~~Checksum~~

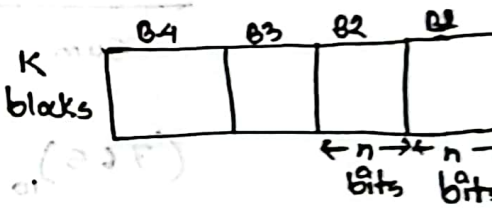
Error Detection Mechanism:

(1) Checksum:

The senden is going to ~~c~~create the massege and calculate the checksum from it. This checksum will be attach with the original massege.

Sender → (Create & Checksum)

1. Breaks the original massege into 'K' blocks.
   each blocks size is n-bit.
2. Sum all the K data box.
3. Add the canny to the sum.
4. Perform 1's complement to the sum.



Example:

Msg 1 : $10011001 = (153)_{10}$

Msg 2 : $11100010 = (226)_{10}$

Msg 3 : $00100100 = (36)_{10}$

Msg 4 : $10000100 = (132)_{10}$

$10\ 001000\ 11 = (547)_{10}$

Canny ← 10 → +10

00100101

1's Complement → $11011010 = (218)_{10}$

Checksum

Receiver: (Validates Checksum)

Receiver collects all the msg bow along with the checksum. then it adds all of them. If all the bits of the sum result is 1, then the massege is error free.

Example:

msg 1 = 153

msg 2 = 226

msg 3 = 36

msg 4 = 132

Checksum = 218

Sum = $(765)_{10}$

$(765)_{10}$ = 10 11111101

Canny ← → +10

11111111

as the sum result is all 1, the msg has no error;

Example: With corrupted bit at 2nd and 32th.

msg 1 = 1 0 0 1 1 0 [1] 0 1 = $(155)_{10}$
msg 2 = 1 1 1 0 0 0 1 0 = $(226)_{10}$
msg 3 = 0 0 1 0 0 1 0 0 = $(36)_{10}$
msg 4 = [0] 0 0 0 0 1 0 0 = $(4)_{10}$
Checksum = 1 1 0 1 1 0 1 0 = $(218)_{10}$

Sum = $(639)_{10}$

* Corrupted bit will be the complement of the main bit.

$(639)_{10}$ = 1 0 0 1 1 1 0 1 1 1

Carry → + 1 0

1 0 0 0 0 0 0 1

As the sum result is not 1, the msg has some error. So the msg will not forword to the receiver. It will be given back to the sender.

② CRC (Cyclic Redundant Code):

In this method sum reductant bit will be generated, which is called crc. A divisor/generator will be used to generate CRC. The protocol that the sender or receiver will use, they will decide what will be the divisor/generator. So the divisor/generator will be known to both senders and receivers.

* CRC Generation Step: (Senden)

1. Find the length of the divisor, L
2. Append (L-1) bits to the original message.
3. Perform binary divisor operation.
4. Remainder of the divisor operation will be the CRC.
5. The CRC must be (L-1) bit

* CRC Generation Step: (Receiver)

1. Perform the divisor operation with the same divisor
2. If the remainder is 0, then there is no error.

Example: $M(u) = u^7 + u^6 + u^4 + u^2 + 1$

$M(u) = 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1$

$G(u) = u^4 + u^3 + u^2 + 1$

$G(u) = 1\ 1\ 1\ 0\ 1$

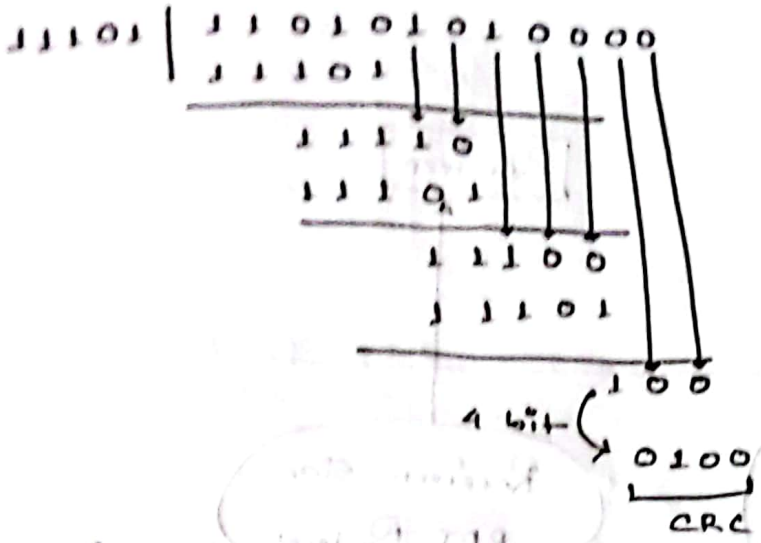
divisor = 5 bit.

Append = 4 bit

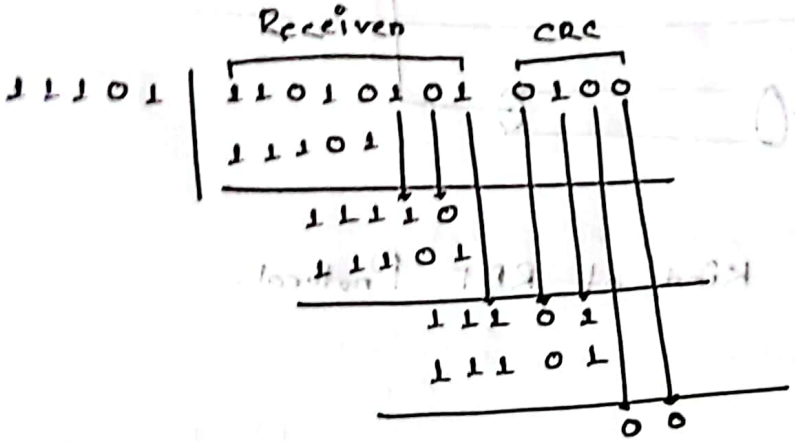
So, $M'(u) = 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ \underbrace{0\ 0\ 0\ 0}$

appended 4 bit

CamScanner

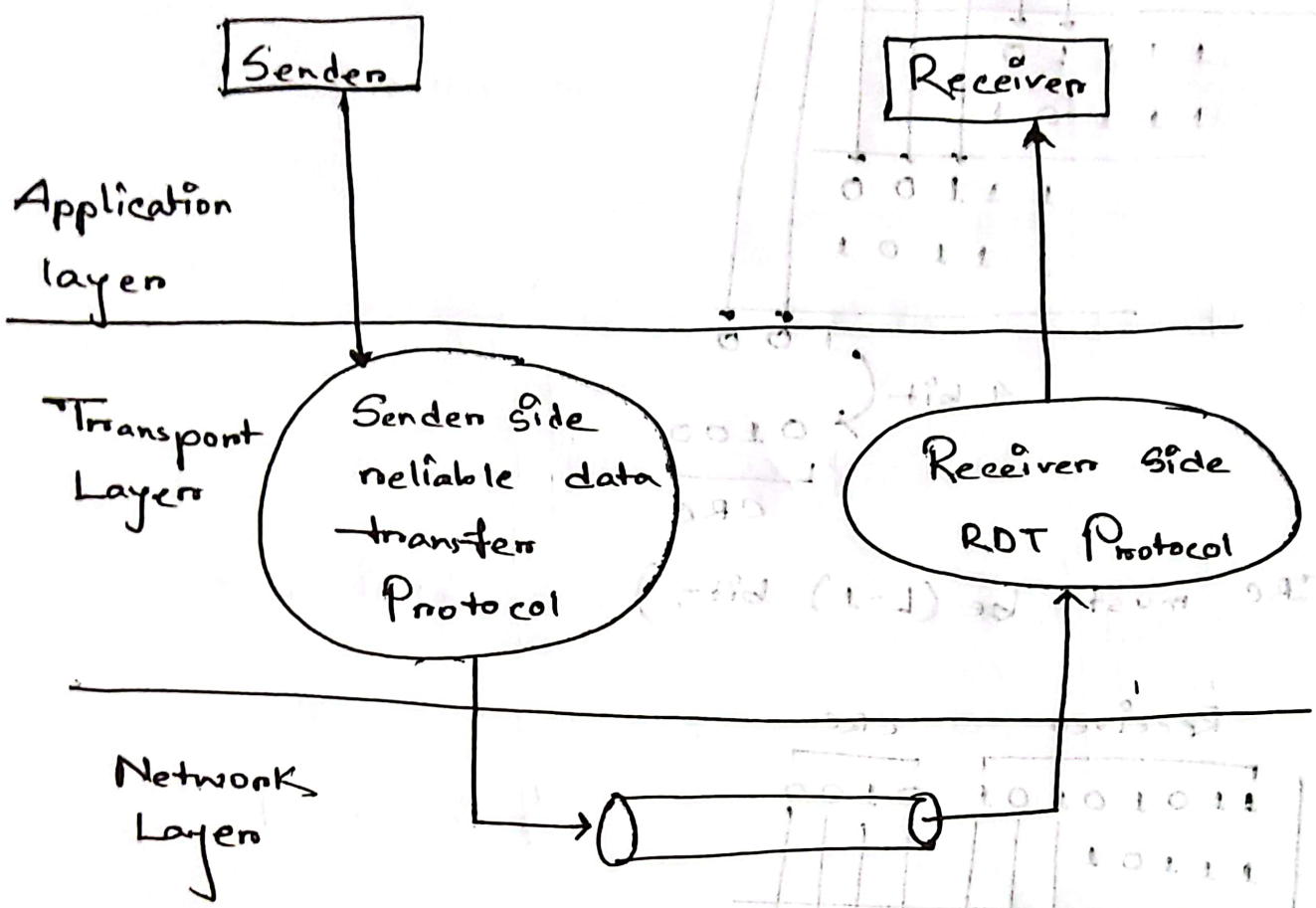


(as CRC must be (L-1) bit.)



As the result is all '0', so there is no error in that massage.

"Computer Networks" (02)



TCP is a Kind of RDT Protocol.

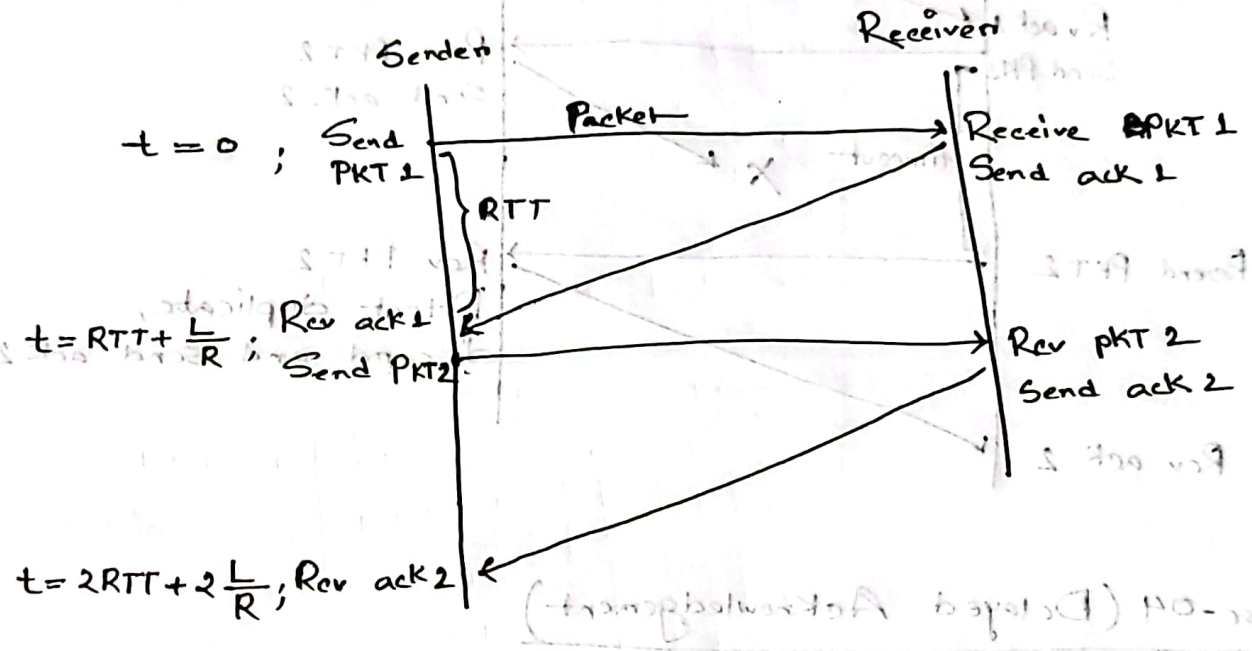
CamScanner

"Computer Networks" (03)

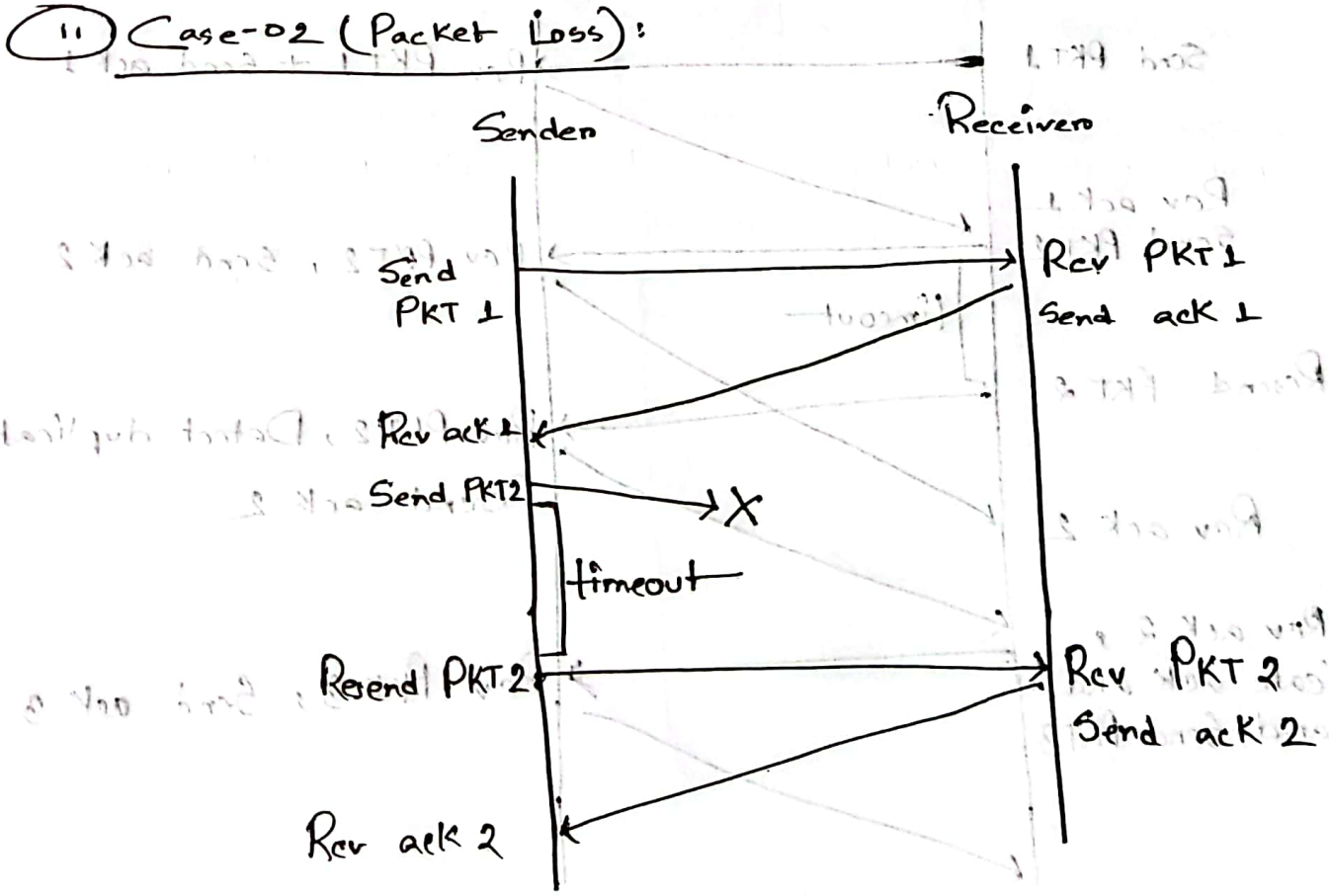
→ Stop and Wait Operation:

Ensures Reliability.

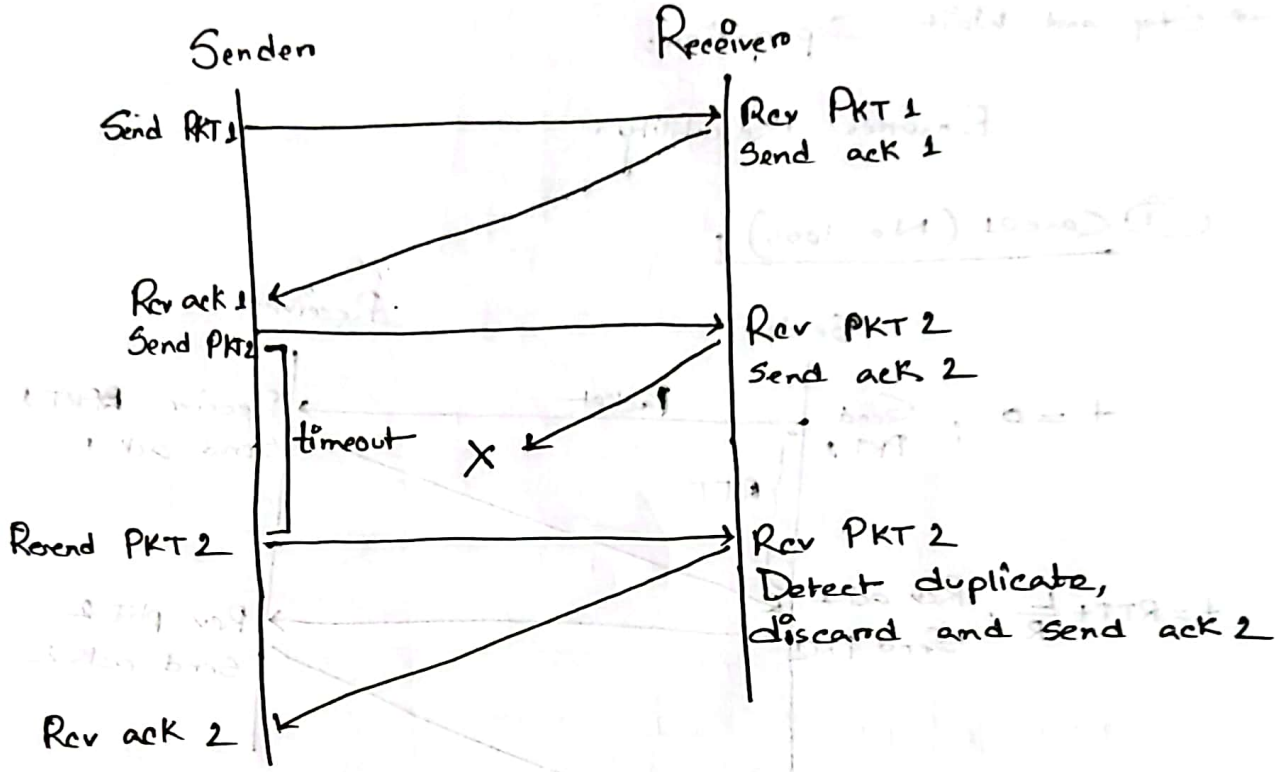
(i) Case-01 (No loss):



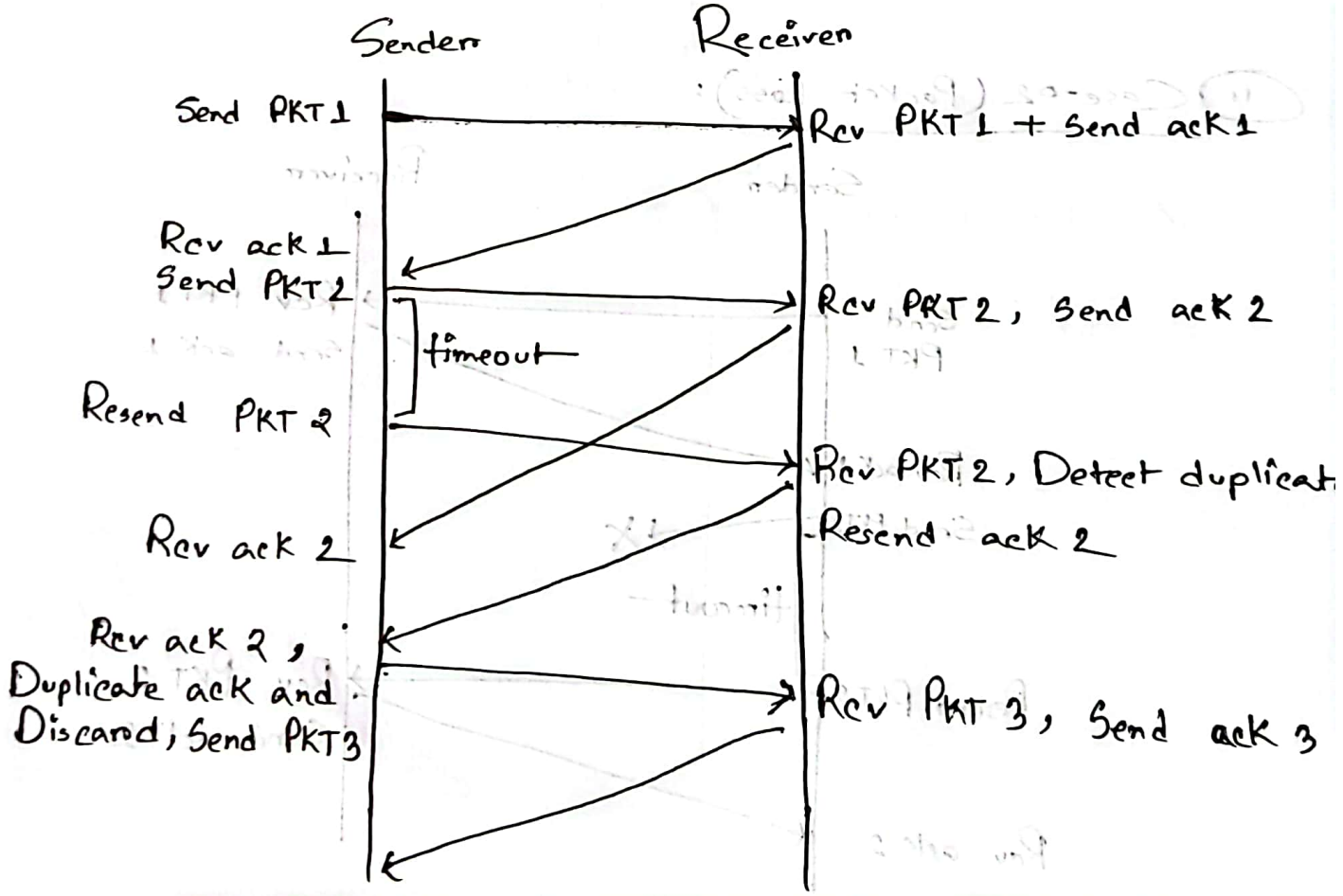
(ii) Case-02 (Packet Loss):



(iii) Case-03 (Acknowledgement Loss):



(iv) Case-04 (Delayed Acknowledgement)



CamScanner

→ Example: Consider,

$L = 8\ B$

$R = 0.5\ Gbps$

$$d_{trans} = \frac{L}{R} = \frac{8 \times 8}{0.5 \times 10^9}$$

$$= 0.128\ \mu sec$$

Total time sending 1 Packet, $= 0.3\ ms + 0.128\ \mu sec$

$= 300.128\ \mu sec.$

$$\text{Utilization at time} = \frac{d_{trans}}{\text{total time}} = \frac{0.128}{300.128}\% = 0.043\%$$

→ Pipe-lining Protocol:

Pipe-Lining is used for better utilization.

It is a technique in which sender sends multiple packets without waiting for the corresponding acknowledgement.

1. Go-Back-N Protocol
2. Selective Repeat Protocol.

→ Example: Consider,

$L = 8$ B

$R = 0.5$ Gbps

$$d_{trans} = \frac{L}{R} = \frac{8 \times 8}{0.5 \times 10^9}$$

$= 0.128$ μsec

Total time sending 1 Packet, $= 0.3$ ms $+ 0.128$ μsec

$= 300.128$ μsec.

$$\text{Utilization of time} = \frac{d_{trans}}{\text{total time}} = \frac{0.128}{300.128}\% = 0.043\%$$

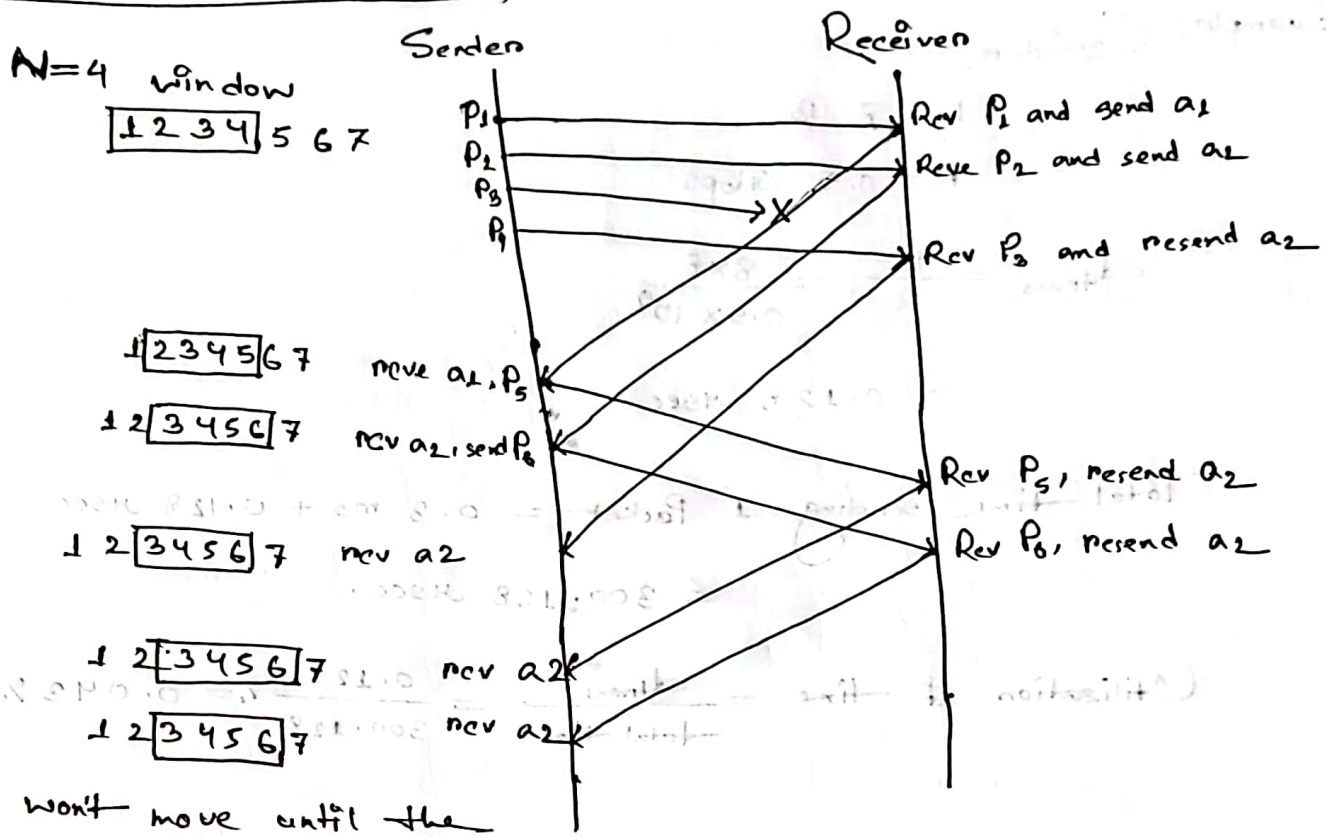
→ Pipe-Lining Protocol:

Pipe-Lining is used for better utilization.

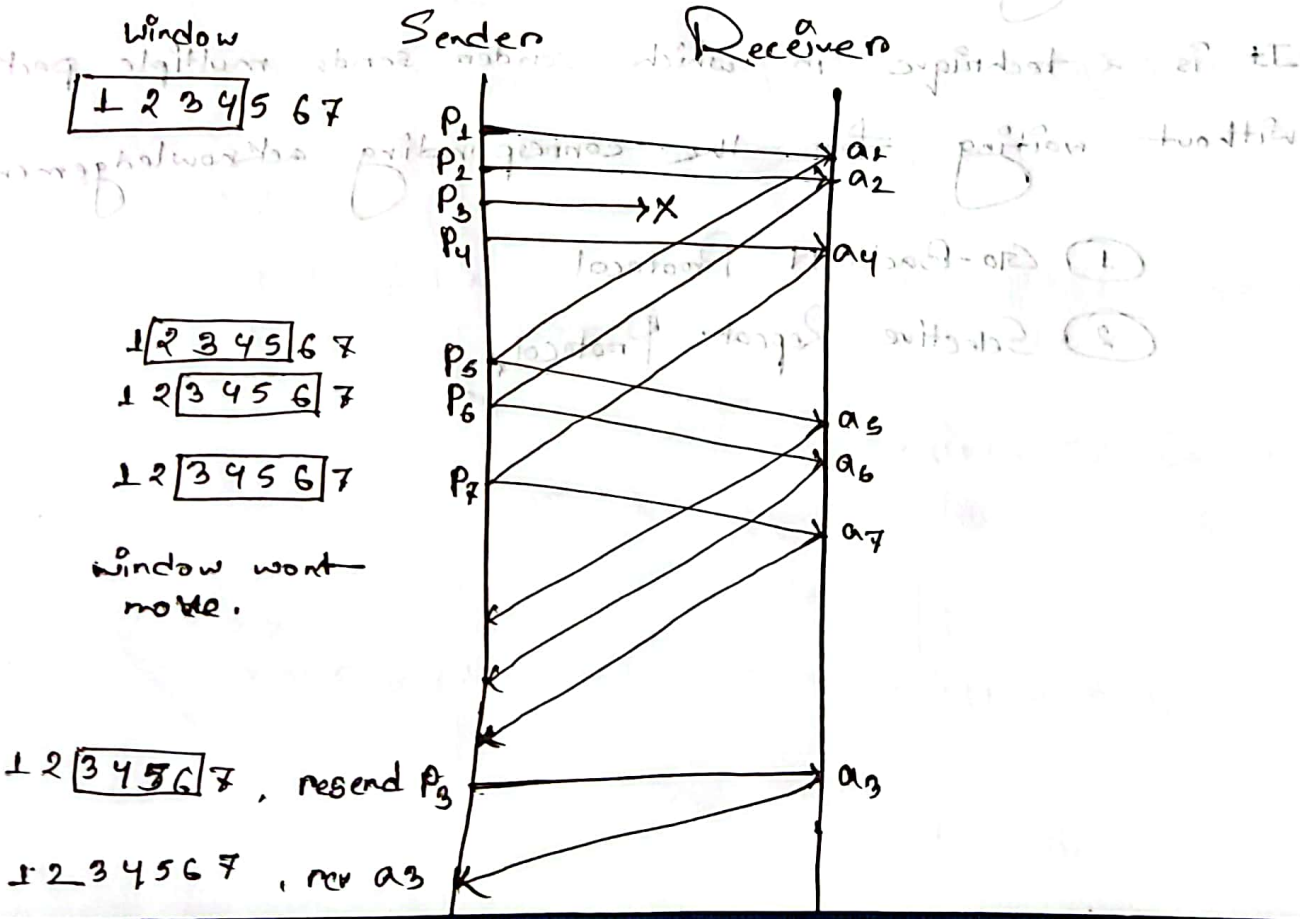
It is a technique in which sender sends multiple packets without waiting for the corresponding acknowledgement.

1. Go-Back-N Protocol
2. Selective Repeat Protocol.

(1) Go-Back-N Protocol:

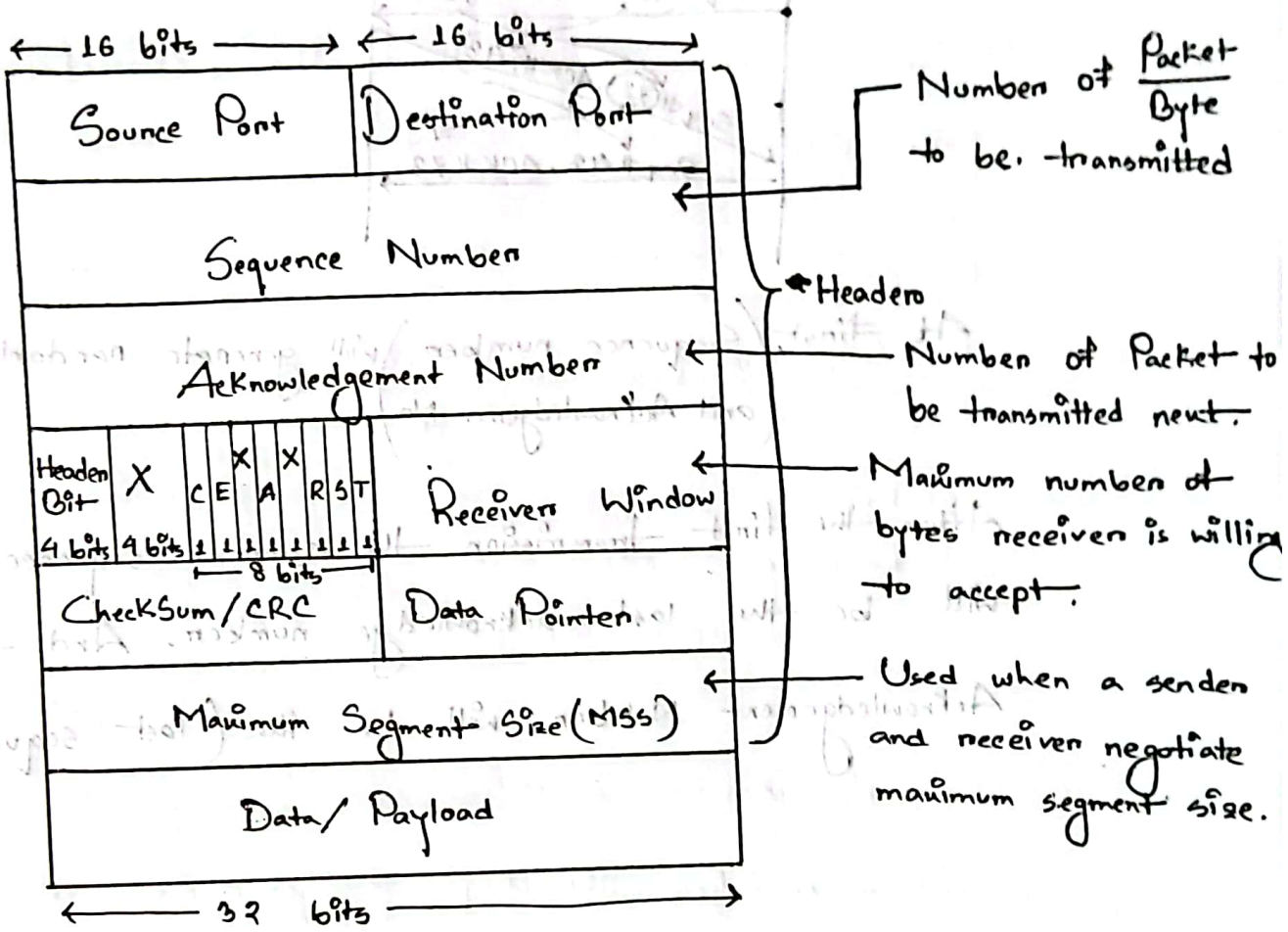


(2) Selective Repeat:



"Computer Networks" (04)

TCP Segment Header : (20 - 60 Bytes)



R = Reset
S = Syncronization
T = Termination

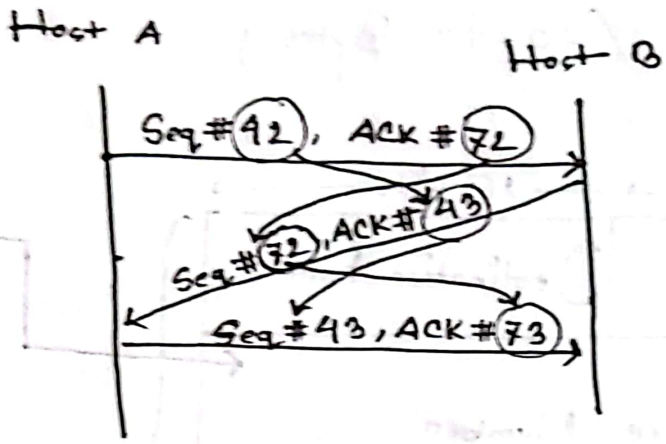
} TCP Connection Establish, mangge and terminate Control.

A = Acknowledgement bit

C, E → Congestion Control

TCP is connection Oriented.

* TCP Sequence and Acknowledgement Number:



At first, (Sequence number and Acknowledgemen No) will generate randomly.

After the first transmission, the current sequence number will be the last acknowledge number. And the current Acknowledgement Number will be the (last sequence number + 1).

3-way Handshaking:

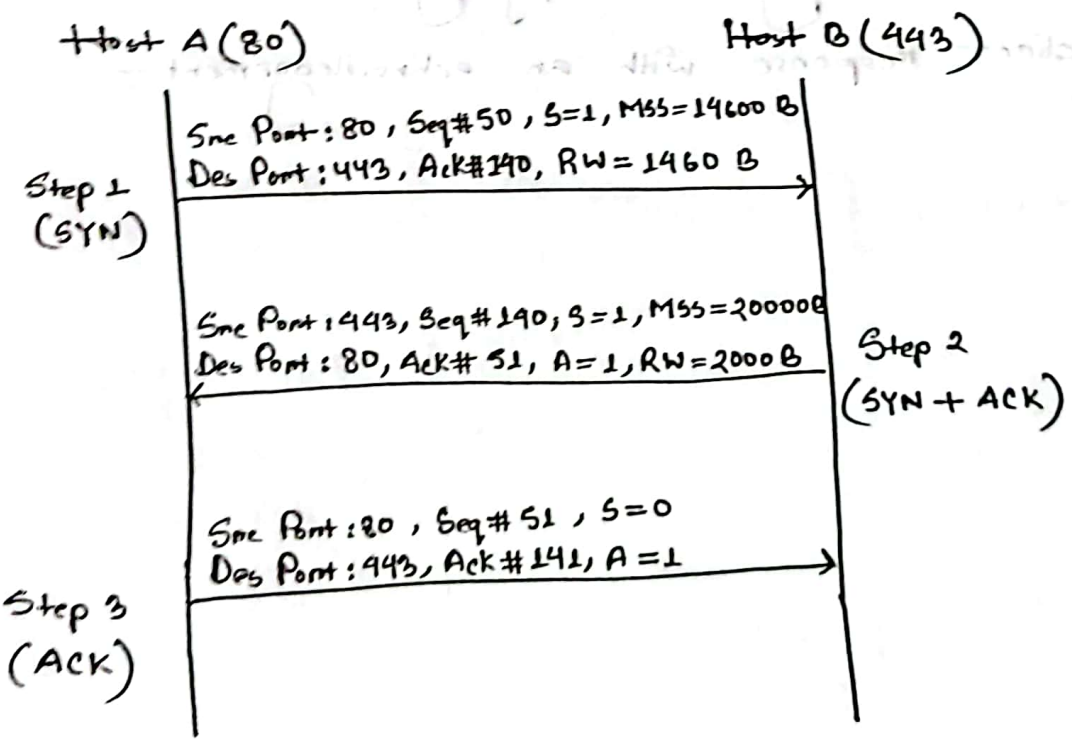
Establish connection between sender and receiver before data transmission.

(1) Step 01 (Sync):

For establishing connection, client sends a segment request to the server. This request segment consists only the TCP Header with an empty data. Then it waits for a replied segment from the server.

Request segment contains the following information:

1. Initial Sequence Number.
2. Sync bit (S) is set to 1.
3. MSS
4. Receiver Window Size.



) Step 02 (SYN+ACK):

After receiving the request segment from the sender, server respond to the sender by sending a reply segment. It informs the client about the perameters of the server site.

Reply segment contains the following information:

1. Innitial Sequence Number.
2. Sync bit (S) set to 1.
3. MSS
4. Receiver Window size
5. Acknowledgement bit (A) set to 1.

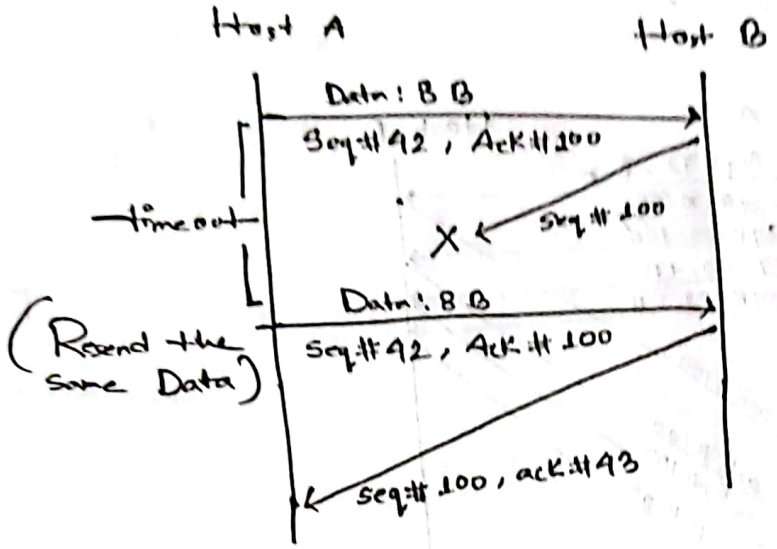
(3) Step 03 (ACK):

After receiving the reply segment from the server, client response with an acknowledgement.

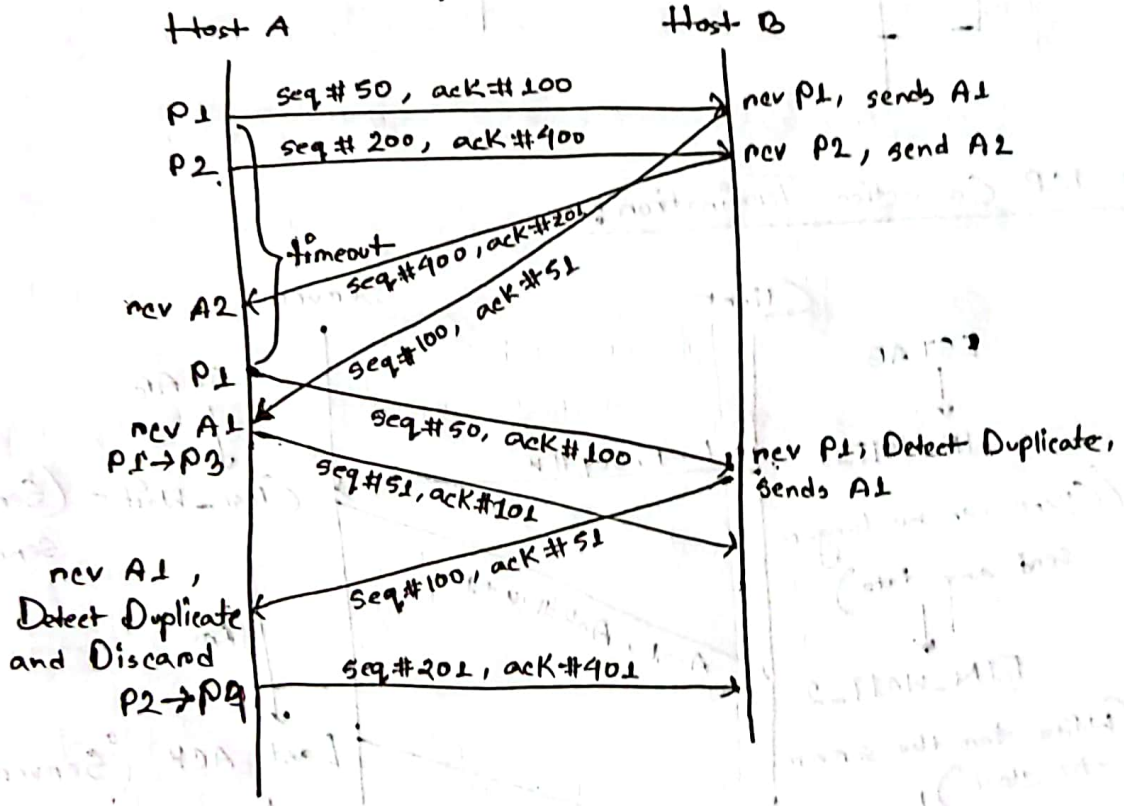
CamScanner

TCP Retransmission Scenario:

* Acknowledgement loss scenario:

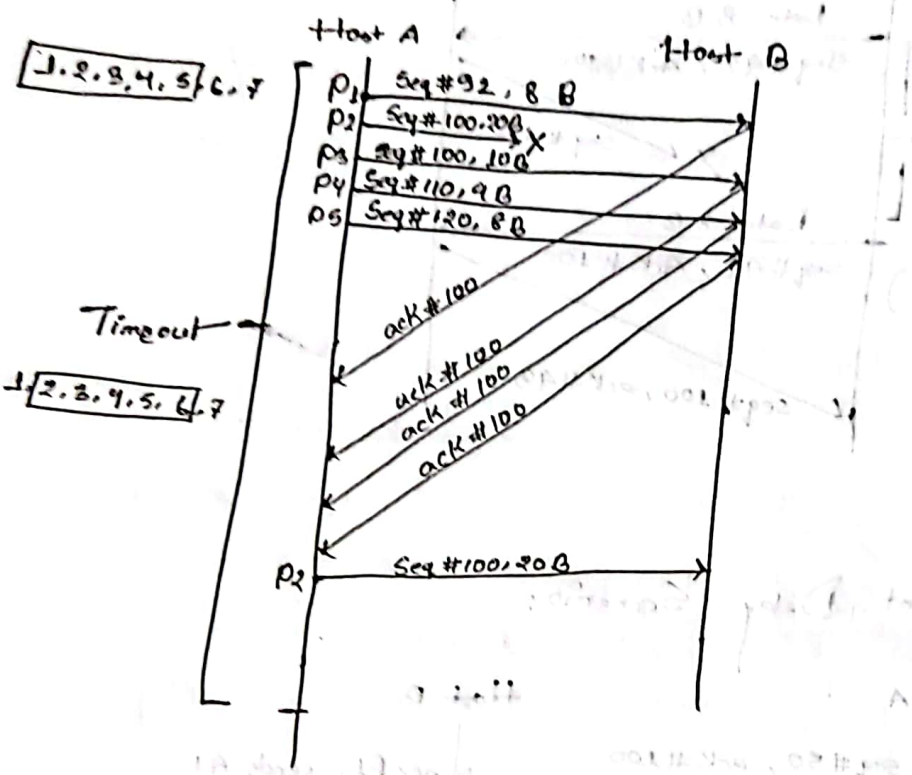


* Acknowledgement Delay Scenario:

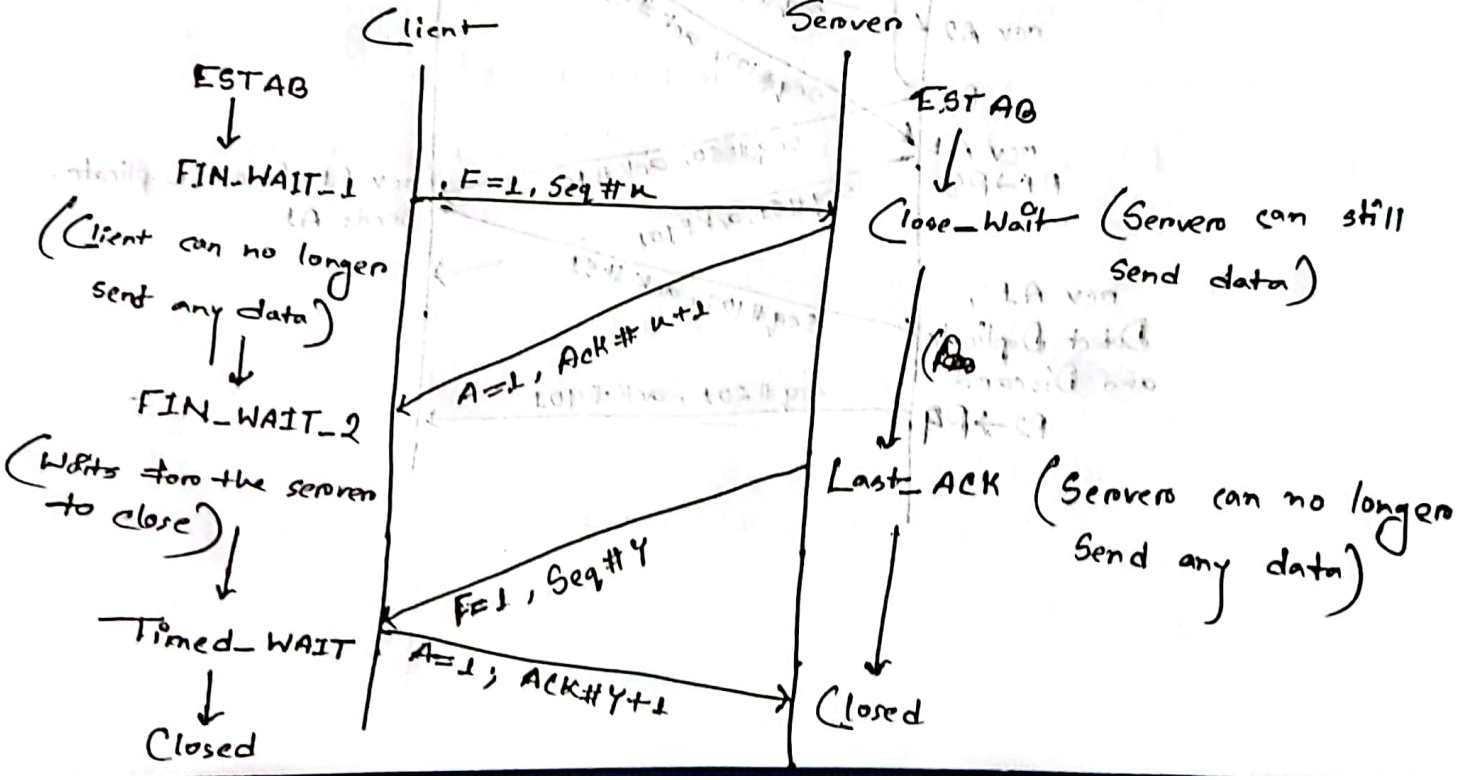


Computer Networking (05)

→ TCP fast retransmit:



→ TCP Connection Termination:



Step 01 (Finish from client):

Suppose the client app decides to terminate the connection. So, client sends a request segment to the server, with the FIN bit (F bit) set to '1'. Then the client enters into 'FIN_WAIT_1' state.

Step 02 (ACK from Server):

When the server receives the request segment from the client, it immidiately sends an acknowledgement to the client.

Step 03 (Client wait):

While in the FIN_WAIT_1 state, the client waits for an acknowledgement from the server. When client receives the acknowledgement, it enters into FIN_WAIT_2 state. While in the FIN_WAIT_2 state, the client waits for a request segment from the server.

Step 04 (Last Ack from Server):

The server sends a request segment with the FIN bit (F bit) set to '1', saying that it has closed the connection and the server can no longer send any data.

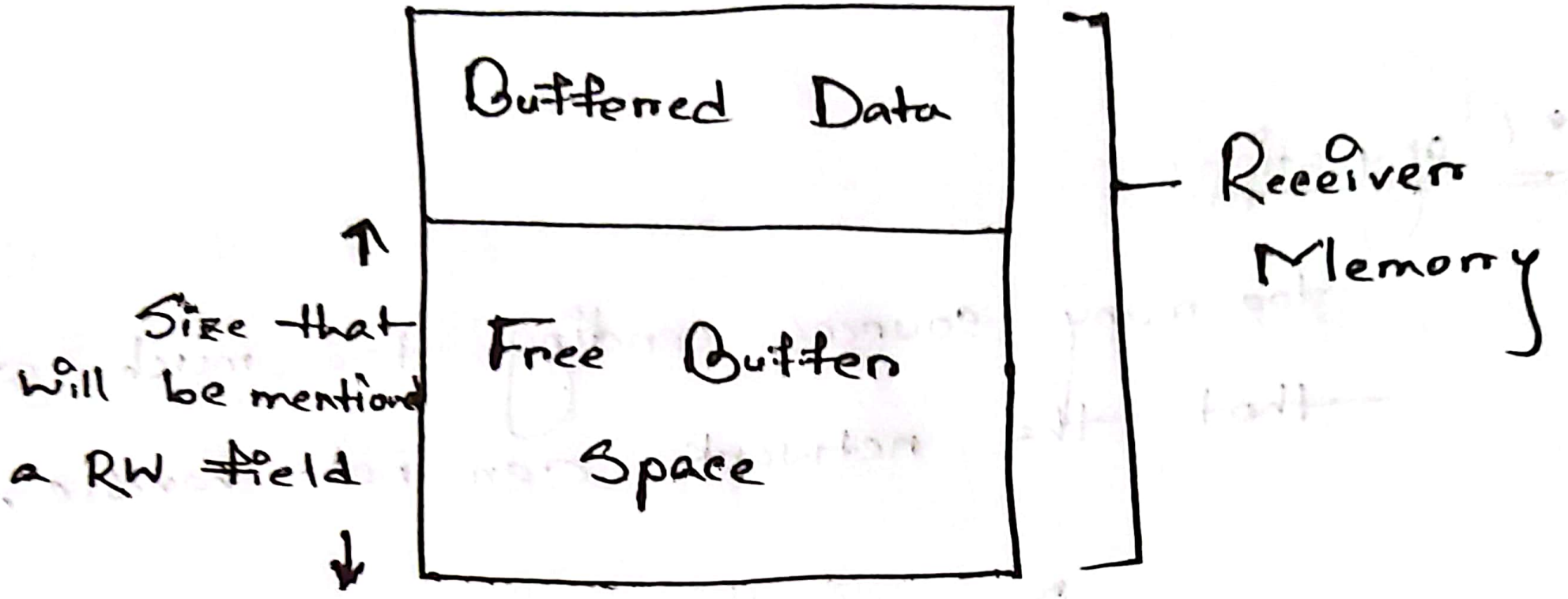
Step 05 (Ack from Client):

When the client receives the request segment from the server, it then enters into TIME_WAIT state. After waiting a certain period of time the client automatically closes the connection.

* TCP flow Control:

It is the process the managing the data trasmition rate between sender and receiver. It is used to prevent a fast sender from over whelming a slow receiver. Receiver will inform sender at what speed and ammount it can handle the data package. Both sender and receivers mutaally agrees on the speed and ammoutnt. It is a responsibility of a sender to transmit at the agreed speed. Receivers advertises the 'free buffer space' in the RW field of the TCP Header.

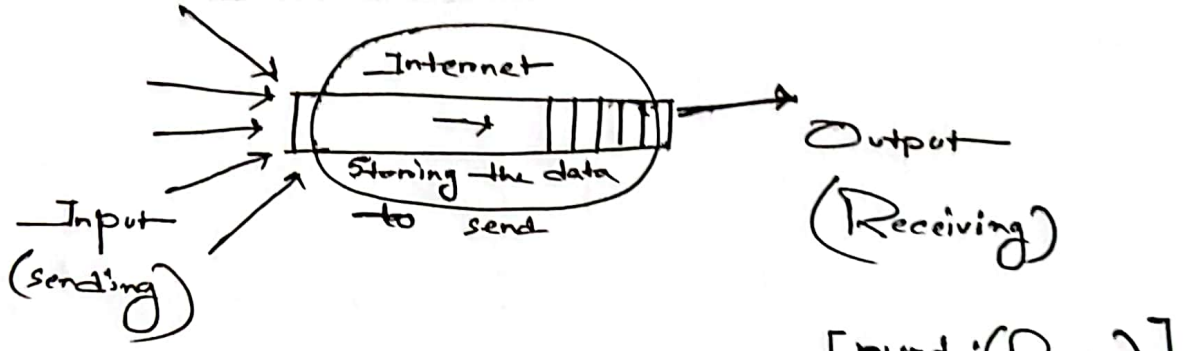
CamScanner


Buffered Data
Free Buffer Space
Receiver Memory
Size that will be mentioned a RW field

Computer Networks (06)

* Congestion:

too many sources sending too much data at too fast, that the network can not handle.



[rwnd: (Public)]

cwnd (Congestion Window): (Private)

It is a variable that limits the amount of data a sender can send into network.

* TCP Congestion Control: (AIMD)

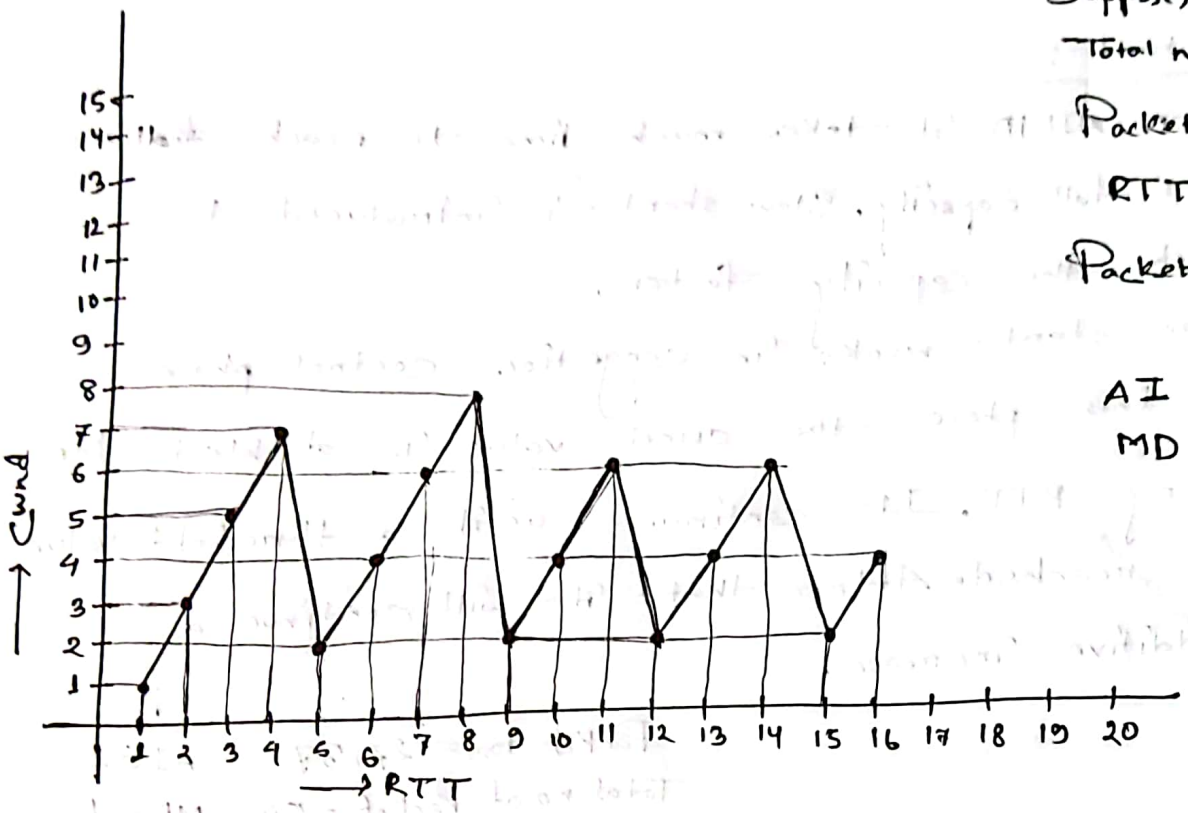
Its an algorithm to control sending rate.

AI (Additive Increase): Increase the sending rate (cwnd) for every RTT, by a "fixed amount" until a loss event occurs.

MD (Multiplicative decrease): decrease the sending rate (cwnd) by a 'multiplicative factor' for every RTT when a loss event occurs.

* AIMD Graph is sawtooth, probing for Bandwidth!"

*Graph:



Suppose,
Total no of Packet = 50
Packet size = 2 Kb
RTT = 100 ms
Packet Loss = 15, 27, 35, 46
AI = 2
MD = 1/3

RTT	Packet Sent
0-1	1
1-2	2, 3, 4
2-3	5, 6, 7, 8, 9
3-4	10, 11, ... (15), 16 ✗
4-5	15, 16
5-6	17, 18, 19, 20
6-7	21, 22, ..., 25, 26
7-8	(27) ✗, 28, ..., 33, 34
8-9	27, 28
9-10	29, 30, 31, 32
10-11	33, 34, (35) ✗, 36, 37, 38
11-12	35, 36
12-13	37, 38, 39, 40
13-14	41, 42, ... 45, (46) ✗
14-15	46, 47
15-16	48, 49, 50.

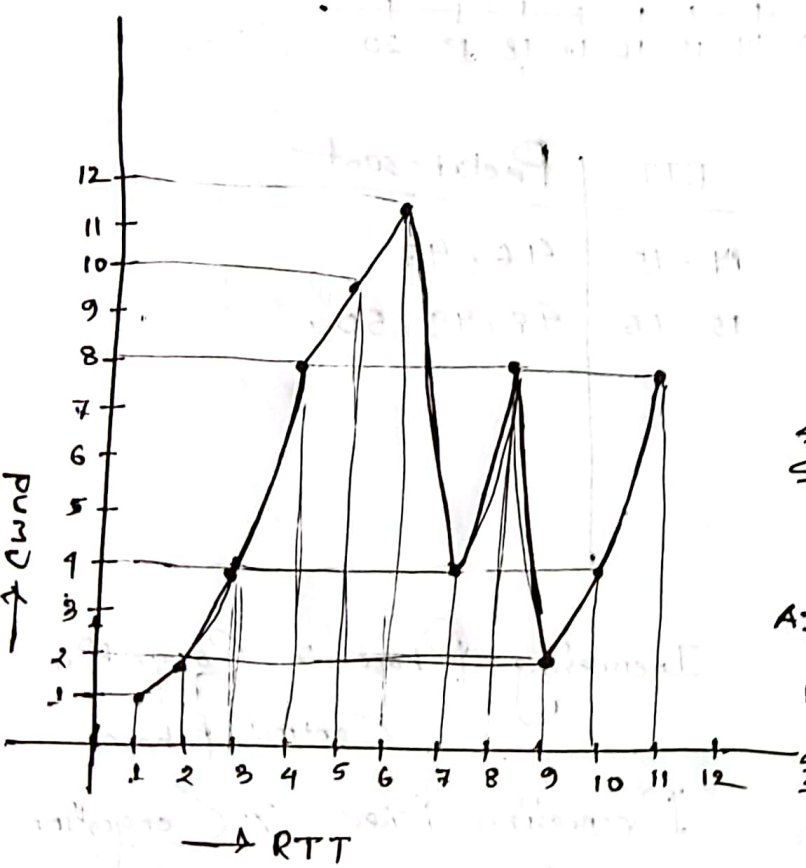
Increasing Phase → Congestion Control Phase
Decreasing Phase → Congestion Avoidance Phase

As the increasing phase requires much time, so we need to make the phase exponential. It is called Slow starter.

TCP Slow Start:

Under AIMD it takes much time to reach the links full capacity. Slow start is introduced to reach the capacity faster.

Slow start works in congestion control phase. In this phase the cwnd value is doubled for every RTT. It continues until a threshold value is reached. After that it will continue as additive increase.



Packet loss = 27, 37 AI = 2

Total no of Packet = 50 MD = 1/3

Packet size = 2KB

RTT = 100 ms

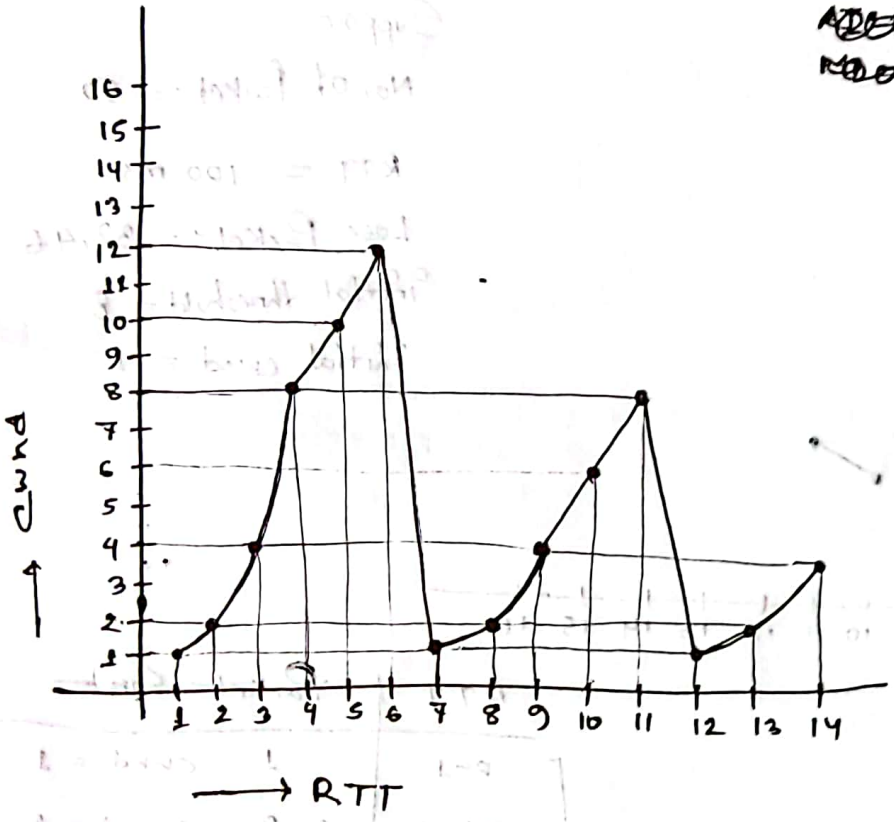
	RTT	Packet Sent
Slow Start	0-1	1
	1-2	2, 3
	2-3	4, 5, 6, 7
	3-4	8, 9, 10, 11 ... 15
AI	4-5	16, 17, 18 .... 24, 25
	5-6	26, 27, 28 37
MD	6-7	27, 28, 29, 30
Slow Start	7-8	31, 32, 33 ..... 37 38
MD	8-9	37, 38
Slow Start	9-10	39, 40, 41, 42
	10-11	43, 44, 45 .... 49, 50

CamScanner

Congestion Control Technique:

1. TCP Tahoe
2. TCP Reno

(1) TCP Tahoe: (Slow Start + AIMD + Fast Retransmit)



Initial cwnd = 1
Initial ssthres = 8
loss Packet = 32, 46

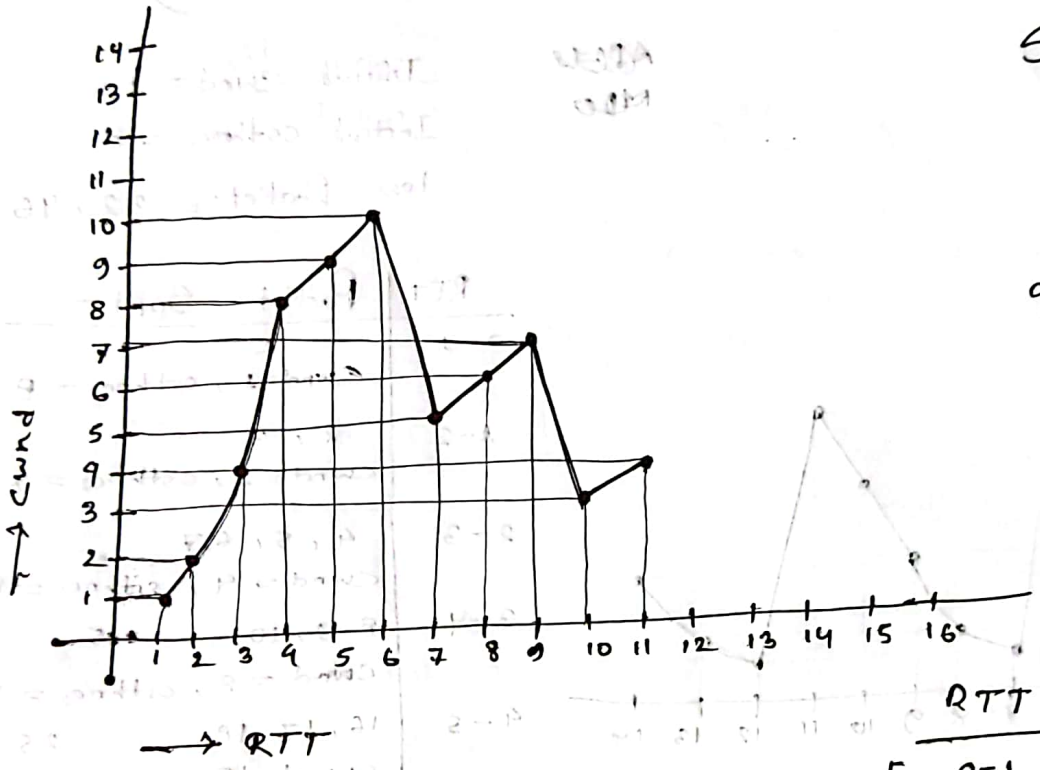
RTT	Packet Sent
0-1	1 cwnd=1, ssthres = 8
1-2	2, 3 cwnd = 2, ssthres = 8
2-3	4, 5, 6, 7 cwnd = 4, ssthres = 8
3-4	8, 9, 10, ... 15 cwnd = 8, ssthres = 8
4-5	16, 17, 18, ... 25 cwnd = 16, ssthres = 8
5-6	26, 27, (32) 36, 37 cwnd = 12, ssthres = 8
6-7	32 cwnd = 1, ssthres = $\frac{1}{2}$(cwnd) $= \frac{1}{2} \times 12 = 6$
7-8	33, 34 cwnd = 2, ssthres = 6
8-9	35, 36, 37, 38 cwnd = 4, ssthres = 6
9-10	39, 40, 41, 42, 43, 44 cwnd = 6, ssthress = 6
10-11	45, (46), 47, 48, 49, 50, 51, 52 cwnd = 8, ssthres = 6
11-12	46 cwnd = 1, ssthres = $\frac{1}{2}$(cwnd) $= \frac{1}{2}(8) = 4$
12-13	47, 48 cwnd = 2, ssthres = 4
13-14	49, 50, 51, 52 cwnd = 4, ssthres = 4

Computer networks (07)

② TCP Reno : (Efficient version of TCP Tahoe)

on loss event, (i) $ssthres = \frac{1}{2}(cwnd)$

(ii) $cwnd = ssthres$



Suppose,

No. of Packet = 50

RTT = 100 ms

Loss Packet = 32, 46

initial threshold = 8

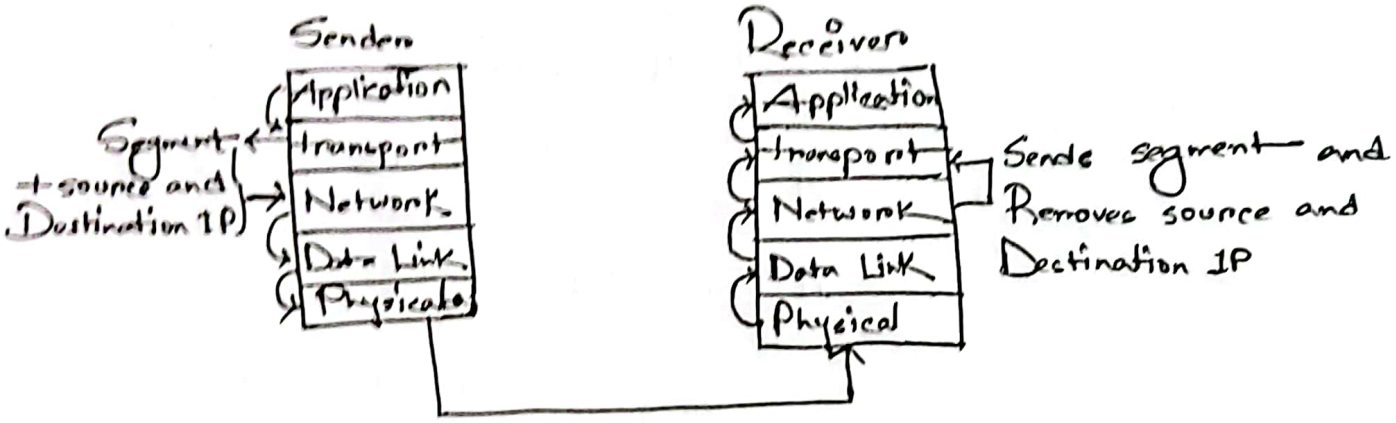
initial cwnd = 1

RTT	Packet sent
Slow Start: 0-1	1 cwnd = 1
Slow Start: 1-2	2, 3 cwnd = 2
Slow Start: 2-3	4, 5, 6, 7 cwnd = 4
Slow Start: 3-4	8, 9, 10 ... 15, cwnd = 8
AI: 4-5	16, 17, ... 24, cwnd = 9
AI: 5-6	25, 26, ... (32) 33, cwnd = 10 (X)
MD: 6-7	32, 33, ... 36 cwnd = 5, ssthres = $\frac{1}{2}$(cwnd) = 5
AI: 7-8	37, 38, 39, 40, 41, 42 cwnd = 6, ssthres = 5
AI: 8-9	43, 44, 45, (46) 47, 48, 49 cwnd = 7 (X)
MD: 9-10	46, 47, 48 ssthres = 3 = cwnd
AI: 10-11	49, 50, cwnd = 4

CamScanner

Computer Networks (08)

Network Layer:



Network Layer works in Host Device and Switch/Router

2 function of Network Layer: (1) Forwarding
(2) Routing

→ DHCP (Dynamic Host Configuration Protocol) works under UDP.

(1) DHCP Server: Port: 67
(2) DHCP Client, Port: 68

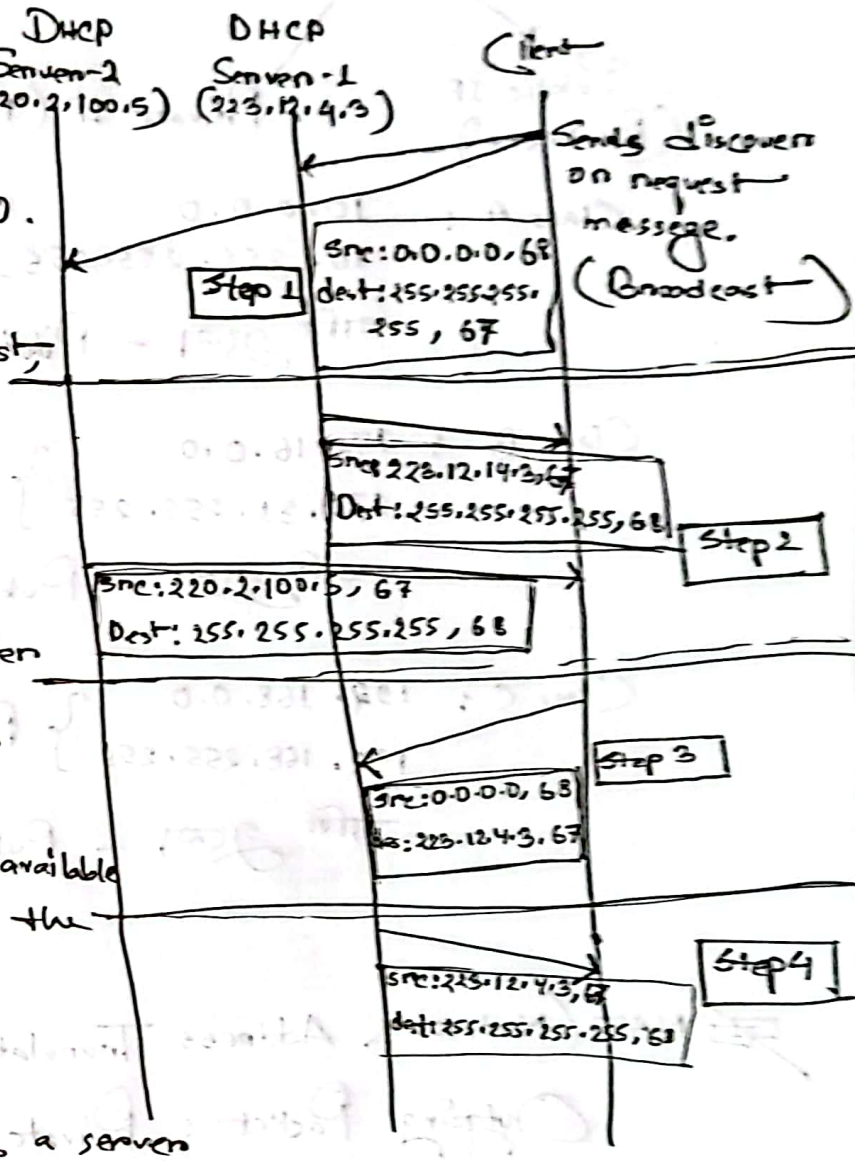
DHCP Setup করার সময়, New Host এর Network এর যা যা লাগে:
(1) IP Address
(2) Subnet Mask
(3) Default Gateway
(4) DNS
— DHCP Server Provides these to the new client.

DHCP is a client-servers Protocol.

* DHCP Protocol is a 4 step process:

(1) DHCP Discover:

As the new client has no IP, so the src IP will be: 0.0.0.0. Again, as the client is Broadcasting the request, the dest IP will be: 255.255.255.255.



(2) DHCP Offer:

Every server will offer the client a IP to add to its server. If a server has no available IP, it will not offer the client.

(3) DHCP Request:

The client will choose a server among the DHCP Offer servers and request to get an ip address.

(4) DHCP Ack:

The selected server will send an ack. to the client, confirming the request parameters.

IP V4

Public IP (Routable)

Private IP (Non-routable)

Class A : 10.0.0.0 } Private IP
10.255.255.255 }
বাকি গুলো - Public IP

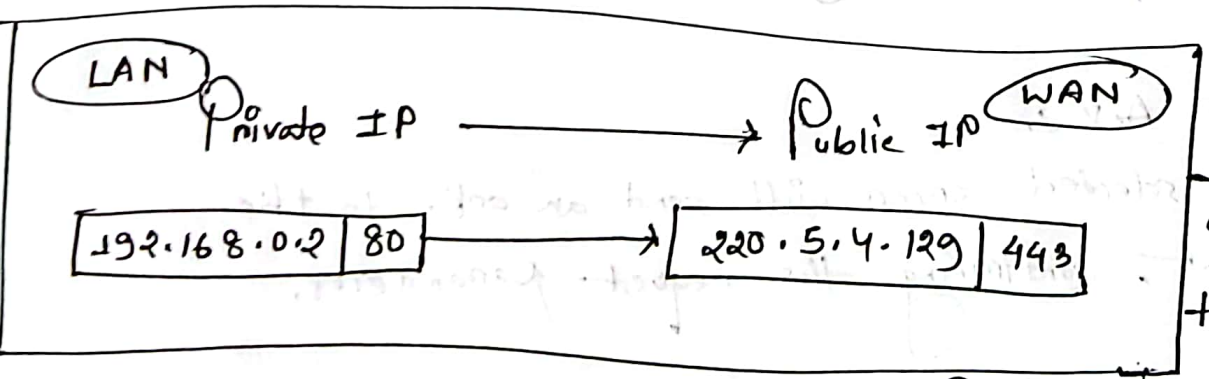
Class B : 172.16.0.0 } Private IP
172.31.255.255 }
বাকি গুলো - Public IP

Class C : 192.168.0.0 } Private IP
192.168.255.255 }
বাকি গুলো - Public IP

NAT (Network Address Translation) :

Outgoing Packet : Private IP converts into Public IP

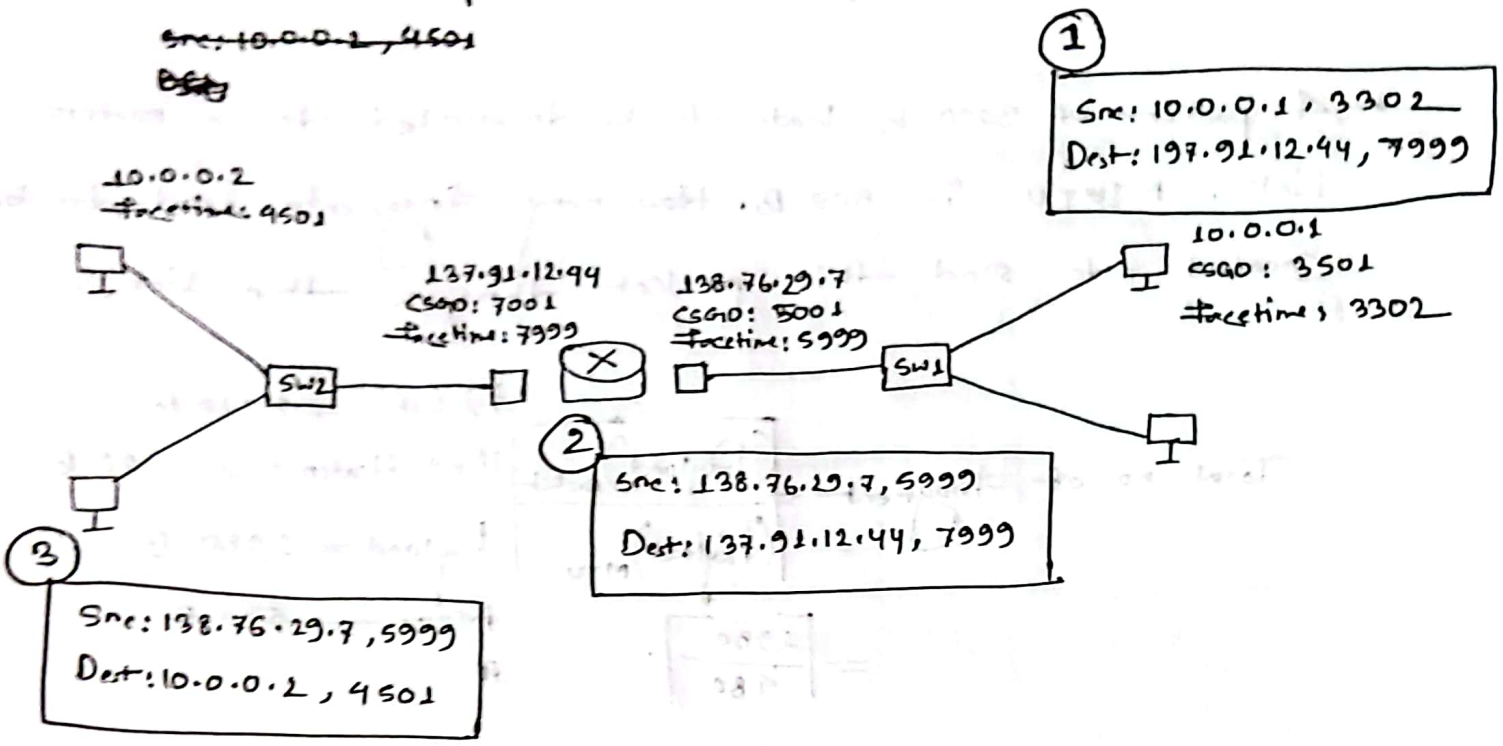
Incoming Packet : Public IP converts into Private IP



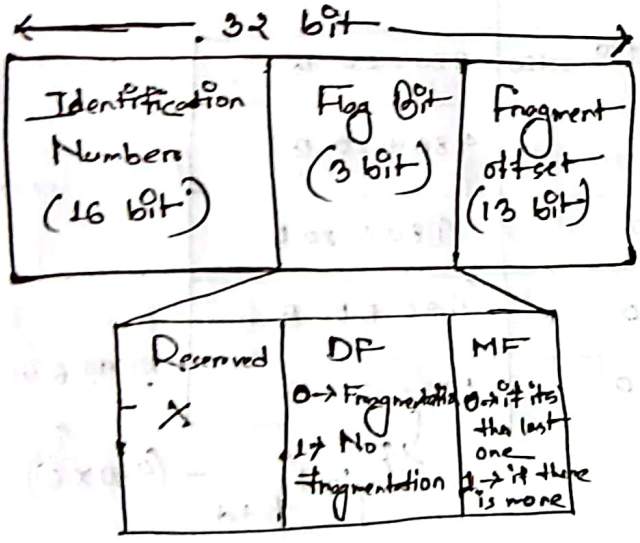
→ Stores the Private IP against the Public IP and the Private Port no against the Public Port no in a NAT Table.

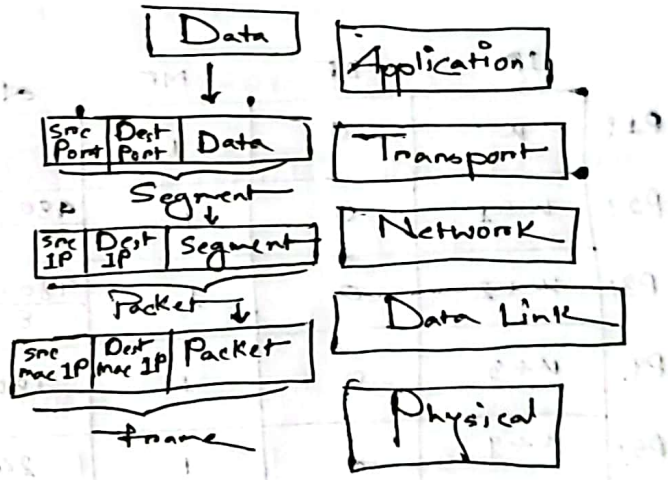
Computer Networks (09)

~~Src: 10.0.0.2, 4501~~
~~Dest~~



~~IP~~ IPv4 Fragmentation:





MTU → Maximum Transfer Unit

IPv4 header size → 20 B

DF → Don't Fragment

MF → More fragment

A packet of 3000 B leads to be forwarded to a routers Link. MTU is 500 B. How many fragments needs to be Created to send this packet through the Link?

Packet = 3000 B

IPv4 Header Size = 20 B

Payload = 2980 B

MTU = 500 B

$$\text{Total no of fragment} = \left\lceil \frac{(\text{Payload})_{\text{packet}}}{(\text{Payload})_{\text{MTU}}} \right\rceil$$

$$= \left\lceil \frac{2980}{480} \right\rceil$$

$$= \lceil 6.28 \rceil$$

$$= 7$$

MTU = 500 B

IPv4 Header = 20 B

Payload = 480 B

	ID	DF	MF	offset	Total Length
P1:	x	0	1	0	480+20 B
P2:	x+1	0	1	$\frac{480}{8} = 60$	480+20 B
P3:	x+2	0	1	$\frac{480+480}{8} = 120$	480+20 B
P4:	x+3	0	1	180	480+20 B
P5:	x+4	0	1	240	480+20 B
P6:	x+5	0	1	300	480+20 B
P7:	x+6	0	0	360	100+20 B

আকারে 6 টা fragment

$(\text{Payload})_{\text{Packet}} - (480 \times 6)$

$= 2980 - 2880$

$= 100$ B

→ Packet = 5000 B

IPv4 = 20 B

Payload = 4980 B

MTU = 1500 B

IPv4 = 20 B

Payload = 1480 B

∴ Total no of fragment $= \left\lceil \frac{(\text{Payload})_{\text{packet}}}{(\text{Payload})_{\text{MTU}}} \right\rceil$

$= \left\lceil \frac{4980}{1480} \right\rceil = \lceil 3.36 \rceil = 4$

	ID	DF	MF	offset	Total length
P1:	x	0	1	0	1480 + 20
P2:	x+1	0	1	185	1480 + 20
P3:	x+2	0	1	3870	1480 + 20
P4:	x+3	0	0	555..	540 + 20

→ Limitations of IPv4:

1. IPv4 address depletion.
2. Lack of end-to-end connection.
3. Increased Network Complexity.

CamScanner

Computer Networks : (10)

IP V6 : (128 bit) → Hexadecimal.

4 bit ← 16 bit → Octet

B2 3A : 0239 : 0027 : BB59 : 0000 : 60AA : 025A : A059

0239 → 239 ; 0027 → 27 ; 0000 → 0 ; 025A → 25A

IP V6 Addressing Format:

1. Omit leading zero's.
2. Double Colon

ex: 5A29 : 0000 : 0000 : 0BEF : 69AA : 0000 : 0000 : FFEB

5A29 : 0 : 0 : BEF : 69AA : 0 : 0 : FFEB → 1st rule

5A29 :: BEF : 69AA :: FFEB → 2nd rule.

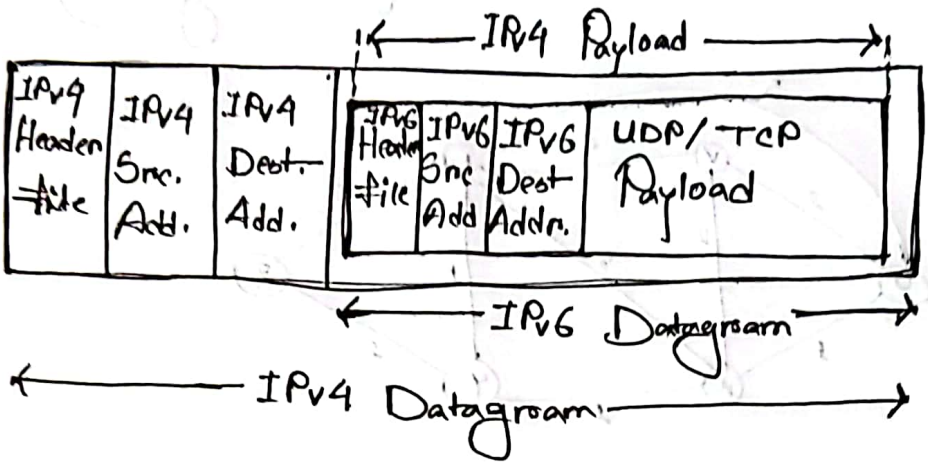
Note: Double colon (::) can only be used once within an address.

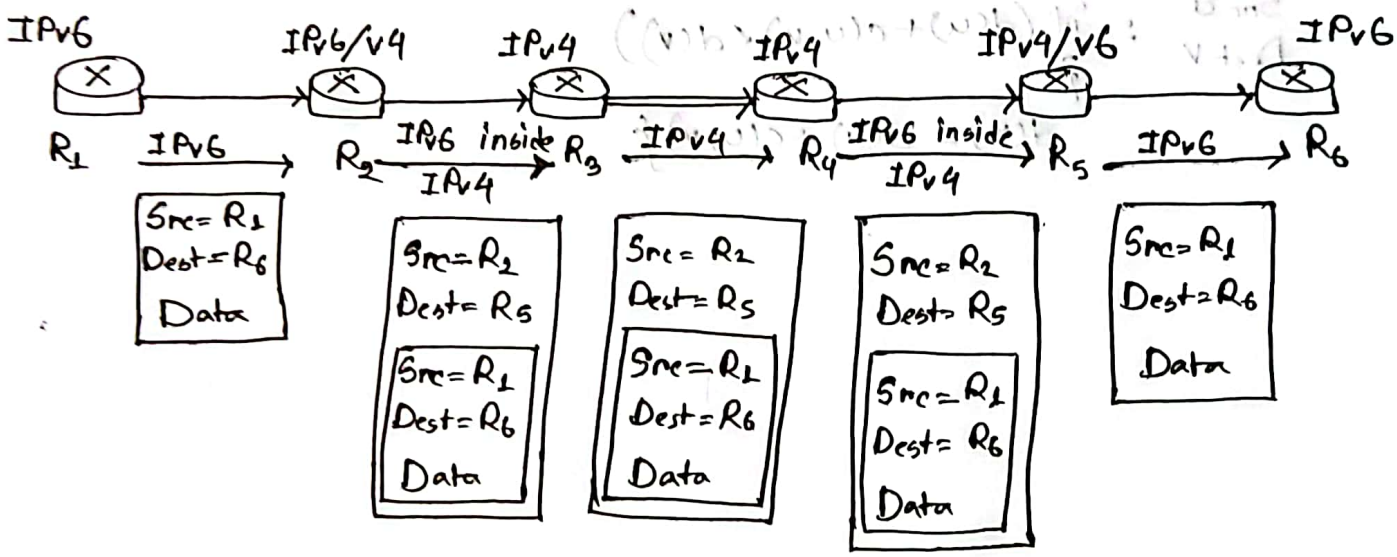
IPv4 and IPv6 Coexistence:

1. Dual Stack → The devices supports both IPv4 and IPv6.
2. Tunneling → Transporting an IPv6 packet over an IPv4 network.
3. Translation → NAT64 allows IPv6-enabled devices to Communicate with IPv4, using a translation technique similars to NAT for IPv4.

CamScanner

Tunneling :



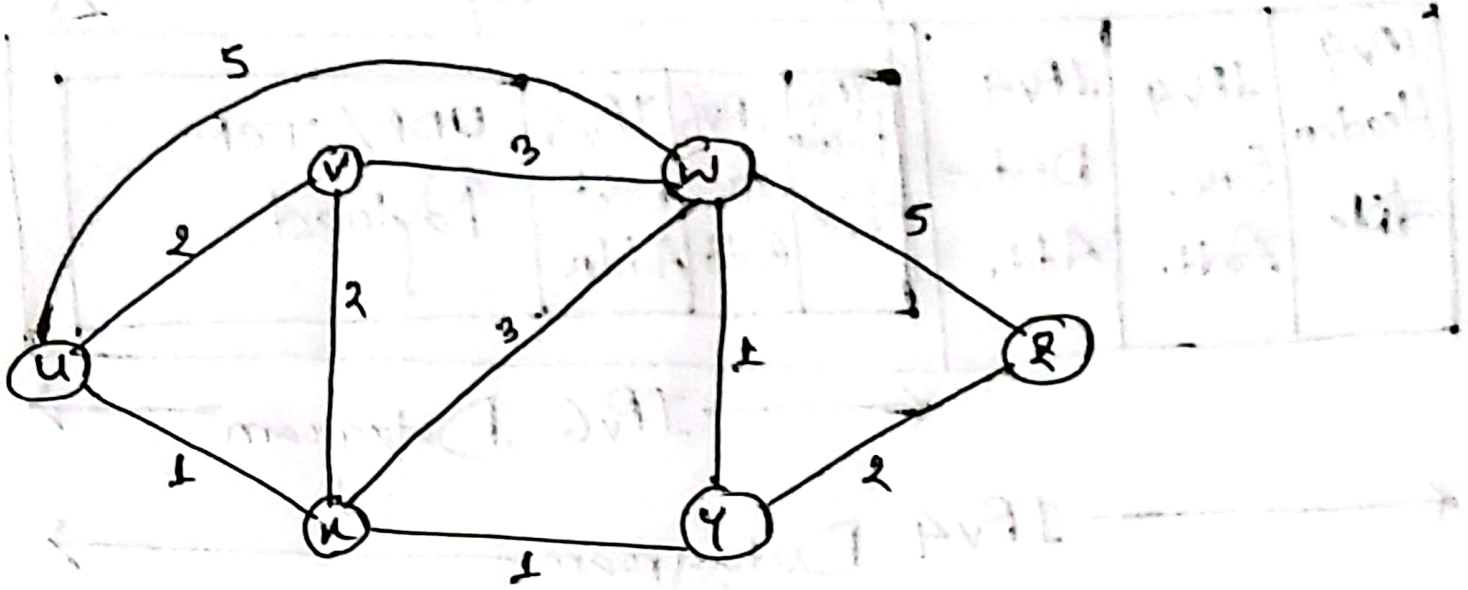


Routing Protocol :

1. Distance Vector Protocol (Bellmen Ford Algorithm)
2. Link-state Vector Protocol (Dijstra's Algorithm)

CamScanner

Link-state Vector Protocol: (Dijstra's Algorithm)



Src u
Dest v : if (d(u) + c(u,v) < d(v))
{
d(v) = d(u) + c(u,v) }