TRANSPORT LAYER

- It is responsible to perform process to process communication for which it uses 16 bit port address.
- Their are following two protocols defined in transport layer.

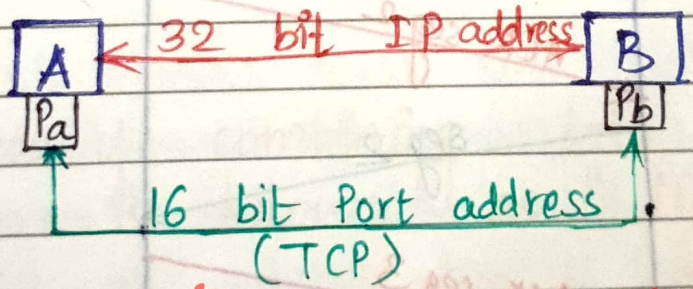
1) TCP - Transmission control Protocol.
2) UDP - User Datagram Protocol.

1) Transmission Control Protocol (TCP) :-

- **TCP services :-**

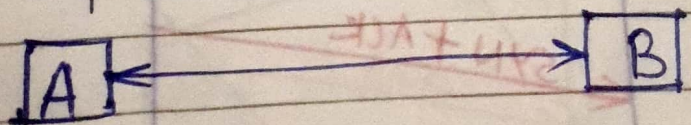
a) Process to Process Communication :-

- When two process communicate with each other, for which it uses 16 bit port address it is known as "Process to Process Communication."



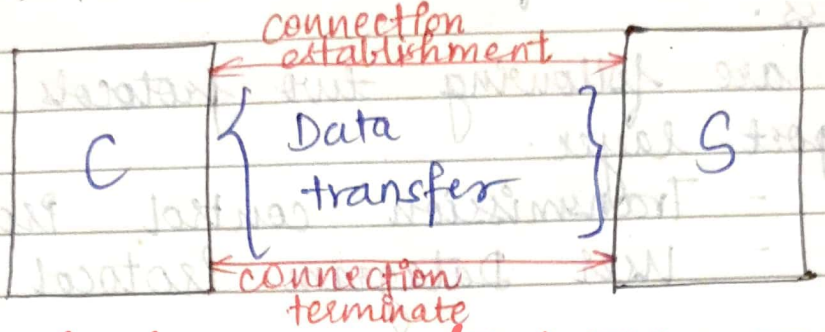
b) Full Duplex Communication :-

- When both the parties communicate with each other at the same time, it is known as "Full duplex communication."



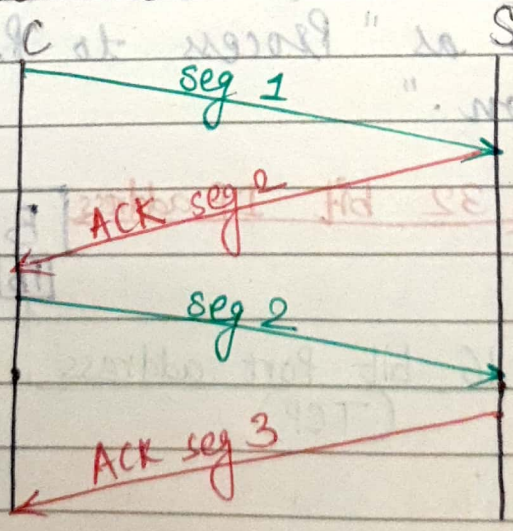
c) Connection Oriented Service :-

- TCP starts the data transfer only after connection establishment, once the data transfer is over, it terminates the connection.



d) Reliable :-

- This service will make sure that the data (segment) is successfully received by the receiver as no segment sender can proceed to send next set of segments until it receives acknowledgement for the previous.

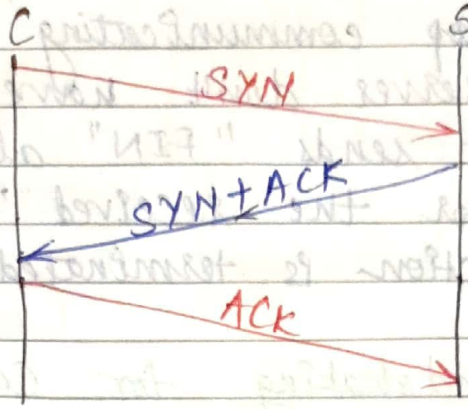


CN NOTES BY PROF. AKN

V.V.V.V.Imp

Connection Management in TCP :-

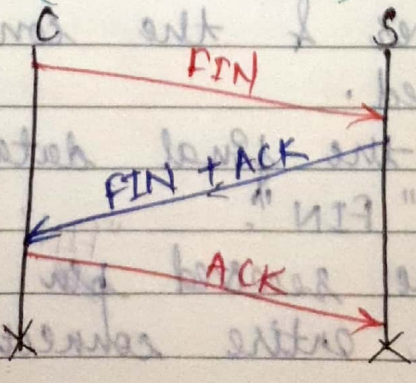
1) 3-way Handshaking for connection Establishment :-



1) Client to establish a connection with server, it sends "SYN" segment to server.
2) If server is ready to accept the connection it sends "ACK" for the receives & to establish connection with client, it sends "SYN" along with "ACK".
3) Client sends an "ACK" for the received "SYN" & the connection is established.

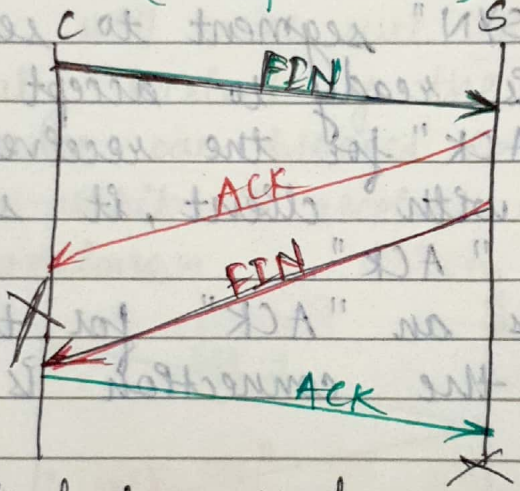
Imp Viva ★ NOTE :- The process of combining data & ACK together is known as "Piggy Backing."

*2) 3-way Handshaking for Connection termination (Full close) :-



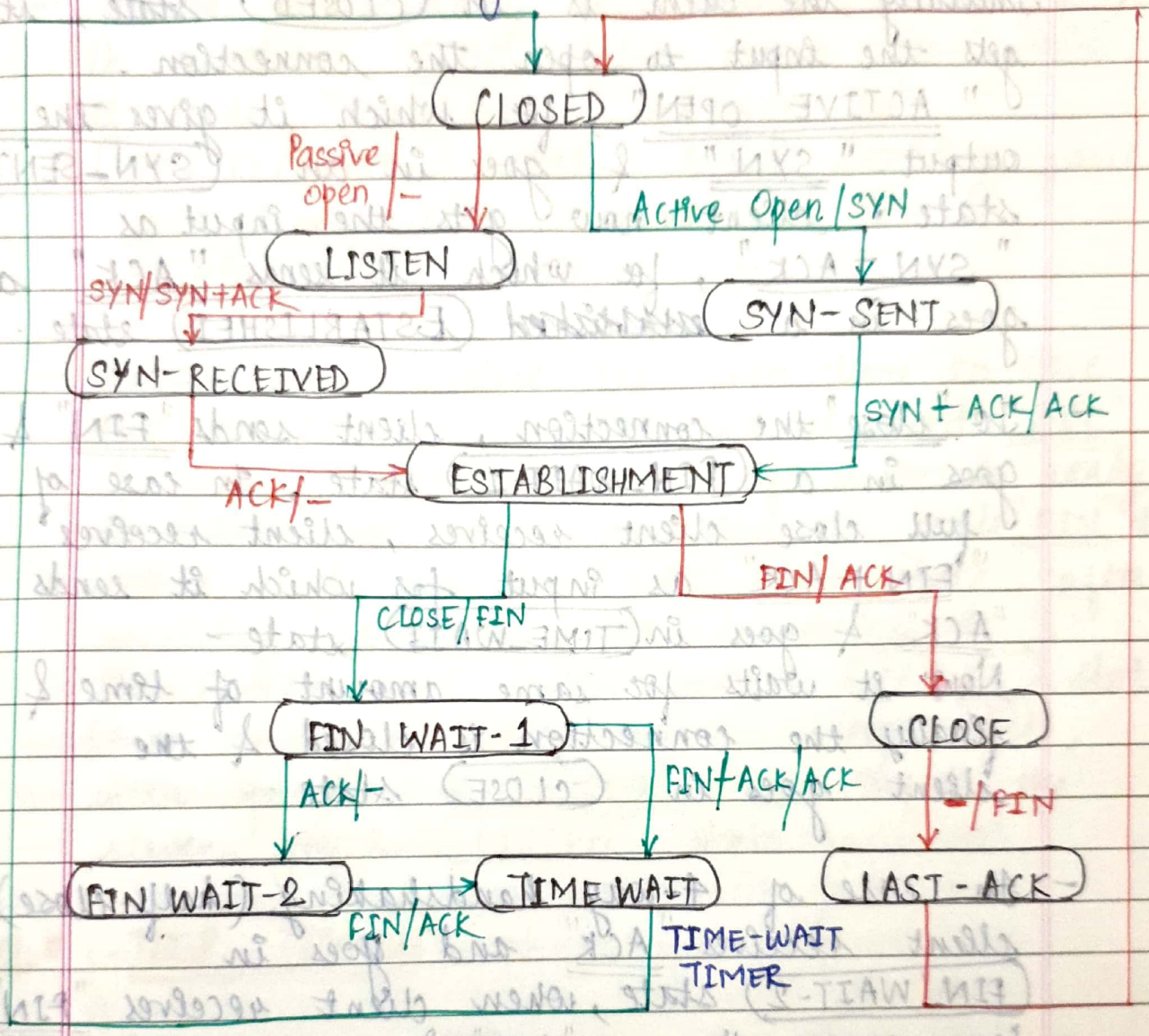
1) Client after sending the entire data, it sends "FIN" to server.
2) If server is ready to terminate the connection it sends "ACK", this will make client to stop communicating with server, & even if server don't have any data to send to client, it sends "FIN" along with "ACK".
3) Now client for the received "FIN" send "ACK" & the connection is terminated at both the ends.

3) 4 way Handshaking for Connection termination (Half Close) :-



1) Assume client has send some data for processing to server & immediately sends "FIN" to terminate the connection.
2) Since, server is not ready to terminate the connection. It sends "ACK" for the received "FIN". Now, client can not send any data to server & the connection is half way closed.
3) After sending the final data to client, server sends "FIN".
4) Client for the received "FIN" sends "ACK" & finally the entire connection is closed.

- State Transition Diagram of TCP connection management :-



— :- Server side

— :- Client side

CN NOTES BY PROF. AKN

Page No.
Date

Client Side Explanation :-

- Initially the client is in (CLOSED) state, it gets the input to open the connection. "ACTIVE OPEN", for which it gives the output "SYN" & goes in an (SYN_SENT) state. client now gets the input as "SYN + ACK", for which it sends "ACK" and goes in a ~~established~~ (ESTABLISHED) state.

- To "close" the connection, client sends "FIN" & goes in a (FIN_WAIT-1) state, In case of full close client receives, client receives "FIN + ACK" as input for which it sends "ACK" & goes in (TIME_WAIT) state - Now it waits for same amount of time & finally the connection is closed & the client goes in (CLOSE) state.

- In case of 4-way handshaking (half-close) client receives "ACK" and goes in (FIN_WAIT-2) state, when client receives "FIN" from server it gets "ACK" & goes in a (TIME_WAIT) state, it wait for some amout of time & the connection is get CLOSE and client goes in (CLOSE) state.

CN NOTES BY PROF. AKN

CN NOTES BY PROF. AKN

- **Server side Explanation:-**

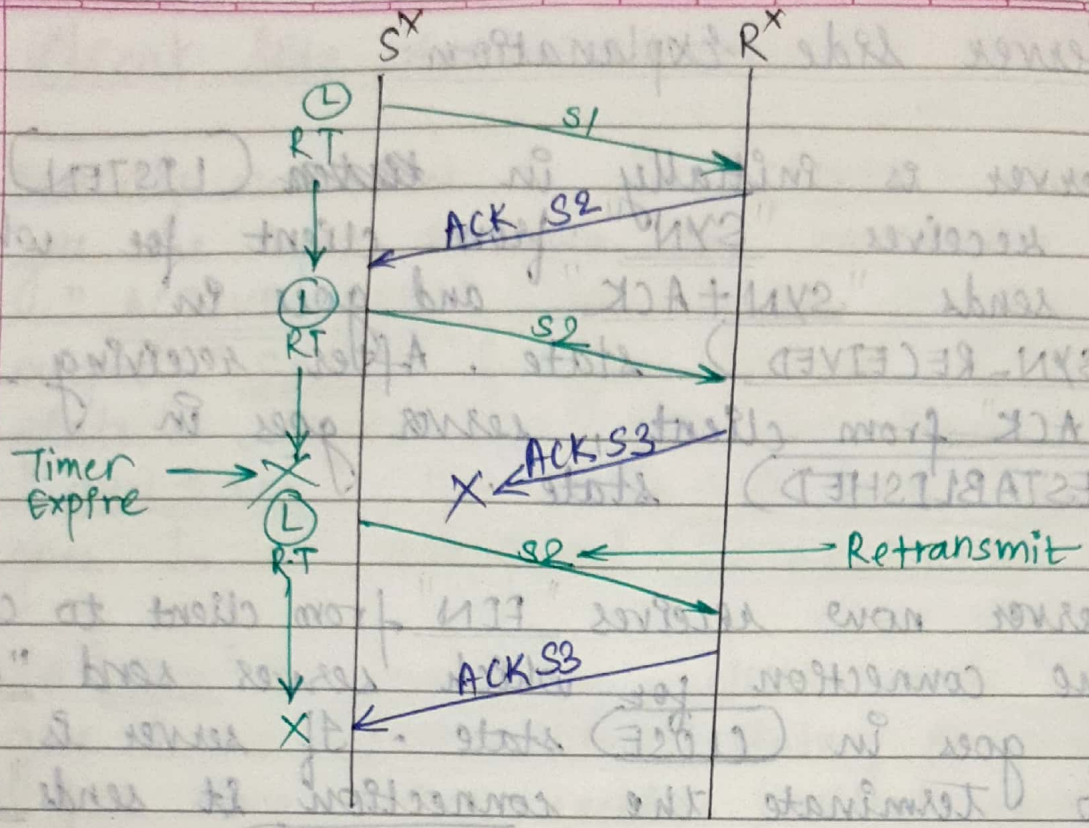
- Server is initially in ~~listen~~ (LISTEN) state, it receives "SYN" from client for which it sends "SYN+ACK" and goes in (SYN-RECEIVED) state. After receiving final "ACK" from client, server goes in (ESTABLISHED) state.

- Server now receives "FIN" from client to close the connection for which server send "ACK" & goes in (CLOSE) state. If server is ready to terminate the connection it sends "FIN" to client and goes in (LAST_ACK) state, after receiving "LAST_ACK", the connection is terminated and server goes is (CLOSED) state; Since server is never in a sleep mode, with ~~passive~~ "PASSIVE OPEN" input, the server goes in (LISTEN) state.

Imp • **TCP Timers :-**

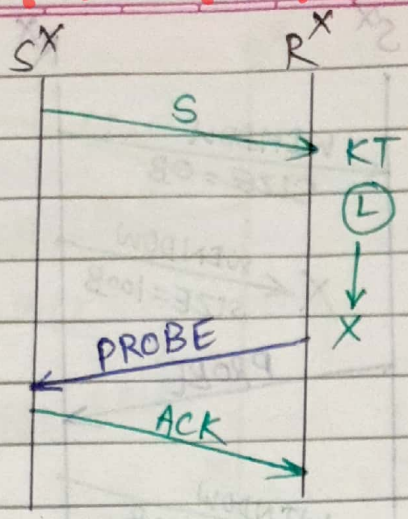
① **Retransmission Timer :-**

- This timer is generally set by the sender after sending a segment, the sender should receive an acknowledgement for the segment before this timer expires, if acknowledgement is not received the segment should be retransmitted.



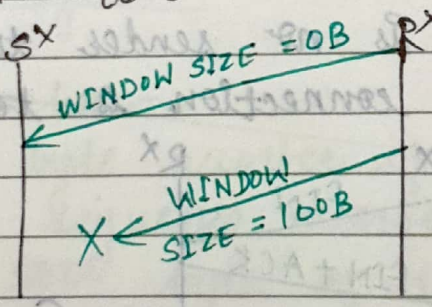
(2) Keep alive Timer :- (KT)

- Whenever the connection is set up between the sender & the receiver & there is no communication from sender for more amount of time, the receiver is confused with the concept why sender is not sending the data.
- To avoid this confusion it starts with keep-alive timer, before this timer expires there should be some segment coming from sender, if not then receiver sends "PROBE" for which sender has to send acknowledgement.

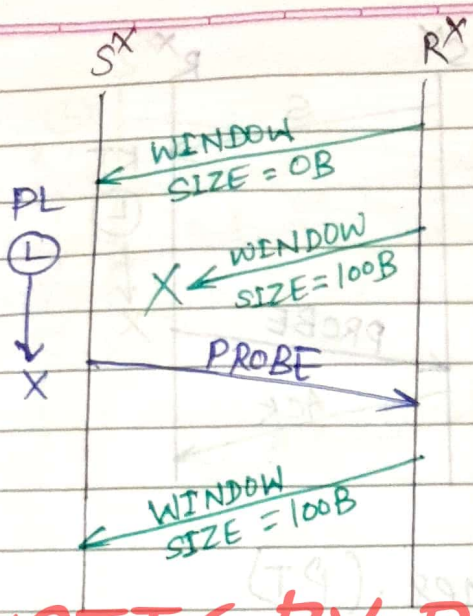


(3) Persistent Timer. (PT)

- Receiver announces window size equal to 0 byte, hence sender stop sending data to receiver.
- Now receiver advertises the new window size which is lost.

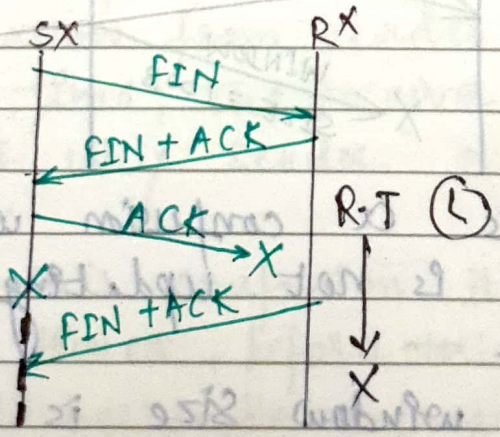


- Now, there is a confusion with sender that why receiver is not updating with the window size.
- So, when the window size is 0 byte is advertise initially, sender should maintain persistent timer.
- Before this timer expires, if there is no update from receiver about ~~receiver size~~ window size, sender would send "PROBE" for which receiver has to reply back with the latest window size.

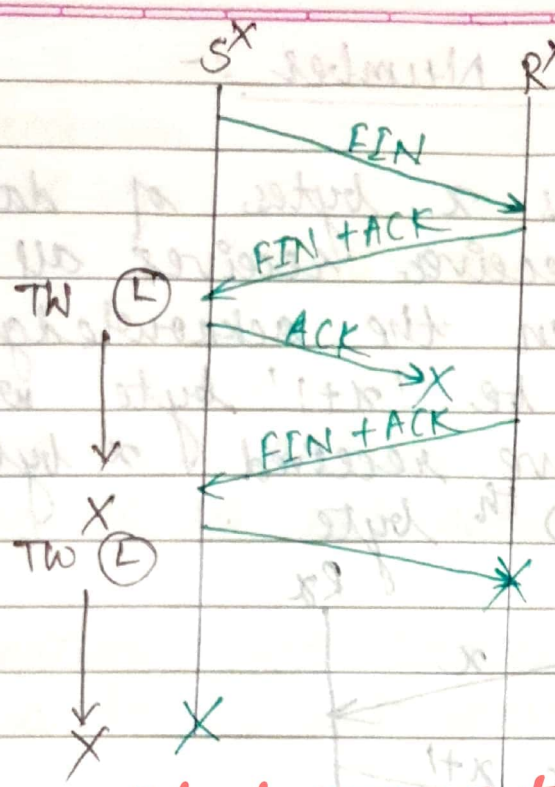


④ TIME WAIT TIMER :-

- Assume sender sends last ACK & terminates the connection if the last ACK is lost, if it receiver is retransmit the FIN+ACK again their is no sender to accept it as sender the connection is terminate the connection.



- Therefore, after sending last ACK sender starts with Time-wait timer & waits for some amount of time to handle such situation.



TCP Features :-

Numbering system :-

Sequence Number :-

The starting byte number of every segment is known as sequence number.

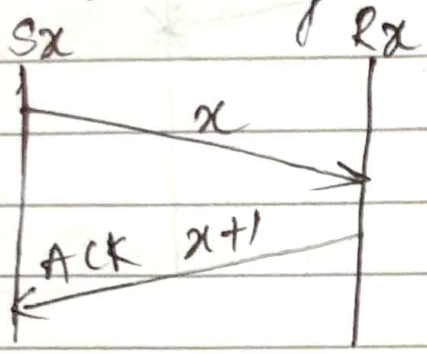
For eg - If 5000 byte of data is to be transmitted in 5 segments each of 1000 bytes & the starting byte number of first segment is 10001, show all 5 segments with sequence number.

S1 : 10,001 to 11,000 sequence no - 10,001.
S2 : 11,001 to 12,000 sequence no. - 11,001
S3 : 12,001 to 13,000 sequence no. - 12,001
S4 : 13,001 to 14,000 sequence no. - 13,001
S5 : 14,001 to 15,000 sequence no. - 14,001

CN NOTES BY PROF. AKN

2) Acknowledgement Number :-

- If sender sends x bytes of data to the receiver and receiver receives all x byte successfully then the acknowledgement number would be 'x+1' byte which indicates I have received x bytes, send me (x+1)th byte.



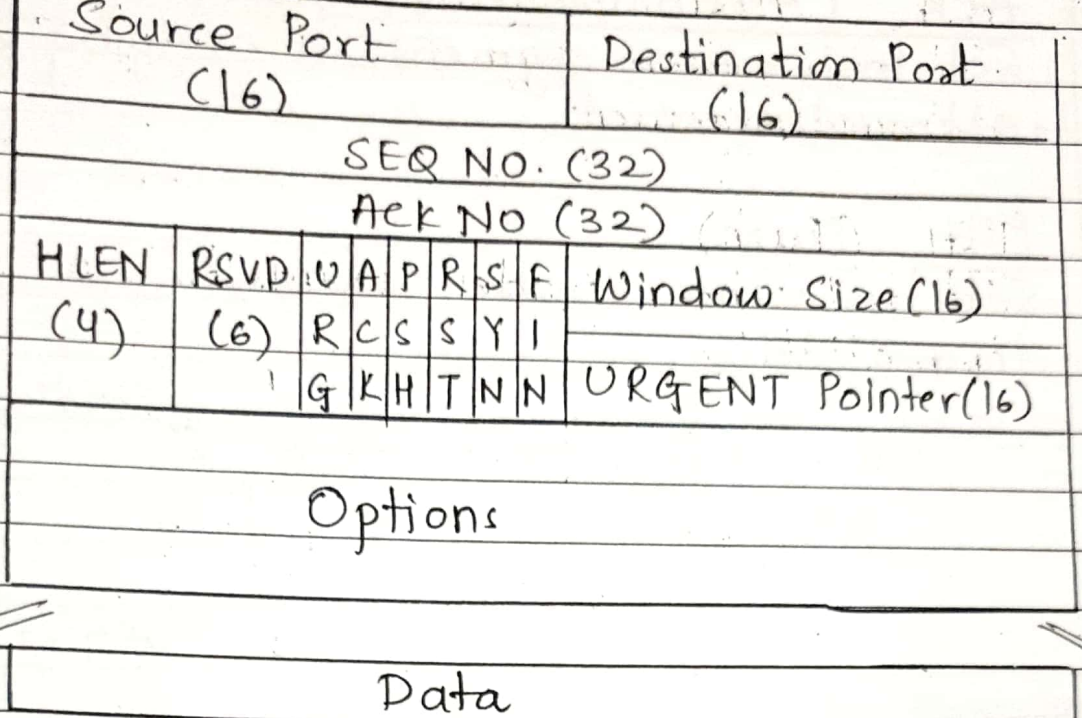
2) Error control :-
3) Flow control :-
4) Congestion control :-

} Refer Further lectures.

CN NOTES BY PROF. AKN

Camlin | Page
Date / /

TCP Header:



TCP Header is of minimum 20 bytes (without option) and 20 bytes to 60 bytes (with opt^n)

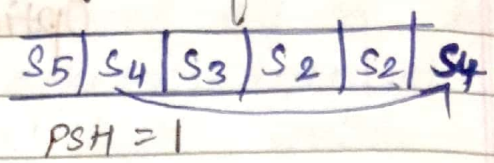
- **Source Port Address :-**
  It is a 16 bit port address of sender process.
- **Destination Port Address :-**
  It is a 16 bit port address of receiver process
- **Sequence Number** (Refer TCP features)
- **Acknowledgement Number** (Refer TCP features)
- **HLEN** (Same as IPV4)
- **Reserved :-**
- **Flags :-**

1) **URG (URGENT FLAG) :-**
- It tells the receiver that their an important data to be read first pointed by the urgent pointer.

2) **ACK (Acknowledgement Flag) :-**
- It tells that the segment is carrying an acknowledgement.

3) **PSH (Push Flag) :-**
- It is used by the sender process to

understand and that their is some segment that needs to be push first from the sender process.



4) RST (Reset Flag) :-
- It is used for resetting the connection.

5) SYN (Synchronization Flag) :-
- It is used for establishing the connection.

6) FIN (Finish Flag) :-
- It is used for terminating the connection.

- **Window Size :-**
- It is the receiver's buffer size that tells sender how much amount of data receiver can accept at a time.

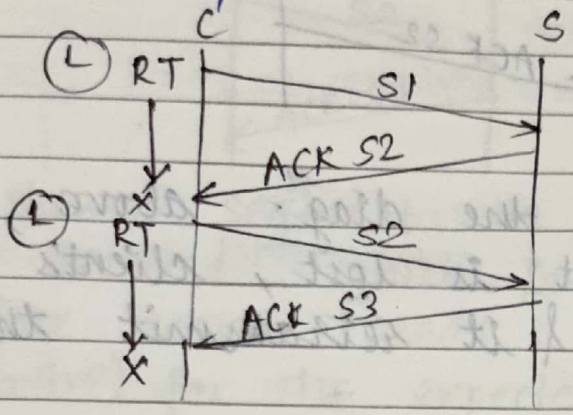
- **Checksum** (same as IPV4)

- **Urgent Pointer :-**
- It is valid only when URG is set to 1 & it points to the important data that is to be read first.

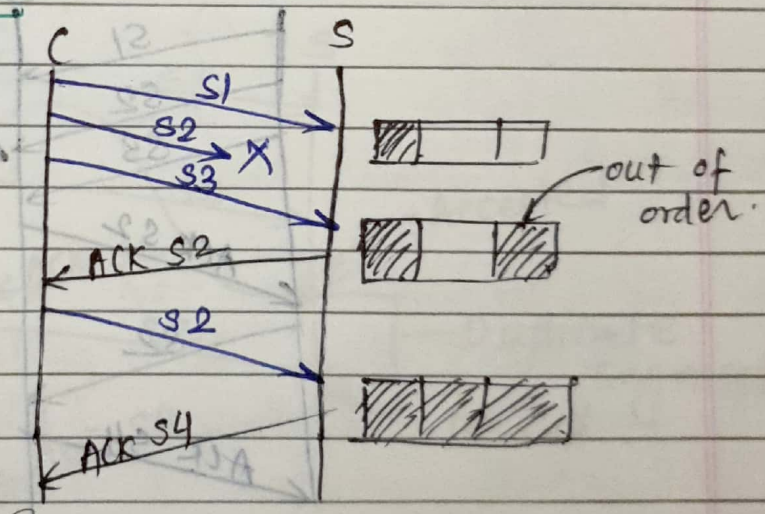
2] Error control :-

There are various types of errors found in network which transport layer possibly take care using following approaches :

- Normal operation :-

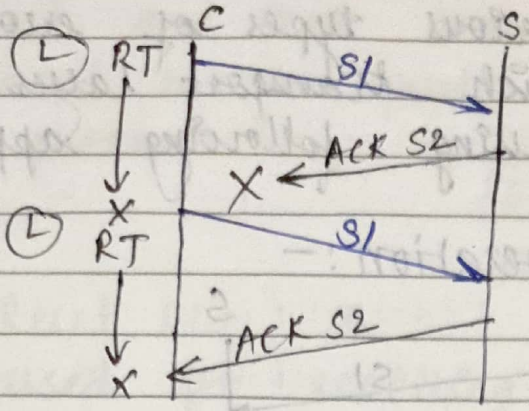


- Lost Segment :



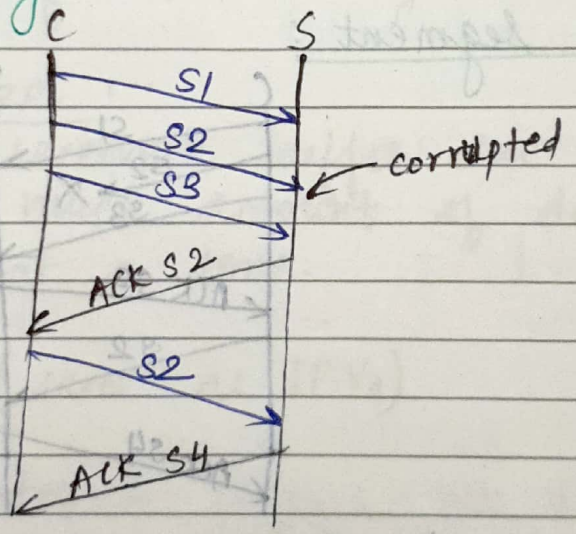
- Client sends 3 segments & S2 is lost, server should immediately send an acknowledgment for the lost segment, client should retransmit that segment again.

2) Lost ACK :-



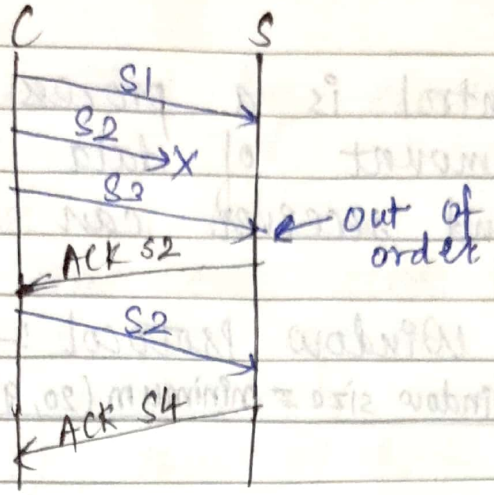
- As shown in the diag. above, acknowledgement is lost, client's retransmition timer expires & it retransmit the segment S1.

3) Corrupted segment :-



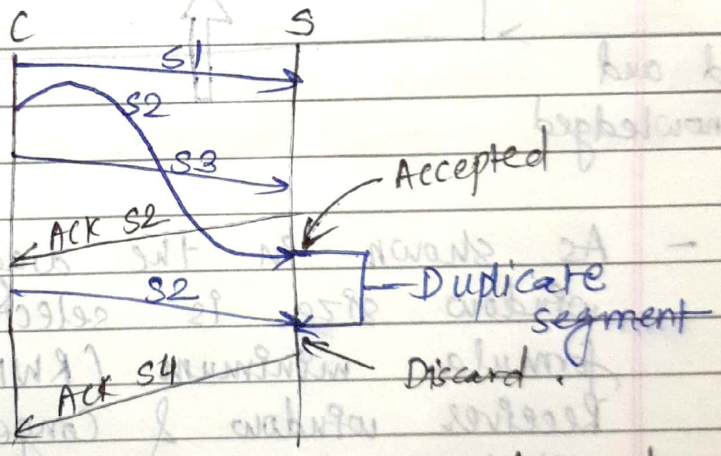
- As shown in the diag. S2 was corrupted, server immediately sends an acknowledgement (Negative) for the corrupted segment & client retransmits that.

4) Out of Order :-



- As shown segment S2 was accepted & S3 is received which is out of order, therefore server sends an acknowledgement (Negative) for the expected segment i.e S2.

5) Duplicate segment :-



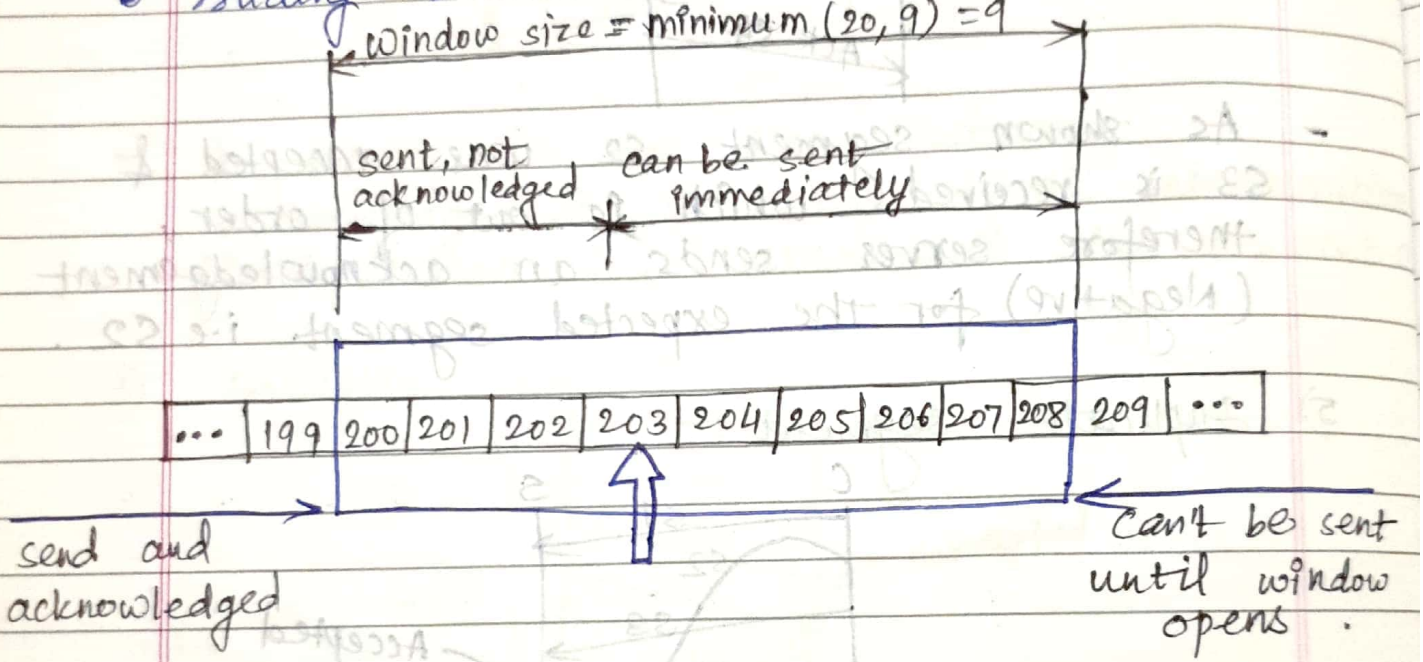
- As shown in the diag. S2 was delayed, server receives S3 which is out of order & hence sends ACK for S2, client retransmit S2 & since S2 was delay it is received twice. Such segment are received more than one which is duplicate segment & receiver should accept the first segment & discard the rest duplicate segment.

Page No.
Date

3) Flow Control :-

- Flow control is a process that regulates the amount of data a sender can send and receiver can receive at a time.

- **Sliding Window Protocol :-**



- As shown in the diag. above the window size is selected based on the formula minimum (RWND & CWND - Receiver window & Congestion window)
- whatever bytes are in the window would be send & continue to remain in the window till their acknowledged.
- when x- bytes are acknowledged the left window moves to right by x-byte (closing) and the right wall move to right side by x bytes (opening).
- Only the movement of right wall to left side (shrinking) should be avoided.

Page No.
Date

- Silly Window Syndrome :-
- Syndrome created by sender :-

- Assume sender has to send 5 bytes of data but at a time only sends 1 byte of data.
- For sending 1 byte of data 40 bytes of header is attached (20 byte TCP + 20 byte IP), thus 41 Bytes of data would be transmitted over the network.
- So such 5 segment each carrying 1 byte of data would be transmitted causing 205 bytes of network data will be utilized.

- If 5 bytes of data was sent in single segment with 40 bytes of header, only 45 bytes of network data would be utilized.

- Solution :-
- Nagle's Solution :-

Step 1 :- Send the 1st segment as it is even if it is of 1 byte.
Step 2 :- Start the buffer & the timer, collect all the segments in the buffer.
Step 3 :- Empty the buffer either when timer expires or buffer is full.

Syndrome created by receiver :-

In this assume receivers buffer size is 1000 bytes & the buffer is full, so it announces window size = 0B to sender & hence sender stop sending the segment. Receiver consumes 1 byte, announces window size = 10 byte to sender & hence sender sends 1B of data with 40 Bytes of header (20BTCP + 20BIP), causing 41 bytes of network data utilization.
Receiver accepts 1 byte of data, the buffer becomes full & story continuous.

Solution :

Clark solution :

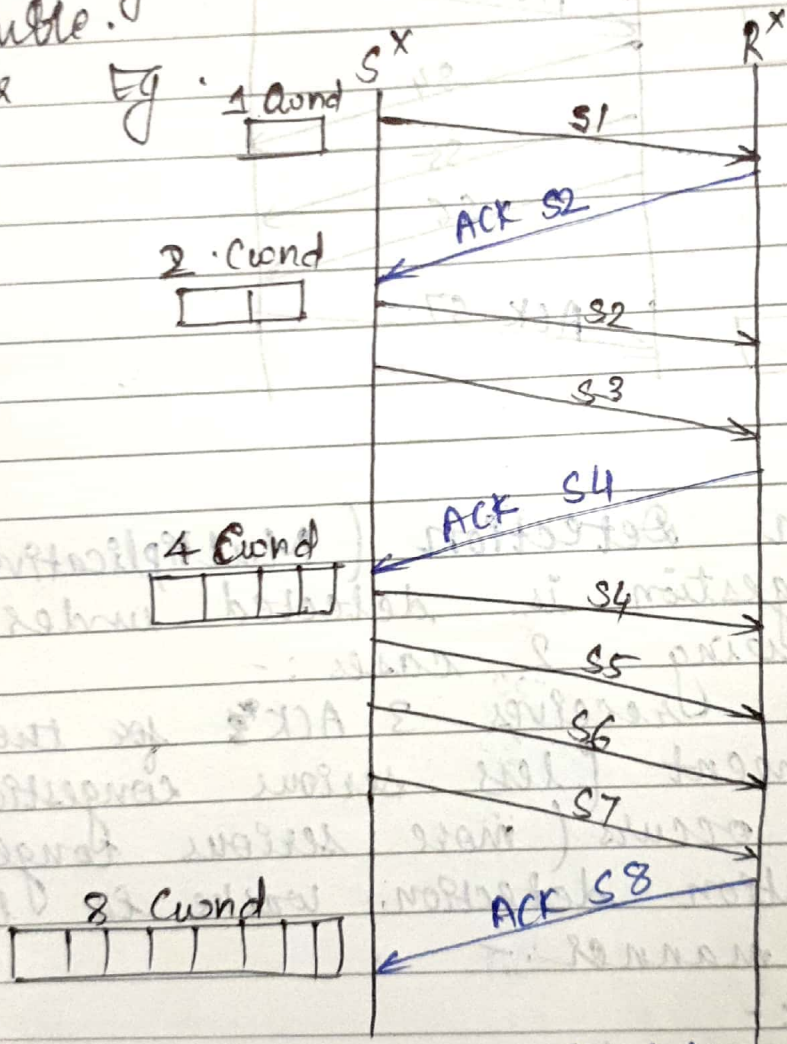
Don't advertise the window size till it is half empty.

CN NOTES BY PROF. AKN

Congestion Control :-

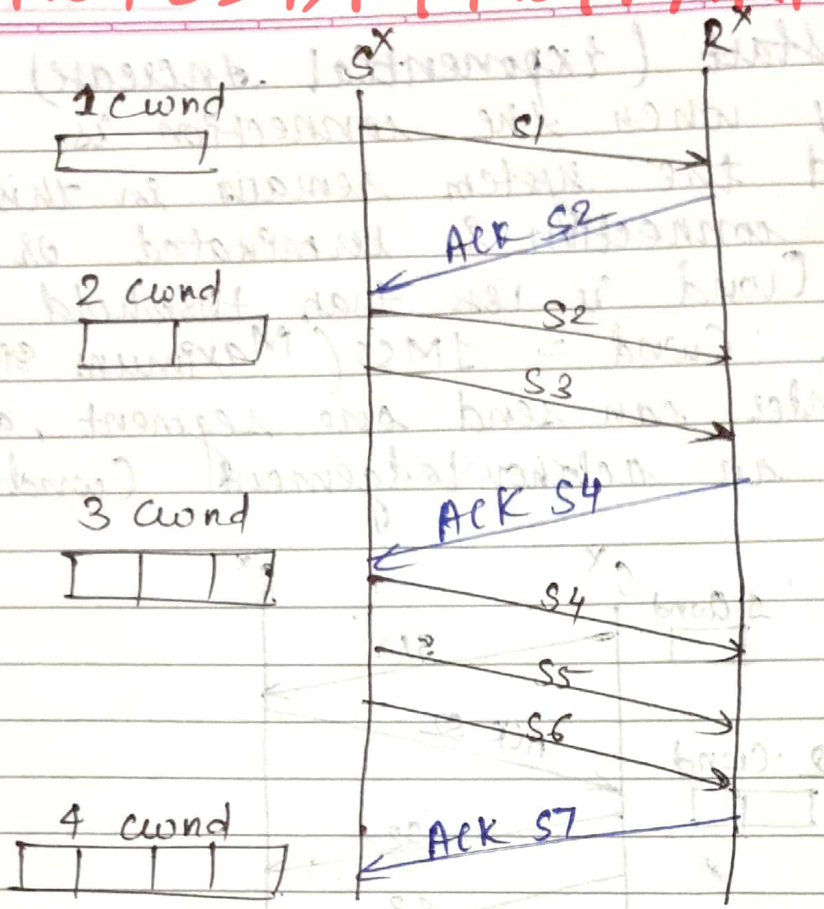
Congestion control focuses on reducing the overall traffic congestion in the network.
For congestion control TCP uses 3 phases :-
1) Slow start (exponential Increase)
2) Congestion Avoidance (Additive Increase)
3) Congestion Detection (Multiplicative Decrease)

1) Slow Start (exponential increase) :-
- Initially when the connection is established the system remains in this phase till the connection is terminated or cwnd is less than threshold value.
- Initially Cwnd = 1MSS (Maximum size segment) so a sender can send one segment, after receiving an acknowledgement Cwnd is double.
- For Eg :



2) Congestion Avoidance (Additive Increase) :-
- In this whenever an acknowledgement is received for the send segment, Cwnd is incremented by 1.
- The sender remains in this phase until connection is terminated, congestion is detected.

Page No.
Date



3) Congestion Detection (Multiplicative Decrease)
The congestion is detected under the following 2 cases :-
a) If sender receives 3 ACK's for the same segment (less serious congestion).
b) Time out occurs (more serious congestion)
- The congestion detection works in the following manner :-

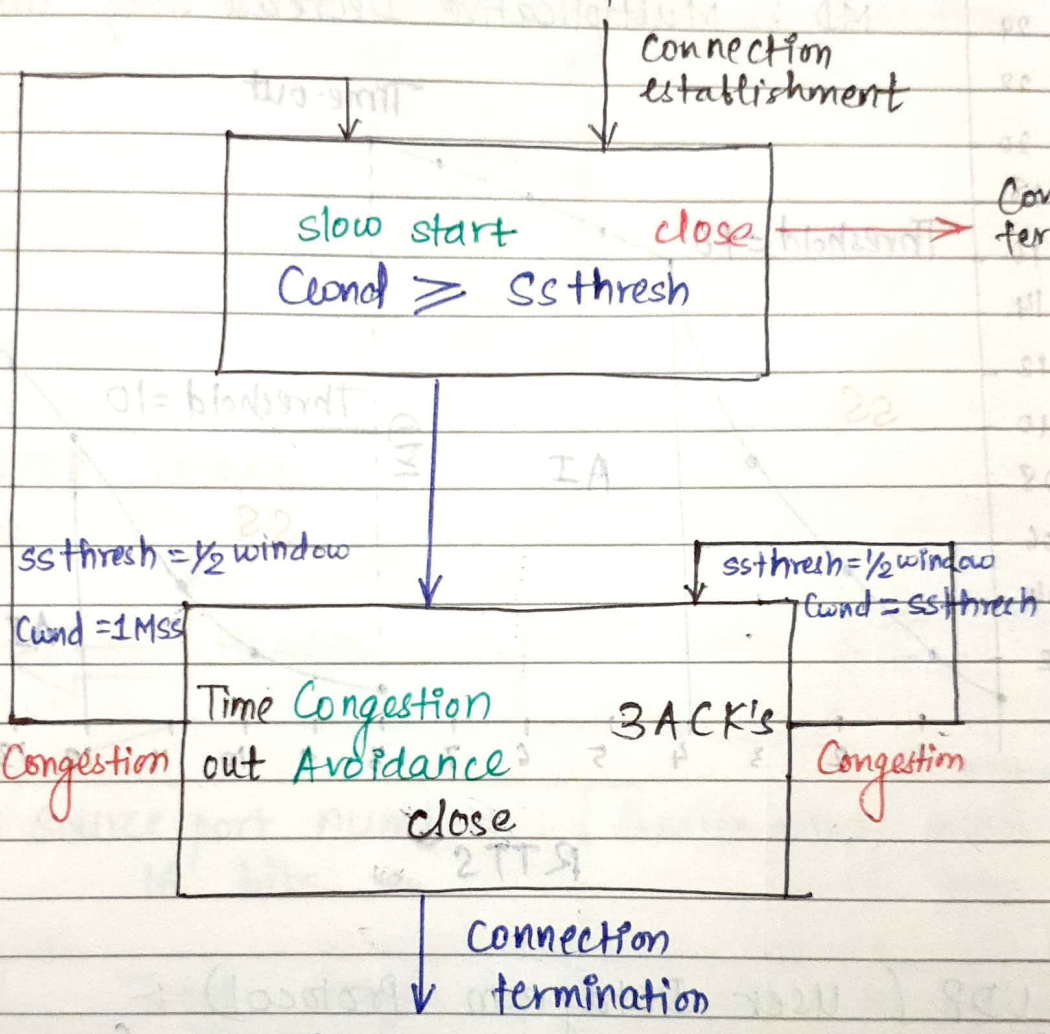
i) 3 ACK :-
Step 1 : Threshold = half Cwnd
Step 2 : Cwnd = Threshold
Step 3 : Start with phase 2.

ii) Time Out :-
~~Step~~

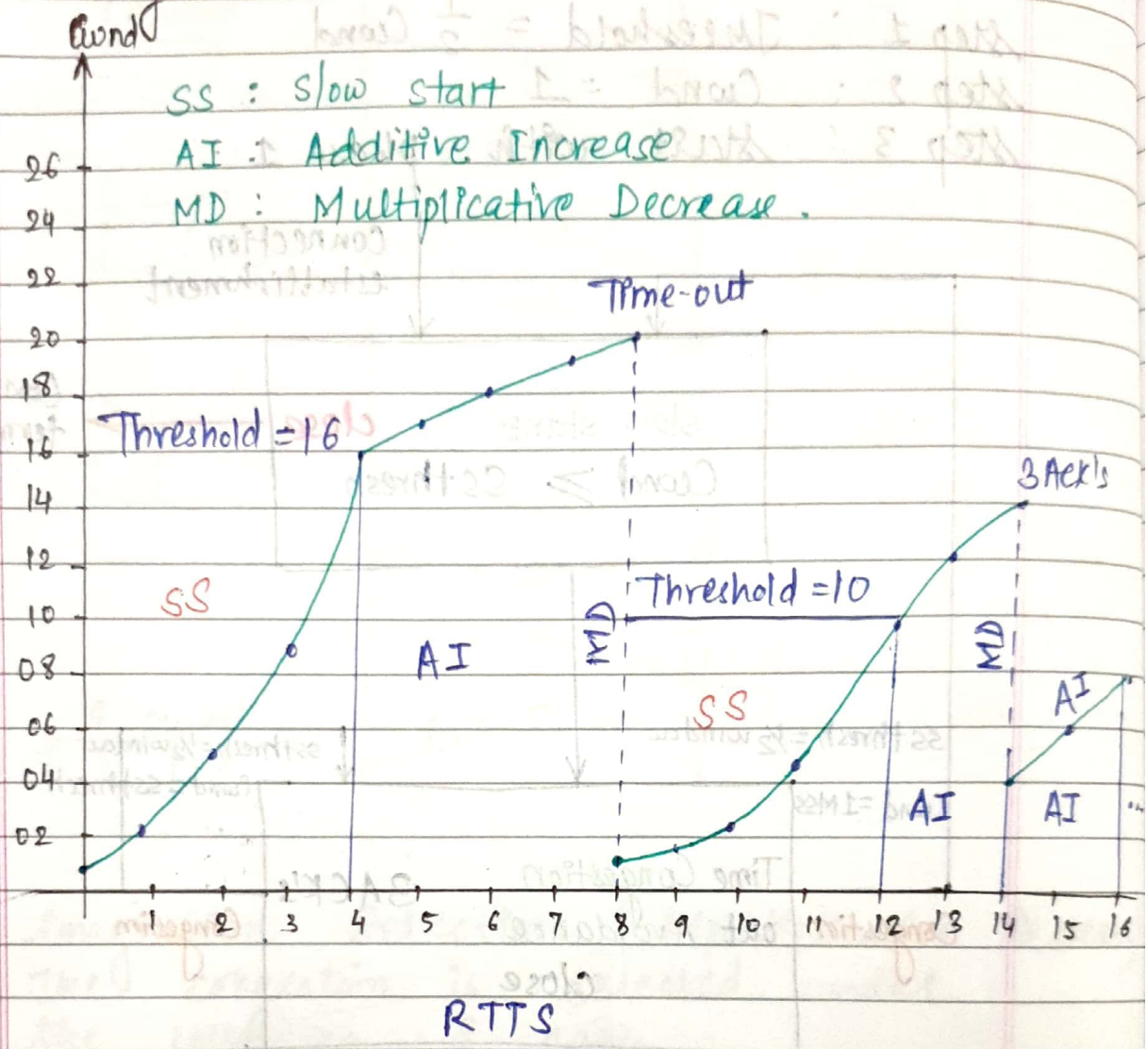
Step 1 : Threshold = $\frac{1}{2}$ Cwnd

Step 2 : Cwnd = 1

Step 3 : Start with phase 1.



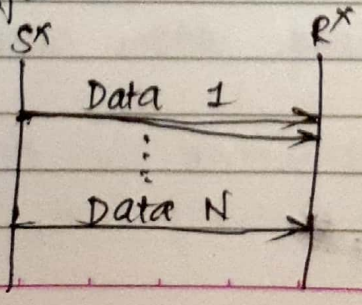
For eg.



- VDP (User Datagram Protocol) :-

1) Process to Process Communication (same as TCP)
2) Full Duplex Communication (same as TCP)
3) Connection-less service :-

— In this sender never establish connection with receiver & directly starts the data transfer.

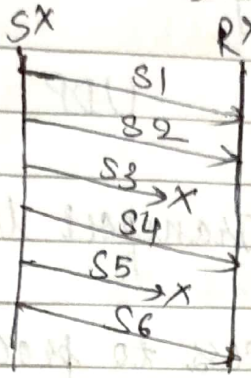


CN NOTES BY PROF. AKN

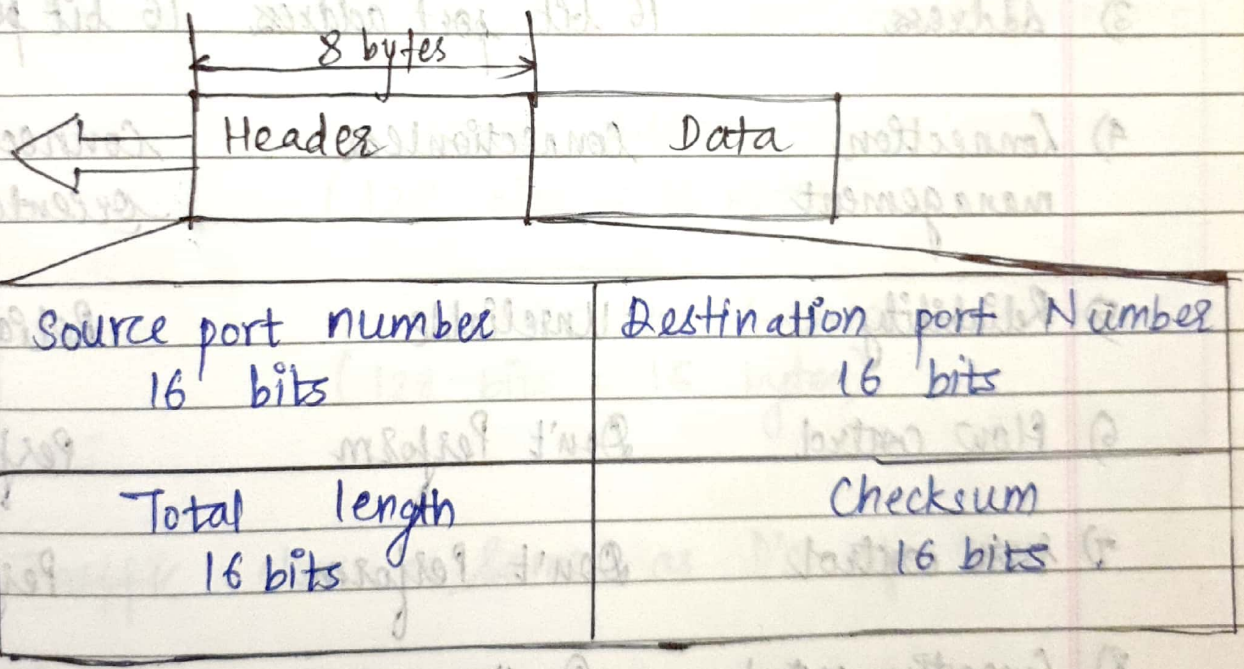
4) Reliable

Unreliable communication :-

- Unlike TCP UDP expects any acknowledgement & continuously send the data without understanding the fact that data received by receiver or not.



- UDP Header :-



1) Source port Number :-

- It is a 16 bit port address of source process of sending side.

2) Destination port Number :-

- It is a 16 bit port address of destination process of receiving side.

3) Total length = Header length + Data length

4) Checksum :- same as IPV4.

- Similarity & Difference between TCP & UDP.

Parameter	UDP	TCP
1) OSI Layer	Transport layer	Transport layer
2) Type of communication	Process to process communication.	Process to process communication.
3) Address	16 bit port address	16 bit port address
4) Connection management	Connectionless	Connection oriented.
5) Reliability	Unreliable	Reliable
6) Flow control	Don't Perform	Performed
7) Error control	Don't Perform	Performed
8) Congestion control	Don't perform	Performed
9) Flags	Not supported	Supported.