GATE CSE NOTES

by

Joyoshish Saha

TCP, UDP

* Transport layer protocols :

1. Transmission control protocol (TCP)
2. User datagram protocol (UDP)

* TCP

→ Characteristics.

1. Reliable protocol : Guarantees the delivery of data packets to its correct destination. After receiving the packet, receiver sends an acknowledgement to the sender. TCP employs retransmission to compensate for packet loss.

2. Congestion & flow control : TCP handles congestion & flow control by controlling the window size. TCP reacts to congestion by reducing the sender window size.

3. Connection oriented protocol : TCP establishes an end to end connection between the source & destination. The connection is established before exchanging the data. The connection is maintained until the application programs at each end finishes exchanging the data.

4. In-order delivery : Sequence numbers are used to coordinate which data has been transmitted & received.

5. Full duplex : TCP connections are full duplex.

6. Collaboration with IP : A TCP connection is uniquely identifi by combination of port numbers & IP addresses of sender & receiver. Port numbers are contained in the TCP header & IP addresses are contained in the IP header. TCP segments are encapsulated into an IP datagram. So, TCP header immediately follows the IP header during transmission.

7. Acknowledgement type : TCP can use both selective ↑ cumulative acknowledgements. TCP uses a combination of selective repeat & Go back N protocols. In TCP, $W_S = W_R$. Out of order packets are accepted by the receiver. When receiver receives an out of order packet, it accepts that packet but sends an acknowledgement for the expected packet. Receiver may choose to send independent acknowledgements or cumulative acknowledgements.

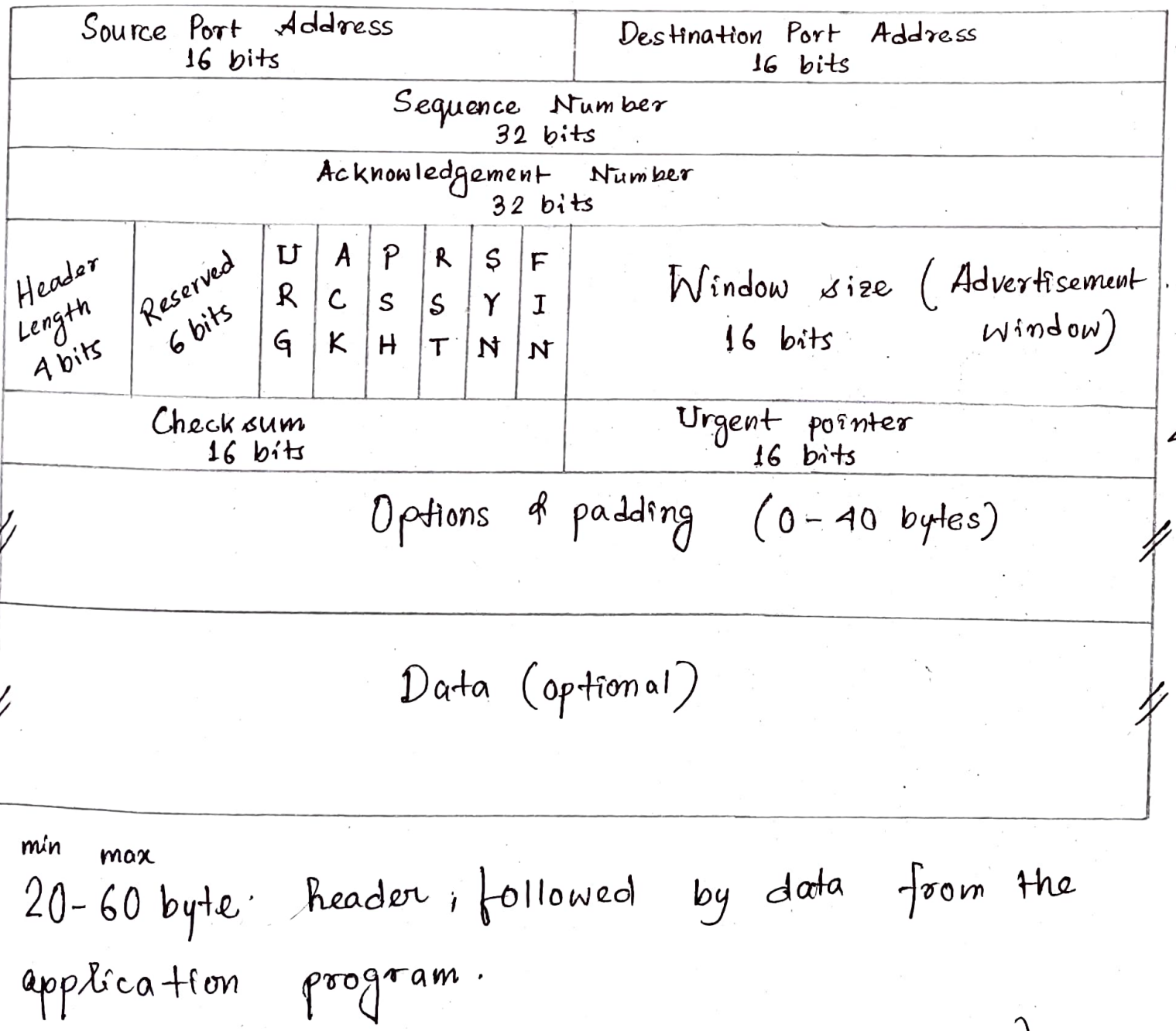
✓ To sum up, TCP is a combination of 75% SR & 25% GBN protocol.

✓ 8. Byte stream protocol. : Application layer sends data to the transport layer without any limitation. TCP divides the data into chunks where each chunk is a collection of bytes. Then, it creates a TCP segment by adding TCP header to the data chunk.

9. Error checking & recovery mechanism :

Using checksum, acknowledgement, retransmission.

→ TCP segment header.



min max
20 - 60 byte header ; followed by data from the application program.

(Unit of transmission in TCP - segment).

1. Source port : Identifies the port of the sending application.

2. Destination port : Identifies the port of the receiving application.

TCP connection is uniquely identified using combination of port numbers & IP addresses.

3. Sequence number : Contains the sequence no. of first data byte. *

assigns unique seq. no. to each byte of data in TCP segment.

4. Acknowledgement no. : Sequence no. of the data byte that receiver expe to receive next from the sender. It is always sequence number of the last received data byte incremented by 1.

5. Header length : Contains the length of TCP header. It helps in knowing from where the actual data begins.

Min-Max HLen.

Length of TCP header always lies in the range [20B, 60B].

Initial 5 rows of the TCP header are alwa used. So, min length of TCP header 5×4 = 20B

Size of options field can go up to 40 B. So, max length (20+40) = 60 B.

* In TCP, we send chunk of data bytes. So, only the seq. no 1st data byte is put in seq. no. field

Scaling factor : As HLen is a 4 bit field, range is [0, 15]. But, range of HLen is [20, 60]. So, to represent the HLen, we use a scaling factor of 4.

✓ Header length = Header length field value × 4 Bytes.

eg. HLen field 0101 ≡ $(5)_{10}$

Header length = 5 × 4 = 20. bytes.

. Range of HLen field value [5, 15].

6. Reserved bits. : Not used.

7. URG bit : Used to treat certain data on an urgent bassis. When set to 1, it indicates the receiver that certain amount of data within the current segment is urgent. Urgent data is pointed out by evaluating the urgent pointer field. The urgent has to be prioritized. Receiver forwards urgent data to the receiving application on a separate channel.

8. ACK bit : Indicates whether acknowledgement no. field is valid or not. When set, ACK no. is valid. For all TCP segments except request segment, ACK bit is set to 1.

✓ Request segment is sent for connection establishment during Three-way Handshake.

9. PSH bit : Used to push the entire buffer immediately to the receiving applicatio

When set, all the segments in the buffer are imm-ediately pushed to the receiving application. No wait is done for filling the entire buffer. This makes the entire buffer to free up immediately.

✓ Unlike URG bit, PSH bit does not prioritize the data. Same order is maintained while pushing, as of the arrival of segments.

It's not a good practice to set PSH to 1, as it disrupts the wooking of receiver's CPU & forces it to take an action immediately.

10. RST bit : Used to reset the TCP connection. When set, it indicates the receiver to terminate the connection immediately. It causes both the sides to release the connection & all its resources abnormally. It may result in the loss of data that is in transit. This is used only when, there are unrecoverable errors. There is no chance of terminating the TCP connection normally then

11. SYN bit : Used to synchronise the sequence numbers. When set, it indicates the ✓ receiver that the sequence no. contained in the TCP header is the initial sequence number. ✓ Request segment sent for connection establishment during Three way handshake contains SYN bit set to 1.

12. FIN bit : Indicates the end of data transmission to finish a TCP connection.

13. Window size : Contains the size of the receiving window of the receiver or sender. It advertises how much data (in bytes) the sender can send without acknowledgement (W_R). Thus, window size is used for flow control.

No. of bytes that a receiver is willing to accept. rwnd is determined by the receiver. The sender must obey the dictation of the receiver in this case.

The window size changes dynamically during data transmission. It usually increases during TCP transmission up to a point where congestion is detected. After congestion is detected, the window size is reduced to avoid having dropped packets.

congestion control

14. Checksum : Verifies the integrity of data in the TCP payload. Sender adds CRC checksum to the checksum field before sending the data.

15. Urgent pointer : Indicates how much data in the current segment counting from the first data byte is urgent. Urgent pointer added to the sequence number indicates the end of urgent data byte. This field is considered valid & evaluated only if URG is set.

urgent bytes = urgent pointer + 1.

✓ End of urgent byte =

Sequence no. of the first byte in the segment + Urgent pointer.

16. Optional fields. :

Used for following purposes -

✓✓ a) Time stamp - When wrap around time is less than life time of a segment, multiple segments having the same sequen number may appear at the receiver side. This makes it difficult for the receiver to identif the correct segment. If time stamp is used, it marks the age of TCP segments. Based on the time stamp, receiver can identify the correct segment.

✓ b) Window size extension - May be used to represent a window size greater than 16 bits.

c) Parameter negotiation - For example, during connection establishment, both sender & receiver have to specify their max. segment size. To specify maximum segment size, there is no special field. So, they specify their MSS using this field & negotiates. (MSS)

d) Padding - Addition of dummy data to fill up unused space in the transmission unit & make it ~~conforme~~ conform to the standard size is padding.

eg. HLen not multiple of 4. Extra zeros are padded in the options field. By doing so, HLen becomes multiple of 4.

HLen = 30 B. 2B dummy data

⇓

HLen 32 B.

HLen field $\frac{32}{4} = 8$

- In worst case, 3 bytes of dummy data might have to be padded to make the HLen a multiple of 4.

→ Well known ports used by TCP :

Port	Protocol	Description.
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that's received
11	Users	Active users.
13	Daytime	Returns the date & the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters.
20	FTP, Data	File transfer protocol (data conn^n)
21	FTP, control	" (control conn^n)
23	TELNET	Tenninal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain name server
67	BOOTP	Bootstrap protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote procedure call

MSS - Window size. (buffer capacity of sender/receiver)

↓

S/R tells their MSS so when datagram comes to them there is no need of segmentation. MSS - options field

---

Assume, sender's window size = W_S Bytes

MSS of sender = 1460 B

MSS of receiver = ~~1460~~ 500 B

} We take the min of them to avoid segmentation

⇓

MSS = 500 B

Receiver's window size = W_R Bytes

∴ Receiver can send maximum of $\frac{W_S}{500}$ packets/segments

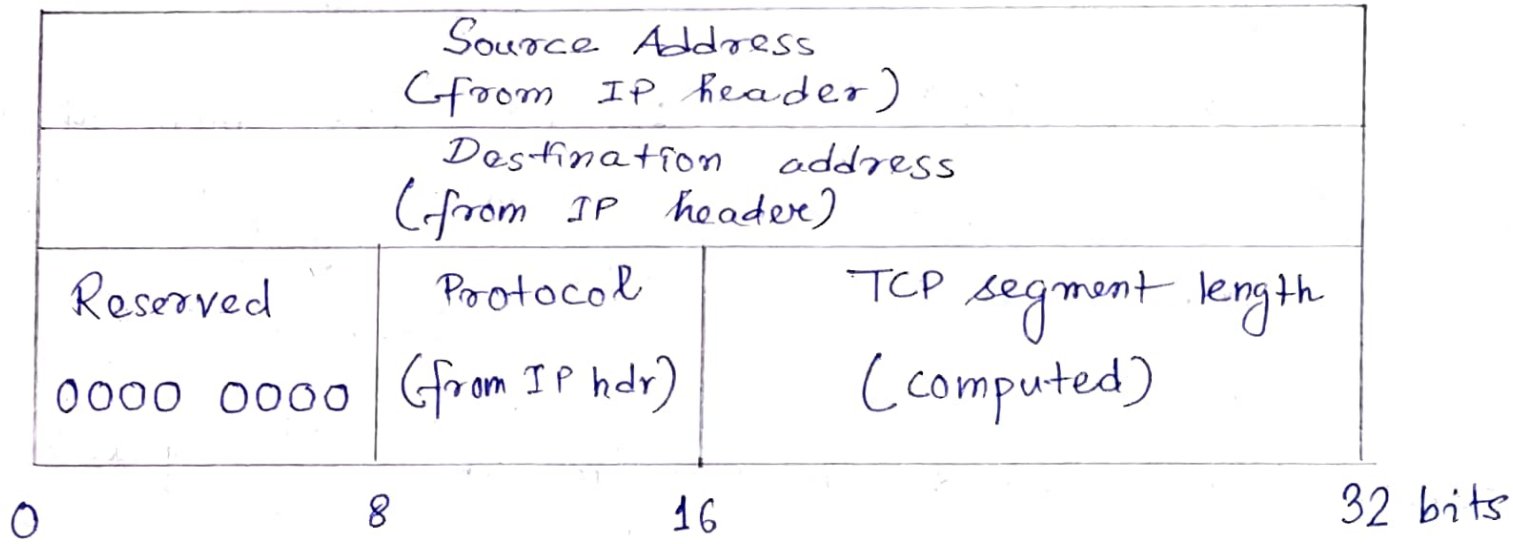
Sender can send maximum of $\frac{W_R}{500}$ segments

→ Receiver's sending window size

→ Sender's sending window size

Checksum (CRC) : 16b, TCP checksum is computed on TCP header, TCP data and pseudoheader.

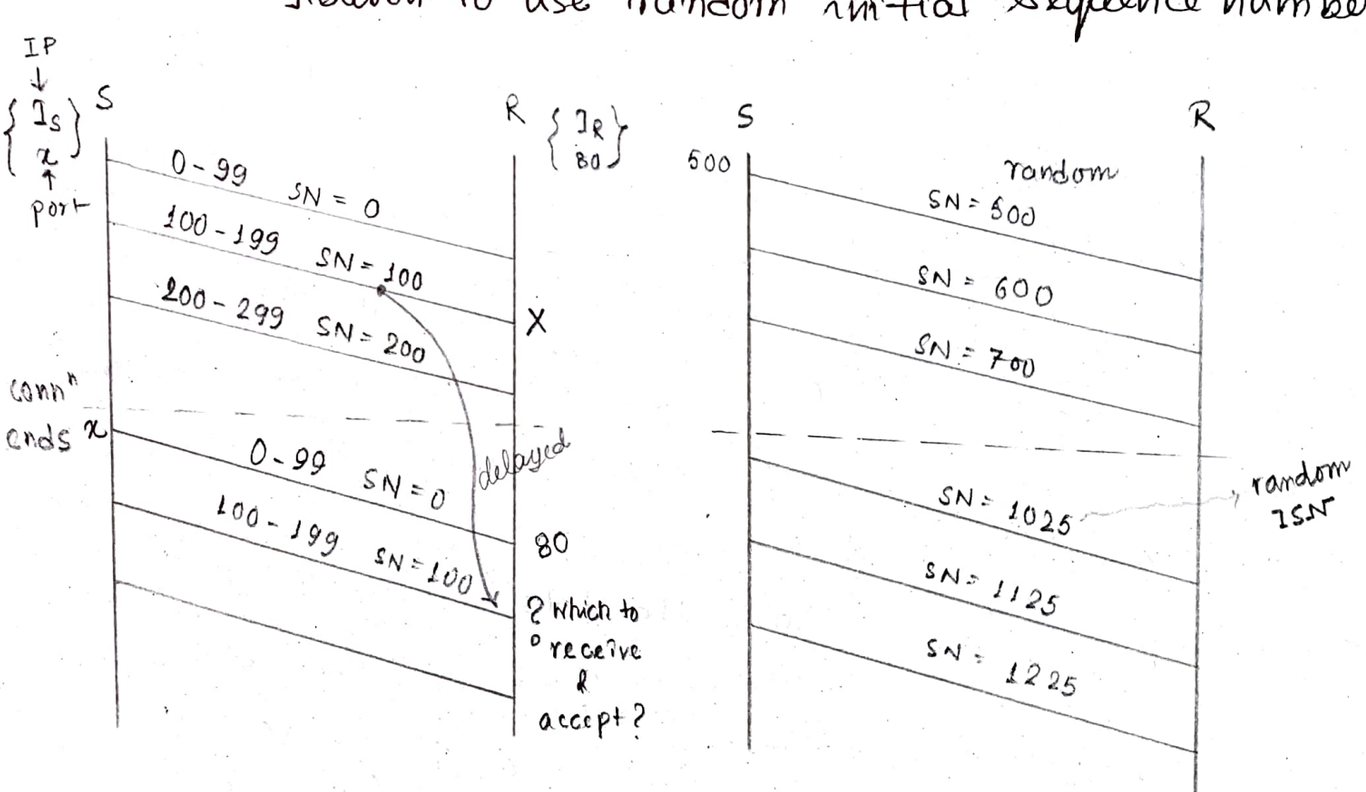
Pseudoheader — (used for double checking)



- **Sequence number, Acknowledgement Number. :**

To ensure connectivity, each byte to be transmitted is numbered. During connection establishment each party uses a random number generator to create initial sequence number (ISN) which is usually different in each direction. TCP sequence number field is 32 bits. So, it has 4 Giga sequence numbers and as TCP is a byte stream protocol, we will be able to send only 4 GB of data with unique sequence number.

Reason to use random initial sequence number.



- **Purpose of sequence numbers:** i) helps to identify each data byte uniquely. ii) helps in the segmentation of data into TCP segments & reassemble them later. iii) Keeping track of how much data has been transferred & received. iv) put the data back into the correct order if it is received in the wrong order. v) Request data when it has been lost in transit.

- Maximum number of possible sequence numbers = 2^{32} = 4G
Concept of wrap around allows to send unlimited data using TCP.

- Wrap around.

After all the 2^{32} sequence numbers are used up & more data is to be sent, the sequence numbers can be wrapped & used again from starting.

In general, if ISN chosen is X, sequence numbers are used from X to $2^{32}-1$ and then from 0 to X-1. Then, seq. no.s are wrapped.

- Wrap around time: Time taken to use up all the 2^{32} sequence numbers. It depends on the bandwidth of the N/W i.e. the rate at which the bytes go out. More the bandwidth, lesser the WAT & vice versa.

$$WAT \propto \frac{1}{BW}$$

$$WAT = \frac{\text{No. of sequence numbers available.}}{\text{Bandwidth in Bytes/sec.}}$$

- Life time of TCP segment: Life time of a TCP segment is 180 seconds (3 m). After sending a TCP segment, it might reach the receiver taking 3 min in worst case.

WAT must be greater than the life time to avoid same sequence numbers for bytes.

When WAT $\geq$ LT, by the time the seq no.s wrap around, there's no probability of existing any segment having the same sequence no.

- To reduce the WAT to the LT, there must exist as many seq. no.s as there are no. of data bytes sent in time equal to the LT of segment.

After previous segment reaches receiver, wrap around to avoid confusion of same SNs

~ No. of seq. no.s needed if WAT is t and BW B bytes/sec = Bt.

~ # bits reqd. in the seq. no. field so that wrap around time becomes equal to lifetime of TCP segment =

$$\log_2 (\text{LT of TCP segment} \times \text{BW}).$$

Extra bits needed are accomodated in the Options field of the TCP header.

Q. BW = 1 MB/s. WAT = ?

→ $$\text{WAT} = \frac{\#\text{ seq. no. available.}}{\text{BW}} = \frac{2^{32}}{10^6 \text{ Bps}} = 1.19 \text{ hr}$$

Q. BW = 1 GBps, how many extra bits have to be appended in the options field so that WAT = LT ?

→ $$\log_2 (\underbrace{180 \times 2^{30}}_{\text{bytes sent}})$$

data ▶ power of 2
BW ▶ power of 10

$\simeq 38$

ans $(38 - 32) = 6$ bits.

BW 1 GBps
bytes transferred in 1s = 1 GB
" " in 180 s = 180 GB
$= 180 \times 2^{30}$ B
∴ # of seq. no reqd = 180×2^{30}

WAT = LT

$\frac{\#\text{seq}}{\text{BW}} = \text{LT}$

$\#\text{seq} = \text{BW} \times \text{LT}$

Q. In a N/W that has a maximum TPDU (transport layer protocol data unit / segment) size of 128 bytes, a max. TPDU life time of 30 sec & 8 bit seq. no., What's the max. data rate per connection?

→ Max. amount of data that can be sent in 30 sec = 2^8 = 256 bytes.

Max data rate per connection = $\frac{256 \text{ bytes}}{30 \text{ s}}$ = 8.5 byte/s.

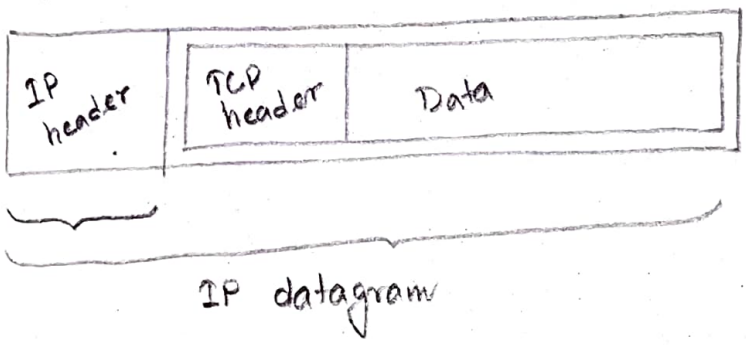
Q. Advertised window is 1 MB long. If a seq. no. is selected at random from the entire seq. no. space, what is the probability that the seq. no. falls inside the advertised window?

→ $\frac{2^{20}}{2^{32}} = \frac{1}{2^{12}}$.

Window size = 2^{20} Bytes

2^{20} seq. no.

→ Header length and calculation of Ack. no.



IP datagram total length be 1000B

IP header 5 B.

TCP header 5B.

Seq. no. of 1st byte in TCP segment 100.

What is the ack no. sent from the receiver?

Header size = 5 x 4 = 20 B [as min header size is 20 B].

Data in TCP segment = (1000 - 20 - 20) = 960 B.

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

Ack no. = 1060 (Ans)

→ TCP connection establishment

(3 way handshake.)

TCP establishes an end to end connection between the sender & receiver. This connection is established between them before exchanging the data.

3 way handshake is a process used for establishing a TCP connection. Client wants to establish a connection with the server. Before 3 way handshake, both client & server are in closed state.

Steps:

1. SYN - For establishing a connection, client sends a request segment to the server. Request segment consists only of TCP header with an empty payload. Then, it waits for a reply segment from the server.

Request segment contains the following information in TCP header —

i) Initial sequence number (RISN)
ii) SYN bit set to 1.
iii) Maximum segment size
iv) Receiving window size of sender

(i) Client sends the ISN to the server. Contained in the sequence no. field. Randomly chosen 32 bit value.

(ii) Client sets SYN bit to 1 which indicates the server that the segment contains the ISN used by the client. It has been sent for synchronising the sequence numbers.

→ SYN flooding attack: A malicious attacker sends a large number of SYN segments to a server, pretending that each of them is coming from a different client by faking the source IP addresses in the datagrams. The server, assuming that the clients are issuing an active open, allocates the necessary resources. This SYN flooding attack belongs to a type of security attack known as a denial-of-service attack, in which an attacker monopolizes a system with so many service requests that the system collapses & denies service to every request.

(iii) MSS dictates the size of the largest data chunk that client can send & receive from the server. Contained in the options field.

(iv) Limit of unacknowledged data that the client can receive.

2. SYN + ACK - After receiving the request segment, server responds to the client by sending the reply segment. It informs the client of the parameters at the server side.

Reply segment contains following information -

(i) ISN ~ Server sends the ISN to the client.

(ii) SYN bit set to 1 ~ Indicates the client the segment contains ISN

(iii) MSS ~ Size of largest chunk that server can send & receive from the client.

(iv) Receiving window size ~ Limit of unack. & buffer data that the server can receive. Contained in the window size field.

(v) Acknowledgement no. ~ Server sends the ISN incremented by 1 as an ack. no.

(vi) ACK bit set to 1 ~ Server sets ACK bit to 1. Indicates the client that the ack. no. field in the current segment is valid.

3. ACK - After receiving the reply segment, client acknowledges the response of server. It acknowledges the server by sending a pure acknowledgement.

This is just an ACK segment. Sequence no. for this segment is that of client's ISN incremented by 1. But, this same sequence will be used for data transfer. Hence, we can say that this ack segment does not consume any sequence number.

- A SYN segment cannot carry data but it consumes one sequence number.

- A SYN + ACK segment cannot carry data, but does consume one sequence no.

- TCP connection establishment phase consumes 1 seq no. of both the sides.

Request (SYN) segment consumes 1 seq. no. of the requester.

Reply segment (SYN + ACK) consumes 1 seq. no. of the respondent.

Pure acknowledgements (ACK) don't consume any seq. no.

- Pure acknowledgement for the reply segment is not necessary. This is because - if client sends the data packet immediately then it will be considered as an ack. It means that on the first two steps only, the full duplex connection is established.

- For all the segments except the request, ACK bit is always set to 1.

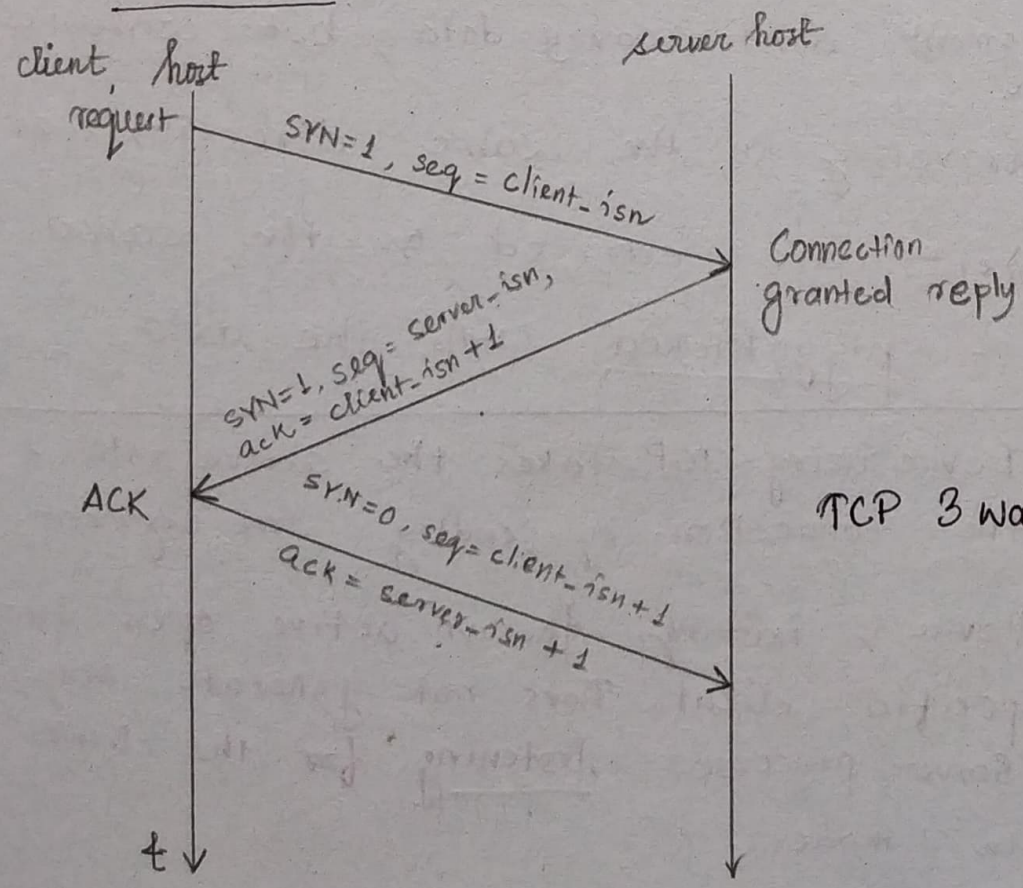
- Certain parameters (window size, MSS, timer values) are negotiated during conn^n establishment.

- In any TCP segment :

SYN	ACK	Segment remark
1	0	Request segment
1	1	Reply segment
0	1	Can be pure ACK or segment meant for data transfer.
0	0	Not possible. (as except SYN request segment ACK bit is always 1).

- There is no dedicated field for MSS in TCP header. This is because MSS has to be informed only once. So, if dedicated field would be present, then sending it each time would not be required. So, MSS is informed once using Options.

- An ack. by TCP sender guarantees data has been delivered to the application.



TCP 3 way handshake.

(namaste!
corona alert ⚠)

- SYN packet consumes 1 sequence no.

ACK=1 consumes 0 seq. no.

FIN=1 consumes 1 seq. no.

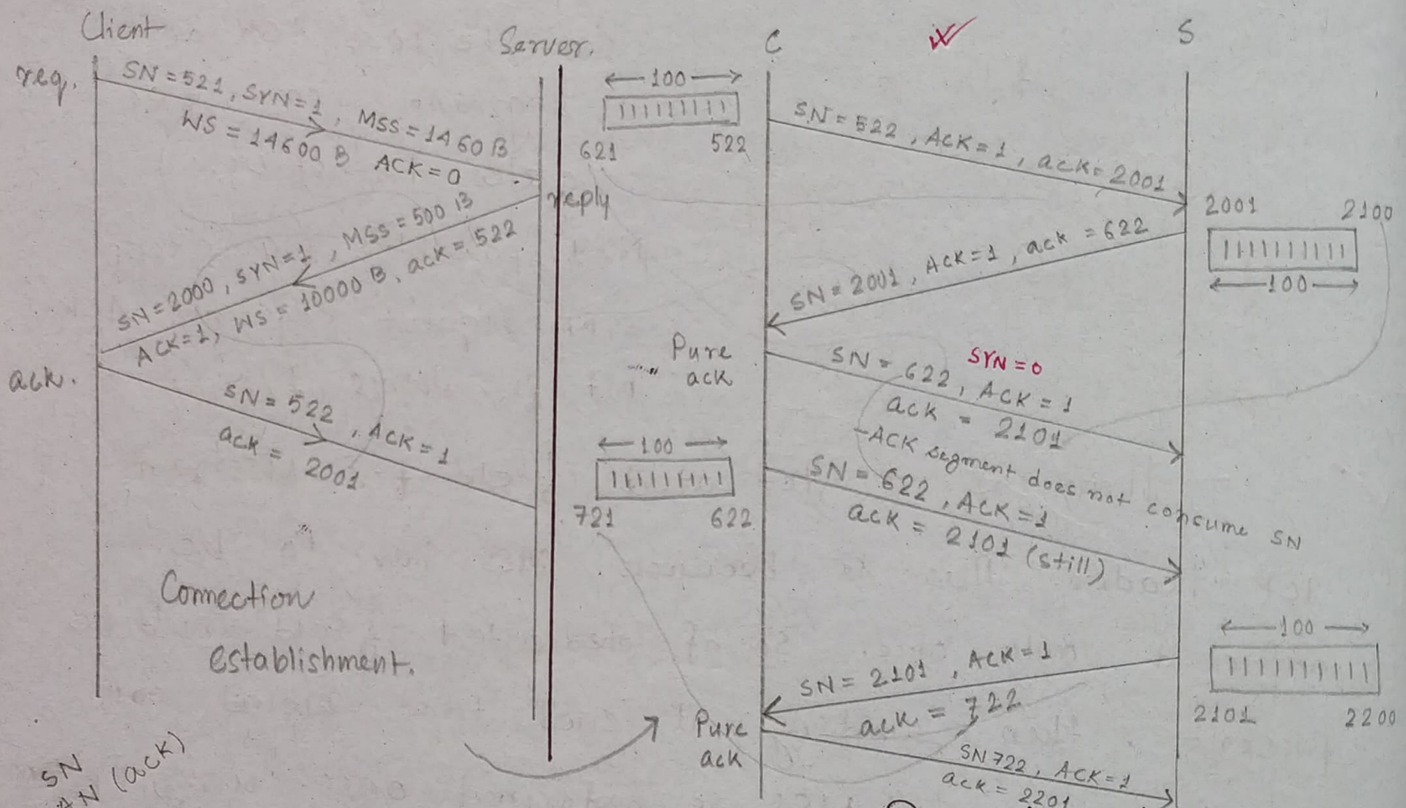
1 data byte consumes 1 seq. no.

Full duplex TCP

Pure ack - does not consume seq. no.

→ Data transfer after connection establishment.

SN - seq. no.
WS - window size



Data transfer.

SN
AN (ack)
SYN
ACK
WS
MSS
FIN

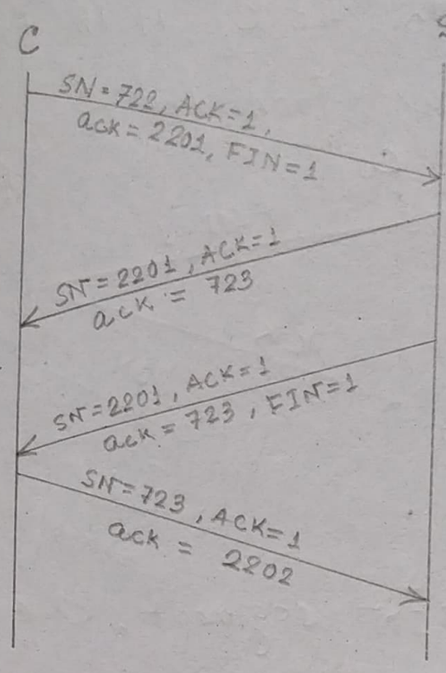
- An ACK segment, if carrying no data, consumes no SN. A SYN or SYN+ACK segment don't carry data, but consumes one SN.

- Data traveling in the same direction as an acknowledgement are carried on the same segment. The ack. is piggybacked with the data.

Active open : Device using TCP takes the active role & initiates the connection by sending SYN segment.

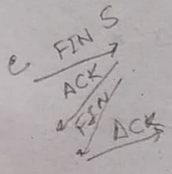
Passive open : Device is waiting for an active open from a specific client. Does not generate any TCP msg segment. Server processes listening for the clients are in passive open mode.

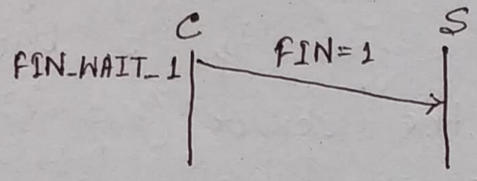
→ TCP connection termination.



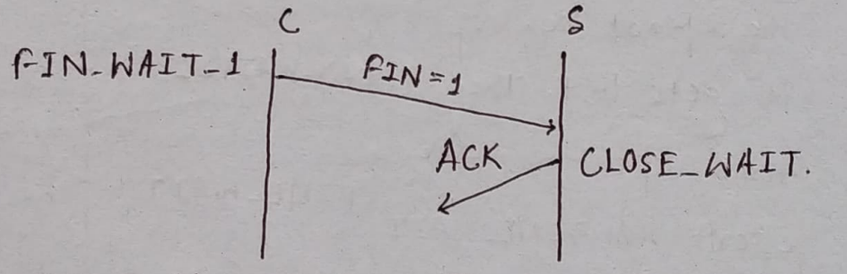
A TCP connection is terminated using FIN segment where FIN bit is set.

Step 1. For terminating a conn^n client sends a FIN segment to the server with FIN bit set. Client enters the FIN_WAIT_1 state as it waits for an ack. from the server.

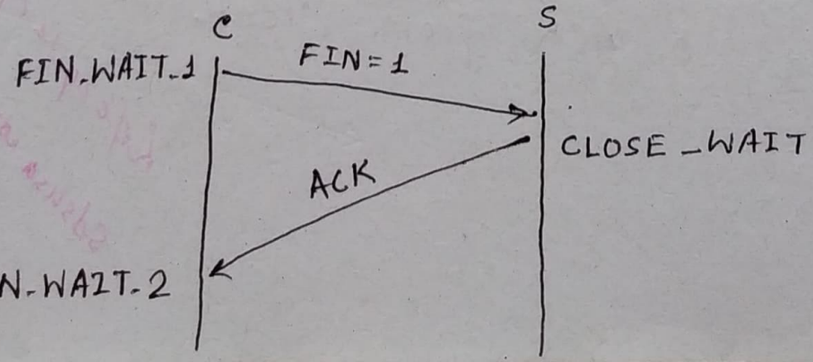




Step 2 After receiving the FIN segment, server frees up its buffers. Server sends an ack to the client. Server enters the CLOSE_WAIT state.

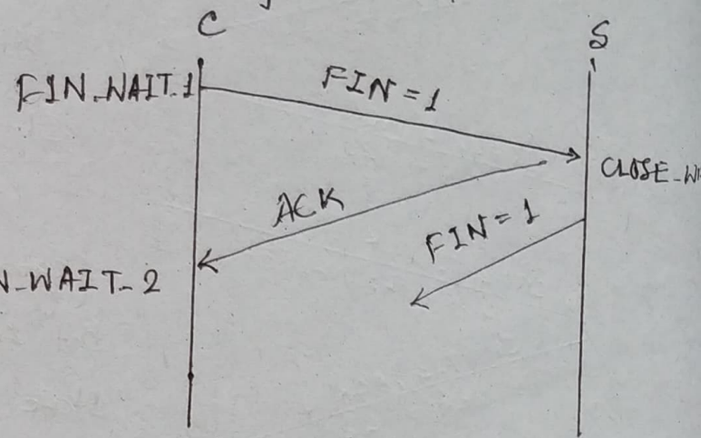


Step 3 After receiving the ack, client enters the FIN_WAIT_2 state. Now, the conn^n from C to S is terminated i.e. one way comm^n is closed. Client can't send any data to the server since server has released its buffers. Pure ack.s can still be sent from the C to S. The comm^n from S to C is still open. Server can send both data & ack.s (not needed as no data from C to S) to the client.



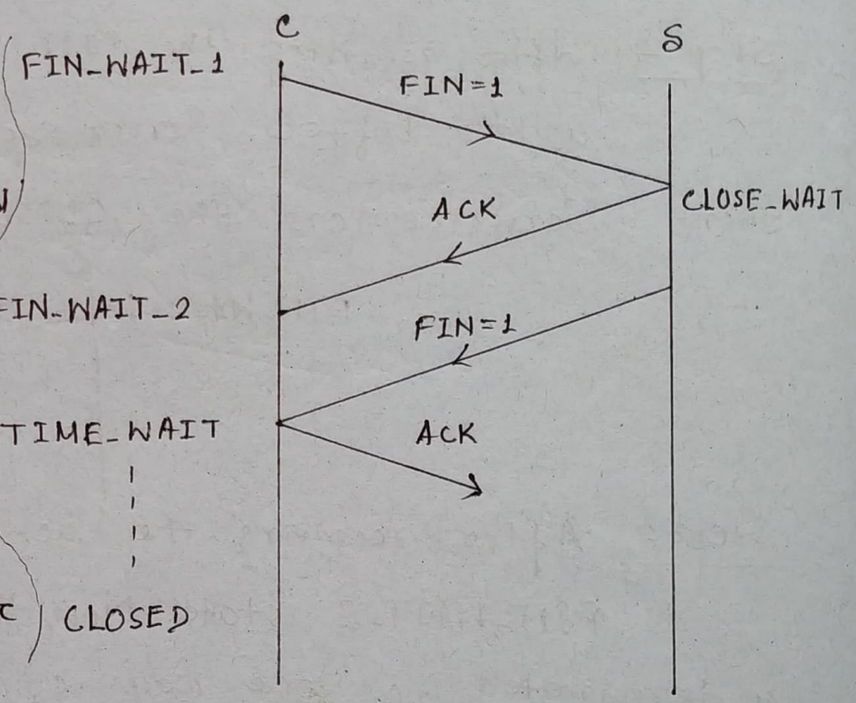
Step 4 Now, if S wants to close the conn^n with C, S sends a FIN segment to the C with FIN bit set. S waits for an ack from the C.

If server wanted, it could have sent the FIN segment along with the previous ack. that it sent to the client. (piggybacking)



Step 5 After receiving the FIN segment, client frees up its buffers. Client sends an ack. to the server (not mandatory). Client enters the TIME_WAIT state.

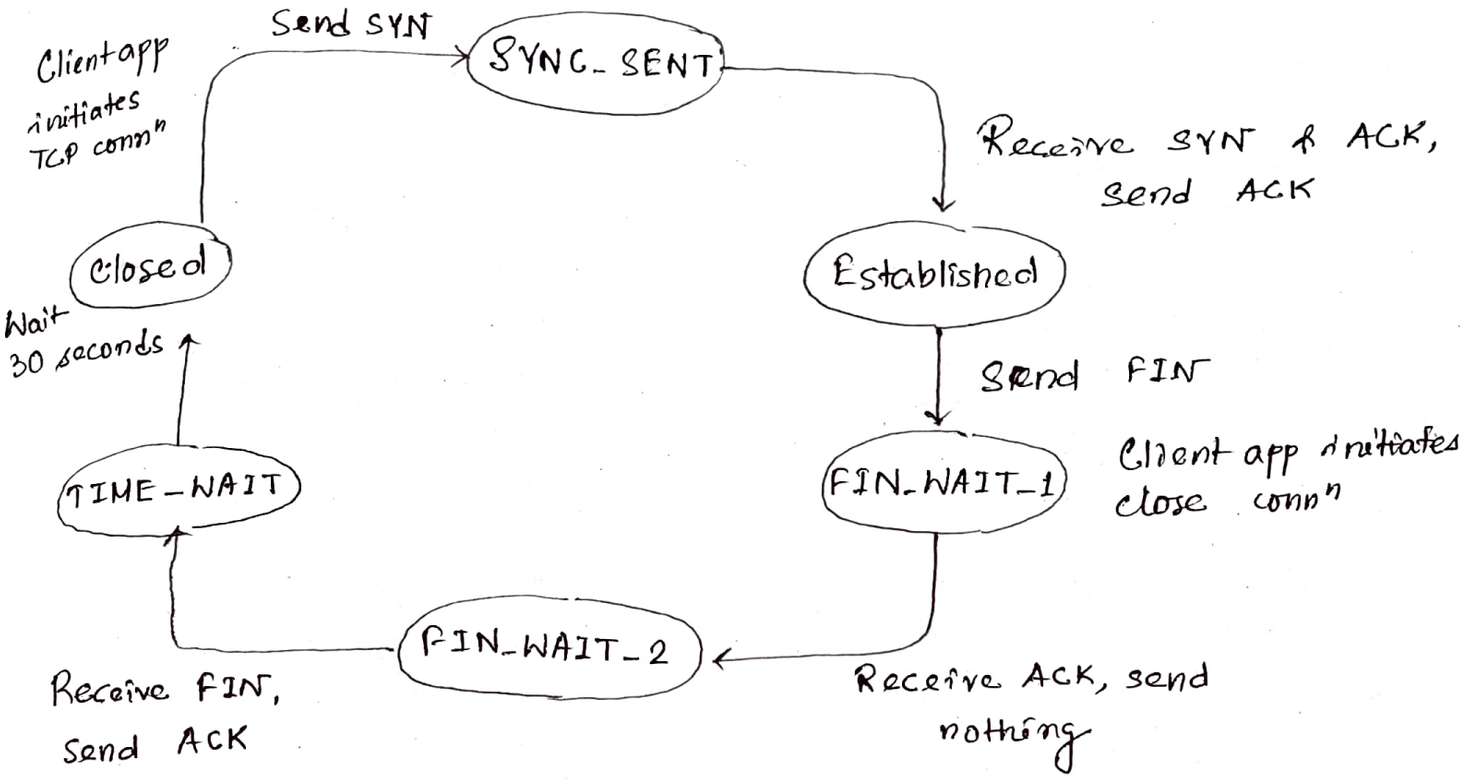
The TIME_WAIT state allows the C to resend the final ack if it gets lost. The time spent by the client in TIME_WAIT state depends on the implementation. After the wait, the conn^n gets formally closed.



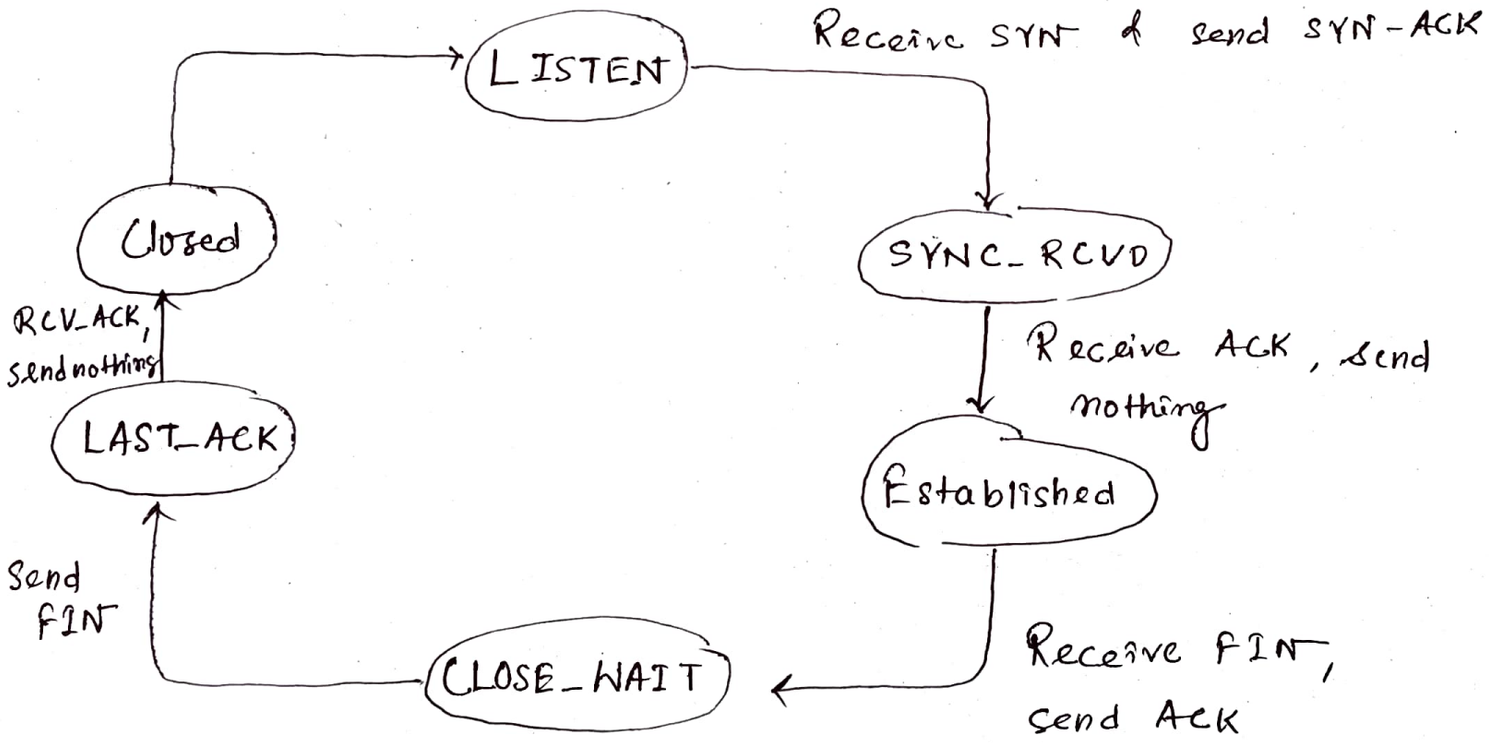
← 1. FIN from client
2. ACK from server
3. Client waiting
4. FIN from server
5. ACK from client.

Life cycle of TCP/IP
sdsusa.com/support/connections/

TCP states visited by Client -



TCP states visited by server -

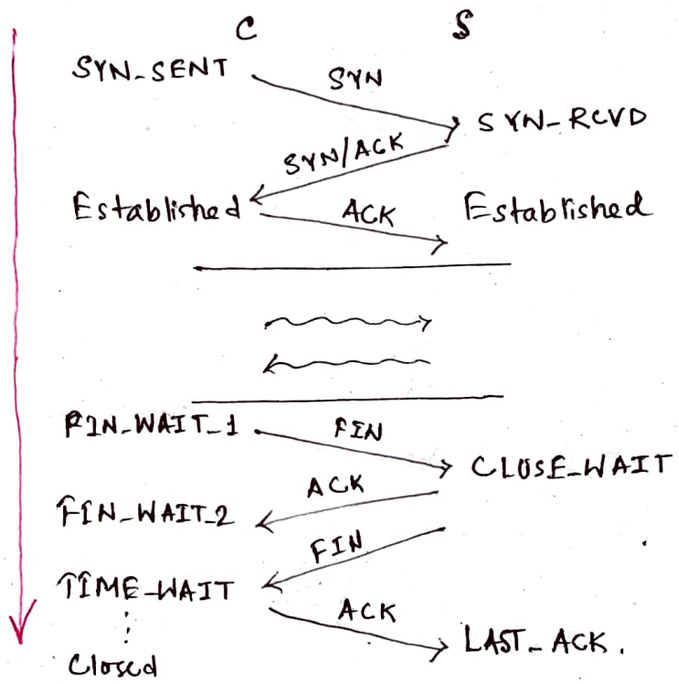


Tanenbaum (4e) 415p

Q. G'17 Consider a TCP client & a TCP server running on 2 different machines. After completing data transfer, the TCP client calls 'close' to terminate the conn^n & a FIN segment is sent to the TCP server. Server responds by sending an ACK which is received by client. As per the TCP connection state diagram (RFC 793), in which state does the client side TCP conn^n wait for the FIN from the server-side TCP?

a) LAST_ACK
b) TIME_WAIT
c) FIN_WAIT_1
✓ d) FIN_WAIT_2.

DDOS - Distributed denial of service (attacks)



TCP
conn estab.
conn termin.

PDU SDU
service data unit

Message AL
Segment TL
Datagram NL
Frame DLL
PDU PL
↳ protocol data unit

segment/datagram
packet
frame
bit (generally symbol)
} wiki

TCP - UDP (contd)

→ PSH flag: Transport layer by default waits for some time for application layer to send enough data equal to maximum segment size so that the no. of packets transmitted on N/W is minimized which is not desirable by some application like interactive ones. Similarly transport layer at receiver end buffers packets & transmit to application layer if it meets certain criteria.

This problem is solved by using PSH. Transport layer sets PSH = 1 & immediately sends the segment to N/W layer as soon as it receives signal from application layer. Receiver transport layer, on seeing PSH = 1 immediately foowards the data to application layer. In general, it tells the receiver to process these packets as they are received instead of buffering them. The sending TCP must not wait for the window to be filled. It must create segment & send it immediately.

→ URG flag: Data inside a segment with URG = 1 is forwarded to application layer immediately even if there are more data to be given to application layer. It is used to notify the receiver to process the urgent packets before processing

For interactive applications (chat) we need to transfer data to the application layer more often to keep the app interactive. Then, the PSH flag is set to 1.

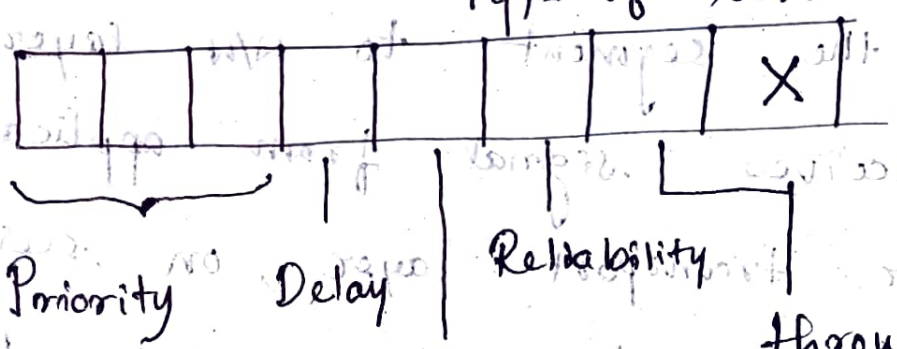
all other packets. The receiver will be notified when all known urgent data has been received.

How IPv4 sets urgent data?

In IPv4, types of service field, first 3 bits are to set priority.

As routers don't have transport layer, to make the data urgent we need to set priority value as 7 (111)

(TCP is transport layer protocol).



Data that is urgent -

(Seq. no.) to (SN + URG pointer value)

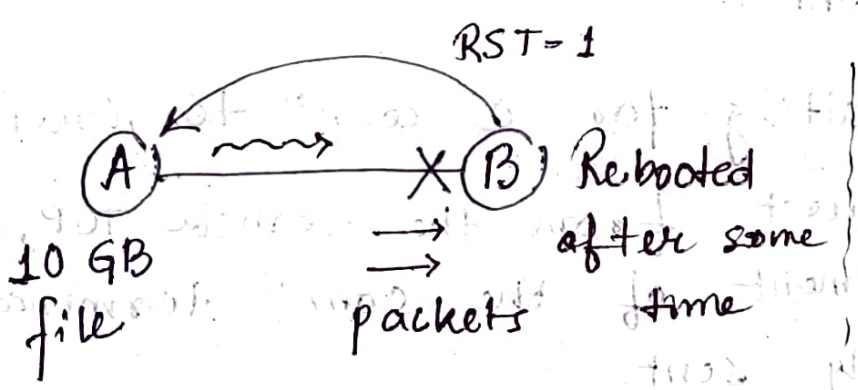
→ PSH

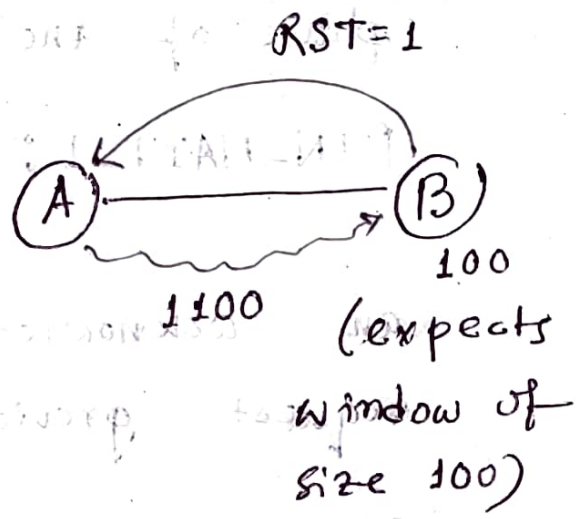
- All data in buffer to be pushed to NL (sender) or AL (receiver).
- Data delivered in order.
- No priority is set.

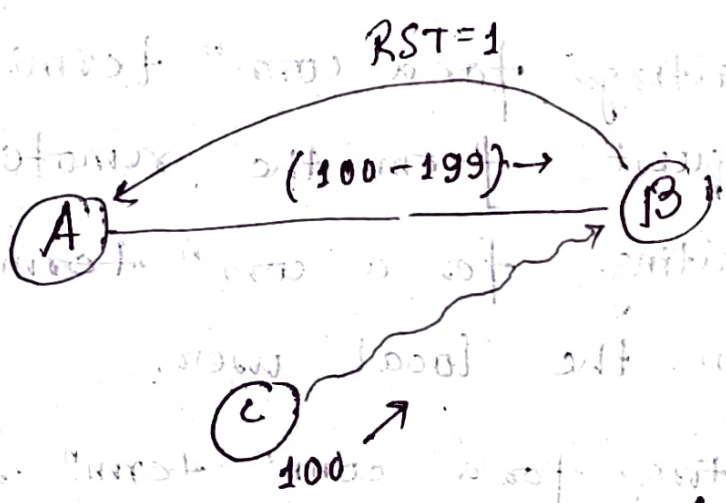
URG

- Only the urgent data to be given to AL immediately.
- Data delivered out of order.
- Priority set.

→ RST flag : Used to terminate the conn^n if the RST sendor feels something is wrong with the TCP conn^n or that the conversation should not exist.







C sending as A (pretends)

→ TCP state transition diagram.

Meaning of states:

LISTEN : Waiting for a connection request from (server) any remote TCP & port.

SYN_SENT : Waiting for a matching conn^n request after having sent a conn^n request.

SYN_RECEIVED : Waiting for a confirming conn^n request acknowledgement after having both received & sent a conn^n request.

Established : An open comm^n, data received can be delivered to the user. The normal state for the data transfer phase of the comm^n.

FIN_WAIT_1 : Waiting for a conn^n termination request from the remote TCP, or an acknowledgement of the comm^n termination request previously sent.

FIN_WAIT_2 : Waiting for a comm^n termination request from the remote TCP.

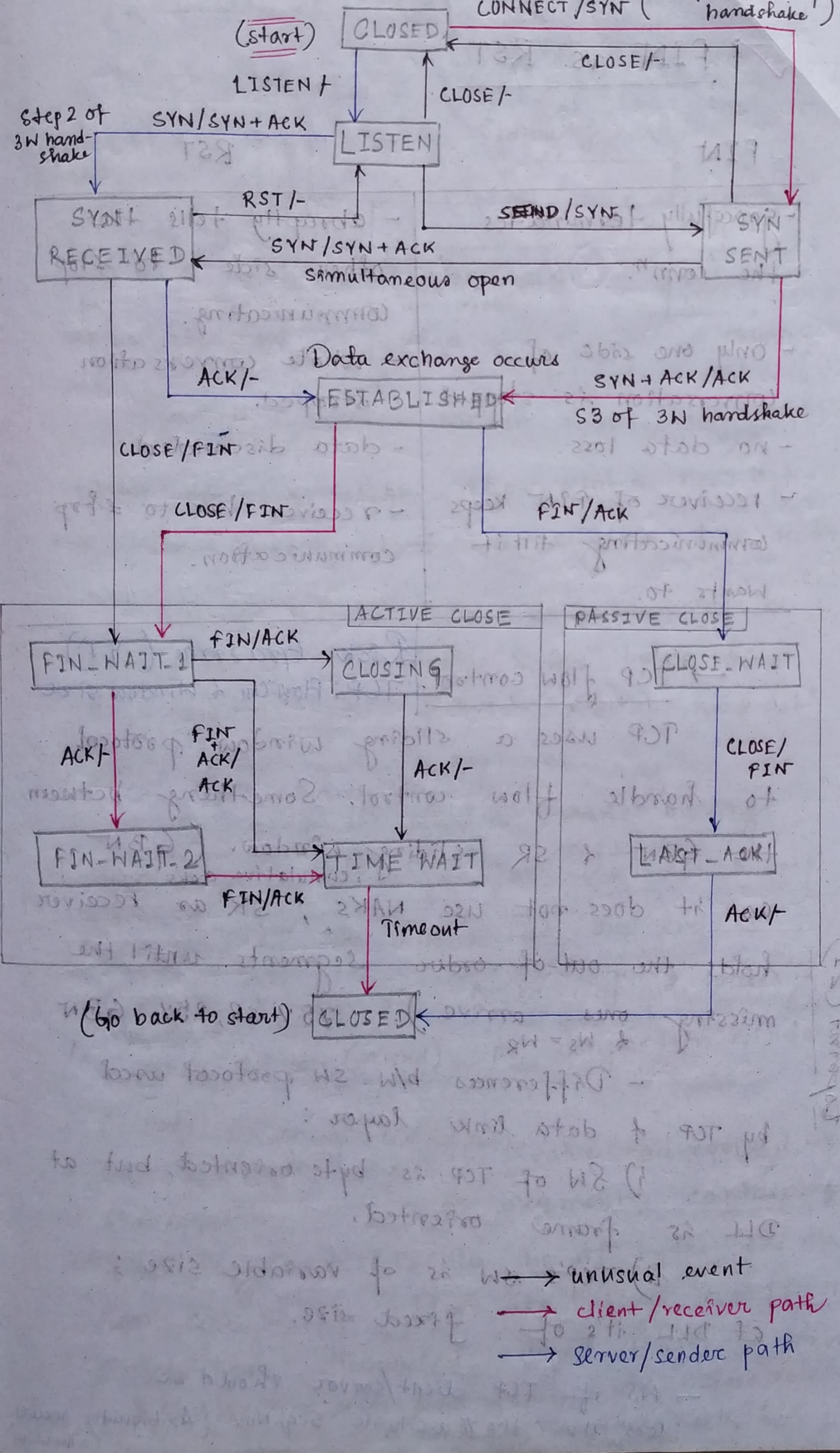
CLOSE_WAIT : Waiting for a comm^n term^n request from the local user.

CLOSING : Waiting for a comm^n term^n request acknowledgement from the remote TCP.

LAST_ACK : Waiting for an ack. of the comm^n term^n request previously sent to the remote TCP (that includes an ack of its comm^n term^n request).

TIME_WAIT : Waiting for enough time to pass to be sure the remote TCP received the ack. of its comm^n term^n request.

CLOSED : No connection state at all.


(start)
CLOSED
CONNECT/SYN (step 1 of 3 way handshake)
CLOSE/-
LISTEN/-
CLOSE/-
Step 2 of 3W handshake
SYN/SYN+ACK
LISTEN
RST/-
SEND/SYN
SYN RECEIVED
SYN SENT
SYN/SYN+ACK
Simultaneous open
Data exchange occurs
ACK/-
ESTABLISHED
SYN+ACK/ACK
S3 of 3W handshake
CLOSE/FIN
CLOSE/FIN
FIN/ACK
ACTIVE CLOSE
PASSIVE CLOSE
FIN_WAIT_1
FIN/ACK
CLOSING
CLOSE_WAIT
ACK/-
FIN+ACK/ACK
ACK/-
CLOSE/FIN
FIN_WAIT_2
TIME WAIT
LAST_ACK
FIN/ACK
Timeout
ACK/-
(Go back to start)
CLOSED
unusual event
client/receiver path
server/sender path

→ FIN vs. RST

FIN	RST
- gracefully terminates the comm^n.	- abruptly tells the other side to stop communicating.
- only one side of conversation is stopped.	- whole conversation stopped.
- no data loss.	- data discarded.
- receiver of FIN keeps communicating till it wants to.	- receiver has to stop communication.

→ TCP flow control.

Rick. Graziani (YT)
TCP: Flow Con. & Window size

TCP uses a sliding window protocol to handle flow control. Something between the GBN & SR sliding window. (GBN as it does not use NAKs, SR as receiver holds the out-of-order segments until the missing ones arrive.). 75% SR, 25% GBN
(cumulative acks & $W_S = W_R$)

- Differences b/w SW protocol used by TCP & data link layer:

i) SW of TCP is byte oriented, but at DLL is frame oriented.

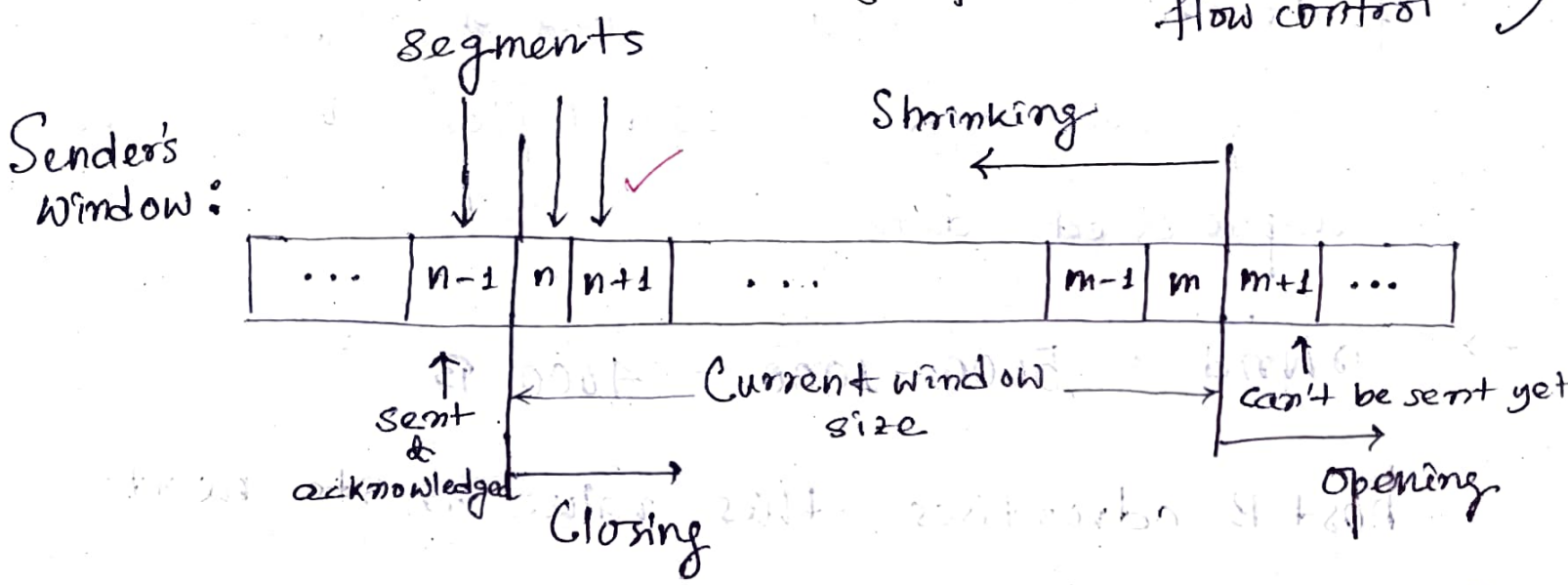
ii) TCP's SW is of variable size; at DLL it's of fixed size.

- HS of TCP client/server should be less than the # available Seq. No. (Ambiguity occurs otherwise)

topoint (21)

The window is opened, closed or shrunk.

(Tutpoint YT: Sliding window flow control)



Opening a window means moving the right wall to the right. Allows more new bytes in the buffer that are eligible for sending

Closing the window means moving the left wall to the right. Some bytes have been acknowledged & the sender need not worry about them anymore.

Shrinking the window means moving the right wall to the left.

- Size of the window at one end is determined by the lesser of 2 values - receiver window (rwnd) or congestion window (cwnd).

(The rwnd is the value advertised by the opposite end in a segment containing acknowledgement. It's the no. of bytes the other end can accept before its buffer overflows & data are discarded. Cwnd limits the amount of data the TCP can send into the N/W before receiving an Ack.)

determined by N/W to avoid congestion.

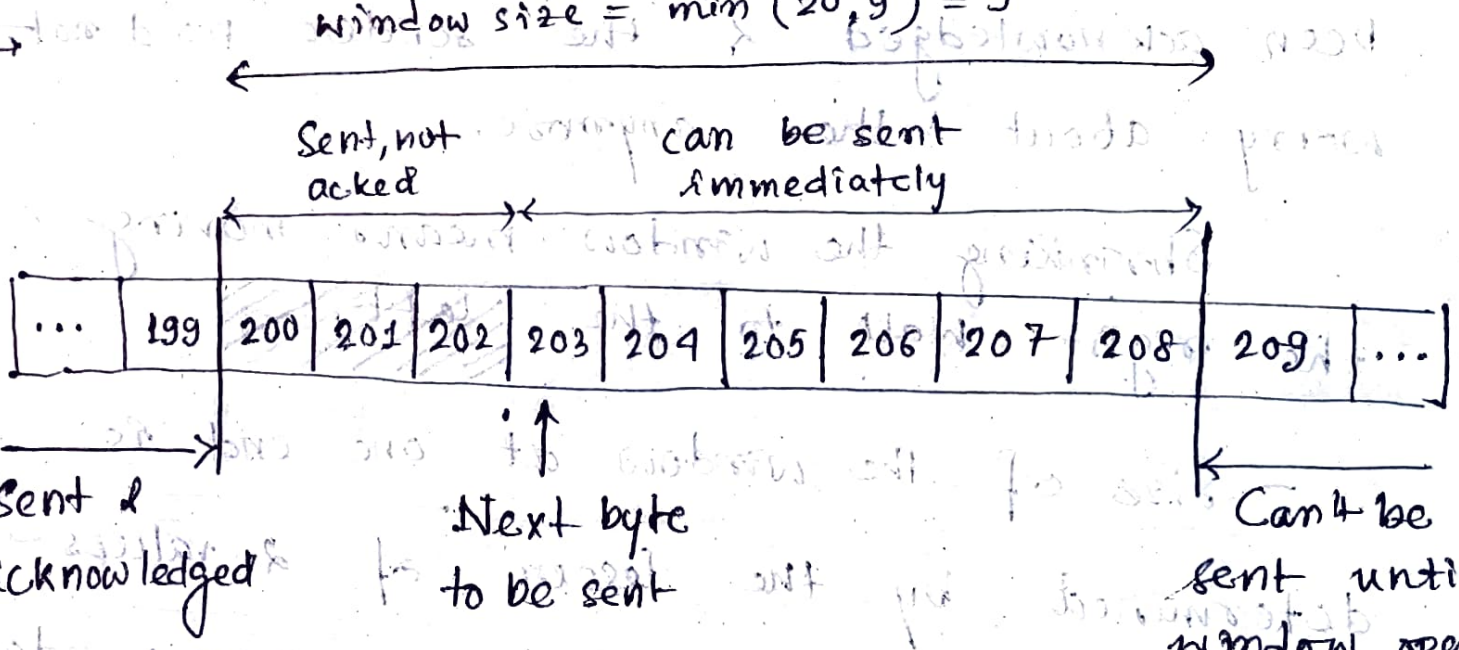
Q What is the value of rwnd for host A if the receiver, host B, has a buffer size of 5000 Bytes & 1000 Bytes of received & unprocessed data?

→ rwnd = 5000 − 1000 = 4000 B

host B advertises this value in its next segment to A.

Q. Sender has sent bytes upto 202. We assume that cwnd is 20. Receiver has sent an ack no. of 200 with an rwnd of 9 bytes. How many bytes can be sent now?

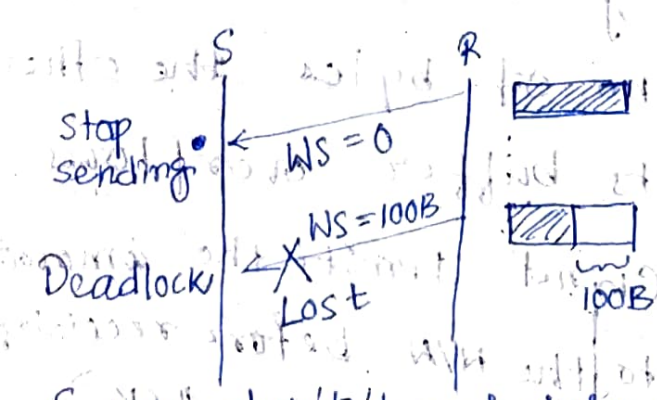
→ window size = min (20, 9) = 9



Bytes 203 to 208 can be sent without worrying about acknowledgement.

---

* Deadlock in TCP:



⇒ Solution: S sets persistent timer & when it goes off, sends a probe (1B size) to R. If R has ws > 0, it advertises in next packet.

S now doesn't send data as it does not know R's ws (as the packet was lost

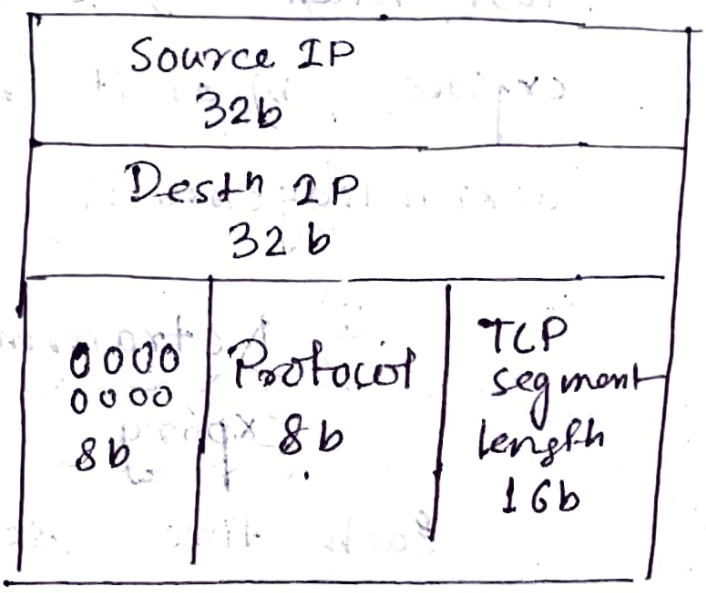
→ TCP Error control.

i) Checksum. : In TCP, the checksum is calculated both on the data & header.

Checksum is calculated on TCP header, TCP data, IP header. (We calculate checksum only on the fields in pseudo IP header).

Pseudo IP header

TCP header
+
TCP data
+
Pseudo IP header



ii) Acknowledgement. : TCP uses ack. to confirm the receipt of data segments. Control segments that carry no data but consume a sequence number are also acknowledged. ACK segments are never acknowledged.

iii) TCP retransmission. : After establishing the connⁿ, sender starts transmitting TCP segments to the receiver. A TCP segment sent by the sender may get lost on the way before reaching the receiver. This causes the receiver to send the ack with same ACK to the sender. As a result, sender

retransmits the same segment to the receiver. This is called TCP retransmission.

When sender discovers that the segment sent by it is lost, it retransmits the same segment to the receiver.

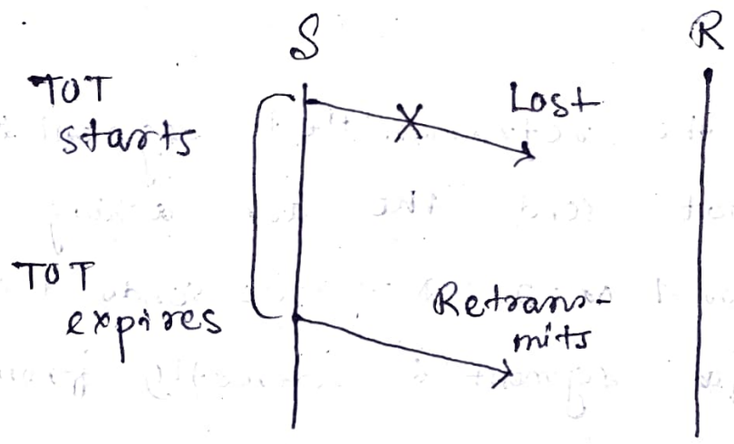
Sender discovers that the segment is lost when a) either time out timer expires, b) or it receives 3 duplicate acknowledgements.

a) Retransmission after TO timer expiry.

Each time sender transmits a TCP segment to the receiver, it starts a TOT. Now, 2 cases may arise —

(-1) Sender receives an ack for the sent segment before the timer goes off. Sender stops the timer.

(-2) Sender does not receive any ack. for the sent segment & timer goes off. In this case, sender assumes that the sent segment is lost. Sender retransmits the same segment to the receiver & resets the timer.

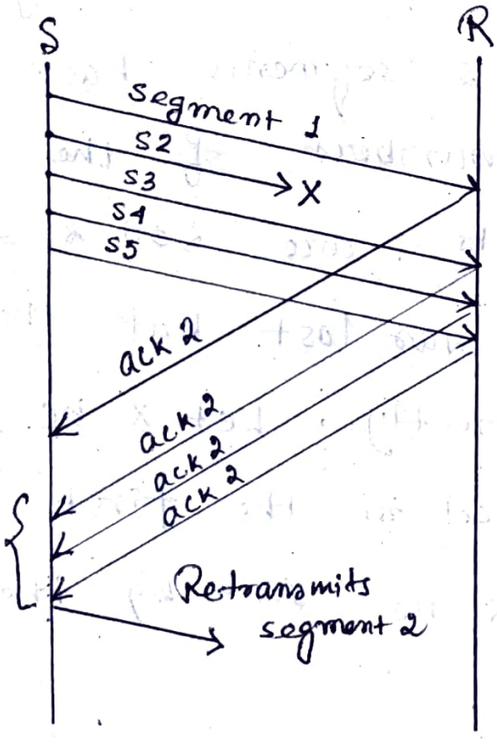


→ If 3 or more duplicate acks are received in a row it's a strong indication that a segment has been lost. (See stackexchange - Why only 3 - duplicate acks...)

b) Retransmission after receiving 3 duplicate acknowledgements.

Sender receives 3 duplicate ack.s for a TCP segment sent by it. Then, sender assumes that the corresponding segment is lost. So, Sender retransmits the ~~the~~ same segment without waiting for its TOT to expire. This is called early or fast retransmission.

eg. Sender sends 5 TCP segments to the receiver. The second TCP segment gets lost before reaching the server.



In TCP we send ack when all segments with SNT ack no. are accepted.

ack 300 ↓ segment 1. 0-99 ✓ 2 100-199 ✓ 3 200-299 ✓ all 3 accepted

After receiving the retransmitted segment 2, receiver does not send the ack. asking for segment-3 or 4 or 5. Receiver sends the ack. asking for segment 6 directly from the sender. This is because previous segments have been already received & acknowledgements for them have been already sent (although wasted in asking for segment 2).

✓ - TOT expires before receiving the ack for a TCP segment → stronger possibility of congestion in the N/W.

✓ Sender receives 3 duplicate ack.s for same segment → weaker possibility of congestion in the N/W.

Q Consider a TCP connection in a state where there are no outstanding ACKs. The sender sends 2 segments back to back. The sequence numbers of the first & second segments are 230 & 290. The first segment was lost but the 2nd segment was received correctly. Let X be the amount of data carried in the first segment & Y be the ACK no. sent by the receiver.

X, Y = ?

Amount of data contained in first segment,

$$X = (290-1) - 230 + 1 = 60 \text{ bytes}$$

Ack. no = Seq. no. of 1st segment = 230

(as first segment was lost).

→ TCP congestion control.

Congestion refers to a N/W state where the traffic becomes so heavy that it slows down the N/W response time. Congestion leads to the loss of packets in transit.

Congestion control refers to techniques & mechanisms that can either prevent congestion before it happens or remove congestion after it has happened.

- TCP reacts to congestion by reducing the sender window size. The size of the sender window is determined by the following 2 factors:

1. Receiver window size: It is an advertisement of how much data (in bytes) the receiver can receive without acknowledgement.

Sendee should not send data greater than receiver ws. Otherwise, it leads to dropping the TCP segments that causes TCP retransmission. So, sender should always send data less than or equal to W_R. Receiver dictates its window size to the sendee through TCP header.

2. Congestion window size (W_c):
Sender should not send data greater than congestion ws.

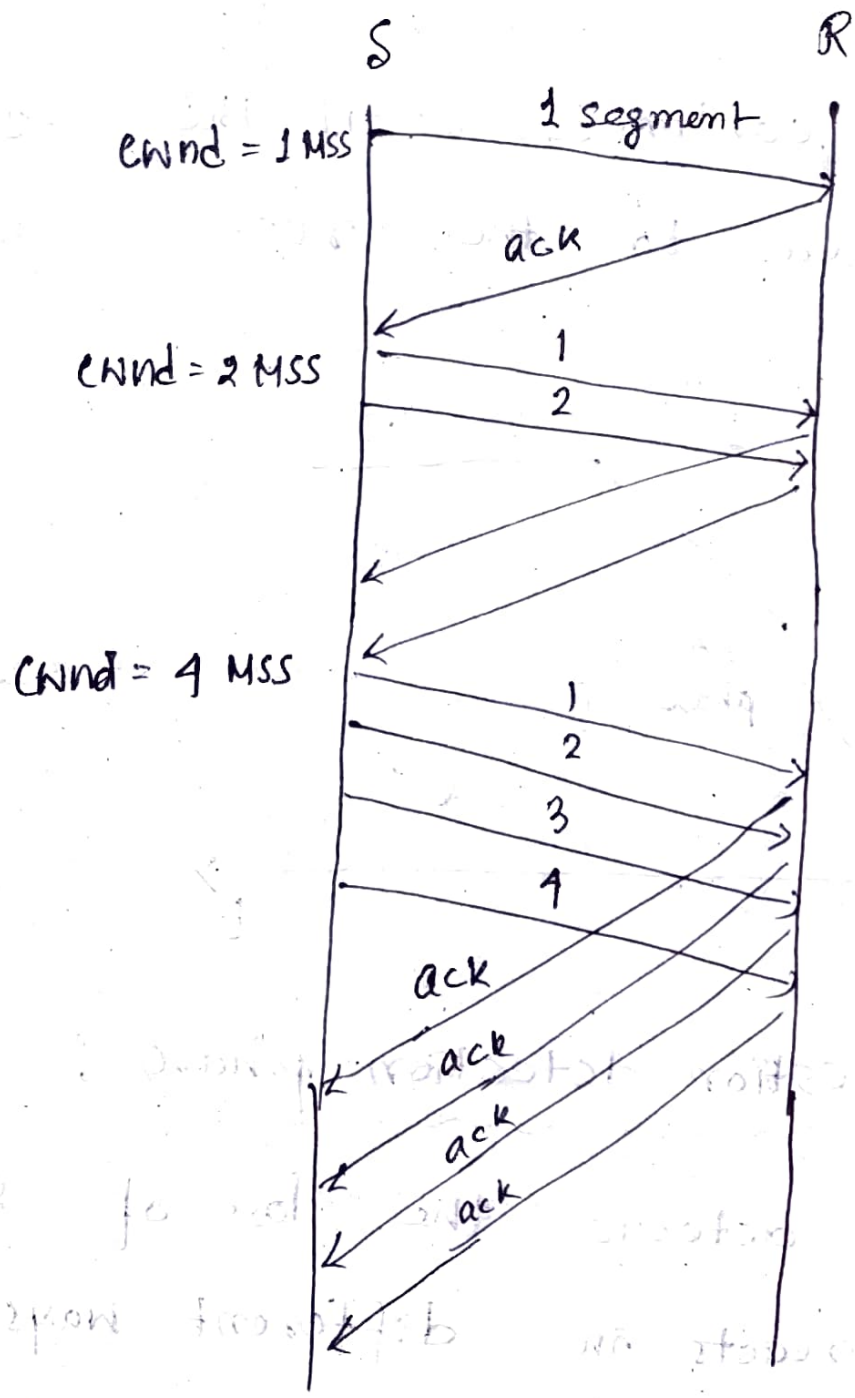
- Sendee window size =
  min (Receiver WS, Congestion WS)

- TCP congestion policy
Has 3 phases -

a) Slow start phase: Sender sets $W_c = MCax$. segment size (1 MSS). After receiving each ack, sender increases the congestion ws (W_c) by 2n MSS. In this phase size of W_c increases exponentially.

(doubles)

$b = 2a$.



After 1 RTT,
$W_c = 2^1 = 2$ MSS

After 2 RTT,
$W_c = 2^2$ MSS

⋮

This phase continues until the W_c reaches the slow start threshold.

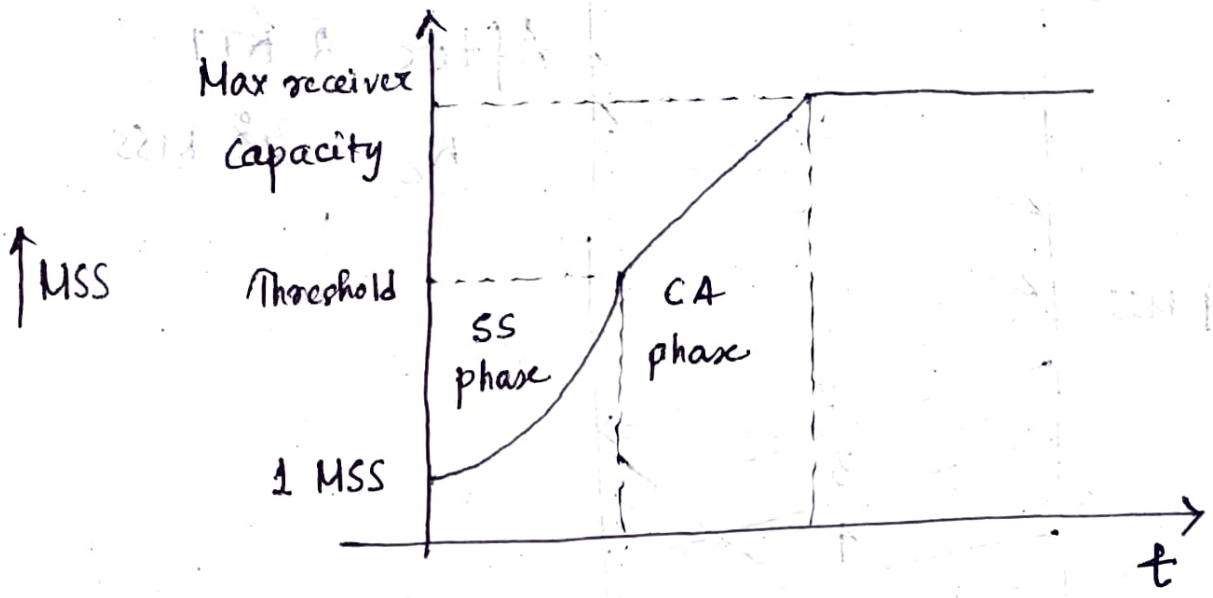
Threshold = $\left(\begin{array}{c}\text{Max. no. of segments that}\\ \text{receiver window can}\\ \text{accomodate}\end{array}\right) / 2$

$= \dfrac{\dfrac{\text{Receiver WS}}{\text{MSS}}}{2}$

b) Congestion avoidance phase:

After reaching the threshold sender increases the W_c linearly to avoid congestion.

This phase continues until the W_c becomes equal to the receiver WS.



c) Congestion detection phase :

When sender detects the loss of segments, it reacts in different ways depending on how the loss is detected.

Case 1 - Detection on time out

Expiry of TO timer. Suggests stronger possibility of congestion.

In this case, sender reacts by setting the slow start threshold to half of the current W_C. Decrease the W_C to 1 MSS and resume the SS phase.

Case 2. - Detection on receiving 3 duplicate acks

Weaker possibility of congestion in the N/N. Chances that a segment has been dropped but few segments sent later may have reached.

Sender reacts by setting the SS threshold to half the current W_c. Decrease the W_c to SS threshold. Resume the CA phase.

Q. Line with 10 msec RTT & no congestion. W_R = 24 KB, MSS = 2KB. How long does it take before the first full window can be sent?

→ W_R = 24 KB = $\frac{24}{2}$ = 12 MSS

SS Threshold = $\frac{12}{2}$ = 6 ~~MSS~~ segments

SS phase: 1 $\xrightarrow{1}$ 2 $\xrightarrow{2}$ 4 $\xrightarrow{3}$ 6

CA phase: 6 $\xrightarrow{4}$ 7 $\xrightarrow{5}$ 8 $\xrightarrow{6}$ 9 $\xrightarrow{7}$ 10 $\xrightarrow{8}$ 11 $\xrightarrow{9}$ 12

9 RTTs taken before first full window sent.

9 x 10 msec = 90 msec. (Ans)

✓ 8 Consider an instance of TCP's additive increase multiplicative decrease (AIMD) algorithm where the W_S at the start of SS phase is 2 MSS, threshold at the start of first transmission is 8 MSS. Assume that a TO occurs during the 5th transmission. Find the W_c at the end of 10th transmission.

→ SS phase: 1 → 2 → 4 → 8 (transmission no. 1, 2, 3)

CA phase: 4 → 5 → 9 → 10 (transmission no. 4, 5)

TO

SS threshold = ~~8~~ / ~~2~~ / 7 / 4. $\frac{10}{2}$ | $\frac{W_c \text{ (current)}}{2}$

$= 5$

$W_c = 2$ MSS (decrease W_c to 2 MSS)

resume SS phase: 6 → 7 → 8 (2 → 4 → 5)

CA phase: 9 → 10 → 11 (6 → 7 → 8)

Ans = 8 MSS.

Q Suppose that the TCP congestion window is set to 18 KB & a TO occurs. How big will the window be if the next four transmission bursts are all successful? Assume that the MSS is 1 KB.

→ $W_c = \frac{18\ KB}{1\ MSS} = \frac{18\ KB}{1\ KB} = 18\ MSS$

TO { SS threshold = $\frac{18}{2} = 9$ MSS
   { W_c = 1 MSS

SS phase. 1 → 2 → 4 → 8 → 9.

Ans = 9 KB.
(9 MSS)

✓ Q On a TCP connection, current W_c is 4 KB.
* Window advertised by the receiver is 6 KB. Last byte sent by the sender is 10240 & the last byte acknowledged by the receiver is 8192.

a) Current WS @ the sender ?

b) Amount of space free in sender window ?

→ a) W_s = min (4 KB, 6 KB)

= 4096 B

b) Bytes from 8193 to 10240 are still present in sender's window. Waiting for their ack. Total bytes present in sender's window

= 10240 − 8193 + 1 B

= 2048 B.

As no ack received after 8192.

From here, amount of free space in sender's window currently

= 4096 − 2048 = 2048 B

→ TCP timers. (RBR 20 TCP timer mgmt)

Used by TCP to avoid excessive delays during communication.

5 important timers ~

a) Time out timer : Used for retransmission of lost segments. Sender starts a TO timer

after transmitting a TCP segment to the receiver. If sender receives an ack before the timer goes off, it stops the timer. If sender does not receive any ack & the timer goes off, then TCP retransmission occurs. Sender retransmits the same segment & resets the timer. The value of TO timer is dynamic & changes with the amount of traffic in the N/W. TO timer is also called as retransmission timer.

b) Time wait timer : TCP uses a time wait timer during connection termination. Sender starts the TW timer after sending the ACK for the second FIN segment. It allows to resend the final ack. If it gets lost. It prevents the just closed port from reopening again quickly to some other application. It ensures all the

segments heading towards the just closed port are discarded. Value of TW timer is usually set to twice the lifetime of a TCP segment.

✓ c) Keep alive timer (Server):

Used to prevent long idle TCP connections. Each time server hears from the client, it resets the KA timer to 2 hrs. If server does not hear from the client for 2 hrs, it sends 10 probe segments to the client. These probe segments are sent at a gap of 75 secs. If server receives no response after sending 10 probe segments, it assumes that the client is down. Then server terminates the conn^{n} automatically.

Client

✓ d) Persistent timer. : To deal with a Zero-window-size deadlock situation. It keeps the window size information flowing even if the other end closes its receiver window.

Situation - sender receives an ack from the receiver with zero window size. This indicates the sender to wait. Later, receiver updates the window size & sends the segment with the update to the sender. This segment gets lost.

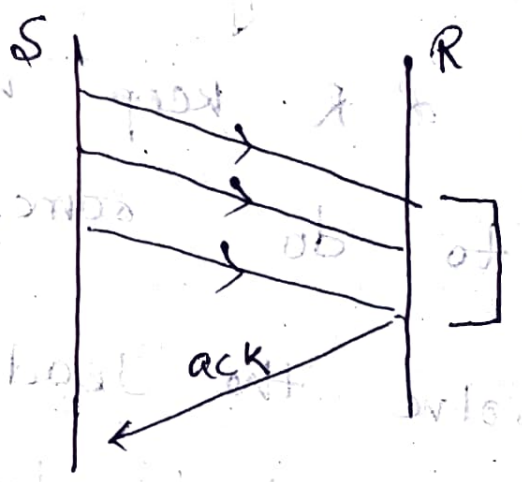
Now, both S & R keep waiting for each other to do something.

To resolve the deadlock, sender starts the persistent timer on receiving an ACK from the receiver with a zero window size. When persistent timer goes off, sender sends a special segment to the receiver. This is called a probe segment & contains only 1 B of new data. Response sent by the R

to the probe segment gives the updated window size. If the updated window size is non-zero, it means data can be sent now. If the updated window size is still zero, the persistent timer is set again & the cycle repeats.

e) Acknowledgement timer.

Cumulative ack & piggy backing.



- Time out timer. (21, 22, 23 rbr)

Consider receiver has sent the ack to the sender. The ack is on its way through the network.

case 1: high traffic. - If there's high traffic in the N/W, the time taken by the ack to reach the

sender will be more. So, as per the high traffic, value of TO timer should be kept large. If the value is kept small, then timer will time out soon. It causes the sender to assume that the segment is lost before reaching the receiver. However, in actual the ack is delayed due to heavy traffic. Sender keeps retransmitting the same segment. This overburdens the N/W & might lead to congestion.

case 2: low traffic - time taken by the ack to reach the sender will be less. So, as per the low traffic, the value of TO timer should be kept small. If the value is kept large, timer will not time out soon. Sender keeps waiting for the ack even when it is actually lost. This causes excessive delay.

✓ - So the value of TO timer should

be such that - it decreases when there is low traffic in the N/W, & increases when there's high traffic.

- Algorithms for computing TO timer value.

All algorithms work on following rules -

1. Value of TOT for the next segment is increased when actual RTT for the previous segment is found to be increased indicating there's high traffic in N/W.

2. Value of TOT for the next segment is decreased when actual RTT for the previous segment is found to be decreased indicating there's low traffic in N/W.

Basic algorithm.

Step 1 - While sending the 1st segment sender assumes any random value of initial RTT say $IRTT_1$. So, after sending the 1st segment, sender expects its ack to arrive in time $IRTT_1$. Sender sets time out timer value (TOT) for the 1st segment to be

✓ $TOT_1 = 2 \times IRTT_1$

Suppose ack for the 1st segment arrives in time $ARTT_1$ (actual RTT for 1st segment).

Step 2 - While sending the 2nd segment, sender computes the value of initial RTT for the 2nd segment using

✓ $IRTT_{n+1} = \alpha(IRTT_n) + (1-\alpha)(ARTT_n)$

α is smoothening factor

$(0 \leq \alpha \leq 1)$

After sending the 2nd segment, sender expects it ack to arrive in time $IRTT_2$. Sender sets TOT value for the 2nd segment to be

✓ $TOT_2 = 2 \times IRTT_2$

Suppose ack for the 2nd segment arrives in time $ARTT_2$

In similar fashion, compute TOT value for further segments.

adv: i) TOT value is flexible to dynamic RTT.

ii) It takes into consideration all the previously sent segments to derive the initial RTT for the current segment.

✓ disadv: i) Always considers TOT value = $2 \times IRTT$.

Jacobson's algorithm.

Modified version of the basic algorithm Better performance.

Step-1 While sending 1st segment, sender assumes any random value of initial RTT say $IRTT_1$. Sender assumes any random value of initial deviation say ID_1. So, after sending the 1st segment, sender expects its ack to arrive in time $IRTT_1$. Sender assumes any random value of initial deviation say ID_1. So, after sending the 1st segment, sender expects there will be a deviation of ID_1 time from $IRTT_1$. Sender sets TOT value for the 1st segment to be —

$$TOT_1 = 4 \times ID_1 + IRTT_1$$

Suppose ack for the 1st segment arrives in time, $ARTT_1$. Here,

$AD_1 = |IRTT_1 - ARTT_1|$. Actual Deviation

Step 2. While sending the 2nd segment, sender computes the value of initial RTT for the 2nd segment using the relation

$$IRTT_{n+1} = \alpha\, IRTT_n + (1-\alpha)\, ARTT_n$$

$$0 \le \alpha \le 1.$$

Sender computes the value of initial deviation for the 2nd segment using the relation —

$$ID_{n+1} = \alpha\, ID_n + (1-\alpha)\, AD_n$$

$$0 \le \alpha \le 1.$$

$$IRTT_2 = \alpha\, IRTT_1 + (1-\alpha)\, ARTT_1$$

$$ID_2 = \alpha\, ID_1 + (1-\alpha)\, AD_1.$$

So, after sending the 2nd segment, sender expects its ack. to arrive in time $IRTT_2$ with deviation of ID_2 time. Sender sets TOT value for the 2nd segment to be —

$$TOT_2 = 4 \times ID_2 + IRTT_2$$

Suppose, ack for the 2nd segment arrives in time $ARTT_2$. Actual deviation from $IRTT_2$ is given by

$$AD_2 = |IRTT_2 - ARTT_2|.$$

In this manner, algo computes the TOT value for all the further segments.

Problems with basic algorithm & Jacobson's algorithm.

To calculate initial RTT, both algos depend on the actual RTT of the previous segment through the rel^n -

$$IRTT_n = \alpha\, IRTT_n + (1-\alpha)\, ARTT_n.$$

Now, consider ack of some segment arrives to the sender after its initial TOT goes off. Then, sender will have to retransmit the segment. Now, for the segment being retransmitted, what should be the initial TOT value is the concern. This is because the ack is delayed & will arrive after TO. So, ARTT is not available. (Resolved by Karn's modification

Karn's Modification

Whenever a segment has to be retransmitted, do not apply either of Basic or Jacobson's since ARTT is not available. Instead, double the TOT whenever the timer times out & make a retransmission.

Q IRTT = 10 msec, acks for the first 3 segments are received in 15, 20, 10 msec. Find TOT value for the first four segments using basic algorithm. $\alpha = 0.5$.

→ 1st segment –

$IRTT_1 = 10$ msec

$TOT_1 = 2 \times IRTT_1 = 20$ msec

$ARTT_1 = 15$ msec

2nd segment –

$IRTT_2 = \alpha IRTT_1 + (1-\alpha) ARTT_1$

$= 12.5$ msec

$TOT_2 = 2 \times IRTT_2 = \underline{25 \text{ msec}}$

$ARTT_2 = 20$ msec.

3rd segment –

$IRTT_2 = \alpha \times IRTT_2 + (1-\alpha) ARTT_2$

$= 16.25$ msec.

$TOT_2 = 2 \times IRTT_3 = \underline{32.5 \text{ msec}}$

$ARTT_3 = 10$ msec

4th segment –

$IRTT_4 = \alpha\, IRTT_3 + (1-\alpha) ARTT_3$

$= 13.125$ msec

$TOT_4 = 2\, IRTT_4 = \underline{26.25 \text{ msec}}$.

Q IRTT = 10 msec, $ID_1 = 5$ msec. Acks for the first 3 segments are received in 20, 30, 10 msec. Find TOT value for first 4 segments using Jacobson's. $\alpha = 0.5$.

→ 1st segment –

$IRTT_1 = 10$ msec

$ID_1 = 5$ msec.

$TOT_1 = 4 \times ID_1 + IRTT_1$

$= \underline{30 \text{ msec.}}$

$ARTT_1 = 20$ msec

Actual deviation = $AD_1 = |IRTT_1 - ARTT_1|$

$= |10 - 20| = 10$ msec

2nd segment –

$$IRTT_2 = \alpha\, IRTT_1 + (1-\alpha)\, ARTT_1$$

$$= 15 \text{ msec}$$

$$ID_2 = \alpha\, ID_1 + (1-\alpha)\, AD_1$$

$$= 7.5 \text{ msec}$$

$$TOT_2 = 4ID_2 + IRTT_2 = \underline{45 \text{ msec}}$$

$$ARTT_2 = 30 \text{ msec}$$

$$AD_2 = |\, IRTT_2 - ARTT_2 \,|$$

$$= 15 \text{ msec}.$$

3rd segment –

$$IRTT_3 = \alpha\, IRTT_2 + (1-\alpha)\, ARTT_2$$

$$= 22.5 \text{ msec}$$

$$ID_2 = \alpha\, ID_2 + (1-\alpha)\, AD_2$$

$$= 11.25 \text{ msec}$$

$$TOT_3 = 4ID_3 + IRTT_3 = \underline{67.5 \text{ msec}}$$

$$ARTT_2 = 10 \text{ msec}.$$

$$AD_2 = |\, IRTT_3 - ARTT_3 \,| = 12.5 \text{ msec}.$$

4th segment –

RTT_4 = 16.25 msec

ID_4 = 11.875 msec

TOT_4 = 63.75 msec

→ topic 8 Silly window syndrome

→ Traffic shaping

Another method of congestion control is to 'shape' the traffic before it enters the N/W. Traffic shaping controls the rate at which packets are sent. During connection establishment, the sender & the carrier negotiate a traffic pattern (shaping).

Leaky bucket algorithm.

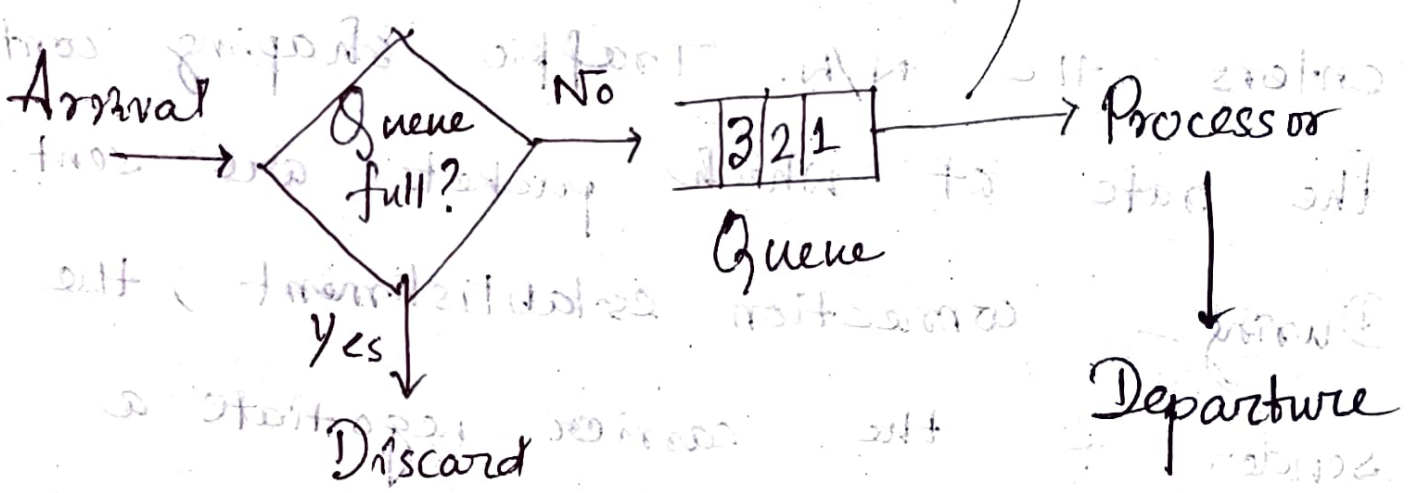
1. When host wants to send packet, packet is thrown into the bucket.

2. The bucket leaks at a constant rate, meaning the N/W interface transmits packets at a constant rate.

(use a queue to keep the data output rate constant even if there's bursty data)

3. Bursty traffic is converted to a uniform traffic by the leaky bucket.

4. In practice, the bucket is a finite queue that outputs at a finite rate.



✓ Token Bucket Algorithm.

The leaky bucket algorithm enforces output pattern at the avg rate, no matter how bursty the traffic is. So, in order to deal with the bursty traffic we need a flexible algorithm so that the data is not lost. Hence, token bucket algo.

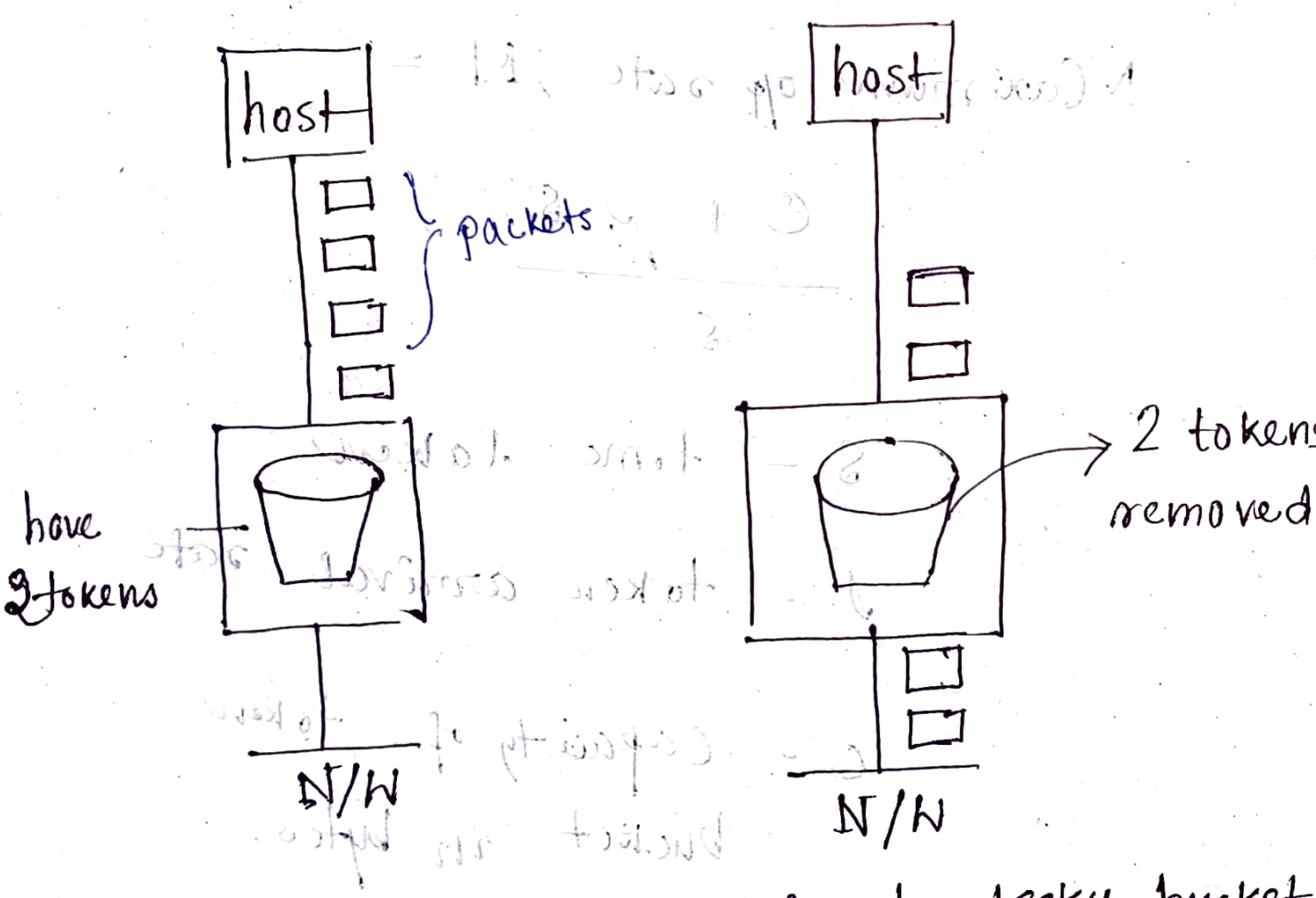
Steps -

1. In regular intervals tokens are thrown into the bucket.

2. The bucket has a max. capacity.

3. If there is a ready packet, a token is removed from the bucket, & the packet is sent.

4. If there is no token in the bucket, the packet cannot be sent.



Token bucket superior to leaky bucket. Flexibility introduced in token bucket algo. Tokens are generated at each tick (up to a limit). For an incoming

Bucket of tokens → Ready packet
the N/W comes → Takes a token → Goes out to

packet to be transmitted, it must capture a token & the transmission takes place at the same rate. Some of the bursty packets are transmitted at the same rate if tokens are available & thus introduces some amount of flexibility in the system.

Maximum o/p rate, $M = \frac{C + \rho \cdot S}{S}$

S - time taken

ρ - token arrival rate

C : capacity of token bucket in bytes.

eg. Token bucket of capacity 250 KB & tokens arrive at rate 2 MB/s. If max. o/p rate is 25 MB/s, what is burst time?

$$\frac{C + \rho \cdot S}{S} = M \Rightarrow S = \frac{C}{M - \rho}$$

$$= \frac{250\ KB}{25\ MB/s - 2\ MB/s}$$

$$= 10.86\ ms$$

H/W Devices - refer RBR videos

Silly Window Syndrome | Nagle's Algorithm

Computer Networks

Silly Window Syndrome-

- Silly Window Syndrome is a problem that arises due to the poor implementation of TCP.
- It degrades the TCP performance and makes the data transmission extremely inefficient.

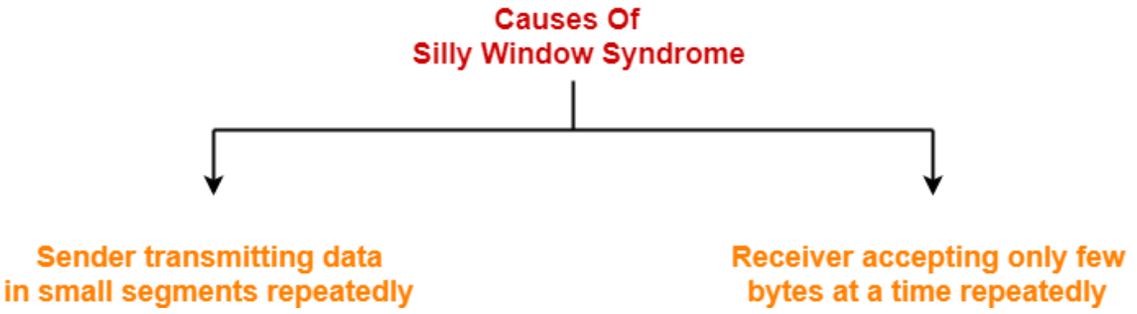
Why This Name?

The problem is called so because-

- It causes the sender window size to shrink to a silly value.
- The window size shrinks to such an extent where the data being transmitted is smaller than **TCP Header**.

Causes-

The problem arises due to following causes-



1. Sender transmitting data in small segments repeatedly
2. Receiver accepting only few bytes at a time repeatedly

Cause-01: Sender Transmitting Data In Small Segments Repeatedly-

- Consider application generates one byte of data to send at a time.
- The poor implementation of TCP causes the sender to send each byte of data in an individual TCP segment.

This problem is solved using Nagle's Algorithm.

Nagle's Algorithm-

Nagle's Algorithm tries to solve the problem

caused by the sender delivering 1 data byte at a time.

Nagle's algorithm suggests-

- Sender should send only the first byte on receiving one byte data from the application.
- Sender should buffer all the rest bytes until the outstanding byte gets acknowledged.
- In other words, sender should wait for 1 RTT.
- After receiving the acknowledgement, sender should send the buffered data in one TCP segment.
- Then, sender should buffer the data again until the previously sent data gets acknowledged.

Cause-02: Receiver Accepting Only Few Bytes Repeatedly-

- Consider the receiver continues to be unable to process all the incoming data.
- In such a case, its window size becomes smaller and smaller.
- A stage arrives when it repeatedly sends the window size of 1 byte to the sender.

This problem is solved using Clark's Solution.

Clark's Solution-

> Clark's Solution tries to solve the problem
>
> caused by the receiver sucking up one data byte at a time.

Clark's solution suggests-

- Receiver should not send a window update for 1 byte.
- Receiver should wait until it has a decent amount of space available.
- Receiver should then advertise that window size to the sender.

Specifically, the receiver should not send a window update-

- Until it can handle the MSS it advertised during **Three Way Handshake**
- Or until its buffer is half empty, whichever is smaller.

Important Notes-

Note-01:

> Nagle's algorithm is turned off for the applications that require data to be sent immediately.

This is because-

- Nagle's algorithm sends only one segment per round trip time.
- This impacts the latency by introducing a delay.

Note-02:

Nagle’s algorithm and Clark’s solution are complementary.

- Both Nagle’s solution and Clark’s solution can work together.
- The ultimate goal is sender should not send the small segments and receiver should not ask for them.

PRACTICE PROBLEM BASED ON NAGLE’S ALGORITHM-

Problem-

A fast typist can do 100 words a minute and each word has an average of 6 characters. Demonstrate Nagle’s algorithm by showing the sequence of TCP segment exchanges between a client with input from our fast typist and a server. Indicate how many characters are contained in each segment sent from the client.

Consider the following two cases-

1. The client and server are in the same LAN and the RTT is 20 ms.
2. The client and server are connected across a WAN and the RTT is 100 ms.

Solution-

Nagle’s algorithm suggests-

- Sender should wait for 1 RTT before sending the data.
- The amount of data received from the application layer in 1 RTT should be sent to the receiver.

Case-01:

Amount of data accumulated in 1 RTT

= (600 characters / 1 minute) x 20 msec

= (600 characters / 60 sec) x 20 msec

= (10 characters / 10^3 msec) x 20 msec

= 0.2 characters

From here, we observe-

- Even if the sender waits for 1 RTT, not even a single character is produced.
- So, sender will have to wait till it receives at least 1 character.
- Then, sender sends it in one segment.

Thus, one character will be sent per segment.

Assuming the TCP header length is 20 bytes, 41 bytes of data will be sent in each segment.

Case-02:

Amount of data accumulated in 1 RTT

= (600 characters / 1 minute) x 100 msec

= (600 characters / 60 sec) x 100 msec

= (10 characters / 10^3 msec) x 100 msec

= 1 character

From here, we observe that one character is produced in 1 RTT.

Thus, one character will be sent per segment.

Assuming the TCP header length is 20 bytes, 41 bytes of data will be sent in each segment.

Next Article- User Datagram Protocol | UDP Header

Get more notes and other study material of **Computer Networks**.

Watch video lectures by visiting our YouTube channel **LearnVidFun.**

Summary

Article Name	Silly Window Syndrome | Nagle's Algorithm
Description	Silly Window Syndrome is a problem that arises due to the poor implementation of TCP. Nagle's Algorithm and Clark's solution tries to solve the problems caused due to silly window syndrome.
Author	Akshay Singhal
Publisher Name	Gate Vidyalay
Publisher Logo	GateVidyalay A temple of learning

Transmission Control Protocol | Practice Problems

Computer Networks

PRACTICE PROBLEMS BASED ON TRANSMISSION CONTROL PROTOCOL-

Problem-01:

How many TCP connections can be opened between two ports?

1. Multiple
2. Single
3. Zero
4. None

Solution-

Option (B) is correct.

Problem-02:

TCP protects itself from miss delivery by IP with the help of-

1. Source IP Address in IP header
2. Destination IP Address in IP header
3. Pseudo header
4. Source port and Destination port

Solution-

Option (C) is correct.

Problem-03:

What addressing system has topological significance?

1. Logical or Network Address
2. LAN or Physical Address
3. Port Addressing System
4. Multicast Addressing System

Solution-

Option (A) is correct.

Problem-04:

If WAN link is 2 Mbps and RTT between source and destination is 300 msec, what would be the optimal TCP window size needed to fully utilize the line?

1. 60,000 bits
2. 75,000 bytes
3. 75,000 bits
4. 60,000 bytes

Solution-

Given-

- Bandwidth = 2 Mbps
- RTT = 300 msec

Optimal TCP Window Size-

Optimal TCP window size

= Maximum amount of data that can be sent in 1 RTT

= 2 Mbps x 300 msec

= 600 x 10^3 bits

= 60,0000 bits

= 75,000 bytes

Thus, Option (B) is correct.

Problem-05:

Suppose host A is sending a large file to host B over a TCP connection. The two end hosts are 10 msec apart (20 msec RTT) connected by a 1 Gbps link. Assume that they are using a packet size of 1000 bytes to transmit the file. For simplicity, ignore ack packets. At least how big would the window size (in packets) have to be for the channel utilization to be greater than 80%?

1. 1000
2. 1500
3. 2000
4. 2500

Solution-

Given-

- RTT = 20 msec
- Bandwidth = 1 Gbps
- Packet size = 1000 bytes
- Efficiency >= 80%

Window Size For 100% Efficiency-

For 100% efficiency,

Window size

= Maximum number of bits that can be transmitted in 1 RTT

= 1 Gbps x 20 msec

= (10^9 bits per sec) x 20 x 10^{-3} sec

= 20 x 10^6 bits

= 2 x 10^7 bits

Window Size For 80% Efficiency-

For 80% efficiency,

Window size

= 0.8 x 2 x 10^7 bits

= 1.6 x 10^7 bits

In terms of packets,

Window size

= 1.6 x 10^7 bits / Packet size

= 1.6 x 10^7 bits / (1000 x 8 bits)

= 0.2 x 10^4 packets

= 2000 packets

Thus, Option (C) is correct.

Problem-06:

A TCP machine is sending windows of 65535 B over a 1 Gbps channel that has a 10 msec one way delay.

1. What is the maximum throughput achievable?

2. What is the line efficiency?

Solution-

Given-

- Window size = 65535 bytes
- Bandwidth = 1 Gbps
- One way delay = 10 msec

Method-01:

Maximum amount of data that can be sent in 1 RTT

= 1 Gbps x (2 x 10 msec)

= (10^9 bits per sec) x 20 x 10^{-3} sec

= 20 x 10^6 bits

= 25 x 10^5 bytes

Amount of data that is actually being sent in 1 RTT = 65535 bytes

Thus,

Line Efficiency(η)

= Amount of data being sent in 1 RTT / Maximum amount of data that can be sent in 1 RTT

= 65535 bytes / 25 x 10^5 bytes

= 0.026214

= 2.62%

Now,

Maximum Achievable Throughput

= Efficiency x Bandwidth

= 0.0262 x 1 Gbps

= 26.214 Mbps

Method-02:

Maximum Achievable Throughput

= Number of bits sent per second

= 65535 B / 20 msec

= (65535 x 8 bits) / (20 x 10^{-3} sec)

= 26.214 Mbps

Now,

Line Efficiency

= Throughput / Bandwidth

= 26.214 Mbps / 1 Gbps

= 26.214×10^{-3}

= 0.026214

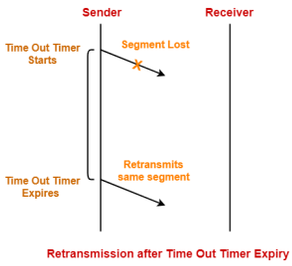
= 2.62%

Next Article- **TCP Congestion Control**

Get more notes and other study material of **Computer Networks**.

Watch video lectures by visiting our YouTube channel **LearnVidFun.**

Summary



Retransmission after Time Out Timer Expiry

Article Name	Transmission Control Protocol | Practice Problems
Description	Practice Problems based on Transmission Control Protocol. TCP Protocol is a transport layer protocol. TCP header format specifies various fields required for transmission. TCP congestion control policy is used for handling congestion.
Author	Akshay Singhal
Publisher Name	Gate Vidyalay
Publisher Logo	GateVidyalay A temple of learning

Liked this article? Share it with your friends and classmates now-

A proposed modification to TCP, the so-called **selective acknowledgment** [RFC 2018], allows a TCP receiver to acknowledge out-of-order segments selectively rather than just cumulatively acknowledging the last correctly received, in-order segment. When combined with selective retransmission—skipping the retransmission of segments that have already been selectively acknowledged by the receiver—TCP looks a lot like our generic SR protocol. Thus, TCP's error-recovery mechanism is probably best categorized as a hybrid of GBN and SR protocols.

3.5.5 Flow Control

Recall that the hosts on each side of a TCP connection set aside a receive buffer for the connection. When the TCP connection receives bytes that are correct and in sequence, it places the data in the receive buffer. The associated application process will read data from this buffer, but not necessarily at the instant the data arrives. Indeed, the receiving application may be busy with some other task and may not even attempt to read the data until long after it has arrived. If the application is relatively slow at reading the data, the sender can very easily overflow the connection's receive buffer by sending too much data too quickly.

TCP provides a **flow-control service** to its applications to eliminate the possibility of the sender overflowing the receiver's buffer. Flow control is thus a speed-matching service—matching the rate at which the sender is sending against the rate at which the receiving application is reading. As noted earlier, a TCP sender can also be throttled due to congestion within the IP network; this form of sender control is referred to as **congestion control**, a topic we will explore in detail in Sections 3.6 and 3.7. Even though the actions taken by flow and congestion control are similar (the throttling of the sender), they are obviously taken for very different reasons. Unfortunately, many authors use the terms interchangeably, and the savvy reader would be wise to distinguish between them. Let's now discuss how TCP provides its flow-control service. In order to see the forest for the trees, we suppose throughout this section that the TCP implementation is such that the TCP receiver discards out-of-order segments.

TCP provides flow control by having the *sender* maintain a variable called the **receive window**. Informally, the receive window is used to give the sender an idea of how much free buffer space is available at the receiver. Because TCP is full-duplex, the sender at each side of the connection maintains a distinct receive window. Let's investigate the receive window in the context of a file transfer. Suppose that Host A is sending a large file to Host B over a TCP connection. Host B allocates a receive buffer to this connection; denote its size by `RcvBuffer`. From time to time, the application process in Host B reads from the buffer. Define the following variables:

- `LastByteRead`: the number of the last byte in the data stream read from the buffer by the application process in B
- `LastByteRcvd`: the number of the last byte in the data stream that has arrived from the network and has been placed in the receive buffer at B

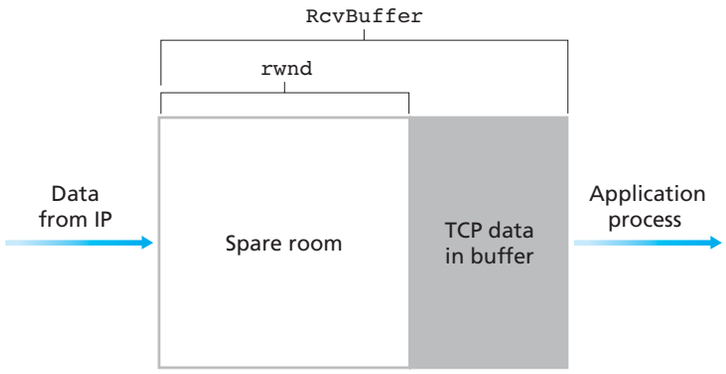


Figure 3.38 ♦ The receive window (`rwnd`) and the receive buffer (`RcvBuffer`)

Because TCP is not permitted to overflow the allocated buffer, we must have

$$\texttt{LastByteRcvd} - \texttt{LastByteRead} \leq \texttt{RcvBuffer}$$

The receive window, denoted `rwnd` is set to the amount of spare room in the buffer:

$$\texttt{rwnd} = \texttt{RcvBuffer} - [\texttt{LastByteRcvd} - \texttt{LastByteRead}]$$

Because the spare room changes with time, `rwnd` is dynamic. The variable `rwnd` is illustrated in Figure 3.38.

How does the connection use the variable `rwnd` to provide the flow-control service? Host B tells Host A how much spare room it has in the connection buffer by placing its current value of `rwnd` in the receive window field of every segment it sends to A. Initially, Host B sets `rwnd` = `RcvBuffer`. Note that to pull this off, Host B must keep track of several connection-specific variables.

Host A in turn keeps track of two variables, `LastByteSent` and `LastByteAcked`, which have obvious meanings. Note that the difference between these two variables, `LastByteSent` – `LastByteAcked`, is the amount of unacknowledged data that A has sent into the connection. By keeping the amount of unacknowledged data less than the value of `rwnd`, Host A is assured that it is not overflowing the receive buffer at Host B. Thus, Host A makes sure throughout the connection's life that

$$\texttt{LastByteSent} - \texttt{LastByteAcked} \leq \texttt{rwnd}$$

There is one minor technical problem with this scheme. To see this, suppose Host B's receive buffer becomes full so that `rwnd` = 0. After advertising `rwnd` = 0 to Host A, also suppose that B has *nothing* to send to A. Now consider what happens. As the application process at B empties the buffer, TCP does not send new segments with new `rwnd` values to Host A; indeed, TCP sends a segment to Host A only if it has data to send or if it has an acknowledgment to send. Therefore, Host A is never informed that some space has opened up in Host B's receive buffer—Host A is blocked and can transmit no more data! To solve this problem, the TCP specification requires Host A to continue to send segments with one data byte when B's receive window is zero. These segments will be acknowledged by the receiver. Eventually the buffer will begin to empty and the acknowledgments will contain a nonzero `rwnd` value.

The online site at http://www.awl.com/kurose-ross for this book provides an interactive Java applet that illustrates the operation of the TCP receive window.

Having described TCP's flow-control service, we briefly mention here that UDP does not provide flow control. To understand the issue, consider sending a series of UDP segments from a process on Host A to a process on Host B. For a typical UDP implementation, UDP will append the segments in a finite-sized buffer that "precedes" the corresponding socket (that is, the door to the process). The process reads one entire segment at a time from the buffer. If the process does not read the segments fast enough from the buffer, the buffer will overflow and segments will get dropped.

3.5.6 TCP Connection Management

In this subsection we take a closer look at how a TCP connection is established and torn down. Although this topic may not seem particularly thrilling, it is important because TCP connection establishment can significantly add to perceived delays (for example, when surfing the Web). Furthermore, many of the most common network attacks—including the incredibly popular SYN flood attack—exploit vulnerabilities in TCP connection management. Let's first take a look at how a TCP connection is established. Suppose a process running in one host (client) wants to initiate a connection with another process in another host (server). The client application process first informs the client TCP that it wants to establish a connection to a process in the server. The TCP in the client then proceeds to establish a TCP connection with the TCP in the server in the following manner:

- *Step 1.* The client-side TCP first sends a special TCP segment to the server-side TCP. This special segment contains no application-layer data. But one of the flag bits in the segment's header (see Figure 3.29), the SYN bit, is set to 1. For this reason, this special segment is referred to as a SYN segment. In addition, the client randomly chooses an initial sequence number (`client_isn`) and puts this number in the sequence number field of the initial TCP SYN segment. This segment is encapsulated within an IP datagram and sent to the server. There has

TCP vs UDP: What's the Difference?

What is TCP?

TCP/IP helps you to determine how a specific computer should be connected to the internet and how you can transmit data between them. It helps you to create a virtual network when multiple computer networks are connected.

TCP/IP stands for Transmission Control Protocol/ Internet Protocol. It is specifically designed as a model to offer highly reliable and end-to-end byte stream over an unreliable internetwork.

In this tutorial, you will learn:

TCP is best suited to be used for applications that require high reliability where timing is less of a concern.

World Wide Web (HTTP, HTTPS),
Secure Shell (SSH),
File Transfer Protocol (FTP),
Email (SMTP, IMAP/POP)

UDP is best suited for applications that require speed and efficiency.

VPN tunneling,
Streaming videos,
Online games,
Live broadcasts,
Domain Name System (DNS),
Voice over Internet Protocol (VoIP),
Trivial File Transfer Protocol (TFTP),
Broadcasting and Multicasting services

What is UDP?

UDP is a Datagram oriented protocol. It is used for broadcast and multicast type of network transmission. The full form of UDP is User Datagram Protocol (A datagram is a transfer unit associated with a packet-switched network.) The UDP protocol works almost similar to TCP, but it throws all the error-checking stuff out, all the back-and-forth communication and deliverability.

How TCP work?

A TCP connection is established with the help of three-way handshake. It is a process of initiating and acknowledging a connection. Once the connection is established, data transfer begins, and when the transmission process is finished, the connection is terminated by the closing of an established virtual circuit.

How UDP work?

UDP uses a simple transmission method without implied hand-shaking dialogues for ordering, reliability, or data integrity. UDP also assumes that error checking and correction is not important or performed in the application, to avoid the overhead of such processing at the network interface level. It is also compatible with packet broadcasts and multicasting.

Features of TCP

Here, are some important features of TCP

- Delivery Acknowledgements
- Re transmission
- Delays transmission when the network is congested
- Easy Error detection

Here, are some important feature of UDP:

- Supports bandwidth-intensive applications that tolerate packet loss
- Less delay
- It sends the bulk quantity of packets.
- Possibility of the Data loss
- Allows small transaction (DNS lookup)

Difference between TCP and UDP

Here, are the differences between TCP and UDP

(/images/1/011720_0714_TCPvsUDPWha1.png)

TCP	UDP
It is a connection-oriented protocol.	It is a connectionless protocol.
TCP reads data as streams of bytes, and the message is transmitted to segment boundaries.	UDP messages contain packets that were sent one by one. It also checks for integrity at the arrival time.
TCP messages make their way across the internet from one computer to another.	It is not connection-based, so one program can send lots of packets to another.
TCP rearranges data packets in the specific order.	UDP protocol has no fixed order because all packets are independent of each other.
The speed for TCP is slower.	UDP is faster as error recovery is not attempted.
Header size is 20 bytes	Header size is 8 bytes.
TCP is heavy-weight. TCP needs three packets to set up a socket connection before any user data can be sent.	UDP is lightweight. There are no tracking connections, ordering of messages, etc.
TCP does error checking and also makes error recovery.	UDP performs error checking, but it discards erroneous packets.
Acknowledgment segments	No Acknowledgment segments
Using handshake protocol like SYN, SYN-ACK, ACK	No handshake (so connectionless protocol)
TCP is reliable as it guarantees delivery of data to the destination router.	The delivery of data to the destination can't be guaranteed in UDP.
TCP offers extensive error checking mechanisms because it provides flow control and acknowledgment of data.	UDP has just a single error checking mechanism which is used for checksums.

Application of TCP

Here, are pros/benefits of using the TCP/IP model:

- It helps you to establish/set up a connection between different types of computers.
- Operates independently of the operating system
- Supports many routing-protocols.
- It enables the internetworking between the organizations.
- It can be operated independently.
- Supports several routing protocols.

- TCP can be used to establish a connection between two computers.

Application of UDP

- UDP method is largely used by time-sensitive applications as well as by servers that answer small queries from a larger client base.
- UDP is compatible with packet broadcasts for sending all over the network and for multicasting sending.
- It is also used in Domain Name System, Voice over IP, and online games.

Advantage of TCP

Here, are pros/benefits of TCP:

- It helps you to establish/set up a connection between different types of computers.
- It operates independently of the operating system.
- It supports many routing-protocols.
- It enables the internetworking between the organizations.
- TCP/IP model has a highly scalable client-server architecture.
- It can be operated independently.
- Supports several routing protocols.
- It can be used to establish a connection between two computers.

Advantage of UDP

Here are the pros/benefits of UDP:

- It never restricts you to a connection-based communication model; that's why startup latency in distributed applications is low.
- The recipient of UDP packets gets them unmanaged, which also includes block boundaries.
- Broadcast and multicast transmission are also available with UDP
- Data loss can be made
- Small transaction (DNS lookup)
- Bandwidth intensive app which endures packet loss

Disadvantages of TCP

Here, are disadvantage of using TCP:

- TCP never conclude a transmission without all data in motion being explicitly asked.
- You can't use for broadcast or multicast transmission.
- TCP has no block boundaries, so you need to create your own.
- TCP offers many features that you don't want. It may waste bandwidth, time, or effort.
- In this, model the transport layer does not guarantee delivery of packets.
- Replacing protocol in TCP/IP is not easy.
- It doesn't offer clear separation from its services, interfaces, and protocols.

Disadvantages of UDP

Here, are important cons/drawback of UDP:

- In UDP protocol, a packet may not be delivered or delivered twice. It may be delivered out of order, so you get no indication.
- Routers are quite careless with UDP, so they never retransmit it if it collides.
- UDP has no Congestion Control, and flow control, so implementation is the job of a user application.
- UDP mostly like to suffer from worse packet loss

When to use UDP and TCP?

- TCP is an ideal choice, and even it has associated overhead, Therefore, when most of the overhead is in the connection, your application stays connected for any length of time.
- UDP is ideal to use with multimedia like VoIP.
- Use TCP sockets when both client and server independently send packets at that time; an occasional delay is acceptable. (e.g., Online Poker).
- You should use user UDP if both client and server may separately send packets, and occasional delay is also not acceptable. (e.g., Multiplayer games).

KEY DIFFERENCES:

- TCP is a connection-oriented protocol, whereas UDP is a connectionless protocol.
- The speed for TCP is slower while the speed of UDP is faster
- TCP uses handshake protocol like SYN, SYN-ACK, ACK while UDP uses no handshake protocols
- TCP does error checking and also makes error recovery, on the other hand, UDP performs error checking, but it discards erroneous packets.
- TCP has acknowledgment segments, but UDP does not have any acknowledgment segment.
- TCP is heavy-weight, and UDP is lightweight.

Prev

Report a Bug

Next

YOU MIGHT LIKE:

ETHICAL HACKING

(/cybercrime-types-tools-examples.html) (/cybercrime-types-tools-examples.html)

What is Cybercrime? Types, Tools, Examples

(/cybercrime-types-tools-examples.html)

ETHICAL HACKING

(/ethical-hacking-interview-questions.html) (/ethical-hacking-interview-questions.html)

Top 25 Ethical Hacking Interview Questions & Answers

(/ethical-hacking-interview-questions.html)

ETHICAL HACKING

(/comptia-certification-guide.html) (/comptia-certification-guide.html)

CompTIA Certification Guide: Career Paths & Study Material

(/comptia-certification-guide.html)

ETHICAL HACKING

(/vulnerability-scanning-tools-websites-network.html) (/vulnerability-scanning-tools-websites-network.html)

13 BEST Vulnerability Assessment Scanners for Websites, Network

(/vulnerability-scanning-tools-websites-network.html)

ETHICAL HACKING

(/best-ethical-hacking-books.html) (/best-ethical-hacking-books.html)

16 BEST Ethical Hacking Books (2020 Update)

(/best-ethical-hacking-books.html)

ETHICAL HACKING

(/ethical-hacking-tutorial-pdf.html) (/ethical-hacking-tutorial-pdf.html)

Ethical Hacking Tutorial for Beginners PDF

(/ethical-hacking-tutorial-pdf.html)

Networking Tutorial

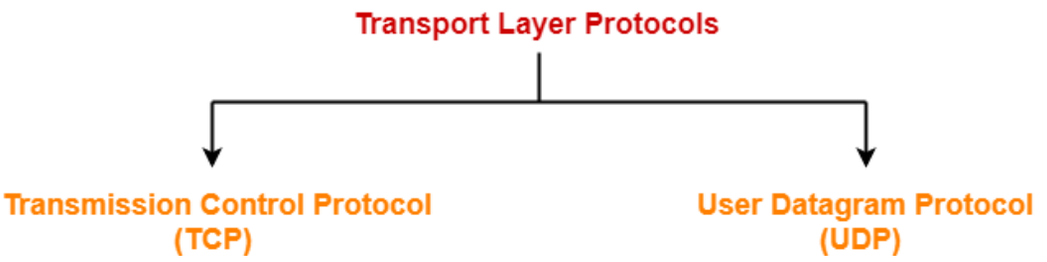
5) TCP 3-Way Handshake (/tcp-3-way-handshake.html)

UDP Protocol | UDP Header | UDP Header Format

Computer Networks

Transport Layer Protocols-

There are mainly two transport layer protocols that are used on the Internet-



1. Transmission Control Protocol (TCP)
2. User Datagram Protocol (UDP)

In this article, we will discuss about User Datagram Protocol (UDP).

Learn about **Transmission Control Protocol**.

UDP Protocol-

- UDP is short for **User Datagram Protocol**.
- It is the simplest transport layer protocol.
- It has been designed to send data packets over the Internet.
- It simply takes the datagram from the network layer, attaches its header and sends it to the user.

Characteristics of UDP-

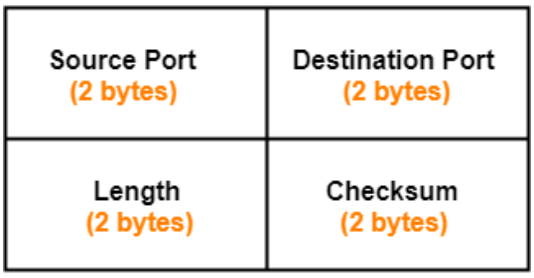
- It is a connectionless protocol.
- It is a stateless protocol.
- It is an unreliable protocol.
- It is a fast protocol.
- It offers the minimal transport service.
- It is almost a null protocol.
- It does not guarantee in order delivery.
- It does not provide congestion control mechanism.
- It is a good protocol for data flowing in one direction.

Need of UDP-

- TCP proves to be an overhead for certain kinds of applications.
- The **Connection Establishment** Phase, **Connection Termination** Phase etc of TCP are time consuming.
- To avoid this overhead, certain applications which require fast speed and less overhead use UDP.

UDP Header-

The following diagram represents the UDP Header Format-



UDP Header

1. Source Port-

- Source Port is a 16 bit field.
- It identifies the port of the sending application.

2. Destination Port-

- Destination Port is a 16 bit field.
- It identifies the port of the receiving application.

3. Length-

- Length is a 16 bit field.
- It identifies the combined length of UDP Header and Encapsulated data.

Length = Length of UDP Header + Length of encapsulated data

4. Checksum-

- **Checksum** is a 16 bit field used for error control.
- It is calculated on UDP Header, encapsulated data and IP pseudo header.
- Checksum calculation is not mandatory in UDP.

Applications Using UDP-

Following applications use UDP-

- Applications which require one response for one request use UDP. Example- **DNS**.
- Routing Protocols like RIP and OSPF use UDP because they have very small amount of data to be transmitted.
- Trivial **File Transfer Protocol** (TFTP) uses UDP to send very small sized files.

- Broadcasting and multicasting applications use UDP.
- Streaming applications like multimedia, video conferencing etc use UDP since they require speed over reliability.
- Real time applications like chatting and online games use UDP.
- Management protocols like SNMP (Simple Network Management Protocol) use UDP.
- Bootp / DHCP uses UDP.
- Other protocols that use UDP are- Kerberos, Network Time Protocol (NTP), Network News Protocol (NNP), Quote of the day protocol etc.

Important Notes-

Note-01:

Size of UDP Header= 8 bytes

- Unlike TCP header, the size of UDP header is fixed.
- This is because in UDP header, all the fields are of definite size.
- Size of UDP Header = Sum of the size of all the fields = 8 bytes.

Note-02:

UDP is almost a null protocol.

This is because-

- UDP provides very limited services.
- The only services it provides are checksumming of data and multiplexing by port number.

Note-03:

UDP is an unreliable protocol.

This is because-

- UDP does not guarantee the delivery of datagram to its respective user (application).
- The lost datagrams are not retransmitted by UDP.

Note-04:

Checksum calculation is not mandatory in UDP.

This is because-

- UDP is already an unreliable protocol and error checking does not make much sense.
- Also, time is saved and transmission becomes faster by avoiding to calculate it.

It may be noted-

- To disable the checksum, the field value is set to all 0's.
- If the computed checksum is zero, the field value is set to all 1's.

Note-05:

> UDP does not guarantee in order delivery.

This is because-

- UDP allows out of order delivery to ensure better performance.
- If some data is lost on the way, it does not call for retransmission and keeps transmitting data.

Note-06:

> Application layer can perform some tasks through UDP.

Application layer can do the following tasks through UDP-

1. Trace Route
2. Record Route
3. Time stamp

When required,

- Application layer conveys to the UDP which conveys to the IP datagram.
- UDP acts like a messenger between the application layer and the IP datagram.

Also Read- TCP Header | IPv4 Header

PRACTICE PROBLEMS BASED ON UDP HEADER-

Problem-01:

Which field is optional in UDP?

1. Checksum
2. Destination port
3. Length
4. None

Solution-

- Checksum calculation is not mandatory in UDP.
- Thus, Option (A) is correct.

Problem-02:

The pseudo header of IP is used in-

1. Only TCP
2. Only UDP
3. Both TCP and UDP
4. None

Solution-

- IP Pseudo header is used in both TCP and UDP while calculating checksum.
- Thus, Option (C) is correct.

Problem-03:

Broadcasting applications like WHOD (who daemon on UNIX) uses what transport layer protocol?

1. TCP
2. UDP
3. Either TCP or UDP
4. IGMP

Solution-

- Broadcasting and multicasting applications use UDP.
- Thus, Option (B) is correct.

To gain better understanding about UDP Header,

Watch this Video Lecture

Next Article- Application Layer Protocols

Networking Devices.

AL, T.L } Gateway.

N.L — Router (multiport)

D.L.L — Bridge / Switch

Ph.L — Repeater / Hub (multiport)

Hub, Repeater ⇒

Neither breaks collision domain nor broadcast domain

Switch, Bridge ⇒

Breaks only collision domain

Router ⇒ Breaks both.

Application Layer Protocols

→

DNS	DN → IP	SMTP	Email
HTTP	Web pages.	POP	Email retrieving.
FTP	files transfer		

* Domain Name Service (DNS). (Port 53) TL

or Domain Name System.

→ Purpose – Converting domain name of the websites to their numerical IP address. (as IP addresses aren't static) (easy to remember DN)

→ Kinds of domain –

i) Generic domain – .com (commercial, .edu (educational), .org (non-profit orgs), .net (similar to commercial), .mil (military).

ii) Country domain – .in, .us, .uk.

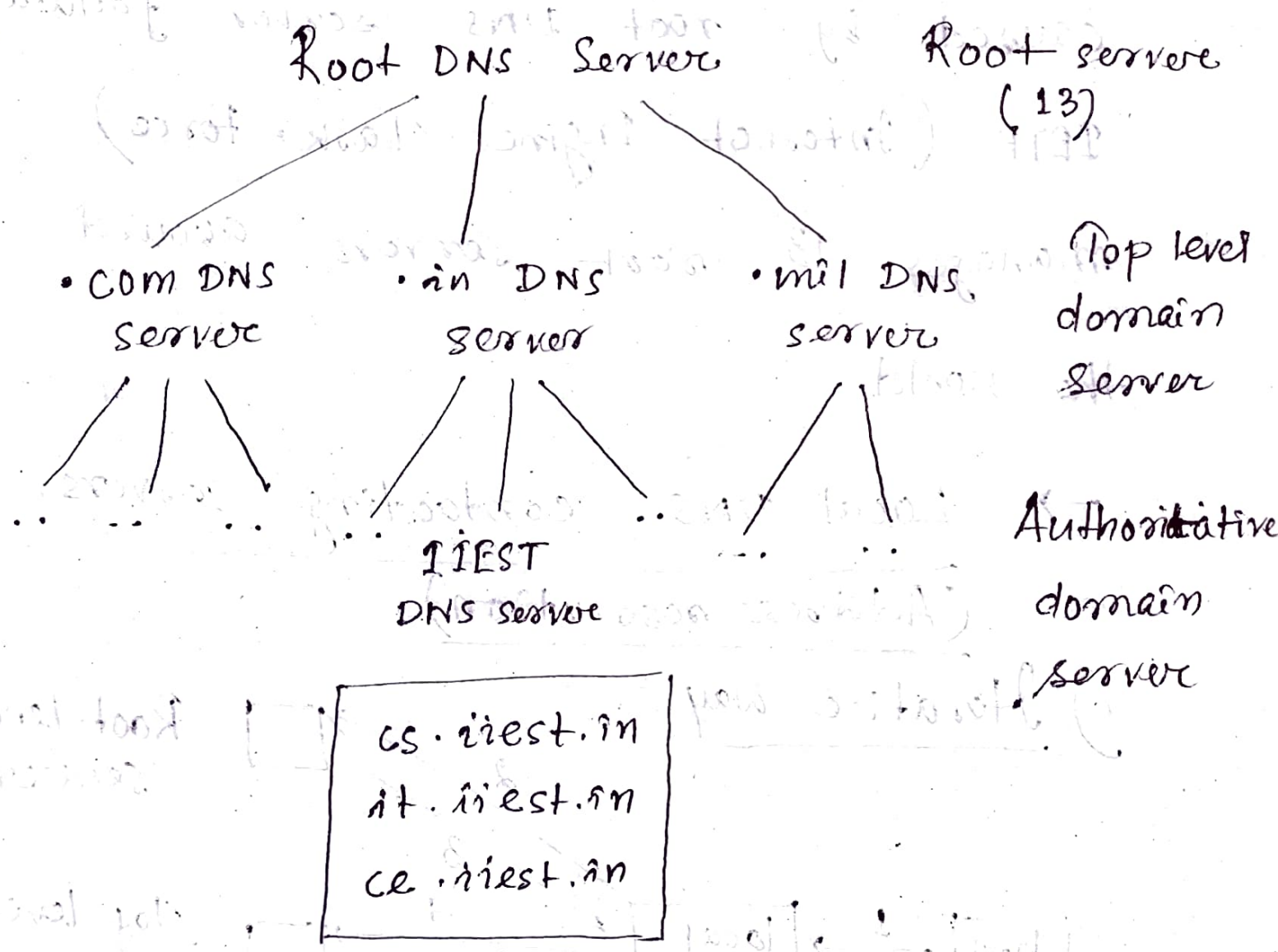
iii) Inverse domain – IP to domain name mapping. DNS can provide both the mapping – DN → IP,

IP → DN. eg. to find ip address of google.com -

$ nslookup google.com

→ DNS is also used for load balancing (redistributing load).

→ Organisation of domain.



DNS record will store domain names, ip addresses, their validity, time to live.

ISP uses local DNS cache to find the IPs of common websites fast.

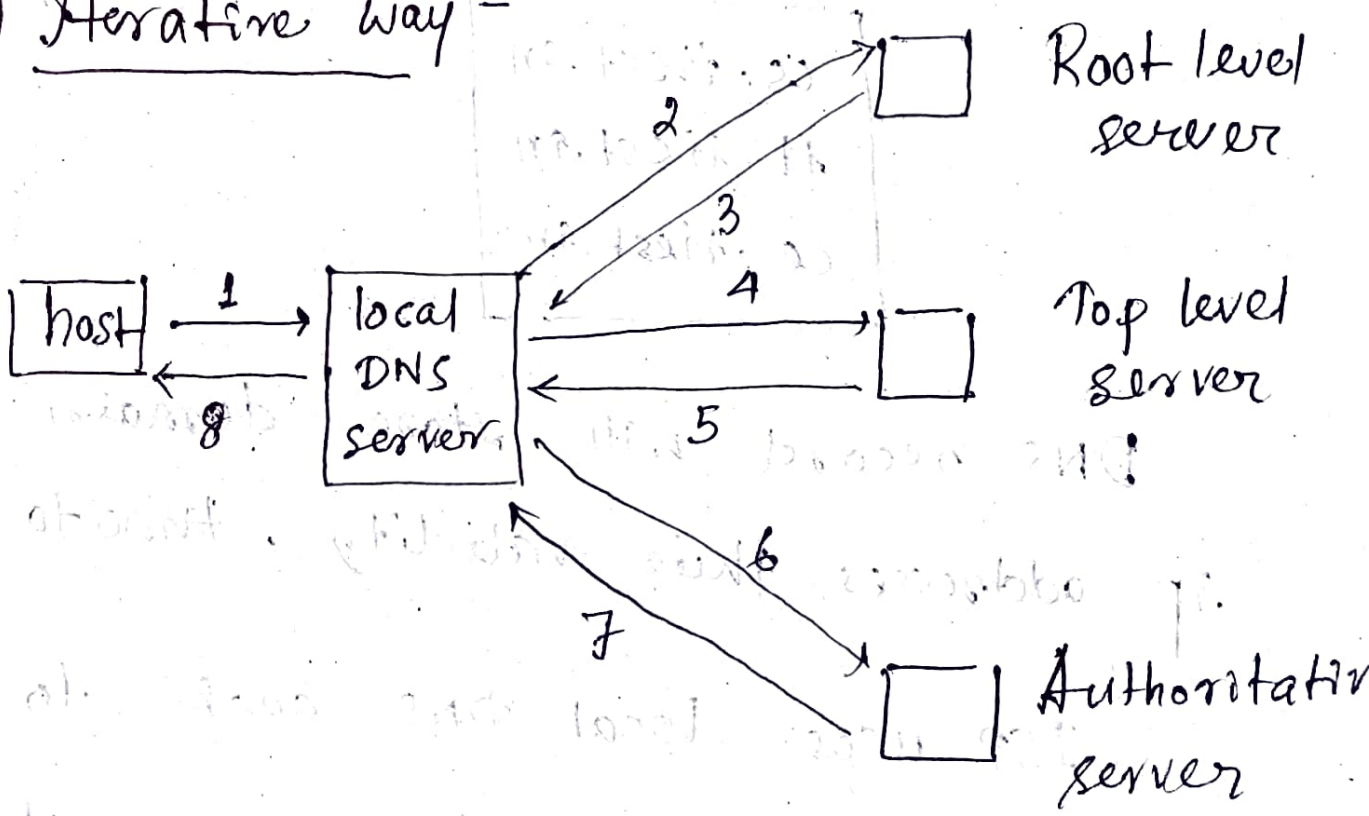
→ DNS overhead – Delay to convert domain name into IP address. More delay for the first time. To reduce delay next time, it is saved in computer log.

→ To prevent problems that may be caused by root DNS server failure, IETF (Internet Engine Task force) manages 13 root servers around the world.

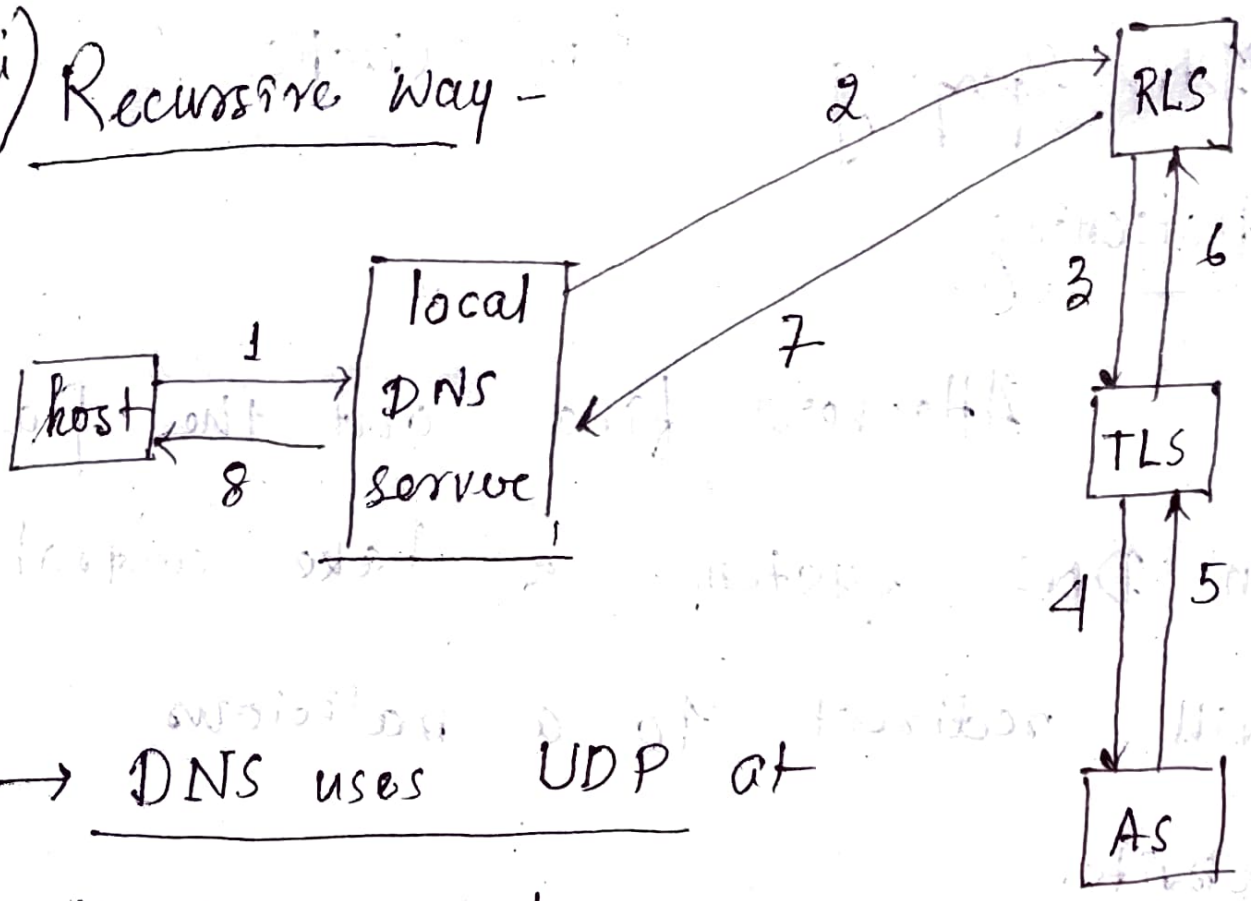
→ Local DNS contacting servers.

(Address resolution)

i) Iterative way –



ii) Recursive way -



→ DNS uses UDP at the transport layer.

Reasons -

i) Much faster than TCP.

ii) DNS requests are very small.

iii) In UDP, reliability can be added by using timeouts & resend at the application layer.

→ DNS is connection less protocol.

DNS is non-persistent.

→ DNS is a stateless protocol.

→ DNS can translate DN into an IP address. Also, it can m IP addr into a DN.

→ DNS Spoofing or DNS Cache poisoning

Attackers find out the flaws in DNS system & take control & will redirect to a malicious website.

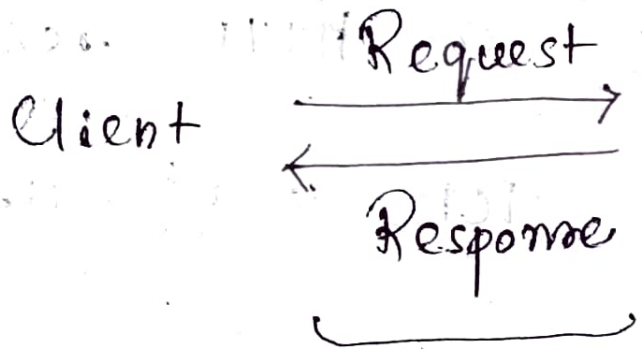
We get a wrong entry or IP of the requested site.

* Hyper Text Transfer Protocol (HTTP). (Port 80)

→ Used for retrieval of data from websites throughout the internet. Works on the top of TCP/IP suite of protocols.

→ HTTP uses a client-server model where - web browser is the client. Client communicates with the web

Server hosting the website.



Whenever a client requests some info to the website server, the browser sents a request message to the HTTP server for the requested objects. Then, HTTP opens a conn^n b/w the client & server through TCP. HTTP sends request to the server which collects the requested data. HTTP sends the response with the objects back to the client. Then, HTTP closes the conn^n.

→ HTTP connections :

1) Non-persistent : Used for serving exactly 1 request & sending 1 response. Here, HTTP server

closes the TCP conn^n automatically after sending a HTTP response. New separate TCP conn^n used for each object.

HTTP 1.0 supports it by default.

eg: Suppose a request has been made for a HTML page that contains 10 images (objects). With non-p conn^n, all 11 objects (1 pg + 10 img) will be sent one by one. For getting each object, a new separate conn^n will be opened & closed.

ii) Persistent: Used for serving multiple responses. Closes TCP conn^n only when it's not used for a certain configurable amount of time. A single TCP conn^n is used for sending multiple

* HTTP is connectionless. Client & server knows about each other during current request & response only.

objects one after the other.

HTTP 1.1 supports persistent conn^ns by default.

eg. All the 11 objects will be sent one after the other using a single TCP connection.

→ HTTP uses TCP at TLayer.

Ensures retransmission of lost packets. (It's imp to know - any service which does not use TCP should have the inbuilt facility for providing reliability. That's why HTTP uses TCP.)

→ Persistent conn^ns improve performance by 20%.

→ HTTP is an in-band protocol. Passes the control data (commands) & main data over the same conn^n. No high priority to cmds

→ HTTP is a stateless protocol.

HTTP server does not maintain any state. Forgets about the client after sending the response. It treats every new request independently. HTTP closes the connⁿ automatically after generating response. Ensures no client can engage with the web server for a long time.

→ Methods used by HTTP

i) Head : Get the header of the webpage (metadata).

ii) Get } Used with forms
iii) Post }

iv) Delete - Deleting object

v) Put - Uploading "

vi) Trace - Tracing servers which are sending data

vii) Options

viii) Connect : HTTPS uses it for authentication.

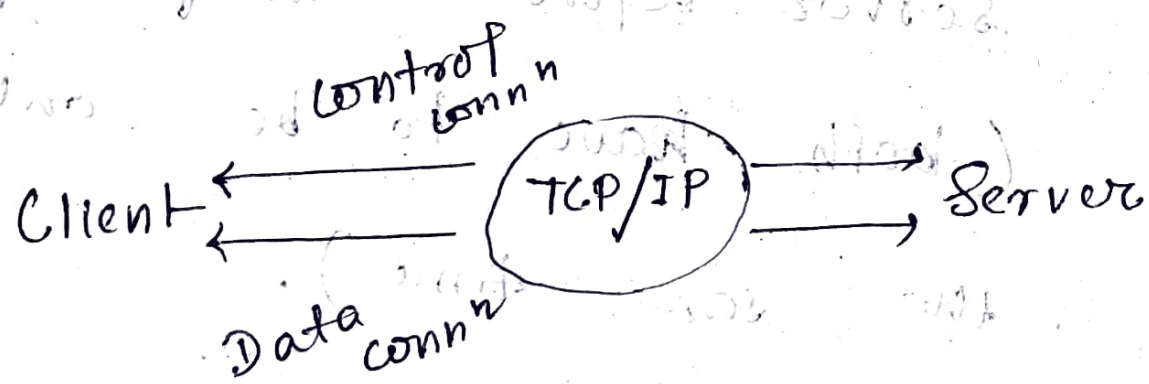
* File Transfer Protocol (FTP)

Port 21 - control conn^n (persistent)

Port 20 - data conn^n (non-p (data transfer)

→ Used for exchanging files over the internet.

✓ → FTP uses 2 TCP connections between the client & server — (One conn^n used for transferring data ; other conn^n is used for transferring control info.)



✓ → FTP uses TCP at T layer.

✓ → FTP uses persistent TCP for control conn^n. Uses non-p conn^n for data conn^n.

→ FTP is connection oriented protocol.

→ FTP is an out of band protocol as data & command info flow over different conn^ns.

→ FTP is a stateful protocol.

→ FTP can transfer one file at a time.

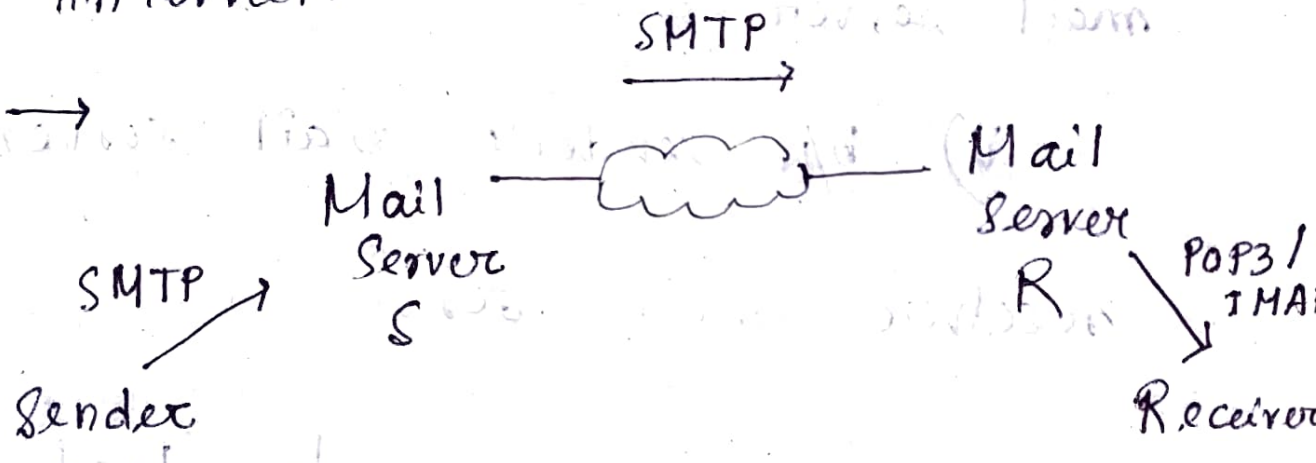
✓ → FTP requires reliability & this is provided by TCP.

✗ → Emails can't be sent using FTP as FTP requires the conn^n establishment b/w the client & server before transferring files (both have to be online at the same time).

* Simple Mail Transfer Protocol (SMTP)

(Port 25)

→ Used for sending emails over the internet.



✓ → SMTP servers are always on a listening mode.

Client initiates a TCP conn^n with the SMTP server. SMTP server listens for a conn^n & initiates a conn^n on that port. Conn^n established Client informs the SMTP server that it would like to send a mail. Client sends mail to its mail server. Client's mail server uses DNS to get the IP address of receiver's mail server. (SMTP transfers mail from sender's mail server to receiver's mail server.)

✓ While sending the mail, SMTP is used 2 times —

— i) between the sender & sender's mail server.

— ii) b/w sender's mail server & receiver mail server.

→ To receive or download the email,

another protocol is needed b/w the receiver's mail server & the receiver. Most used protocols are POP3 & IMAP.

Post office protocol.

Internet message access protocol

→ SMTP is a push protocol (pushing emails).

→ SMTP uses TCP at Tlayer.

✓ Uses persistent TCP connⁿ, so it can send multiple emails at once.

→ SMTP is an inband, stateless, connection oriented protocol.

✓ → SMTP is pure text based protocol. Only handles the messages containing 7 bit ASCII text. Can't transfer other types of data like image, video etc. Can't transfer the text data of other languages like Chinese, Japanese etc (as they are represented in 8 bit codes).

✓ - MIME (Multipurpose Internet Email Extensions) is an extension to the internet email protocol. Enables to send & receive graphics, audio etc.

✓ → Sender & receiver can't run SMTP b/w their machines. Because, machines can't always be on. Mail server receives the mail on behalf on its client & manages the mail box of the client.

✓ → SMTP doesn't require authentication. It allows anyone to send emails to anyone; SMTP auth has been provided for authentication.

* Post Office Protocol. (POP)

(Port 110)

→ Message access protocol.

Enables client to receive or download the emails from their machine remote mail server. POP v3 is most popular.

→ POP is a pull protocol.

→ POP uses TCP at T layer.

POP uses persistent TCP connⁿ.

POP is connⁿ oriented.

(POP is in band & stateful

until the mail is downloaded or

well as stateless across sessions.)

• IMAP (Port 143)

Pull protocol, uses TCP,

persistent TCP connⁿ, connⁿ oriented,

inband, stateful, distributes mail

boxes across multiple servers.

	DNS	HTTP	SMTP	POP	FTP
Stateful / Stateless	Stateless	Stateless	Stateless	Stateful	Stateful
Transport Protocol Used	UDP	TCP	TCP	TCP	TCP
Connectionless / Connection Oriented	Connectionless	Connectionless	Connection Oriented	Connection Oriented	Connection Oriented
Persistent / Non-persistent	Non-persistent	HTTP 1.0 is non-persistent. HTTP 1.1 is persistent.	Persistent	Persistent	Control connection is persistent. Data connection is non-persistent.
Port Number Used	53	80	25	110	20 for data connection. 21 for control connection.
In band / Out-of-band	In band	In band	In band	In band	Out-of-band

Next Article- Cryptography and Network Security

Get more notes and other study material of **Computer Networks**.

Watch video lectures by visiting our YouTube channel **LearnVidFun.**

Summary

IPv6 & WiFi

* Need for IPv6.

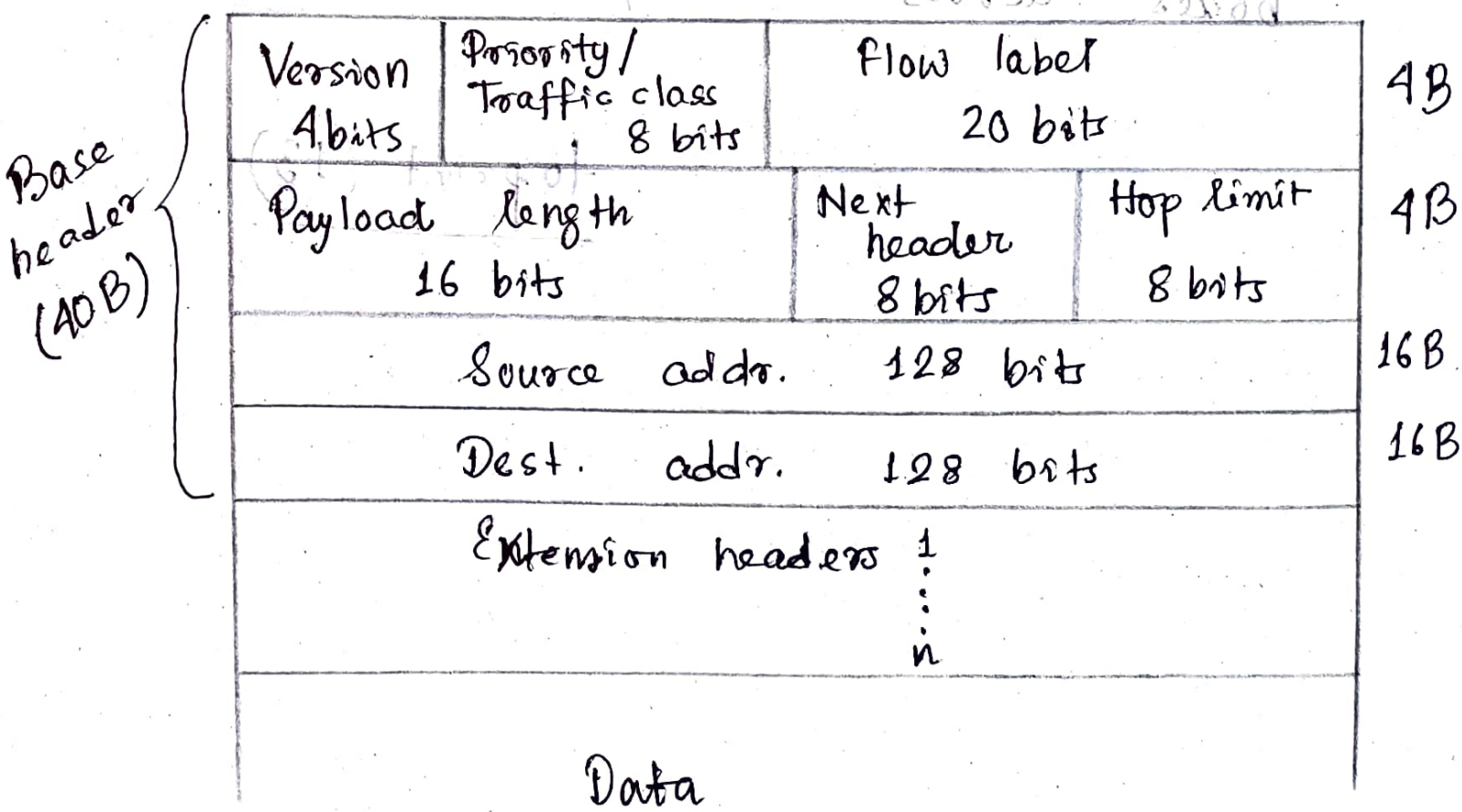
1. IPv4 address depletion
2. Real time audio / video transmission
3. Encryption
4. Authentication
5. Fast processing
6. Additional functionalities.
   (Extended headers).

→ IPv6 sometimes called IPng (next generation)

→ IPv6 has 2^{128} address space.

128 bits in IPv6 address.

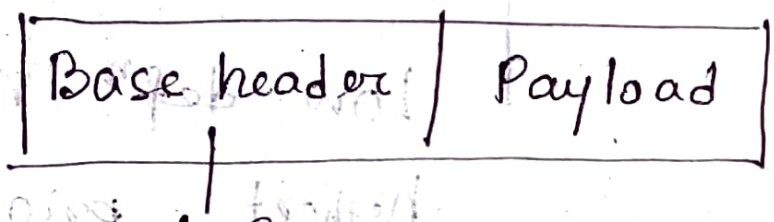
* IPv6 header format.



- Version : Version of IP which contains bit sequence 0110 (for IPv6), 0100 (for IPv4)

- In IPv6, header size is fixed. So, we don't need header length field.

IPv6 packet :



- Priority / Traffic class (RFC)

(8-bits acc. to RFC)

Traffic can be classified into -

a) Congestion controlled traffic :

The protocols that are sending the data observes there will be congestion & they will adjust their sizes (window size) appropriately.

eg. TCP.

b) Uncontrolled traffic : No facility as of congestion controlled traffic.

eg. audio - video traffic (live streaming)

→ First 4 bits of traffic field are given for setting priority.

4 bits - (0-15)

(0-7) congestion controlled

(8-15) uncontrolled

- & least, 15 highest prio
- Low defn video given highest prio, high defn video given lowest prio.

Uncontrolled traffic is mainly used for audio/video. So, we give higher prio to uncontrolled data traffic.

Priority	Meaning.
0	No specific traffic
1	Background data (news tranomission)
2	Unattended data traffic (email data)
3	Reserved
4	Attend bulk data traffic FTP, HTTP
5	Reserved.

6 . . Interactive traffic
(Remote login, putty, ssh)

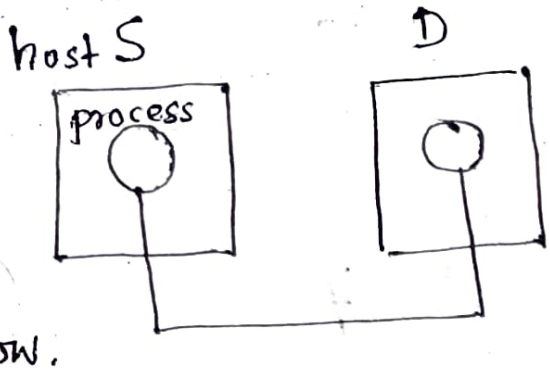
7 Control traffic
(highest priority among all)
SNMP (Simple N/W mgmt protocol), routing data

- The priorities are set by the upper layers (above N/W layer) & IPv6 should provide an interface to the upper layers by which priorities can be set.

- Source & routers can change priority.

- "Dest" should not assume the packet received is of same priority as set by source as the nodes (routers, gateways) inbetween may change priority.

– Flow label

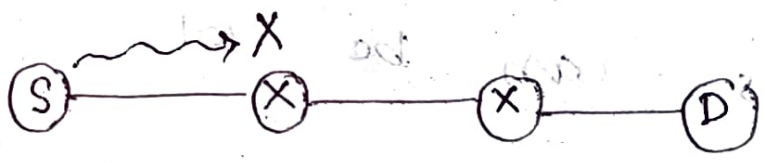


Stream of packets that go from source to destination is called flow.

All the packets that belong to same flow have the same requirements. –

i) source address
ii) destination address
iii) priority
iv) extension headers

– these are same for all the packets in the flow.



In IPV4 packets may be dropped if buffer, CPU or bandwidth is not enough. So, whenever we send data through flow, before sending inform all the intermediate routers to allocate resources for a flow. We can inform the routers in 2 ways –

1. Control packets

   a) Resource reservation protocol

   b) Real time transport n

2. Extension headers.

For all the packets in the flow we use a label called flow label & routers will form flow table to store info about what to allocate for packets of certain flow label.

Flow table –

Source addr.	Flow label	Information
A	10	BW, buffer, CPU

How much to allocate & what

one entry for a combination of source addr. & flow label is enough for all packets.

Hashing on the combination of source addr. & flow label is used on the flow table.

· Between a source & destination there can be more than one flow. For different processes, we have different flows.

· All the packets need not have flow label. In such case A packet can have flow label = 0.

· Routers using IPv4 or not having the facility, may choose to ignore flow labels.

· Each router have to be informed about flow life time; that how much the router have to provide this flow label facility.

· Every flow should have different flow label. Flow labels should be randomly generated.

· We should not send any flow until the previously sent flows get expired (in case of the

connection get interrupted).

Or in case of interruption, to prevent same flow labels, we can store the flow labels in permanent storage.

- Payload length.

Entire part present after base header is payload.

✓ Payload contains -

extension header, upper layer packets (like TCP header, TCP data).

- Next Header

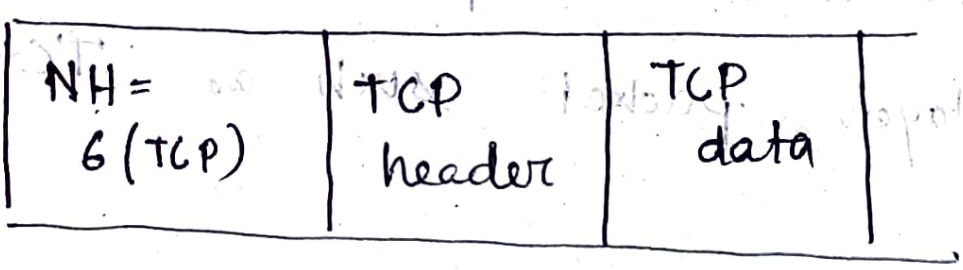
Indicates type of of extension header (if present) immediately following the IPv6 header. In some cases, it indicates the protocols contained within upper layer packet such as TCP, UDP.

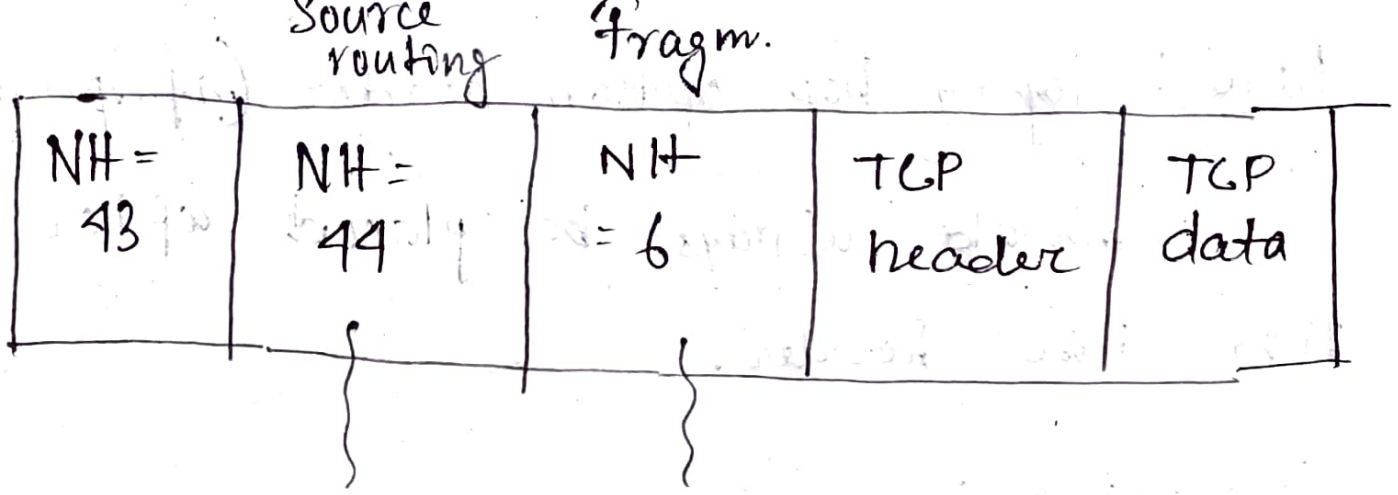
When no extension header is used, TCP header will follow the IPv6 header.

Next header codes

Code	next header
0	hop by hop option
2	ICMP
6	TCP
17	UDP
43	Source routing
44	Fragmentation
50	Encrypted security payload
51	Authentication
59	Null (no next header)
60	Destination option

eg. IPV6





Extension headers.

— Ordering of extension headers

IPv6 packet may contain one or more extension headers but they should be present in their recommended order:

Order	Header type	Next header code.
1	Basic IPv6 header	-
2	Hop by hop options	0
3	Destination options (with routing option)	60
4	Routing header	43
5	Fragment "	44
6	Authentication header	51
7	Encapsulation security payload	50
8	Destination options	60
9	Mobility header	135
	No next header	59
Upper layer	TCP	6
	UDP	17
	ICMPv6	58

Rule : Hop by hop option header (if present) should always be placed after IPv6 base header.

Conventions :

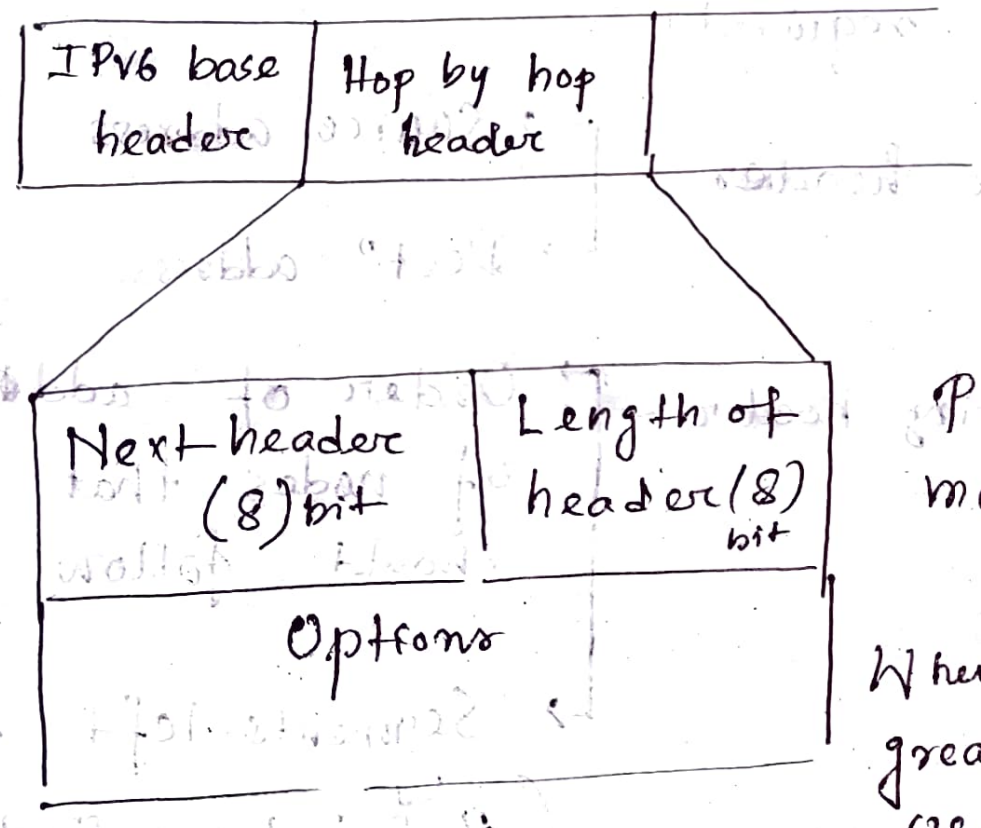
i) Any ext. header can apper at most once except destination header because dest^n header is present 2 times in the list itself.

ii) If dest^n header is present before routing header then it will be examined by all intermediate nodes specified in routing header.

iii) If dest^n header is present just above upper layer then it will be examined only by destination node.

Hop by hop options header

If it is present as extension header, packet should be examined by all the intermediate routers & the destination if any function are specified in the extension header that must be carried out.



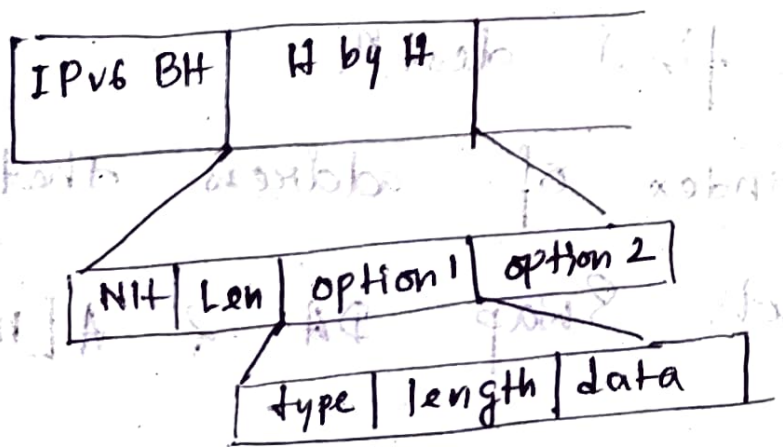
payload = 16b
max = $2^{16} - 1$
= 65,535

When payload greater than this, we use Jumbo packet

Options :

i) Jumbo packet

ii) Pad i : Options field if < 8 Byte the we use padding. pad 2 ⇒ pad 2 bytes.



option field

i) option type

ii) option length

iii) option data.

• Routing extension header

Source takes the control over the path in which its packet should be transmitted. (Source routing).

IPv6 packet contains the base header & the routing extension header (when required) –

base header → Source address
            → Dest^n address

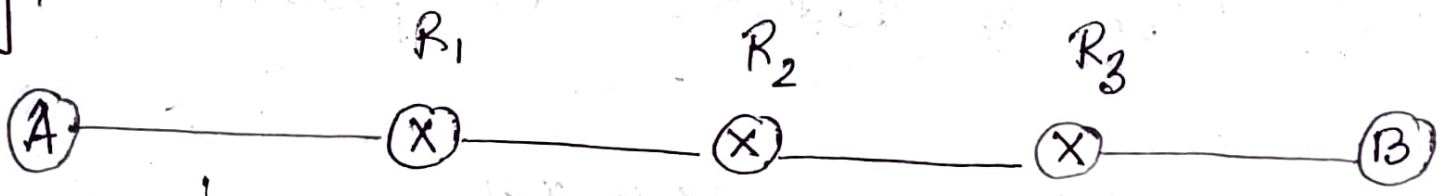
routing header → Order of addresses of nodes that packet should follow
               → Segments left field. (SL)
(# of intermediate nodes that are left to be visited before packet reaches final dest^n).

* n - no of ext. headers (addresses) that are present in ext header

i - no. of nodes that must be visited before we reach final dest^n.

n-i gives the index of address that has to be visited next. Swap DA & A[n-i].

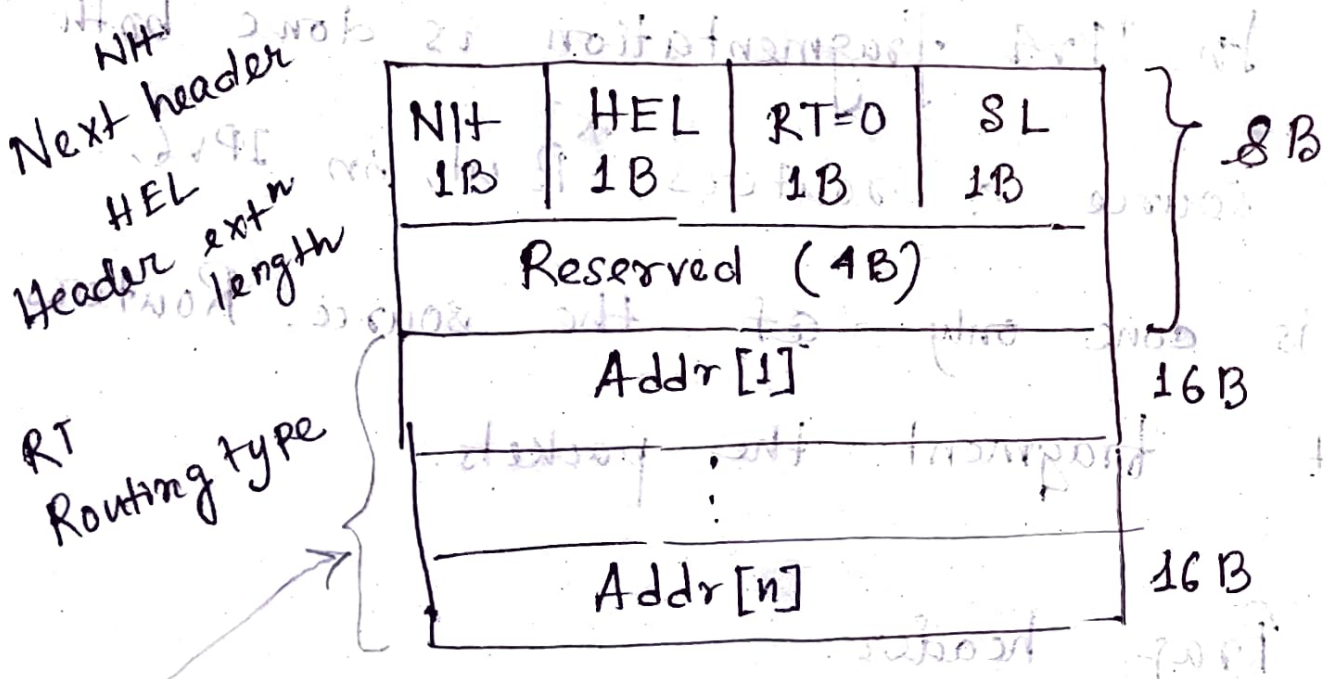
Eg.



SA = A	SA = A	SA = A	SA = A	SA = A
DA = R_1	DA = ~~R_1~~ R_2	DA = ~~R_2~~ R_3	DA = ~~R_3~~ B	DA = B
SL = 3	SL = ~~3~~ 2	SL = ~~2~~ 1	SL = ~~1~~ 0	SL = 0
A[1] = R_2	A[1] = ~~R_2~~ R_1	A[1] = R_1	A[1] = R_1	A[1] = R_1
A[2] = R_3	A[2] = R_3	A[2] = ~~R_3~~ R_2	A[2] = R_2	A[2] = R_2
A[3] = B	A[3] = B	A[3] = B	A[3] = ~~B~~ R_3	A[3] = R_3
	n-i = 3-2 = 1	n-i = 3-1 = 2	n-i = 3-0 = 3	

recordroute alike

~ Routing ext^n header –



HEL represents the size of Routing header in multiples of 8; (excluding the 8B part NH - HEL - RT - SL).

~ HEL → # of addresses

Scaling factor 8

Say HEL = 8

$\frac{16 \times n}{8} = 8 \Rightarrow n = 4$

Each address contribute 2 to the HEL.

# of addresses = $\frac{8}{2} = 4$.

~ If n addresses are present

$$\frac{8B}{x} + \frac{16B \times n}{\frac{16B \times n}{8}} \quad (2n)B$$

x exclude

We place this value in the HEL field.

~ # IPv6 addresses = $\frac{HEL}{2}$.

— Fragmentation extension header

In IPv4 fragmentation is done both at source & routers. But, in IPv6, it is done only at the source. Routers don't fragment the packets.

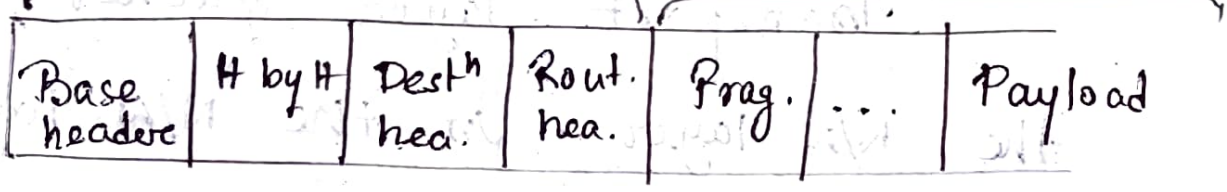
Frag. header.

NH 8b	Reserved = 0 8b	Frag. offset 13b	Reserved 2b	More fragments 1b
Identification 32b				

Identification: same for all fragments of one packet

IPV6 packet can be visualised to be consisting of 2 parts —

unfragmentable & fragmentable



We don't fragment the unfragmentable part.

— Need for security at N/W layer

1. Confidentiality.
2. Message authentication / Message integrity
   - Receiver should be able to find if the message is changed on the way.
3. Source authentication
   - Whether message is from the genuine source.
4. Replay attack prevention
   - copies the same packet over & again.

Protocol used to provide these features — IP security (IPsec) - collection of protocols.

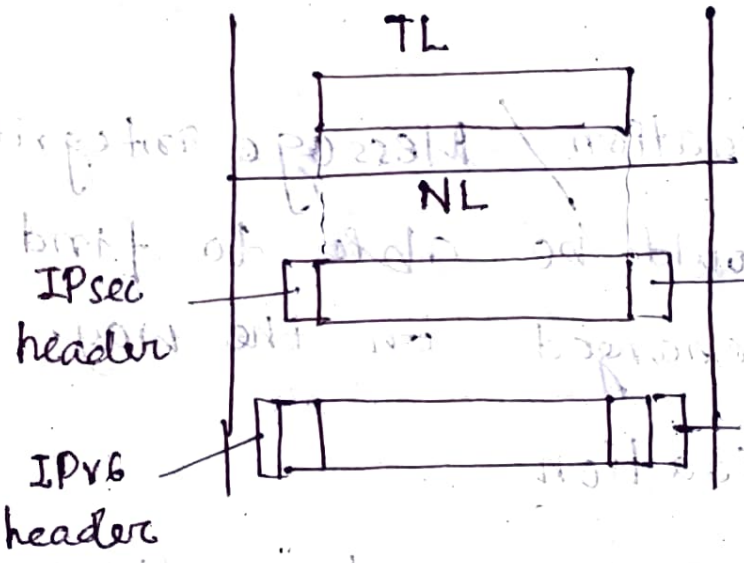
IPsec can be used with IPv6 as well as with IPv4.

IPsec
- Authentication header protocol (AH)
- Encapsulating security protocols (ESP)

- Modes of IP security

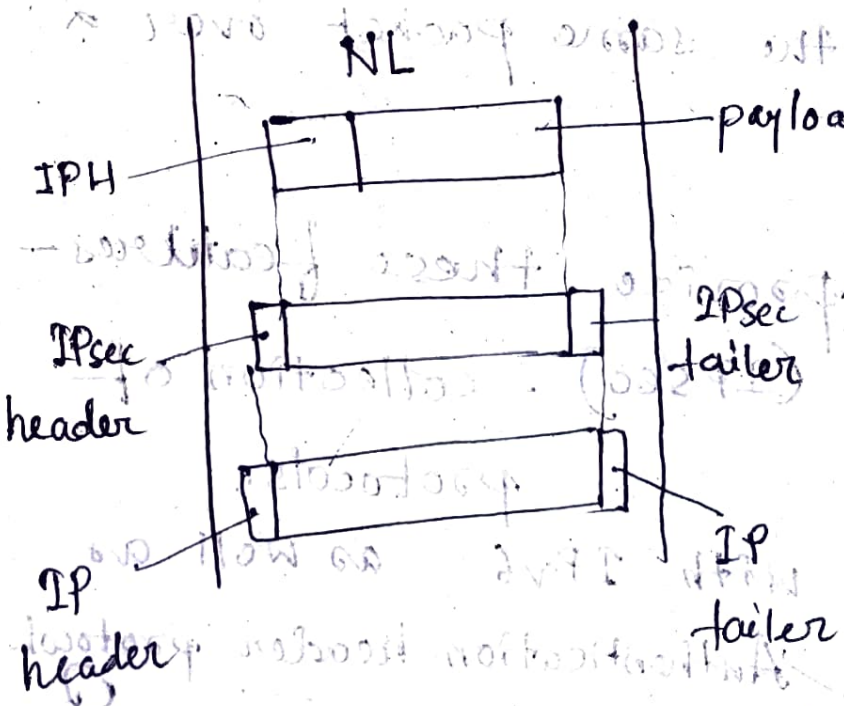
a) Transport mode :

Transport layer packet passed to the N/W layer. In the N/W layer, packet is first received by IPsec & it adds IPsec header & IPsec tailer & again it's passed to the IP in N/W layer. IP adds IP header & tailer.



Modification done only on the payload but not on the transport/N/W layer headers.

b) Tunnel mode



Modification on both the header & payload.

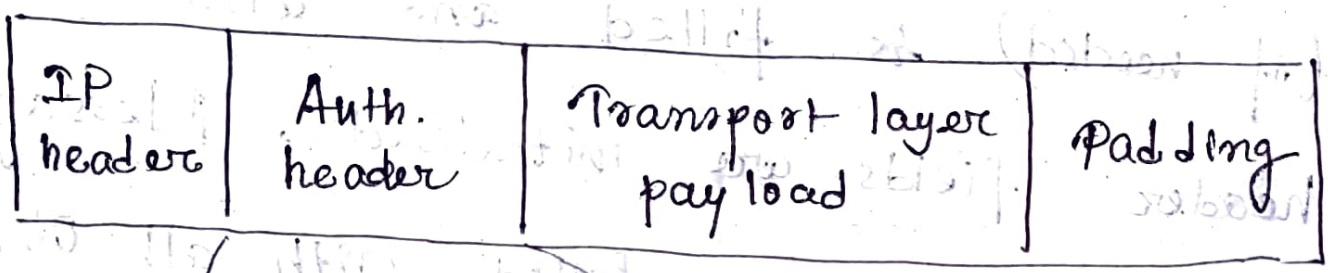
Tunneling as an IP packe is kept inside another IP packet & 2 IP headers need not be sam

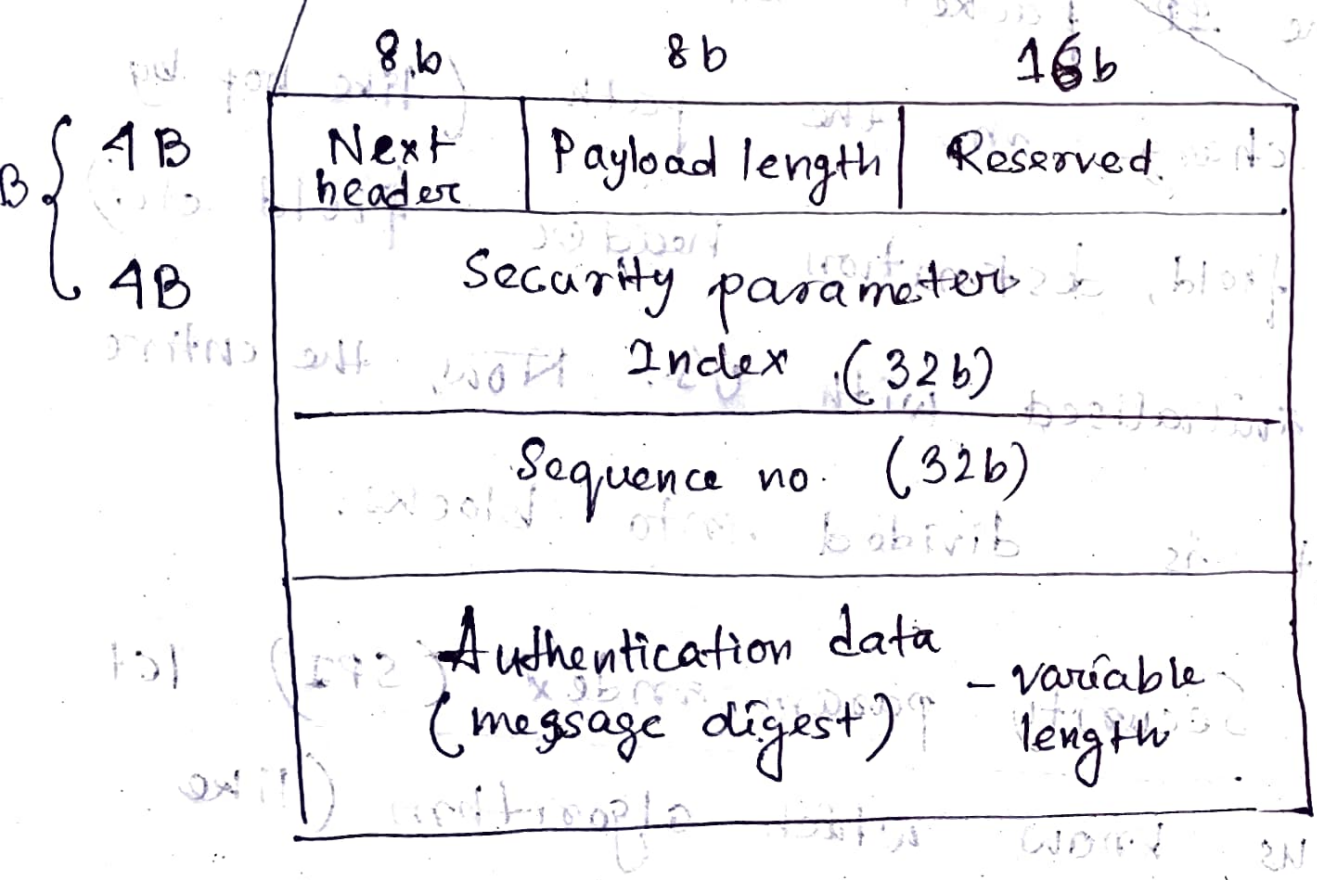
In IPv6, tunneling is used in B/W routers.

(Transport mode only, ✗ tunnel mode)

— Authentication header

- Message integrity
- Source authentication
- Replay attack protection.





Payload length → Auth. header size.

x

$\Downarrow$

$(x+2) \times 4$ Bytes.

First 8B part. $\leftarrow \frac{8}{4} = 2$

Payload length calculation ⇒

$$x = \left(\frac{\text{Auth. header size}}{4} - 2 \right) \text{ Bytes}$$

(at sender & receiver)

Message digest is generated to check message integrity on the receiver's side.

Working ~

First IP header field, TL payload, padding (if needed) is filled and also auth. header fields are initialised. Message digest field is filled with all 0's.

In the IP packet all the headers that may change in the path (like hop by hop field, destination header field etc.) are initialised with 0's. Now, the entire packet is divided into blocks.

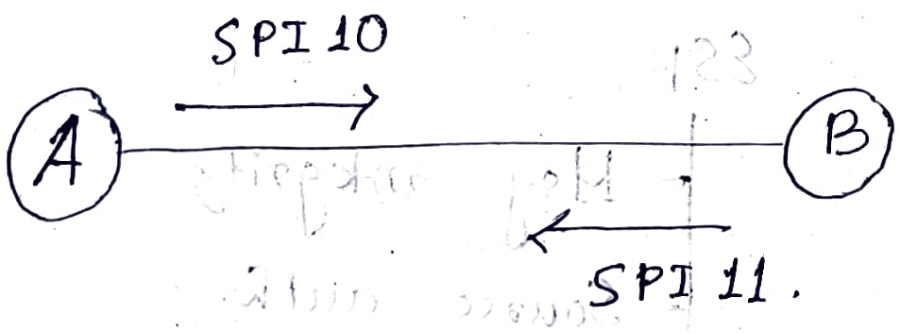
Security param. index (SPI) let the us know which algorithm (like SHA-1 hashing) and key to use to generate the message digest. Then, using security association database (SADB) (where there are field SPI, algo, key; for source to destn and destn to source. SPI changes)

we form the message digest and attach it with the packet to be checked at the receiver's side.

Sequence number field is used to prevent replay attack (copying same packet & sending them over & again). When all seq. no.s are exhausted, we build a new connection between nodes and restart sending packets.

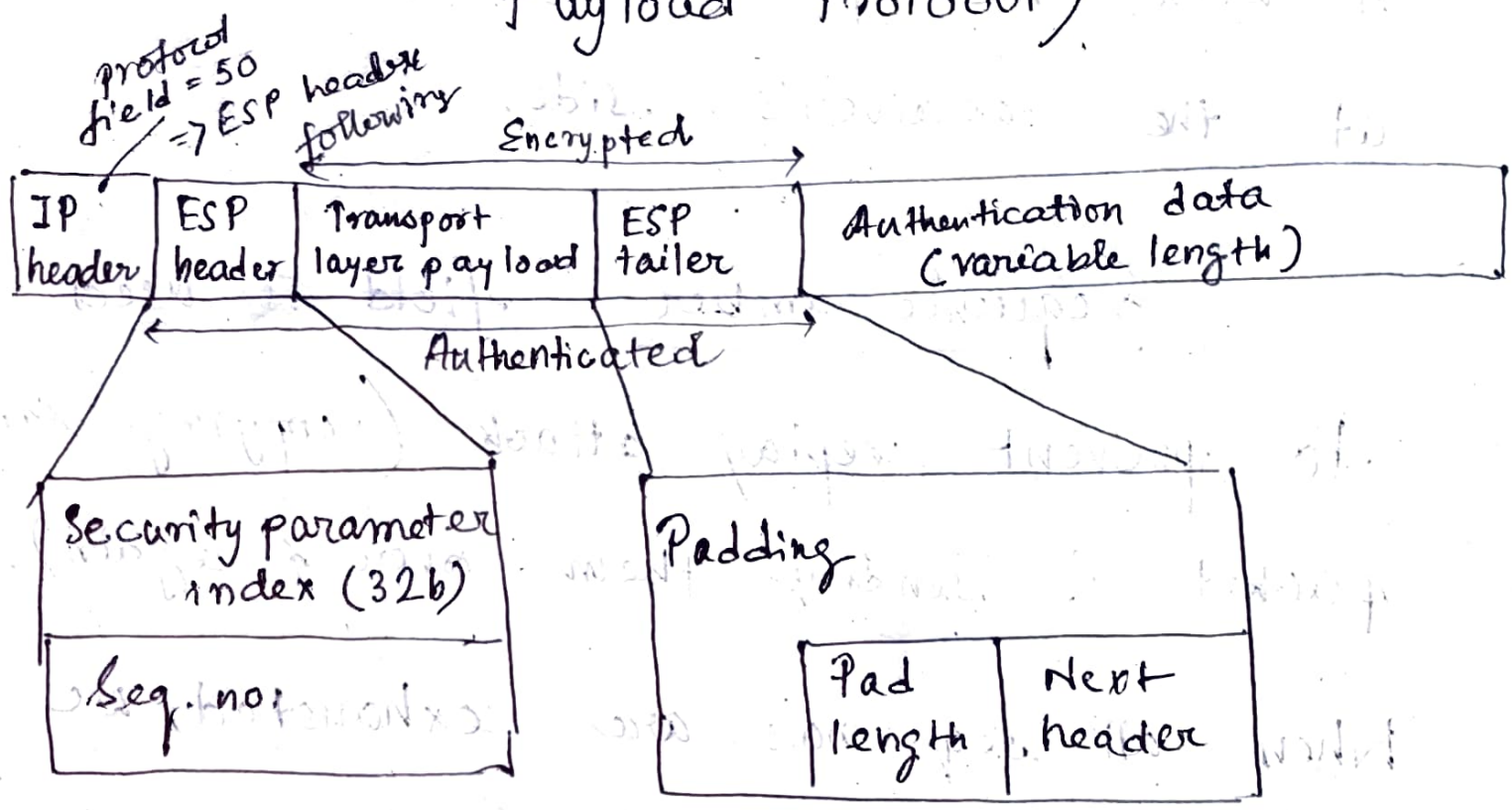
SADB

SPI	Algo	Key
10	SHA-1	K1
11	SHA-1	K2



(Transport mode)

— ESP (Encapsulation Security Payload Protocol)



ESP provides all facilities as of Auth. header along with encapsulation (In AH the MD was not encrypted).

DES algo used for encryption. (Data Encryption Standard)

SHA-1 used for authentication.

ESP
- Msg integrity
- Source auth
- Replay attack
- Encryption.

* IPV6 addresses.

- Unicast
- Multicast
- Anycast

✓ Broadcasting not defined in IPV6.

Unicast addresses: Identifies a single N/W interface. A packet sent to unicast address is delivered to the interface identified by that address.

Multicast address.: Used by multiple hosts, called as group, acquires a multicast destination address. These hosts need not be geographically together. If any packet is sent to this multicast address, it will be distributed to all interfaces corresponding to that multicast address.

✓ **Anycast address**: Assigned to a group of interfaces. Any packet sent to anycast address will be delivered to only one member interface (nearest host possible).

→ Colon hexadecimal representation

- 8 2 Bytes segment in IPv6 address HEX

ABCD : EF01 : 2345 : 6789 : ABCD : EF01 : 2345 : 6789

2 Bytes

- If 4 0's in a segment, put one 0 for them.
- Delete preceding 0's
- If we have continuous 0's in between —

ABCD :: 1234

↓

all 0's except 1st & 8th segment.

- ABCD :: 1234 :0:0: 1234.

↓

3 all 0's segment (calculate from ↑)

or

ABCD:0:0:0: 1234 :: 1234

↑ 2 all 0's segment

HEX BIN 16

A

→ Types of IPv6 addresses

Prefix	Type.
0000 0000	Reserved
0000 0001	UA
0000 001	UA
0000 010	UA
0000 011	UA
0000 1	UA
0001	Reserved
001	Reserved
010	Provider based unicast address
011	Unassigned (UA)
001	Geographical based unicast address.

Prefix	Type
101	UA
110	UA
1110	UA
1111 0	UA
1111 10	UA
1111 110	UA
1111 1110	UA
1111 1110 10	Link local addresses
1111 1110 11	Site local addresses
1111 1111	Multicast address.

- Unicast address can be of 2 types - provider based and geographical based. (ISP)

- Provider based unicast address

3b	5b	n bits	56-n bits	64 bits
010	Registry ID	Provider ID	Subscriber ID	Intra subscriber

Registry

10000	Multiregional (IANA)
01000	RIPE NCC
11000	INTERNIC
00100	APNIC

• Geography based unicast address

3b.	45 bits	16 bits	64 bits
001	Global routing prefix	Subnet ID	Interface ID

Latitude, Longitude.

instead of host id as of IPv4.
(We provide IPv6 address to interface).

- Multicasting

1111 1111	Flag	Scope	Group ID

Flag	
0000	Permanent
0001	Transient

Scope	
0000	Reserved
0001	Node local
0010	Link local
0101	Site local
1000	Organisational
1110	Global
1111	Reserved

— Anycast

An IPv6 anycast address is an address that can be assigned to more than one interfaces (typically different devices). Multiple devices can have the same anycast address. A packet sent to an anycast address is routed to the nearest interface having that address, according to the router's routing table.

✓No special prefix for an IPv6 anycast address.

→ Some special addresses.

Unspecified

0000 0000	120 0's
8b	120b

Loopback

0000 0000	119 0's	1

IPv4 compatible

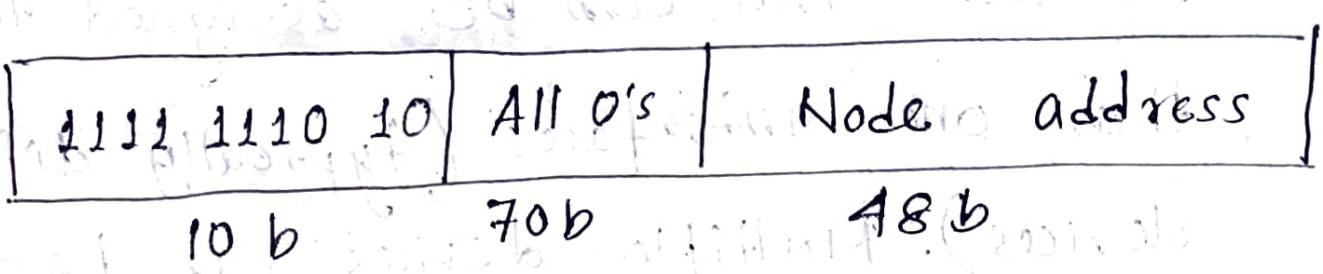
0000 0000	88 0's	IPv4 address
		32b

IPv4 mapped

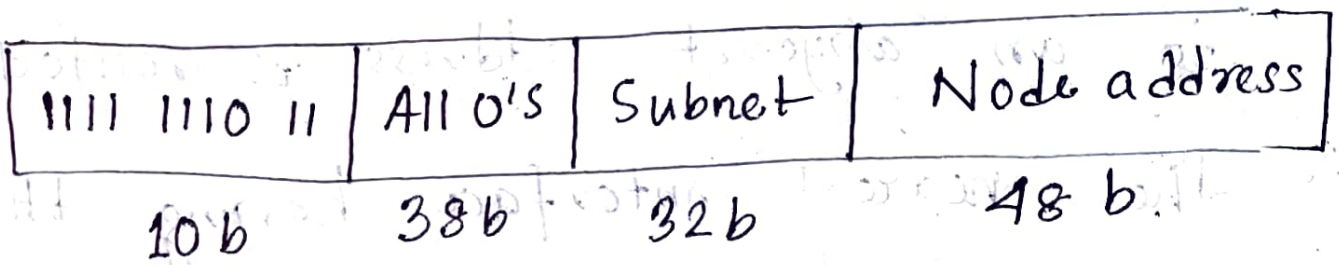
0000 0000	72 0's	16 1's	IPv4 addr
			32b

- Local addresses

Link local (LAN, - no routers)



Site local (have routers)



- IPv4 options vs IPv6 extension header

1. No operation & end of options in IPv4 is replaced by PAD 1 and PAD N options in IPv6.

2. No record route option in IPv6. (can be done w routing ext. h

3. No timestamp " "

4. The source route option is called source route extension in IPv6.

5. Fragmentation fields of IPv4 base header are moved to fragm. ext. header in IPv6.

6. The authentication extension header is new in IPv6.

7. The encrypted security payload extension header is new in IPv6.

→

IPv4	vs.	IPv6
1. Addr. space 2^{32}		1. 2^{128}
2. Len. of header 20-60B		2. 40B
3. No. of header field 12.		3. 8
4. Checksum used.		4. Field eliminated.
5. IPsec is optional.		5. Mandatory
6. Does not identify packet flow for QoS handling.		6. Contains flow label field that specifies packet flow for QoS handlin
7. Fragm" done by sending host & router both.		7. Done by sender only.

IPv4	IPv6
8. Clients have to approach DHCP whenever connect to a N/W.	8. Clients do not have to approach any server as they are given permanent address.
9. Connectionless.	9. Can be made connn oriented using flow label (generally connnle
10. Broadcast Yes Multicast Yes	10. NO Yes
11. Stateful.	11. Stateless.

IPv4 vs IPv6: What's the Difference?

What is IP?

An Internet Protocol address is also known as IP address. It is a numerical label which assigned to each device connected to a computer network which uses the IP for communication.

IP address act as an identifier for a specific machine on a particular network. The IP address is also called IP number and internet address. IP address specifies the technical format of the addressing and packets scheme. Most networks combine IP with a TCP (Transmission Control Protocol). It also allows developing a virtual connection between a destination and a source.

What is IPv4?

IPv4 was the first version of IP. It was deployed for production in the ARPANET in 1983. Today it is most widely used IP version. It is used to identify devices on a network using an addressing system.

The IPv4 uses a 32-bit address scheme allowing to store 2^32 addresses which is more than 4 billion addresses. Till date, it is considered the primary Internet Protocol and carries 94% of Internet traffic.

What is IPv6?

It is the most recent version of the Internet Protocol. Internet Engineer Taskforce initiated it in early 1994. The design and development of that suite is now called IPv6.

This new IP address version is being deployed to fulfill the need for more Internet addresses. It was aimed to resolve issues which are associated with IPv4. With 128-bit address space, it allows 340 undecillion unique address space. IPv6 also called IPng (Internet Protocol next generation).

Features of IPv4

- Connectionless Protocol
- Allow creating a simple virtual communication layer over diversified devices
- It requires less memory, and ease of remembering addresses
- Already supported protocol by millions of devices
- Offers video libraries and conferences

Features of IPv6

- Hierarchical addressing and routing infrastructure
- Stateful and Stateless configuration
- Support for quality of service (QoS)
- An ideal protocol for neighboring node interaction

Example: 127.255.255.255

Example: *2001:0db8:85a3:0000:0000:8a2e:0370:7334*

(/images/1/053018_0657_IPv4vsIPv6W1.png)

Difference Between IPv4 and IPv6 Addresses

IPv4 & IPv6 are both IP addresses that are binary numbers. IPv4 is 32 bit binary number while IPv6 is 128 bit binary number address. IPv4 address are separated by periods while IPv6 address are separated by colons.

Both are used to identify machines connected to a network. In principle, they are the same, but they are different in how they work.

Basis for differences	**IPv4**	**IPv6**
Size of IP address	IPv4 is a 32-Bit IP Address.	IPv6 is 128 Bit IP Address.
Addressing method	IPv4 is a numeric address, and its binary bits are separated by a dot (.)	IPv6 is an alphanumeric address whose binary bits are separated by a colon (:). It also contains hexadecimal.
Number of header fields	12	8
Length of header filed	20	40
Checksum	Has checksum fields	Does not have checksum fields
Example	12.244.233.165	2001:0db8:0000:0000:0000:ff00:0042:7879
Type of Addresses	Unicast, broadcast, and multicast.	Unicast, multicast, and anycast.
Number of classes	IPv4 offers five different classes of IP Address. Class A to E.	lPv6 allows storing an unlimited number of IP Address.
Configuration	You have to configure a newly installed system before it can communicate with other systems.	In IPv6, the configuration is optional, depending upon on functions needed.
VLSM support	IPv4 support VLSM (Virtual Length Subnet Mask).	IPv6 does not offer support for VLSM.
Fragmentation	Fragmentation is done by sending and forwarding routes.	Fragmentation is done by the sender.
Routing Information Protocol (RIP)	RIP is a routing protocol supported by the routed daemon.	RIP does not support IPv6. It uses static routes.
Network Configuration	Networks need to be configured either manually or with DHCP. IPv4 had several overlays to handle Internet growth, which require more maintenance efforts.	IPv6 support autoconfiguration capabilities.

Basis for differences	IPv4	IPv6
Best feature	Widespread use of NAT (Network address translation) devices which allows single NAT address can mask thousands of non-routable addresses, making end-to-end integrity achievable.	It allows direct addressing because of vast address Space.
Address Mask	Use for the designated network from host portion.	Not used.
SNMP	SNMP is a protocol used for system management.	SNMP does not support IPv6.
Mobility & Interoperability	Relatively constrained network topologies to which move restrict mobility and interoperability capabilities.	IPv6 provides interoperability and mobility capabilities which are embedded in network devices.
Security	Security is dependent on applications - IPv4 was not designed with security in mind.	IPSec(Internet Protocol Security) is built into the IPv6 protocol, usable with a proper key infrastructure.
Packet size	Packet size 576 bytes required, fragmentation optional	1208 bytes required without fragmentation
Packet fragmentation	Allows from routers and sending host	Sending hosts only
Packet header	Does not identify packet flow for QoS handling which includes checksum options.	Packet head contains Flow Label field that specifies packet flow for QoS handling
DNS records	Address (A) records, maps hostnames	Address (AAAA) records, maps hostnames
Address configuration	Manual or via DHCP	Stateless address autoconfiguration using Internet Control Message Protocol version 6 (ICMPv6) or DHCPv6
IP to MAC resolution	Broadcast ARP	Multicast Neighbour Solicitation
Local subnet Group management	Internet Group Management Protocol GMP)	Multicast Listener Discovery (MLD)
Optional Fields	Has Optional Fields	Does not have optional fields. But Extension headers are available.
IPSec	Internet Protocol Security (IPSec) concerning network security is optional	Internet Protocol Security (IPSec) Concerning network security is mandatory
Dynamic host configuration Server	Clients have approach DHCS (Dynamic Host Configuration server) whenever they want to connect to a network.	A Client does not have to approach any such server as they are given permanent addresses.
Mapping	Uses ARP(Address Resolution Protocol) to map to MAC address	Uses NDP(Neighbour Discovery Protocol) to map to MAC address

Basis for differences	IPv4	IPv6
Combability with mobile devices	IPv4 address uses the dot-decimal notation. That's why it is not suitable for mobile networks.	IPv6 address is represented in hexadecimal, colon- separated notation. IPv6 is better suited to mobile networks.

IPv4 and IPv6 cannot communicate with other but can exist together on the same network. This is known as **Dual Stack.**

KEY DIFFERENCE

- IPv4 is 32-Bit IP address whereas IPv6 is a 128-Bit IP address.
- IPv4 is a numeric addressing method whereas IPv6 is an alphanumeric addressing method.
- IPv4 binary bits are separated by a dot(.) whereas IPv6 binary bits are separated by a colon(:).
- IPv4 offers 12 header fields whereas IPv6 offers 8 header fields.
- IPv4 supports broadcast whereas IPv6 doesn't support broadcast.
- IPv4 has checksum fields while IPv6 doesn't have checksum fields
- IPv4 supports VLSM (Virtual Length Subnet Mask) whereas IPv6 doesn't support VLSM.
- IPv4 uses ARP (Address Resolution Protocol) to map to MAC address whereas IPv6 uses NDP (Neighbour Discovery Protocol) to map to MAC address.

Prev (/introduction-ccna.html) Report a Bug Next (/difference-http-vs-https.html)

CCNA Tutorial

WiFi Basics

WiFi is the marketing name for IEEE standard 802.11. It is a standard for both Level 1 (physical) and Level 2 (data link) of a wireless data transmission protocol.

802.11 defines many Level 1 variants. 802.11g is the new, high-speed Level 1 standard, versus 802.11b, the first generation WiFi. The radio frequency band is around 3 Gigahertz, same as a microwave oven. As you can see in your WiFi setup, there are channels within the band. These channels overlap so that at most 3 channels can be in use at the same time.

WiFi Concepts

- There are two general types of WiFi transmission: DCF (Distributed Coordination Function) and PCF (Point Coordination Function). DCF is *ethernet in the air*. It employs a very similar packet structure, and many of the same concepts. There are two problems that make wireless different then wired.

- The hidden substation problem.
- High error rate.

These problems demand that a DCF WiFI be a CSMA/CA network (Collision Avoidance) rather than a CSMA/CD network (Collision Detect). The result are the following protocol elements,

- Positive Acknowledgement. Every packet sent is positively acknowledged by the receiver. The next packet is not sent until receiving a positive acknowledgement for the previous packet.
- Channel clearning. A transmission begins with a RTS (Request to Send) and the destination or receiver responds with a CTS (Clear to Send). Then the data packets flow. For the channel is cleared by these two messages. All that hear the CTS squelch. This helps with the hidden substation problem.
- Channel reservation. Each packet has a NAV (Network Allocation Vector) containing a number X. The channel is reserved to the correspondents (the sender and receiver of this packet) for an additonal X milliseconds after this packet. Once you have the channel, you can hold it with the NAV. The last ACK contains NAV zero, to immediately release the channel.

As for PCF, it is a polling, token-ring type communcation system. We shall skip the details. It isn't much used.

Network topologies, bridging

A group of corresponding stations is called a BSS (Basic Service Set). The BSS can be organized in several ways.

- Independent BSS, or ad hoc. The network is only the members of the BSS, they talk between themselves directly, they self-organize, there is not central autority.
- Infrastructure BSS. The BSS is organized around an Access Point which can bridge traffic out the BSS onto a distribution network. Members of the BSS talk to the AP only. You can often understand a domain by answering the question "who will hear a broadcast". A BSS (data link layer) is defined by who will hear a broadcast from the AP (but not by a station which is not an AP, because of the hidden station problem!).
- ESS (Extended Service Set). A bunch of BSS's connected by a distribution network. The distribution network connects the Access Points. WiFi doesn't specify the protocol that builds ESS's.

We will skip ad hoc and concentrate on Infrastructure.

Since this is ethernet on the air, each transmitter/receiver has a 48 bit MAC consistent with the ethernet address. That is, same address space, OUI's, and so on. An AP is a bridge between wired and unwired ethernet, so it has two interfaces. As the leader of a BSS, it gives names the BSS by the ethernet address of its air interface. This is called the BSSID.

An ESS is given a name, called the SSID (Service Set ID). This is the thing you type into your network configuration to join a WiFi network.

A packet on the air will have three addresses, source, destination and BSSID (access point address, essentially). The AP takes traffic it receives off the air that has its address and drops it onto its wired interface, eliting its own address. That is, on the wired side, only the source and destination addresses are seen. The address of the AP is not used, either its wired or unwired addresses.

When an AP sends a packet into the air, it uses the source and destination address of the packet it is bridging as found, and adds its own wireless address as the BSSID. A wired station sending to a wireless station uses the wireless stations's ethernet address just as if it were a wired station. The AP picks the packet off the wire, carries it across to its wireless interface, inserting its wireless address as the extra, third address, and sends it out to the destination.

Association and so on

Definitions:

BSS
: Basic Service Set. A bunch of machines forming a cell.

ESS
: Extended Service Set. Using WiFi beyond a BSS, gluing together several BSS

BSSID
: A 48 bit identifier for a BSS. If an infrastructure BSS, it is the MAC of the 802.11 side of the Acess Point. Else the local bit is set and a 48-bit identifier is randomly selected.

SSID
: Service set Identifier. An character string identifier for a ESS.

NAV
: Network Access Vector. A time slot reservation, in microseconds.

RTS/CTS
: Request To Send, Clear To Send. Reservation mechanism. Source,

Quick description

1. WiFi is standard 802.11, with various letters added. The standard includes a large number of physical variants.
2. The link levels can either be an Independent BSS (IBSS) or an infrastructure BSS. An infra. BSS can be contention based or coordinated (Point Coordination Function).
3. Infrastructure BSS uses AP (access points) and a distribution medium, e.g. ethernet (802.2), either the AP acting as bridges. In a simple example, the packet has three addresses, the two "transparent" endpoints and the BBSID, which is the way-point for the packet between air and wire.
4. In an infra. BSS, the AP either Beacons or responds to a probe from a node. An association (after possible authentication) occurs, so that traffic from the host is bridged onto the distribution by the AP.
5. Gratuitous ARPs open up switches to L2 forward traffic to associated AP.
6. There are mobility issues, which are not part of the standard.
7. It is a positive acknowledge system. Each packet is positively acknowledge (an ACK packet) before the next packet is sent.
8. Packets carry NAV's which hold the channel clear for the time stated in the NAV.
9. RTS/CTS pair with NAV covering up to the end of the ACK of the data packet are exchanged. The ACK has a NAV of 0.
10. There are not NACKs.

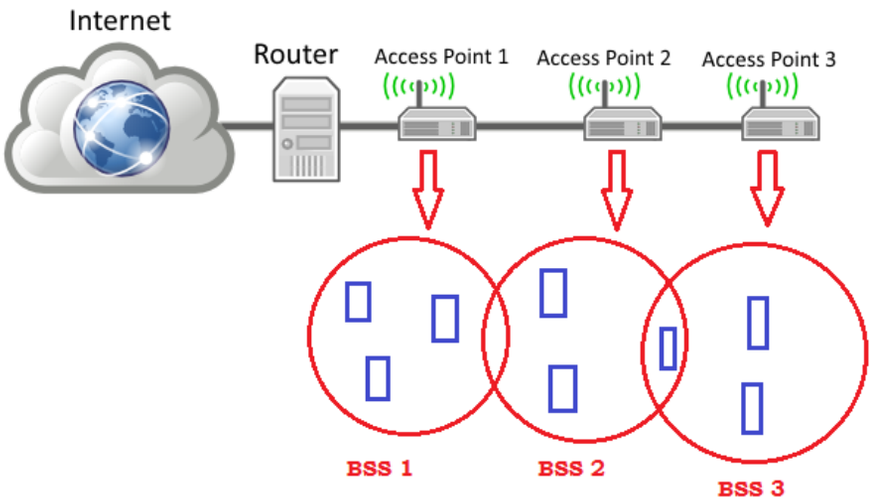
Basics of Wi-Fi

January 29, 2018 Anup Patel CN, Resources

WiFi stands for **Wireless Fidelity**.

It is Based on 802.11 and primarily a LAN Technology. Wi-Fi operates at the physical and data link layers of the OSI model

- **Wi-Fi is Half Duplex.**
- Ethernet is more secure than WiFi
- There is **Collision avoidance in wifi and collision detection in Ethernet**.



(Represent Devices)

Important Points

- Devices in Same BSS can connect Directly.
- Devices in different BSS can connect through Access Point.
- 802.11a uses the 5 GHz U-NII band, which offers at least 23 non-overlapping channels rather than the 2.4 GHz ISM frequency band which offer only three non-overlapping channels.
- RTS and CTS mechanism is used for Collision Avoidance.
- RTS and CTS Also solve **hidden node problem** or **hidden terminal problem.**
- Access point consisting of antenna and routers are main source that transmit and receive radio waves.

Bandwidth :

- **25 MHZ for 802.11b**
- **20 MHz for 802.11a**

According to distributed control function , a wifi station will transmit only when channel is clear. All transmissions are Acknowledged. So, if a station does not receive an ACK , it assume collision and retry after random amount of time.

There are 2 General type of Wifi Transmission

1. DCF (Distributed Coordination Function)
2. PCF (Point coordination Function) – Not much used

There are 2 Problem that makes wireless different than Wired :

1. Hidden Substation Problem
2. High Error Rate

Important Notes

- Every packet sent is positively acknowledged by receiver. Next packet is not sent until receiving a positive acknowledgment for previous Packet.
- A Transmission begin with a RTS (Request to send) & Receiver respond with CTS (Clear to send). then the data packet flow. This help in hidden substation problem.

A group of corresponding station is called a BSS (Basic Service set).

Independent BSS, or ad hoc : The network is only the members of the BSS, they talk between themselves directly, they self-organize, **there is no central autority**.

Infrastructure BSS : The BSS is organized around an Access Point which can bridge traffic out the BSS onto a distribution network. Members of the BSS talk to the AP(Access Point) only. A BSS (data link layer) is defined by who will hear a broadcast from the AP (but not by a station which is not an AP, because of the hidden station problem!).

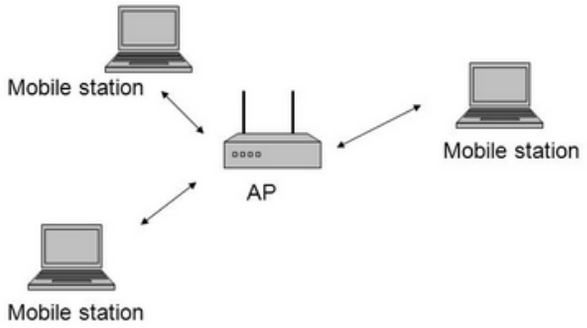
COAP 2020 Details and Important Dates March 18, 2020

IISc Bangalore Interview Experience by Eklavya Sharma| AIR 86 GATE CS 2018 March 18, 2020

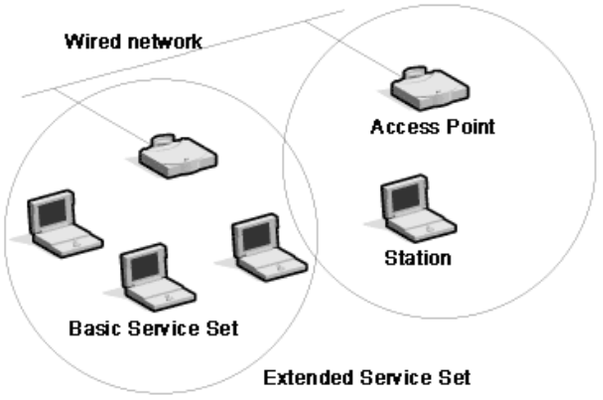
IISc Bangalore Interview Experience by Ravi Raja | AIR 888 GATE CS 2019 March 18, 2020

IIT's and IISc Cutoff Gate 2019 March 18, 2020

IISc Bangalore GATE 2019 Cutoffs March 18, 2020



ESS (Extended Service Set) : A bunch of BSS's connected by a distribution network. The distribution network connects the Access Points. WiFi doesn't specify the protocol that builds ESS's.



When a mobile host move beyond the range of one base station and into range of another. It will change its point of attachment into larger network . This is Handoff .

References

- Wikipedia
- http://www.cs.miami.edu/home/burt/learning/Csc524.052/notes/wifi.html
- Kurose

Check Also : IPv4 Vs IPv6

If you like GatePoint and would like to contribute, you can also write an article and mail your article to **gatecsepoint@gmail.com**. See your article appearing on the GatePoint main page and help other Gate Aspirants.

Anup Patel
M.Tech Student at Indian Institute of Science
AIR 2 ISRO SC Written Test Dec 2017
AIR 142 GATE 2018

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Network Security

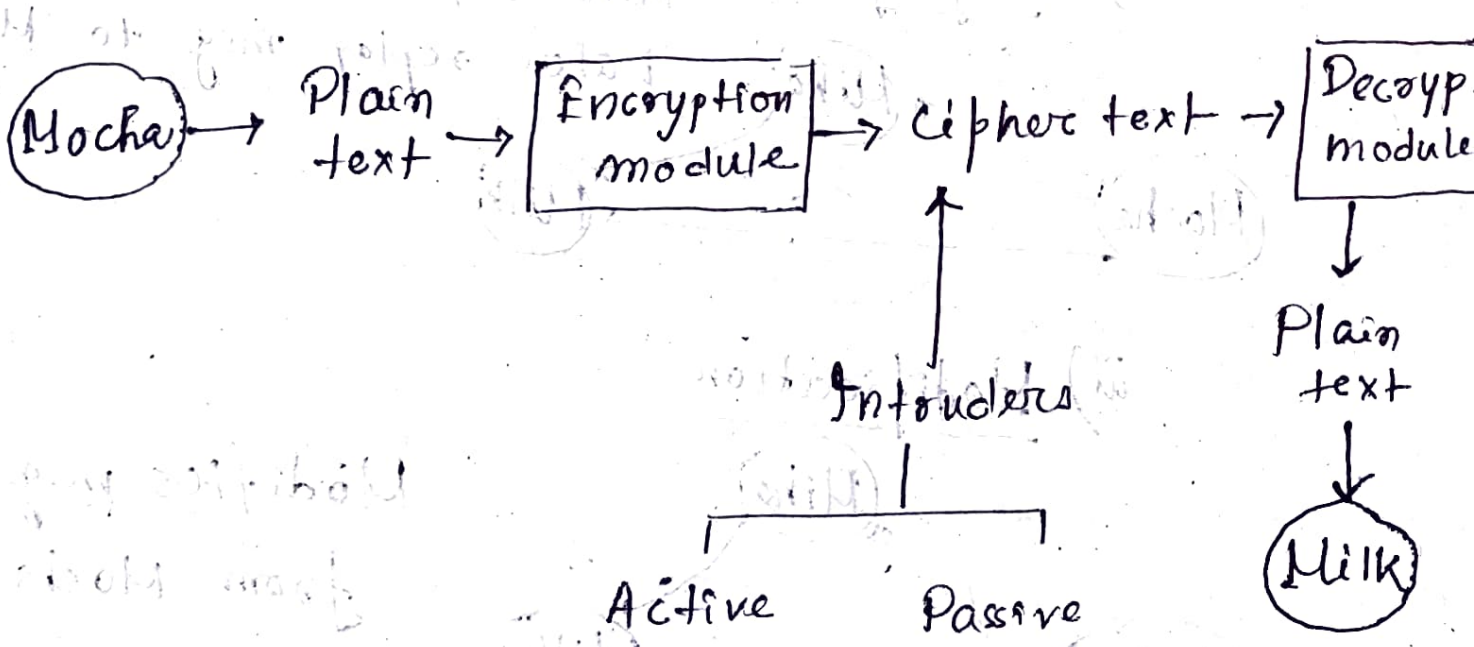
- Plain text : Actual message (Readable)

  Cipher text : Converted message (Unreadable)

  Encryption : Conversion from plain text to cipher text

  Decryption : Conversion from cipher text to plain text.

- Cryptography : Method of storing & transmitting data in a particular form so that only those for whom it is intended can read & process it



- **Passive attacker**

Can only read the data, can't change the content of the data.

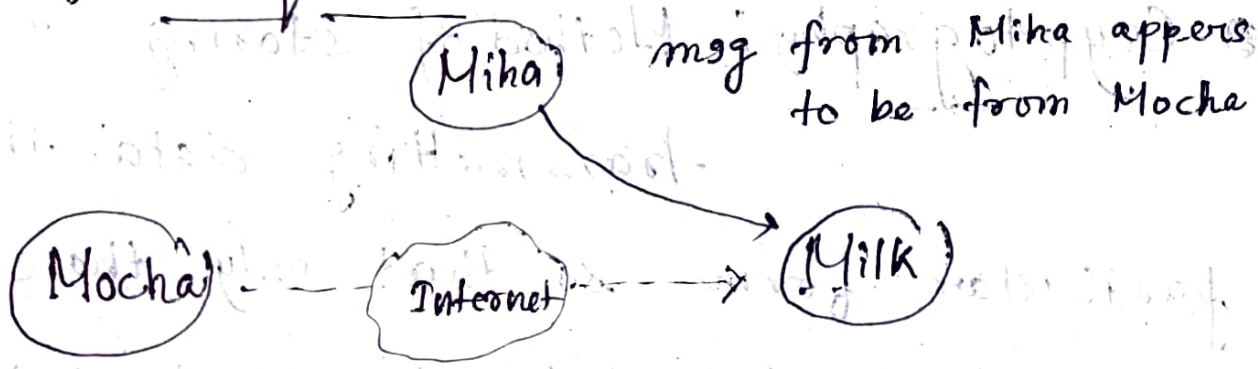
i) Reading message content.

ii) Traffic analysis

- **Active attacker**

Can read & modify the data.

i) Masquerade



ii) Replay

Miha captures msg. Later replay msg to Milk

iii) Modification

Modifies msg from Mocha

iv) Denial of service

Disrupts service by sending lots of requests.

- **Authentication** : Assurance that the communicating entity is the one that it claims to be.

Data integrity : The assurance that data received is exactly as sent by an authorized entity.

Access control : The prevention of unauthorized use of resources.

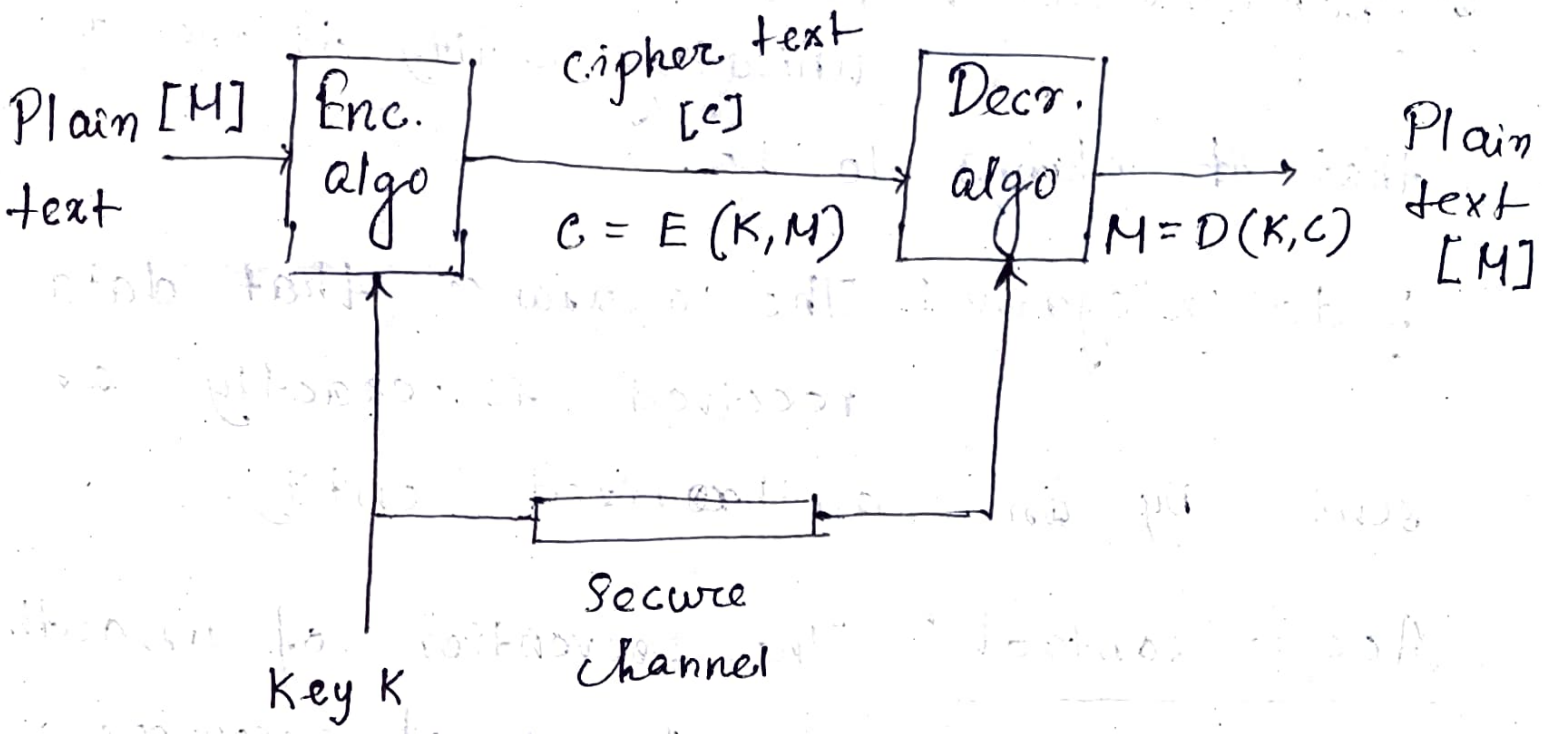
Data confidentiality : The protection of data from unauthorised disclosure.

Non repudation : Provides protection against denial by one of the entities involved in a comm^{n} of having participated in all or part of communication.

- Cryptography → Symmetric
  → Asymmetric

* **Symmetric Key Cryptography**

Both sender & receiver uses a common key to encrypt & decrypt the message. Secret key is known only to the S & R. Also called secret key cryptography.



Symmetric encryption algorithms —

a) Data enc. standard (DES)

b) Advanced enc. standard (AES)

Adv.: Efficient, take less time to encrypt & decrypt.

Disadv.:

i) Each pair of users require a unique secret key. If N people in the world wants to use this technique, we may consider a complete graph with N nodes. Each person will require N-1 keys to communicate with other N-1 persons. No. of edges = No. of unique keys required = $^{N}C_2 = \frac{N(N-1)}{2}$. So, the no. of

keys is very large.

ii) While sharing the key, attackers might intrude. (Diffie Hellman Key exchange algo is used to overcome this.)

* Modular Arithmetic.

- $a \equiv b \bmod n$: a & b leave same remainder when divided by n.

  eg. $24 \equiv 12 \bmod 12$. | a divided by n ⇓ remainder b

- $a \equiv b \bmod n$: n divides $(a-b)$

- If $a \equiv b \bmod n$ & $c \equiv d \bmod n$ then
  $(a+c) \equiv (b+d) \bmod n$.

- If $a \equiv b \bmod n$ & $c \equiv d \bmod n$ then
  $a \times c \equiv (b \times d) \bmod n$

- * If $a \equiv (b \times c) \bmod n$ then
  $a \equiv (b \bmod n \times c \bmod n) \bmod n$

- If $a \equiv (b+c) \bmod n$ then
  $a \equiv (b \bmod n + c \bmod n) \bmod n$

• Multiplicative inverse

For each $a \neq 0 \bmod p$ [p is a prime]

there is a 'b' such that

$$ab \equiv 1 \bmod p$$

b is the multiplicative inverse of a.

- $ab \equiv 1 \bmod p$

$$b = a^{-1} \bmod p$$

eg: $2 \neq 0 \bmod 7$

$$2 \times 4 \equiv 1 \bmod 7$$

$4 \equiv 2^{-1} \bmod 7$ (or $2 = 4^{-1} \bmod 7$)

4 is multiplicative inverse of 2 mod

- If a & n have no common factors, then 'a' has a multiplicative inverse mod n

$$\gcd(a, n) = 1$$

eg. There is no x s.t. $3 \times x \equiv 1 \bmod 6$

$$\gcd(3, 6) \neq 1$$

eg. There is a s.t. $5 \times x = 1 \bmod 9$

$$x = 2$$

– Euler's Theorem.

If n is a +ve integer & a, n are coprime $(\gcd(a,n) = 1)$ then

$$a^{\phi(n)} \equiv 1 \bmod n \quad [\phi(n) \text{ is Euler's totient fun}^n]$$

eg. $a = 8 \qquad n = 165$

$$a^{\phi(n)} \equiv 1 \bmod n \qquad (\because \gcd(a,n) = 1)$$

$$\phi(165) = \phi(15) \times \phi(11)$$

$$= \phi(3) \times \phi(5) \times \phi(11)$$

$$= \overset{3-1}{2} \times \overset{5-1}{4} \times \overset{11-1}{10} = 80$$

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

$$\rightsquigarrow a^{\phi(n)\,t} \equiv 1 \bmod n \quad [t \in \mathbb{N}]$$

– Fermat's Theorem : Special case of Euler's theorem.

For any prime number n and $a \not\equiv 0 \bmod n$

$$a^{n-1} \equiv 1 \bmod n$$

$\rightsquigarrow$ If n is a +ve integer & a, n are coprime then

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

$$\Rightarrow a^{\phi(n)t+1} \equiv a \bmod n$$

eg. $a = 9, n = 13$

$\phi(13) = 12$

$\Rightarrow 9^{12} \equiv 1 \mod 13$

$\Rightarrow 9^{13} \equiv 9 \mod 13$

$\Rightarrow 9^{25} \equiv 9 \mod 13$

~ If n is a positive integer & a, n are coprime, $b \equiv 1 \mod \phi(n)$. then

$$a^b \equiv a \mod n$$

eg. $a = 5$, $n = 8 = 2^3 \Rightarrow \phi(n) = 8\left(1 - \frac{1}{2}\right) = 4.$

Let $b = 17$ $[17 = 1 \mod 4]$

$5^{17} = 5 \mod 8$

— Euler's totient function $\phi(n)$

The totient $\phi(n)$ of a +ve int n $[n \geq 1]$ is defined to be the number of +ve integers less than n that are coprime to n.

eg. $\phi(3) = 2$ $\{1, 2\}$

$\phi(14) = 6$ $\{1, 3, 5, 9, 11, 13\}$

~ When n is a prime number, $\phi(n) = n - 1$

~ When m & n are coprime, $\phi(mn) = \phi(m) \times \phi(n)$

eg. $\phi(15) = \phi(3) \times \phi(5)$ $[\gcd(3,5) = 1]$

$= 2 \times 4$

$= 8$

~ If the prime factorisation of n is given by

$$n = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_n^{e_n}.$$

then $$\phi(n) = n\left(1-\frac{1}{p_1}\right) \times \left(1-\frac{1}{p_2}\right) \times \cdots \times \left(1-\frac{1}{p_n}\right)$$

eg: $36 = 3^2 \times 2^2$

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

$$= 12.$$

* Asymmetric encryption algorithms.:
  - RSA algo
  - Diffie-Hellman Key exchange.

* Primitive root-

The number b in $a \equiv b \bmod n$ is called the residue of $a \bmod n$.

The residue classes of $f(a) \bmod n$ are all possible values of residue $f(a) \bmod n$.

eg. Residue classes of $a^2 \bmod 6$ are {0,1,3,

$0^2 \equiv 0 \bmod 6$
$1^2 \equiv 1 \bmod 6$
$2^2 \equiv 4 \bmod 6$
$3^2 \equiv 3 \bmod 6$
$4^2 \equiv 4 \bmod 6$
$5^2 \equiv 1 \bmod 6$
$6^2 \equiv 0 \bmod 6$

~ Let p be a prime then b ~~be~~ is a primitive root for 'p' if powers of b

$\sim 1, b, b^2, b^3, \ldots$

include all the residue classes of mod p.

~ If p is a prime, then there is a primitive root for p.

~ The powers of b form a repeating cycle, & that cycle can't be longer than p-1 [$a^{p-1} \equiv 1 \bmod n$: Fermat's theorem] so b is a primitive root if the cycle is as long as p-1.

eg. If p=7, then 3 is primitive root for p.

$3^0 \equiv 1 \bmod 7$

$3^1 \equiv 3 \bmod 7$

$3^2 \equiv 2 \bmod 7$

$3^3 \equiv 6 \bmod 7$

$3^4 \equiv 4 \bmod 7$

$3^5 \equiv 5 \bmod 7$

$3^6 \equiv 1 \bmod 7$

repeats.

~ Given an integer 'a' & a +ve integer 'n' with gcd (a, n) = 1, the multiplicative ~~inverse~~ order of a mod n is the smallest +ve integer k with

$$a^k \equiv 1 \bmod n$$

~ m is a primitive root modulo n if & only if the multiplicative order of m is $\phi(n)$.

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

$$3^6 \equiv 1 \bmod 7$$

~ Apart from 1, 2, 4 the only numbers with primitive root 3 are of the shape p^k, $2p^k$ where p is an odd prime number.

eg. 3, 5, 7, 6, 10, 14, 2×7^2 etc

- **Discrete Logarithm problem.**

(base of DH key exchange)

Problem of finding x such that $a^x \equiv b \bmod p$ [p is a prime number & a, b are non zero integers] is called discrete logarithm problem. (NP hard problem)

~ By assuming $0 \leq x < n$ where $n = ord_p(a)$ [$a^n \equiv 1 \bmod p \Rightarrow n = ord_p(a)$] we need to search for the value of x in this range.

eg. $p = 11,\ a = 2,\ b = 9$

$2^x \equiv 9 \bmod 11$) It's a one-way f^n.

$2^6 \equiv 9 \bmod 11$

$x = 6.$

~ one-way f^n

A function $f(x)$ is called one-way function if $f(x)$ is easy to compute but, given y it is computationally infeasible to find x, with $y = f(x)$.

* Diffie Hellman key exchange toppoint(18)

B		C
Public keys P, G		P, G
Private key a		b
key generated $x = G^a \bmod P$		$y = G^b \bmod P$
	Exchange of generated keys.	
Received y		x
Generated secret key $K_a = y^a \bmod P$		$K_b = x^b \bmod P$

Algebraically it can be shown $K_a = K_b$

Users now have symmetric secret key to encrypt.

* Modulo Arithmetic.

dividing a with n leaves remainder b

- $a \equiv b \pmod n \iff \exists k$, s.t. $a = nk + b$

Congruence $\iff$ $(a-b)$ is divisible by n

$(a-b) \mid n$.

eg: $38 \equiv 14 \mod 12$

$-3 \equiv -8 \mod 5$

- Congruence is an equivalence relation.

aRb under $(\mod n)$

- Properties.

i) $a \equiv b \mod n$ and $c \equiv d \mod n$

$\Rightarrow (a+c) \equiv (b+d) \mod n$. —(1)

$a = nk + b$, $c = nl + d$

$a + c = nk + b + nl + d = n(k+l) + (b+d)$

ii) Similarly, $(a-c) \equiv (b-d) \mod n$ —(2)

iii) $a_1 \equiv b_1 \mod n$ and $a_2 = b_2 \mod n$

$\Rightarrow a_1 a_2 = b_1 b_2 \mod n$ —(3)

$a_1^k = b_1^k \mod n$ $\forall$ non-ve k —(4)

iv) $p(x)$: polynomial function $\mid a \equiv b \bmod n$

$$p(a) \equiv p(b) \bmod n \quad \text{—(5)}$$

$$\rightarrow (a \equiv b \bmod n) \not\Rightarrow (k^a \equiv k^b \bmod n)$$

Not always.

v) $\{a \equiv (b+c) \bmod n\} \Rightarrow \{a \equiv (b \bmod n + c \bmod n) \bmod n\}$ —(6)

$\{a \equiv bc \bmod n\} \Rightarrow \{a \equiv (b \bmod n \cdot c \bmod n) \bmod n\}$ —(7)

- **Modular Multiplicative Inverse.**

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

a & x are MMI of each other

- **Co-prime** (relative primes)

$$\gcd(x, y) = 1. \quad \text{eg. } x = 12,\ y = 17$$

✓ • If x & y are coprimes, then $\exists\ z \in I$ s.t.

$$xz \equiv 1 \pmod y$$

(x & z are MMI).

eg. $12 \cdot 10 \equiv 1 \pmod{17}$

(12 = x, 10 = z, 17 = y)

- **Euler's Totient Function.**

$\phi(n)$ = # of +ve integers upto n that are coprime to n

eg. $\phi(9) = 6$

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

✓ → If m & n are coprimes then

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

Chinese Remainder Theorem

→ If n is a prime number, then

$$\phi(n) = n-1$$

→ If p is a prime, & $k \geq 1$, then there are exactly p^k/p numbers between 1 and p^k that are divisible by p; which gives us

$$\phi(p^k) = p^k - p^{k-1}$$

→ Euler's product formula :

If $n = p_1^{a_1} p_2^{a_2} \cdots p_K^{a_K}$ where p_i are prime factors of n (product of prime factors),

$$\phi(n) = \phi(p_1^{a_1}) \phi(p_2^{a_2}) \cdots \phi(p_K^{a_K})$$

$$\phi(n) = (p_1^{a_1} - p_1^{a_1-1})(p_2^{a_2} - p_2^{a_2-1}) \cdots (p_K - p_K^{a_K-1})$$

$$\phi(n) = p_1^{a_1}\left(1 - \frac{1}{p_1}\right) \cdots p_K^{a_K}\left(1 - \frac{1}{p_K}\right)$$

$$\checkmark \quad \phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_K}\right)$$

eg. $36 = 2^2 3^2$

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

- Euler's Theorem for co-primes.

If $n \in I^+$ & a, n are co-primes, then

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

eg. $n = 165$

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

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

$$= 80.$$

$$a = 8 \quad \& \quad \gcd(8, 165) = 1$$

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

- Fermat's little theorem

If $n \in I^+$ & a, n are coprimes then

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

$$\begin{cases} a^{\phi(n)} \equiv 1 \bmod n \\ a \equiv a \bmod n \end{cases}$$

✓ → If $n \in I^+$ & a, n are coprimes & $b \equiv 1 \pmod{\phi(n)}$ then

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

Proof →

$$b = 1 \pmod{\phi(n)}$$

$$\Rightarrow b = K\,\phi(n) + 1$$

$$a^b = a^{K\phi(n)+1} = \left(a^{\phi(n)}\right)^K \cdot a$$

$$\left(a^{\phi(n)}\right)^K \equiv (1^K) \bmod n$$

$$\equiv 1 \bmod n \quad \text{from Euler's theorem for coprimes}$$

$$a \equiv a \bmod n$$

$$\Rightarrow a^b \equiv a \bmod n.$$

* RSA Algorithm. (Rivest – Shamir – Adleman)

(S) ⟶ (R). K_u^R = R's public key

K_r^R = R's private key

$K_u^R = (e, n)$ combination of 2 keys

$K_r^R = (d, n)$ only known to R.

Now, p = plain text

$c = p^e \bmod n$

$0 \le p < n$

@ S

$p = c^d \bmod n$

$= (p^e \bmod n)^d \bmod n$

$= p^{ed} \bmod n.$

@ R

We want to pick e, d s.t.

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

$$\Rightarrow \underline{ed \equiv 1 \bmod \phi(n)}$$

1. given (e, n) computing d must be very hard.

$\Downarrow$

computing $\phi(n)$ must be hard.

Picking n, e, d.

1. p_1, p_2 be large prime numbers.

$$\boxed{n = p_1 p_2}$$ (n is large 256 bit number)

$$\phi(n) = \phi(p_1)\,\phi(p_2) = (p_1 - 1)(p_2 - 1)$$

2. Pick $1 < e < \phi(n)$ s.t. $\gcd(e, \phi(n)) = 1$.
i.e e & $\phi(n)$ are coprimes

$\Rightarrow \exists\, d \in I^+$ s.t. $ed \equiv 1 \bmod \phi(n)$ by property 5
(modulo mult. inverse)
e & d.

eg. $p_1 = 53 \quad p_2 = 61$

$n = 3233. \Rightarrow \phi(n) = 52 \times 60 = 3120$

$e = 17$ as $\gcd(e, \phi(n)) = 1$.

then $d = 2753$.

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

$17d - 1 = K \cdot 3120$

$2753 \quad\quad 15$

So, $K_u^R = (17, 3233) \qquad K_r^R = (2753, 3233)$

- **Residue.**

$a \equiv b \bmod n$

b is residue of $a \bmod n$

Residue class / convergence class.

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

↳ set of all integers that are congruent to $a \bmod n$.

$\equiv a \bmod n$

✓ eg. Residue class of $x^2 \bmod 6$ are –

$0^2 \bmod 6 = 0$

$1^2 \bmod 6 = 1$

$2^2 \bmod 6 = 4$

$3^2 \bmod 6 = 3$

$4^2 \bmod 6 = 4$

$5^2 \bmod 6 = 1$

$\bar{0}_6, \bar{1}_6, \bar{3}_6, \bar{4}_6$

- **Primitive root of a prime.**

p be a prime number.

g is the primitive root of n if $g^0, g^1, g^2, \ldots$ include all convergence classes of p

$\{\bar{1}_p, \bar{2}_p, \bar{3}_p, \ldots \bar{P}_p\}$.

eg. $p = 7$. $g = 3$

$3^0 \bmod 7 = 1$ $\quad$ $3^4 \bmod 7 = 4$

$3^1 \bmod 7 = 3$ $\quad$ $3^5 \bmod 7 = 5$

$3^2 \bmod 7 = 2$ $\quad$ $3^6 \bmod 7 = 1$. } repeats

$3^3 \bmod 7 = 6$

Also, for $g = 5$.

$5^0 \bmod 7 = 1$ $\quad$ $5^4 \bmod 7 = 2$

$5^1 \bmod 7 = 5$ $\quad$ $5^5 \bmod 7 = 3$

$5^2 \bmod 7 = 4$ $\quad$ $5^6 \bmod 7 = 1$. } repeats

$5^3 \bmod 7 = 6$

$7 - 1 = 6$

So, 7 has 2 primitive roots → 3, 5.

- Fermat's little theorem.

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

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

* Multiplicative order.

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

$$a^k \equiv 1 \bmod n$$

Generalised def^n: a is the primitive root of n iff multiplicative order of $a = \phi(n)$

→ primitive roots

n	$g(n)$ - pr. roots
2	1
3	2
4	3
5	2,3
6	5
7	3,5
~~8~~	
9	2,5
10	3,7
11	2,6,7,8
13	2,6,7,11

Except 1, 2, 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

- given a, x, p computing b is easy
- given a, b, p computing x is very hard.

One-way functions

$f(x) = b$ easy to compute

$g(b) = x$ very hard to compute

* Diffie-Hellman Key Exchange.

Symmetric key encryption : key exchange

S	R.
Public keys P, G	Public keys P, G
Private key a	Private key b
key generated = $x = G^a \bmod P$	key generated - $y = G^b \bmod P$

--- Exchange of generated keys ---

key received y	key received x
Generated secret key = $y^a \bmod P$	Generated secret key = $x^b \bmod P$

-- Algebraically , $K_a = K_b$ --

Users now have a secret symmetric key to encrypt data.

Secret key generated @ R,

$$x^b \bmod P = (G^a \bmod P)^b \bmod P$$
$$= G^{ab} \bmod P$$
$$= (G^b)^a \bmod P$$
$$= (G^b \bmod P)^a \bmod P$$
$$= y^a \bmod P.$$

Public Key Cryptography | RSA Algorithm Example

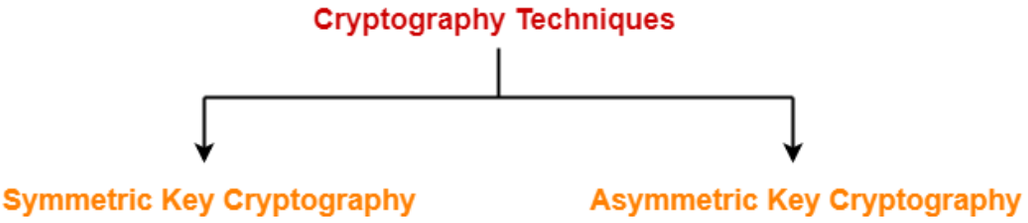
Computer Networks

Cryptography in Network Security-

Before you go through this article, make sure that you have gone through the previous article on **Cryptography**.

We have discussed-

- Cryptography is a method of storing and transmitting data in a particular form.
- Cryptography techniques are-



In this article, we will discuss about Asymmetric Key Cryptography.

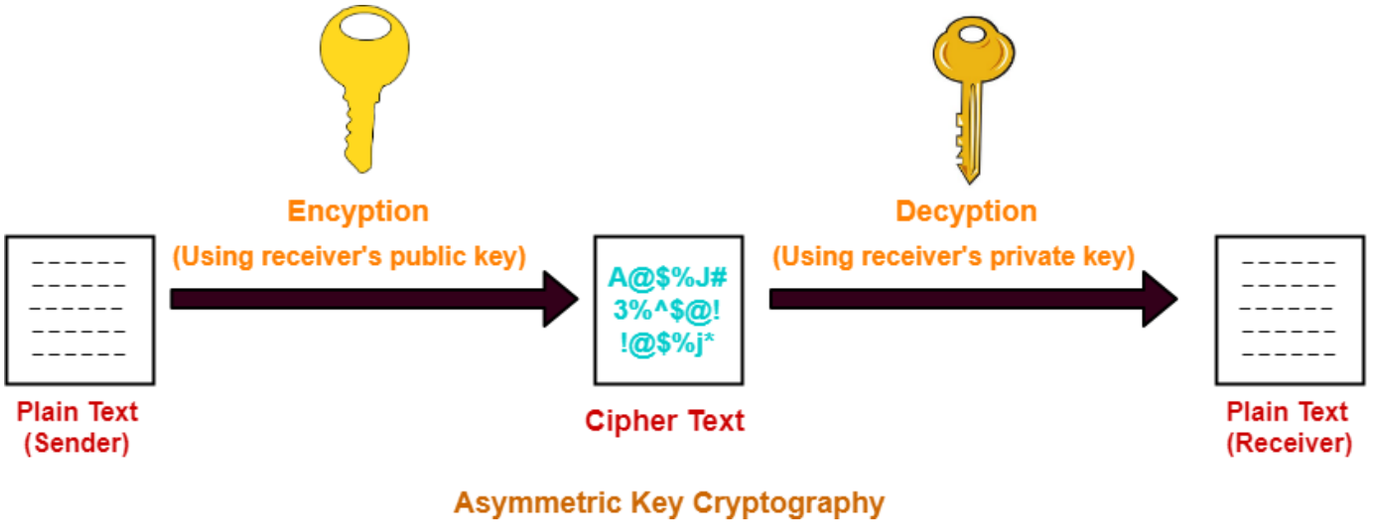
Asymmetric Key Cryptography-

In this technique,

- Sender and receiver use different keys to encrypt and decrypt the message.
- It is called so because sender and receiver use different keys.
- It is also called as **public key cryptography**.

Working-

The message exchange using public key cryptography involves the following steps-



Step-01:

At sender side,

- Sender encrypts the message using receiver's public key.
- The public key of receiver is publicly available and known to everyone.
- Encryption converts the message into a cipher text.
- This cipher text can be decrypted only using the receiver's private key.

Step-02:

- The cipher text is sent to the receiver over the communication channel.

Step-03:

At receiver side,

- Receiver decrypts the cipher text using his private key.
- The private key of the receiver is known only to the receiver.
- Using the public key, it is not possible for anyone to determine the receiver's private key.
- After decryption, cipher text converts back into a readable format.

Advantages-

The advantages of public key cryptography are-

- It is more robust.
- It is less susceptible to third-party security breach attempts.

Disadvantages-

The disadvantages of public key cryptography are-

- It involves high computational requirements.
- It is slower than symmetric key cryptography.

Number of Keys Required-

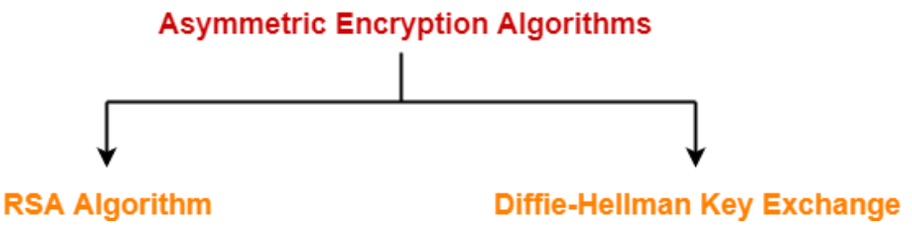
To use public key cryptography,

- Each individual requires two keys- one public key and one private key.

- For n individuals to communicate, number of keys required = 2 x n = 2n keys.

Asymmetric Encryption Algorithms-

The famous asymmetric encryption algorithms are-



1. RSA Algorithm
2. Diffie-Hellman Key Exchange

In this article, we will discuss about RSA Algorithm.

RSA Algorithm-

Let-

- Public key of the receiver = (e , n)
- Private key of the receiver = (d , n)

Then, RSA Algorithm works in the following steps-

Step-01:

At sender side,

- Sender represents the message to be sent as an integer between 0 and n-1.
- Sender encrypts the message using the public key of receiver.
- It raises the plain text message ‘P’ to the e^{th} power modulo n.
- This converts the message into cipher text ‘C’.

$$C = P^e \bmod n$$

Step-02:

- The cipher text ‘C’ is sent to the receiver over the communication channel.

Step-03:

At receiver side,

- Receiver decrypts the cipher text using his private key.
- It raises the cipher text ‘C’ to the d^{th} power modulo n.
- This converts the cipher text back into the plain text ‘P’.

$$\mathbf{P = C^d \bmod n}$$

NOTE-

‘e’ and ‘d’ must be multiplicative inverses modulo Ø(n)

After decryption, receiver must have-

$$P = C^d \bmod n$$

$$P = (P^e \bmod n)^d \bmod n$$

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

For this equation to be true, by Euler’s Theorem, we must have-

$$ed = 1 \bmod Ø(n)$$

OR

$$ed = kØ(n) + 1$$

Thus, e and d must be multiplicative inverses modulo Ø(n).

Steps to Generate Public Key And Private Key-

An individual can generate his public key and private key using the following steps-

Step-01:

Choose any two prime numbers p and q such that-

- They are different.
- They are very large.

Step-02:

Calculate ‘n’ and toilent function Ø(n) where-

- n = p x q
- Ø(n) = (p-1) x (q-1)

Step-03:

Choose any value of ‘e’ such that-

- 1 < e < Ø(n)
- gcd (e, Ø(n)) = 1

Step-04:

Determine ‘d’ such that-

$$ed = 1 \bmod Ø(n)$$

OR

$$d = \frac{1 + k\,Ø(n)}{e}$$

- You already know the value of ‘e’ and Ø(n).
- Choose the least positive integer value of ‘k’ which gives the integer value of ‘d’ as a result.
- Use trial and error method.
- Start substituting different values of ‘k’ from 0.

PRACTICE PROBLEMS BASED ON RSA ALGORITHM-

Problem-01:

In a RSA cryptosystem, a participant A uses two prime numbers p = 13 and q = 17 to generate her public and private keys. If the public key of A is 35, then the private key of A is _____.

Solution-

Given-

- Prime numbers p = 13 and q = 17
- Public key = 35

Step-01:

Calculate ‘n’ and toilent function Ø(n).

Value of n,

n = p x q

n = 13 x 17

∴ n = 221

Toilent function,

Ø(n) = (p-1) x (q-1)

Ø(n) = (13-1) x (17-1)

∴ Ø(n) = 192

Step-02:

- We are already given the value of e = 35.
- Thus, public key = (e , n) = (35 , 221)

Step-03:

Determine ‘d’ such that-

$$d = \frac{1 + k\,Ø(n)}{e}$$

$$d = \frac{1 + k \times 192}{35}$$

Here,

- The least value of ‘k’ which gives the integer value of ‘d’ is k = 2.
- On substituting k = 2, we get d = 11.

Thus, private key of participant A = (d , n) = (11, 221).

Problem-02:

In the RSA public key cryptosystem, the private and public keys are (e, n) and (d, n) respectively, where n = p x q and p and q are large primes. Besides, n is public and p and q are private. Let M be an integer such that 0 < M < n and f(n) = (p-1)(q-1).

Now consider the following equations-

I. $M' = M^e \bmod n$ and $M = (M')^d \bmod n$

II. $ed \equiv 1 \bmod n$

III. $ed = 1 \bmod f(n)$

IV. $M' = M^e \bmod f(n)$ and $M = (M')^d \bmod f(n)$

Which of the above equations correctly represent RSA cryptosystem?

1. I and II
2. I and III
3. II and IV
4. III and IV

Solution-

Clearly, Option (B) is correct.

To gain better understanding about RSA Algorithm,

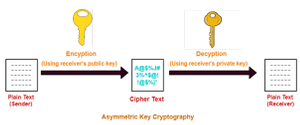
Watch this Video Lecture

Next Article- **Diffie Hellman Key Exchange Algorithm**

Get more notes and other study material of **Computer Networks**.

Watch video lectures by visiting our YouTube channel **LearnVidFun.**

Summary



Article Name	Public Key Cryptography | RSA Algorithm Example
Description	Public key cryptography or Asymmetric key cryptography use different keys for encryption and decryption. RSA Algorithm Examples. RSA Algorithm and Diffie Hellman Key Exchange are asymmetric key algorithms.
Author	Akshay Singhal
Publisher Name	Gate Vidyalay
Publisher Logo	GateVidyalay A temple of learning

Liked this article? Share it with your friends and classmates now-

Diffie Hellman Key Exchange | Asymmetric Encryption

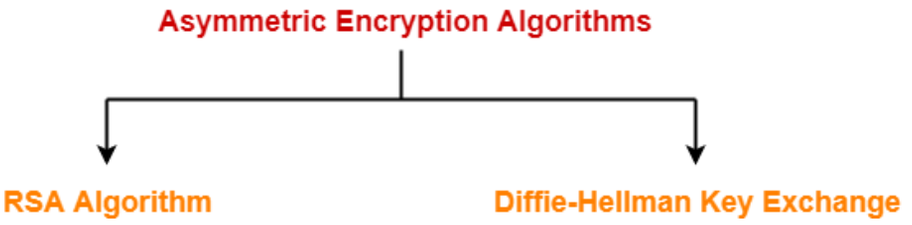
Computer Networks

Asymmetric Encryption-

Before you go through this article, make sure that you have gone through the previous article on **Asymmetric Key Cryptography**.

In asymmetric encryption,

- Sender and receiver use different keys to encrypt and decrypt the message.
- The famous asymmetric encryption algorithms are-



In this article, we will discuss about Diffie Hellman Key Exchange Algorithm.

Symmetric Key Cryptography-

In symmetric key cryptography,

- Both sender and receiver use a common secret key to encrypt and decrypt the message.
- The major issue is exchanging the secret key between the sender and the receiver.
- Attackers might intrude and know the secret key while exchanging it.

Read More- **Symmetric Key Cryptography**

Diffie Hellman Key Exchange-

As the name suggests,

- This algorithm is used to exchange the secret key between the sender and the receiver.
- This algorithm facilitates the exchange of secret key without actually transmitting it.

Diffie Hellman Key Exchange Algorithm-

Let-

- Private key of the sender = X_S
- Public key of the sender = Y_S
- Private key of the receiver = X_r
- Public key of the receiver = Y_r

Using Diffie Hellman Algorithm, the key is exchanged in the following steps-

Step-01:

- One of the parties choose two numbers ‘a’ and ‘n’ and exchange with the other party.
- ‘a’ is the primitive root of prime number ‘n’.
- After this exchange, both the parties know the value of ‘a’ and ‘n’.

Step-02:

- Both the parties already know their own private key.
- Both the parties calculate the value of their public key and exchange with each other.

Sender calculate its public key as-

$$\mathbf{Y_s = a^{X_s} \bmod n}$$

Receiver calculate its public key as-

$$\mathbf{Y_r = a^{X_r} \bmod n}$$

Step-03:

- Both the parties receive public key of each other.
- Now, both the parties calculate the value of secret key.

Sender calculates secret key as-

$$\mathbf{Secret\ key = (Y_r)^{X_s} \bmod n}$$

Receiver calculates secret key as-

$$\mathbf{Secret\ key = (Y_s)^{X_r} \bmod n}$$

Finally, both the parties obtain the same value of secret key.

PRACTICE PROBLEMS BASED ON DIFFIE HELLMAN KEY EXCHANGE-

Problem-01:

Suppose that two parties A and B wish to set up a common secret key (D-H key) between themselves using the Diffie Hellman key exchange technique. They agree on 7 as the modulus and 3 as the primitive root. Party A chooses 2 and party B chooses 5 as their respective secrets. Their D-H key is-

1. 3
2. 4
3. 5
4. 6

Solution-

Given-

- n = 7
- a = 3
- Private key of A = 2
- Private key of B = 5

Step-01:

Both the parties calculate the value of their public key and exchange with each other.

Public key of A

$= 3^{\text{private key of A}} \bmod 7$

$= 3^2 \bmod 7$

$= 2$

Public key of B

$= 3^{\text{private key of B}} \bmod 7$

$= 3^5 \bmod 7$

$= 5$

Step-02:

Both the parties calculate the value of secret key at their respective side.

Secret key obtained by A

$= 5^{\text{private key of A}} \bmod 7$

$= 5^2 \bmod 7$

$= 4$

Secret key obtained by B

$= 2^{\text{private key of B}} \bmod 7$

$= 2^5 \bmod 7$

$= 4$

Finally, both the parties obtain the same value of secret key.

The value of common secret key = 4.

Thus, Option (B) is correct.

Problem-02:

In a Diffie-Hellman Key Exchange, Alice and Bob have chosen prime value q = 17 and primitive root = 5. If Alice's secret key is 4 and Bob's secret key is 6, what is the secret key they exchanged?

1. 16
2. 17
3. 18
4. 19

Solution-

Given-

- n = 17
- a = 5
- Private key of Alice = 4
- Private key of Bob = 6

Step-01:

Both Alice and Bob calculate the value of their public key and exchange with each other.

Public key of Alice

$= 5^{\text{private key of Alice}} \bmod 17$

$= 5^4 \bmod 17$

$= 13$

Public key of Bob

$= 5^{\text{private key of Bob}} \bmod 17$

$= 5^6 \bmod 17$

$= 2$

Step-02:

Both the parties calculate the value of secret key at their respective side.

Secret key obtained by Alice

$= 2^{\text{private key of Alice}} \bmod 7$

$= 2^4 \bmod 17$

$= 16$

Secret key obtained by Bob

$= 13^{\text{private key of Bob}} \bmod 7$

$= 13^6 \bmod 17$

$= 16$

Finally, both the parties obtain the same value of secret key.

The value of common secret key = 16.

Thus, Option (A) is correct.

To gain better understanding about Diffie Hellman Key Exchange Algorithm,

Watch this Video Lecture

Next Article- Digital Signatures

Get more notes and other study material of **Computer Networks**.

Watch video lectures by visiting our YouTube channel **LearnVidFun.**

Summary

How Digital Signature Works | Algorithm

Computer Networks

Digital Signatures-

- The signature on a document is the proof to the receiver that the document is coming from the correct entity.
- A digital signature guarantees the authenticity of an electronic document in digital communication.

How Digital Signature Works?

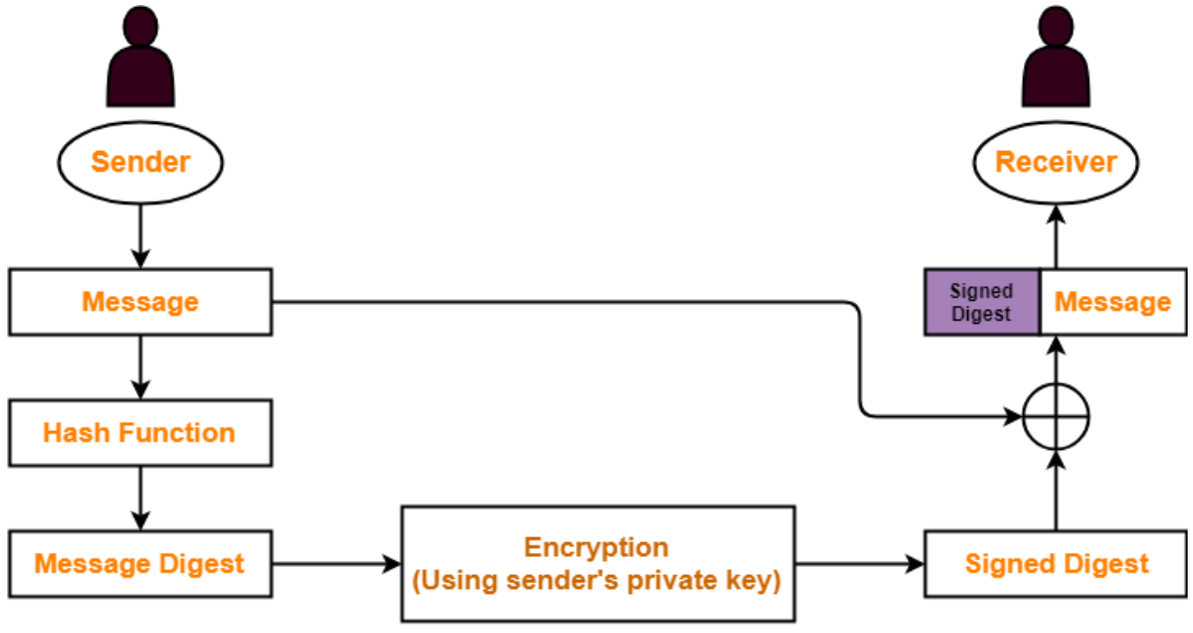
- The sender of the document digitally signs the document.
- The receiver of the document verifies the signature.

The steps involved in the digital signature algorithm are-

At Sender Side-

At sender side,

- Using a hash function, sender converts the message to be sent into a digested form.
- There are various hash functions that may be used like SHA-1, MD5 etc.
- The message in digested form is called as **message digest**.
- Sender encrypts the message digest using his private key.
- The encrypted message digest is called as **signed digest** or **signature** of the sender.
- Sender sends the signed digest along with the original message to the receiver.



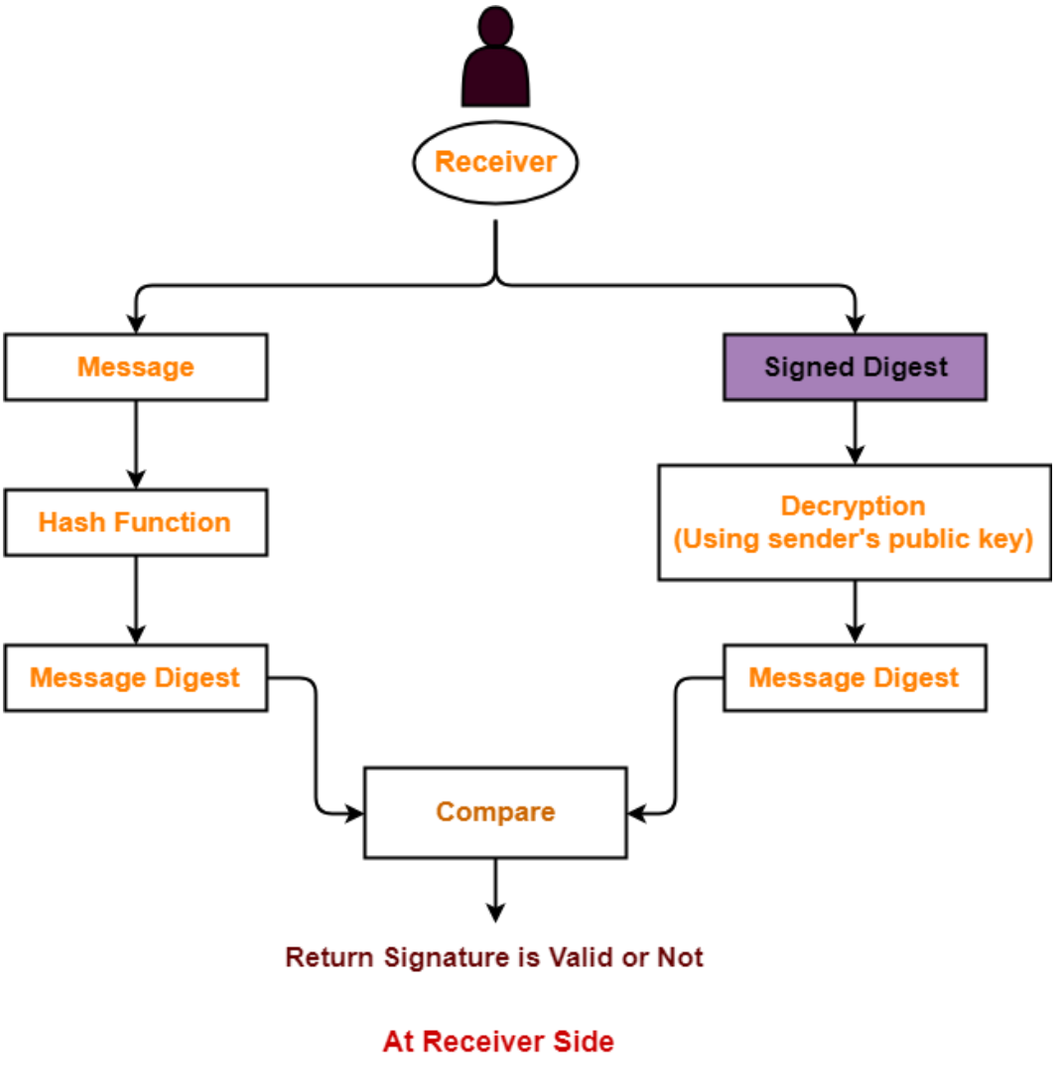
At Sender Side

At Receiver Side-

At receiver side,

- Receiver receives the original message and the signed digest.

- Using a hash function, receiver converts the original message into a message digest.
- Also, receiver decrypts the received signed digest using the sender's public key.
- On decryption, receiver obtains the message digest.
- Now, receiver compares both the message digests.
- If they are same, then it is proved that the document is coming from the correct entity.



At Receiver Side

Also Read- [RSA Algorithm]

Important Points-

Point-01:

After digitally signing the document, sender sends the following two things to the receiver-

- Signed digest or signature
- Original message

Point-02:

- Sender uses his private key to digitally sign the document.
- Receiver uses the sender's public key to verify the signature.

Point-03:

- Digital signature of a person varies from document to document.
- This ensures authenticity of the document.

Point-04:

In digital signature,

- There is one to one relationship between a message and a signature.
- Each message has its own signature.

Point-05:

Digital signature verifies-

- Authenticity
- Integrity
- Non-repudiation

Also Read- **Diffie Hellman Key Exchange Algorithm**

PRACTICE PROBLEMS BASED ON DIGITAL SIGNATURES-

Problem-01:

Anarkali digitally signs a message and sends it to Salim. Verification of the signature by Salim requires-

1. Anarkali's public key
2. Salim's public key
3. Salim's private key
4. Anarkali's private key

Solution-

Clearly, Option (A) is correct.

Problem-02:

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 by 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?

1. $\{m, K_B^+(H(m))\}$
2. $\{m, K_B^-(H(m))\}$
3. $\{m, K_A^-(H(m))\}$
4. $\{m, K_A^+(H(m))\}$

Solution-

Clearly, Option (B) is correct.

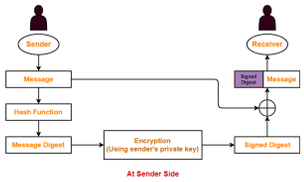
To gain better understanding about Digital Signatures,

Watch this Video Lecture

Get more notes and other study material of **Computer Networks**.

Watch video lectures by visiting our YouTube channel **LearnVidFun.**

Summary



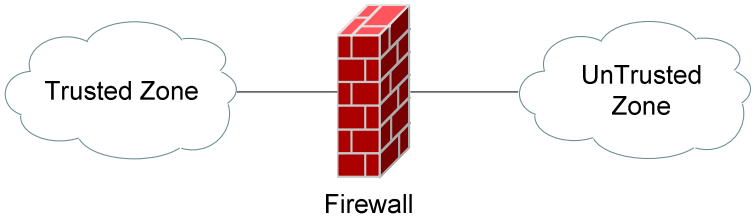
Article Name	How Digital Signature Works | Algorithm
Description	A digital signature guarantees the authenticity of an electronic document. How digital signature works? The digital signature algorithm describes how digital signature works.
Author	Akshay Singhal
Publisher Name	Gate Vidyalay
Publisher Logo	GateVidyalay A temple of learning

Liked this article? Share it with your friends and classmates now-

- Introduction to Firewalls -

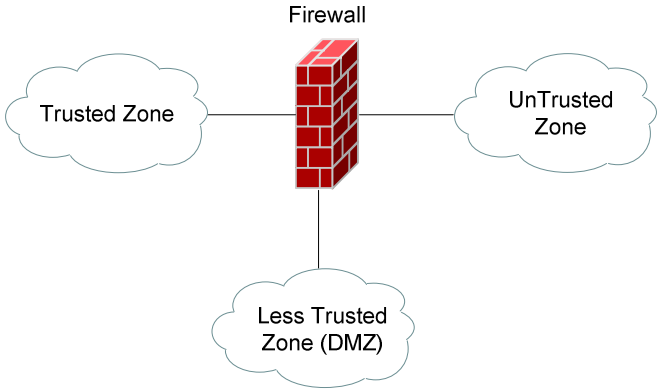
Firewall Basics

Traditionally, a firewall is defined as any device (or software) used to filter or control the flow of traffic. Firewalls are typically implemented on the network perimeter, and function by defining **trusted** and **untrusted zones**:



Most firewalls will **permit** traffic from the *trusted* zone to the *untrusted* zone, **without** any explicit configuration. However, traffic from the *untrusted* zone to the *trusted* zone must be **explicitly permitted.** Thus, any traffic that is not explicitly permitted from the untrusted to trusted zone will be **implicitly denied** (by default on *most* firewall systems)**.**

A firewall is not limited to only two zones, but can contain multiple 'less trusted' zones, often referred to as **Demilitarized Zones (DMZ's).**



To control the *trust* value of each zone, each firewall interface is assigned a *security level*, which is often represented as a numerical value or even color. For example, in the above diagram, the Trusted Zone could be assigned a security value of 100, the Less Trusted Zone a value of 75, and the Untrusted Zone a value of 0.

As stated previously, traffic from a *higher* security to *lower* security zone is (generally) allowed by default, while traffic from a *lower* security to *higher* security zone requires explicit permission.

* * *

Firewall Services

Firewalls perform the following services:

- **Packet Filtering**
- **Stateful Packet Inspection**
- **Proxying**
- **Network Address Translation (NAT)**

Each will be covered in some detail in this guide.

Packet Filtering

Packet Filtering is one of the core services provided by firewalls. Packets can be filtered (*permitted* or *denied*) based on a wide range of criteria:

- Source address
- Destination address
- Protocol Type (IP, TCP, UDP, ICMP, ESP, etc.)
- Source Port
- Destination Port

Packet filtering is implemented as a **rule-list**:

Number	Action	Protocol	Source Add.	Source Port	Destination Add.	Destination Port
1.	Deny	TCP	Any	Any	172.16.1.5	666
2.	Permit	IP	Any	Any	172.16.1.5	Any
3.	Permit	TCP	Any	Any	172.16.1.1	443
4.	Permit	TCP	Any	Any	172.16.1.1	80
5.	Permit	TCP	Any	Any	172.16.1.10	25
6.	Deny	TCP	66.1.1.5	Any	172.16.1.10	110
7.	Permit	TCP	Any	Any	172.16.1.10	110

The *order* of the rule-list is a critical consideration. The rule-list is *always* parsed from **top-to-bottom**. Thus, more specific rules should always be placed near the *top* of the rule-list, otherwise they may be negated by a previous, more encompassing rule.

Also, an implicit 'deny any' rule usually exists at the bottom of a rule-list, which often can't be removed. Thus, rule-lists that contain **only deny statements** will **prevent all traffic**.

* * *

Stateful Packet Inspection

Stateful packet inspection provides services beyond simple packet-filtering, by additionally *tracking* TCP or UDP sessions between devices.

For example, stateful inspection can track connections that originate from the *trusted* network. This session information is kept in a **state session table**, which allows temporary holes to be opened in the firewall for the *return traffic*, which might otherwise be denied.

Connections from the *untrusted* network to the *trusted* network are also monitored, to prevent Denial of Service (DoS) attacks. If a high number of **half-open sessions** are detected, the firewall can be configured to drop the session (and even block the source), or send an alert message indicating an attack is occurring.

A **half-open TCP** session indicates that the three-way handshake has not yet completed. A **half-open UDP** session indicates that no return UDP traffic has been detected. A large number of half-opened sessions will chew up resources, while preventing legitimate connections from being established.

Proxy Services

A proxy server, by definition, is used to make a request *on behalf* of another device. It essentially serves as a middle-man for communication between devices.

This provides an element of security, by hiding the actual requesting source. All traffic will seem to be originated from the proxy itself.

Traditionally, proxy servers were used to **cache** a local copy of requested external data. This improved performance in limited-bandwidth environments, allowing clients to request data from the *proxy*, instead of the actual *external source*.

Other services that proxy servers can provide:

- Logging
- Content Filtering
- Authentication

* * *

NAT (Network Address Translation)

The rapid growth of the Internet resulted in a shortage of IPv4 addresses. In response, the powers that be designated a specific subset of the IPv4 address space to be *private,* to temporarily alleviate this problem.

A **public address** can be routed on the Internet. Thus, devices that should be Internet accessible (such web or email servers) must be configured with public addresses.

A **private address** is only intended for use within an organization, and can never be routed on the internet. Three private addressing ranges were allocated, one for each IPv4 class:

- Class A - **10.x.x.x**
- Class B - **172.16-31.x.x**
- Class C - **192.168.x.x**

NAT (Network Address Translation) is used to translate between private addresses and public addresses. NAT allows devices configured with a private address to be *stamped* with a public address, thus allowing those devices to communicate across the Internet.

NAT is *not* restricted to just public-to-private address translations, though this is the most common application of NAT. NAT can perform a public-to-public address translation, or a private-to-private address translation as well.

NAT provides an additional benefit – hiding the specific addresses and addressing structure of the internal network.

(Reference: http://www.cisco.com/en/US/tech/tk648/tk361/technologies_white_paper09186a0080194af8.shtml)

* * *

Types of NAT

NAT can be implemented using one of three methods:

Static NAT – performs a static one-to-one translation between two addresses, or between a *port* on one address to a port on another address. Static NAT is most often used to assign a public address to a device behind a NAT-enabled firewall/router.

Dynamic NAT – utilizes a **pool** of global addresses to dynamically translate the outbound traffic of clients behind a NAT-enabled device.

NAT Overload or **Port Address Translation (PAT) –** translates the outbound traffic of clients to unique port numbers off of a *single* global address. PAT is necessary when the number of internal clients exceeds the available global addresses.

NAT Terminology

Specific terms are used to identify the various NAT addresses:

- **Inside Local –** the specific IP address assigned to an *inside* host behind a NAT-enabled device (usually a *private* address).
- **Inside Global –** the address that identifies an *inside* host to the *outside* world (usually a *public* address). Essentially, this is the dynamically or statically-assigned public address assigned to a private host.
- **Outside Global** – the address assigned to an *outside* host (usually a *public* address).
- **Outside Local** – the address that identifies an *outside* host to the *inside* network. Often, this is the **same** address as the Outside Global. However, it is occasionally necessary to translate an outside (usually *public*) address to an inside (usually *private*) address.

For simplicity sake, it is generally acceptable to associate **global** addresses with **public** addresses, and **local** addresses with **private** addresses. However, remember that public-to-public and private-to-private translation is still possible. **Inside** hosts are within the local network, while **outside** hosts are external to the local network.

* * *

NAT Terminology Example

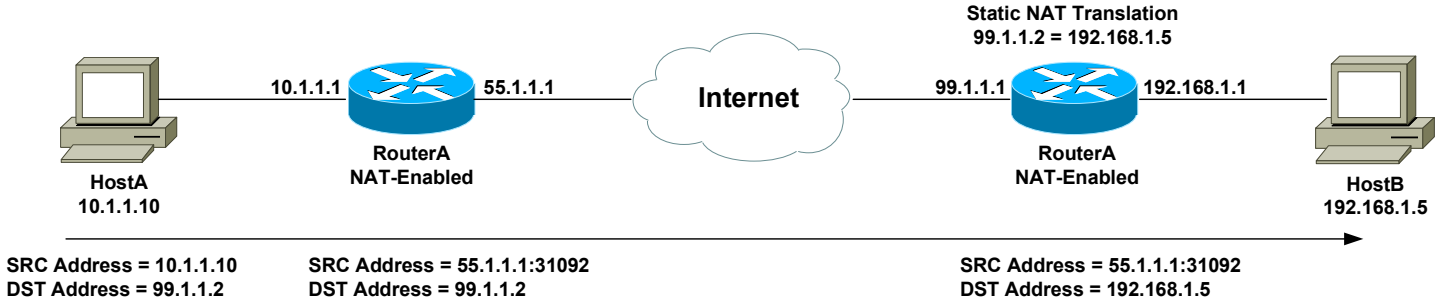
Consider the above example. For a connection *from* HostA *to* HostB, the NAT addresses are identified as follows:

- **Inside Local Address** - 10.1.1.10
- **Inside Global Address** - 55.1.1.1
- **Outside Global Address** – 99.1.1.2
- **Outside Local Address –** 99.1.1.2

HostA's configured address is *10.1.1.10*, and is identified as its *Inside Local* address. When HostA communicates with the Internet, it is stamped with RouterA's public address, using PAT. Thus, HostA's *Inside Global* address will become *55.1.1.1*.

When HostA communicates with HostB, it will access HostB's *Outside Global* address of *99.1.1.2*. In this instance, the *Outside Local* address is also *99.1.1.2*. HostA is never aware of HostB's configured address.

It is possible to map an address from the local network (such as 10.1.1.5) to the global address of the remote device (in this case, 99.1.1.2). This may be required if a legacy device exists that will only communicate with the local subnet. In this instance, the *Outside Local* address would be *10.1.1.5*.



The above example demonstrates how the source (SRC) and destination (DST) IP addresses within the Network-Layer header are translated by NAT.

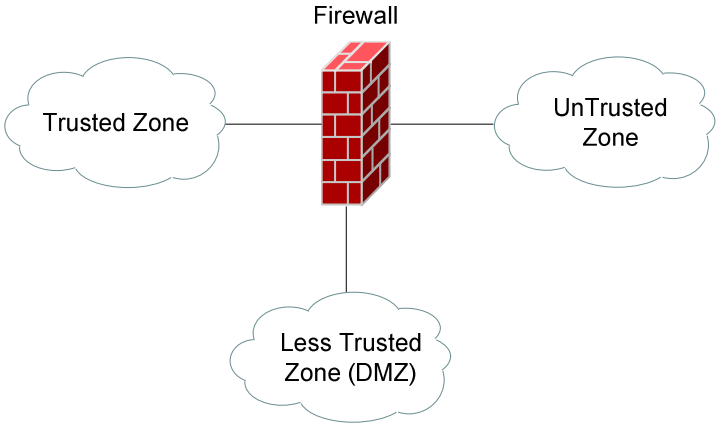
(Reference: *http://www.cisco.com/warp/public/556/8.html*)

* * *

Implementing a DMZ

As briefly described earlier, a DMZ is essentially a **less trusted zone** that sits between the **trusted zone** (generally the LAN) and the **untrusted zone** (generally the Internet). Devices that provide services to the untrusted world are generally placed in the DMZ, to provide separation from the trusted network.

A single firewall with multiple ports can be used to implement a logical DMZ:



A more secure DMZ (referred to as a **screened subnet)** utilizes multiple firewalls:

* * *