod n\}$

$\{a \equiv (b \cdot c) \bmod n\} \rightarrow \{a \equiv (b \bmod n \cdot c \bmod n) \bmod n\}$

Modular multiplicative Inverse :

$ax \equiv 1 \pmod n \iff x \equiv a^{-1} \pmod n$

a and x multiplicative Inverses of each other under modulo m.

Primes and co-primes (relative-primes) [Number theory]

$\{3, 5, 7, 11\}$

$\gcd(x, y) = 1$

$\gcd(5, 17) = 1$

So, they are co-primes.

* If x and y are co-primes then $\exists\ z \in I$ s.t

$xz \equiv 1 \pmod y$

x and z are modulo multiplative Inverse of each other

e.g. $x = 12, y = 17$

$\Rightarrow xz \equiv 1 \bmod y$

$\Rightarrow 12x \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ 10 \end{pmatrix} \equiv 1 \bmod 17 \qquad z = 10$

→ Try for 1, 2, ...

Euler Totient function :-

$\phi(n)$ = Number of +ve integers upto n that are co-primes to n.

Eg: $\phi(9) = 6$

1, 2, ~~3~~, 4, 5, ~~6~~, 7, 8, ~~9~~

Total = 6

* If n and m are co-prime then

$\phi(m)\,\phi(n) = \phi(n \cdot m)$

* If $n = p_1^{k_1} \cdot p_2^{k_2} \cdot p_3^{k_3} \cdots p_r^{k_r}$

then

$$\phi(n) = n\left[1-\frac{1}{p_1}\right]\left[1-\frac{1}{p_2}\right]\cdots\left[1-\frac{1}{p_r}\right]$$

↑ Euler product formula

Eg: $n = 36$

$= 2^2 \cdot 3^2$

$\phi(n) = 36\left(1-\frac{1}{2}\right)\left(1-\frac{1}{3}\right) = 36 \cdot \frac{1}{2} \cdot \frac{2}{3} = 12$

Euler's Theorem : [co-primes]

if $n \in I^+$ & a, n are co-prime

then $a^{\phi(n)} \equiv 1 \pmod n$

e.g: $n = 165$

$\phi(n) = \phi(165) = \phi(15)\,\phi(11)$

$= 15\left(1-\frac{1}{3}\right)\left(1-\frac{1}{5}\right) \cdot 11\left(1-\frac{1}{11}\right)$

$\phi(n) = 8 \cdot 10 = 80$

$8^{80} \equiv 1 \bmod 165$

Fermat's little theorem: Simple extension

→ If $n \in I^+$ & a, n are co-primes

then $a^{\phi(n)} + 1 \equiv a \pmod{n}$

$$\begin{cases} a^{\phi(n)} \equiv 1 \bmod n & \rightarrow \text{Euler} \checkmark \\ a \equiv a \pmod{n} & \rightarrow \text{Trivial} \checkmark \end{cases}$$

Another Extension:

If $n \in I^+$ & a, n are co-prime & $b \equiv 1 \pmod{\phi(n)}$

then $a^b \equiv a \pmod{n}$

Proof: $b \equiv 1 \pmod{\phi(n)}$

$b = k \cdot \phi(n) + 1$

$a^b = a^{k\phi(n)+1} = (a^{\phi(n)})^k \cdot a$

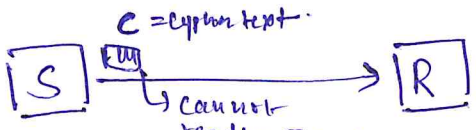
$a^{\phi(n)^k} \equiv 1^k \equiv 1 \bmod$

$a \equiv a \pmod{n}$

↳ Trivial

RSA (Rivest, Shamir, Adelman)

[In RSA ~~sender encrypt~~ with receiver public key and receiver decrypt with own private key]



K_u^R = R's public key

K_{pr}^R = R's private key

$K_u^R = (e, n)$

P = plain text

$C = P^e \bmod n$

$0 \le P < n$

$K_{pr}^R = (d, n)$

$P = C^d \bmod n$

$\Rightarrow P = (P^e \bmod n)^d = P^{ed} \bmod n$

$P < n$ → means [1234 | 1234] → ~~If~~ If we have bigger plain text we divide in blocks for sending.

* R's public key known to everyone = K_u^R
* R's private key only known to R.
*

~~Operation at R =~~

* We want n, e, d to satisfy $P^{ed} \bmod n = P$.
  - At receiver side to again get plain text
* We want to pick e, d s.t

$$P = P^{ed} \bmod n$$

$$ed = 1 \bmod \phi(n)$$ → Theorem.

① Given (e, n) computing d must be very ~~too~~ hard.

(e, n) → K_u^R

⇓

Computing $\phi(n)$ must be hard

n, e, d in RSA

① P_1, P_2 be large prime numbers.

$n = P_1 \cdot P_2$. ⇒ n is very large [256 - bit numbers]

$\phi(n) = \phi(P_1) \cdot \phi(P_2) = (P_1 - 1)(P_2 - 1)$

② Pick $1 \le e \le \phi(n)$ s.t $\gcd(e, \phi(n)) = 1$ (e, $\phi(n)$ are co-prime)

⇒ $\exists d \in I^+$ s.t $e \cdot d \equiv 1 \bmod \phi(n)$

(Modulo multi Inverse)

*

Eg. $P_1 = 53$ $P_2 = 61$

$n = 53 \times 61 = 3233$

$\phi(n) = \phi(53)\,\phi(61) = 52 \times 60 = 3120$

$e = 17$, pick some e

$\gcd(17, 3120) = 1$.

$e \cdot d \equiv 1 \bmod \phi(n)$

17 ↓ 3120 ↓

Then, $d = 2753$

$K_u R = (17, 3233)$ (e, n)

$K_R R = (2753, 3233)$ (d, n)

Residue:

$a \equiv b \bmod n \iff b$ is residue (remainder) of $a \bmod n$

e.g: $38 \equiv 14 \bmod 12 \iff 14$ is residue of $38 \bmod 12$

Residue-class / Congruence class

Defn: $\bar{a}_n = \{\ldots, a-2n, a-n, a, a+n, a+2n, a+3n \ldots\}$

↳ Set of all integers that are $\equiv a \bmod n$

e.g. The residue class of $x^2 \bmod 6$ are

$0^2 \bmod 6 = 0$

$1^2 \bmod 6 = 1$

$2^2 \bmod 6 = 4$

$3^2 \bmod 6 = 3$

$4^2 \bmod 6 = 4$

$5^2 \bmod 6 = 1$

The residue here → $\bar{0}_6, \bar{1}_6, \bar{3}_6, \bar{4}_6$

↓

You will get only these residue pattern

Primitive root of prime :-

P : prime number

g is set to primitive root of n if $g^0, g^1, g^2 \ldots$ (exponent of g)

includes all congruence-classes of P $\{\bar{1}_p, \bar{2}_p, \bar{3}_p, \bar{4}_p, \bar{5}_p \ldots\}$

e.g. P=7 g=3

$3^0 \bmod 7 = 1$
$3^1 \bmod 7 = 3$
$3^2 \bmod 7 = 2$
$3^3 \bmod 7 = 6$

$3^4 \bmod 7 = 4$
$3^5 \bmod 7 = 5$
$3^6 \bmod 7 = 1$
⋮

3 is the primitive root of 7

→ 7 has multiple primitive roots not that one.

e.g. P=7 g=5

$5^0 \bmod 7 = 1$
$5^1 \bmod 7 = 5$
$5^2 \bmod 7 = 4$
$5^3 \bmod 7 = 6$

$5^4 \bmod 7 = 2$
$5^5 \bmod 7 = 3$
$5^6 \bmod 7 = 1$
$5^7 \bmod 7 = 5$ } repeat

for P=7 3, 5 primitive roots.

Fermat's little theorem :

$$g^{p-1} \equiv \text{mod } p \quad \text{as} \quad \phi(p) = p-1.$$

* Powers formed by g can't have cycle longer than $p-1$ if cycle-length $= p-1$ then g is a primitive root.

Alternative defn with cycle length

Multiplicative order :

Coprimes a, n, the multiplicative order of a mod n is the smallest $k \in I^+$ s.t

$$a^k \equiv 1 \text{ mod } n.$$

Generalised defn : { a is primitive root of n iff Multiplicative order of $a = \phi(n)$ }

Let n is prime $\phi(n) = n-1$

* The concept of multiplicative order and cycle length both are related.

n	g(n)	
2	1	
3	2	
4	3	
5	2, 3	$\rightarrow 5^1$
6	5	$\rightarrow 2 \cdot 3^1$
7	3, 5	$\rightarrow 7^1$
9	2, 5	$\rightarrow 3^2$
10	3, 7	$\rightarrow 2 \cdot 5^1$
11	2, 6, 7, 8	$\rightarrow 11^1$
13	2, 6, 7, 11	$\rightarrow 13^1$

Except 1, 2 and 4 all the other integers with primitive roots are of the form p^k or $2p^k$ where p is an odd prime.

Discrete logarithm problem :-

Given a, b, p find x s.t $a^x \equiv b \bmod p$

↳ large prime (256 bit.)

* given a, x and p computing b is easy. [Try for all b]
* given a, b and p computing x is very hard.

↳ computationally hard

One-way funtions:

$f(x) = b$ → easy

→ easy to compute

a, p

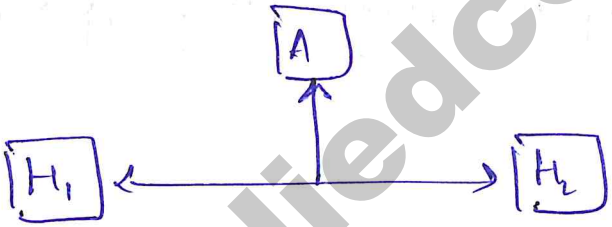
$g(b) = x$ → v. hard to compute

a, p

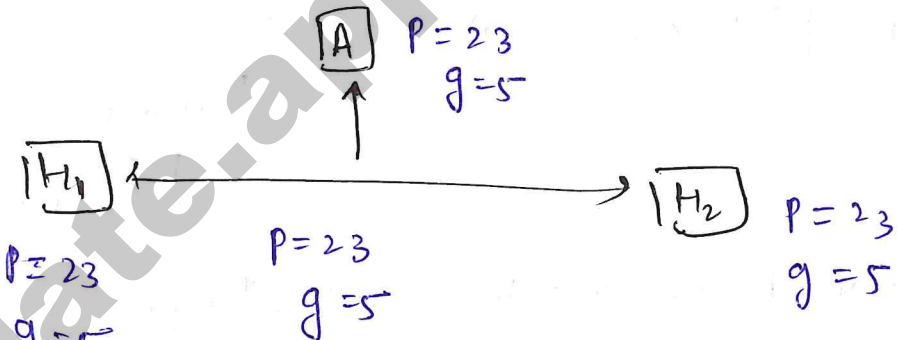
Diffie-Hellman key exchange :- (SSL/TL)

- Symmetric Encryption : key exchange.

K : Key



Step-1 (1)



{ g is a primitive root of prime p }

P and g known to everyone

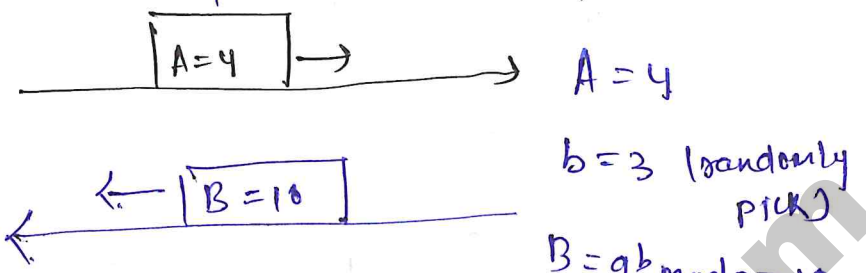
(2) ~~Host sends~~ a = 4 (randomly)

Attacker also knows this

$A = g^a \bmod p = 4$

Computes this

$B = 10$



* a is the secret known to H1
* b is the secret known to H2

Host A knows

A = 4
a = 4
B = 10
p = 23
g = 5

$K = B^a \bmod p$

↓

Attacker can't compute this

Attacker knows

p = 23
g = 5
A = 4
B = 10

↓

Attack can't figure out a, b

$g^{(a)} = A \bmod p$ → finding exp is hard

Host B knows

p = 23
g = 5
b = 3
A = 4
B = 10

$K = A^b \bmod p$

↓

Attacker can't compute

$B^a \bmod p = A^b \bmod p$ → equal

Proof: $B^a \bmod p = (g^b \bmod p)^a \bmod p = (g^b)^a \bmod p$

$= (g^a)^b \bmod p$

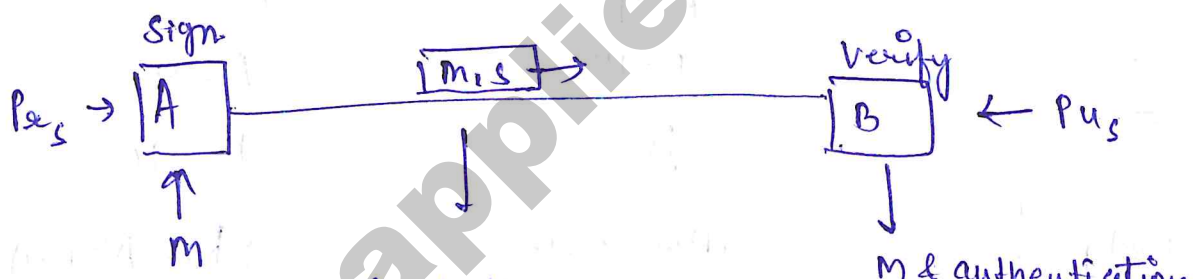
$= (g^a \bmod p)^b \bmod p$

$= A^b \bmod p$.

eq used.

Digital signature (authenticity - verification)

- Using Public key Cryptography.
- Proof of document authenticity we use digital signature.



A is the one who signed the signature. that B should know.

* Sender Encrypt with its own private key
* Public key of sender known to everyone.
* Private key of sender is only known to sender.

→ We encrypt using private key of sender

$$S = \text{encrypt}\ (M, Pr_S)$$

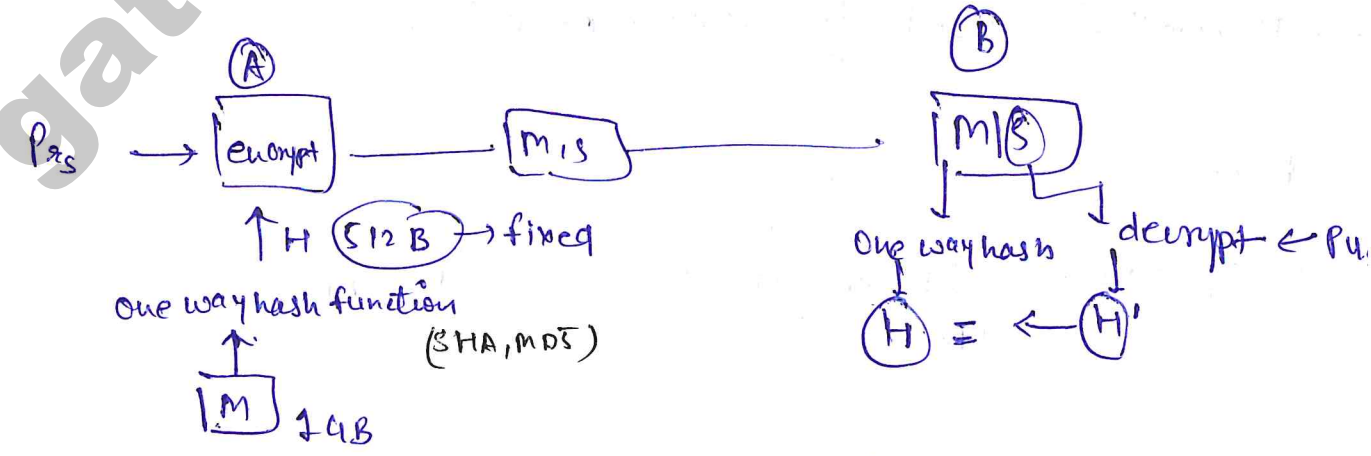
→ B has public key of sender, After decrypting with public key of A sender got to know this message is send by A.

$$M = M' = \text{decrypt}\ (S, Pu_S)$$

* Problem → Here problem is we are sending bigger amount of data. Encrypting it takes much time.

To Solve this issue:

→ We will compute the one way hash function using message. It will generate 512 B hash value.

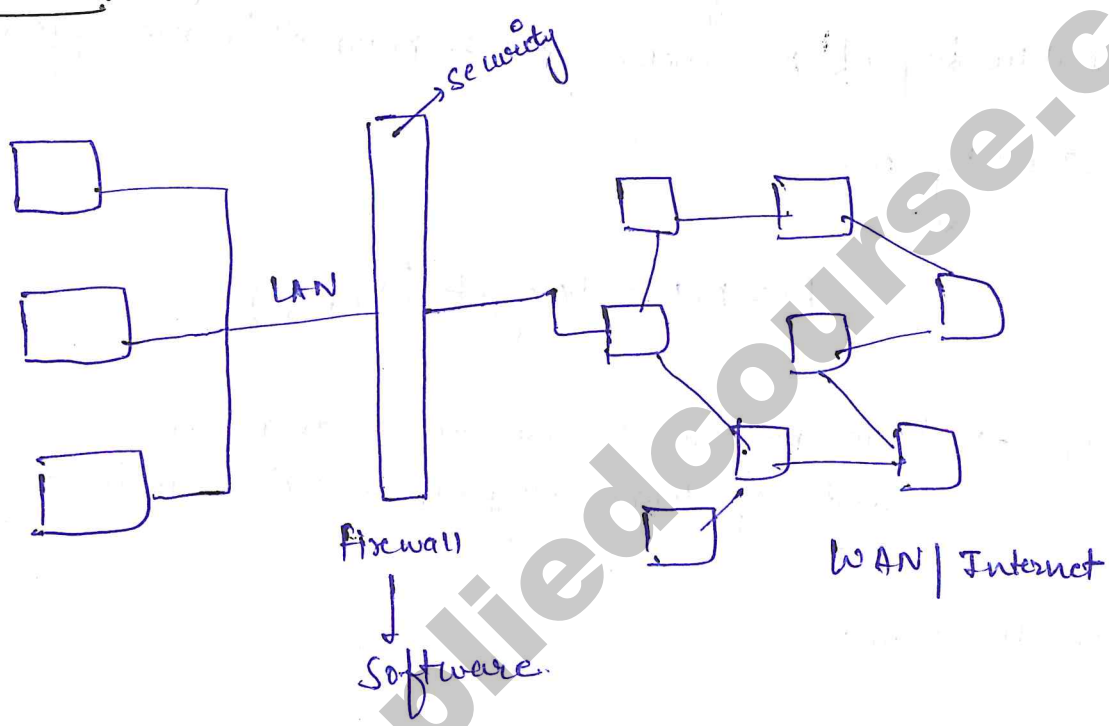


* We encrypt hash value, so it will become small size.

* We decrypt using public key
* We compute hash on m

If H = H' then it is verified.

Firewalls:



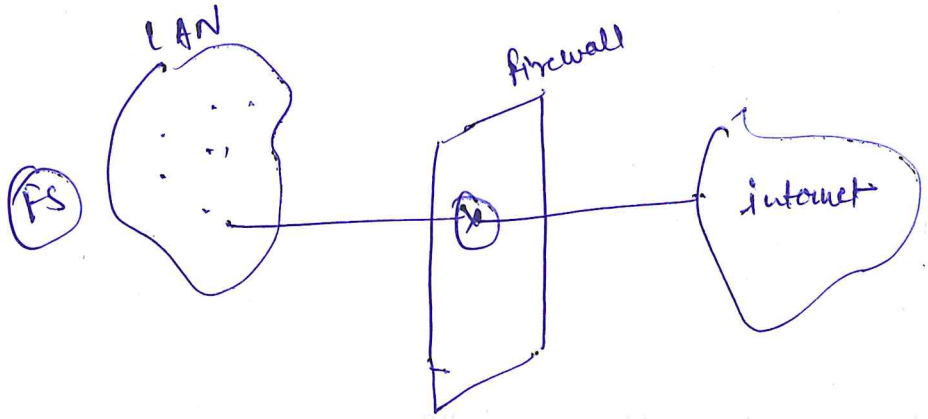
→ To block unnecessary packets to go from LAN to outside and to block unnecessary packets from outside network to LAN.

→ Firewall is used for security mechanism

Packet filter firewall:

→ operates @ IP and TCP layer ← NOTAL
(N/w) TCP

→ Filter based on IP and port address.



e.g: Block port 21 ←

↓

→ No computer from outside can access file system of LAN. We are blocking port 21

→ Block 121.62.1.x

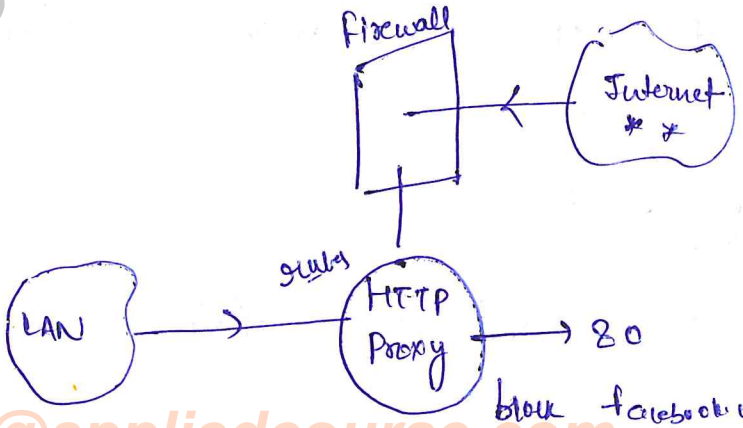
No packet from LAN can go to 121.62.1.x

↓

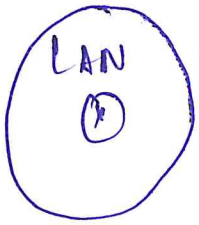
Hacked sites / ~~fb~~ social network fb etc

Application Layer | proxy firewalls

→ It works on Application layer.



→ Any flow which is HTTP packet will first receive by HTTP proxy.



→ HTTP header checked by HTTP proxy. The blocked websites we can not access further.

External proxy servers:

The websites are blocked by local proxy server we can use external proxy server to access the website that are blocked by local proxy server.

Solved problems:

Que: In RSA cryptosystem, the value of the public modulus parameter n is 3007. If it is also known that $\phi(n) = 2880$, where $\phi()$ denotes Eulers totient function, then the prime factors of n which is greater than 50 is ____?

Ans: $n = 3007 = p \cdot q$ —①

$\phi(n) = 2880 = (p-1)(q-1)$ —②

$$2880 = pq - p - q + 1$$

$$2880 = 3007 - p - q + 1$$

$$p + q = 128 \quad \text{—(3)}$$

$$p = \frac{3007}{q}.$$

$$3007 + q^2 = 128q$$

$$q = 31, (97) \rightarrow \text{greater than } 50$$

Que A sender S sends a message m to the receiver R, which is digitally signed by S with its private key. In this scenario, one or more of the following security violations can take place.

Using Birthday Attack, some fraudulent message can be generated which has the same hash value (due to hash collisions) and digital signature as the original message.

(I) S can launch birthday attack to replace m with the fraudulent message.

(II) A third party attacker can launch a birthday attack to replace m with a fraudulent message.

(IV) R can launch a birthday attack to replace m with a fraudulent message.

(A) I and II only

(B) I only

(C) II only

(D) (II) and (III) only.

Ans: S —— m, sign. ——> R

* In digital signatures we pass message m and by hash function we generate hash value.

m —hash→ (H)

m' —hash→ (H)

→ (1) Yes can send message m' by using same hash value.

→ (2) Not possible

→ 3 Not possible

Que: In RSA cryptosystem a particular A uses two prime numbers $p=13$ and $q=17$ to generate her public and private keys. If a public key of A is 35. Then the private key of A is ——.

Ans:

$p \cdot q = n = 13 \times 17$

$\phi(n) = 12 \times 16 = 192$

e, n, d

$e = 35$: Public

$d = Pr = ?$

$\gcd(35, 192) = 1 \quad 1 < e < \phi(n)$

$1 < d < \phi(n)$

$\Rightarrow ed \bmod \phi(n) = 1$

$\Rightarrow 35 \cdot d \bmod 192 = 1$

$d = 11$

Que. Consider that B wants to send a message m that is digitally signed to A. Let the pair of private and public keys for A and B be denoted K_x^- and K_x^+ for $x = A, B$, respectively. Let $K_x(M)$ represent the operation of encrypting m with a key K_x and $H(m)$ represent the message digest. Which one of the following indicates the CORRECT way of sending the message m along with the digital signature to A?

A. $\{m, k_B^{+}(H(m))\}$

B. $\{m, k_B^{-}(H(m))\}$

C. $\{m, k_A^{-}(H(m))\}$

D. $\{m, k_A^{+}(m)\}$

Ans: $-$ → private
$+$ → public

$m \xrightarrow{h} (H)$ → message digest

B wants to send a message.

B ⟶ A

$[m, k_B^{-}H(m)]$ → Ans

We use B private key

Que: Anarkali digitally signs a message and sends it to salim. verification of the signature by salim requires.

(A) Anarkali's public key
(B) Salim's public key
(C) Salim's private key
(D) Anarkali's private key

→ In authentication reciever decrypts with sender's public key.

A ⟶ S

Pr_A ↓ public key of Anarkali