DLL → Hop to Hop
NL → Source host to dest host
TL → process to process

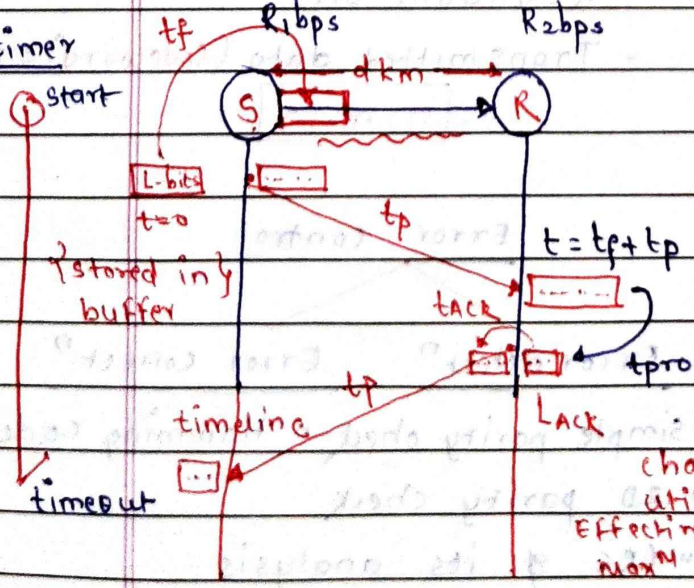
< CN >

- protocols : Set of rules/logics known by each node in n/w.
- layer Architecture :
- Addressing mode :
  - DLL 1. MAC addr - (48 bits) — used to identify host in n/w
  - NL 2. IP addr - (32 bits) — used to identify n/w
  - TL 3. Port addr - (16 bits)
  - AL 4. Specific addr - (-)

	packet Name
AL	Data/message
PL	
SL	
TL	Segment/datagram
NL	Datagram/packet
DLL	frame
PL	stream of bits

DLL (Data link layer) :

1. Framing & bit stuffing.
   - creation of frames
   - Flags/special bits → [0111 | Data | 0111]
   - ↳ if flag pattern appears inside the data ⇒ we add some extra bits.
   - eg. flag = 0111
     AD = 10111011111101
     TD = 101111001110111001.
2. Flow control.
   - ① Stop & wait
   - Sliding window
     - (GBN) Go-back-N
     - Selective Repeat (SR)



$t_f = \frac{L}{R_1}$

$t_p = \frac{d}{s}$

$t_{ACK} = \frac{L_{ACK}}{R_2}$

$\eta = \frac{t_f}{t_f + 2t_p}$ — channel utilization / efficiency

$R_E = \frac{L}{t_f + 2t_p} = \eta \cdot R_1$ — Effective bandwidth / max^M datarate

Bandwidth	m bps	k bps	g bps
Delay	d sec (10^-3)	d sec	d sec (10^-9)
	✓	Σ	5

(1024 x 1024 x 1024)

- Total time to deliver one frame successfully = (tf + tp) + (tpro + tACK + tp)
  = tf + 2tp.

② Sliding Window:

Suppose 'N' no. of frames are transmitted:

$$\eta_{N=N} = \frac{N \cdot t_f}{t_f + 2t_p}$$

'N' will be max^m iff $\eta = 100\% = 1$

$$\therefore N \leq \frac{t_f + 2t_p}{t_f} \quad \text{if } 0 \leq \eta \leq 1$$

No. of frames transmitted by Sender

OR window size Sender

$$R_E = \frac{NL}{t_f + 2t_p} = \eta \cdot R_{N=N} = \eta \cdot N \cdot R_{N=1}$$

- Sequence No. inside the frame at a given time must be unique, but the same frame can be repeated again.

$$m = \lceil \log_2 N \rceil$$

m → min no. of bits reqd to mark each frame uniquely

N → Sender window size OR Sequence no. bits.

- Mode of communication:

Simplex: data flow is one directional

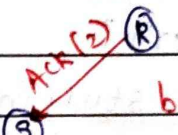
half duplex: data can flow in both direct^n but in one at a time

full duplex: data can flow in both direct^n at a same time.

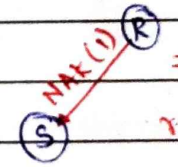
- 'm' should be same on both sides.

SW

GBN	SR
$SW < 2^m$	$SW = RW \leq 2^{m-1}$
$RW = 1$	
$N < 2^m$	$N \leq 2^{m-1}$
$m > \log_2 N$	$m \geq \log_2 N + 1$
$m = \lceil \log_2 N \rceil$	$m = \lceil \log_2 N \rceil + 1$



⇒ Frame No. upto '1' has been received successfully & next expected frame is f2.



⇒ f1 has been lost, retransmit it again.

3. Error Control:

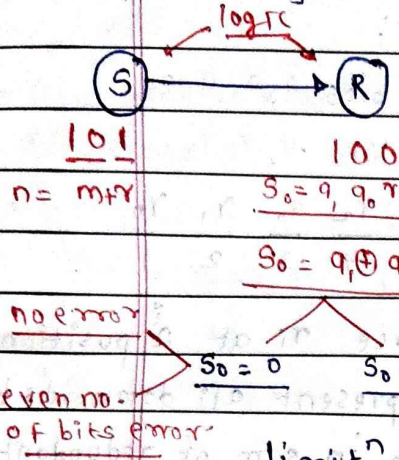
- Datawords ⇒ m
- redundant bits ⇒ r
- Transmitted data (codeword) = n

$$\therefore n = m + r$$

Error control

Error detect^n	Error correct^n
• simple parity check	• Hamming code
• 2D parity check	
• CRC & its analysis	
• checksum	

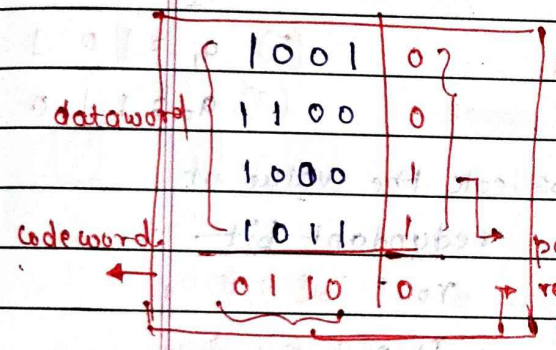
1. Simple parity / 1-D parity :



m = 2 bits	logic
n = m + r	choose 'r' in such a
= 00 0	way so that codeword
01 1	will have even no. of 1's
10 1	
11 0	sender side ⇒ $r_0 = q_1 \oplus q_0$
q_1 q_0 r_0	generate redundant bits.

limitⁿ : can detect only 1 bit errors.

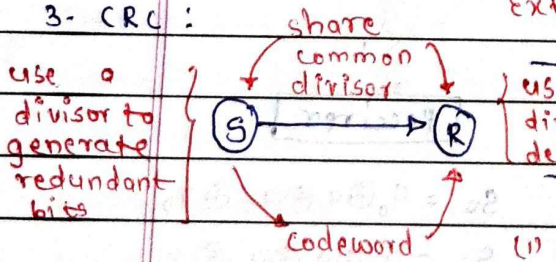
2. 2D parity check :



	parity bits affected	
	max	min
case 1 : single bit error :	2	2
case 2 : double bit error :	4	2
case 3 : triple bit error :	6	2
case 4 : four bit error :	sometimes system fails to detect error.	

limitⁿ : can detect upto 3 bit errors.

3- CRC :



eg. divisor = 1101

dataword = 10001101 1

M r

1101 | 10001101 000

⊕ 1101

01011

⊕ 1101

01101

⊕ 1101

001100

⊕ 1101

010

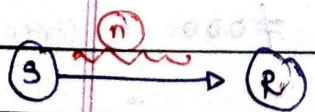
codeword = m+r

n = 10001101 010

m r

size of redundant bits = | size of divisor bits | - 1

= 4 - 1 = (3) = 000



Again XOR

$S_2 S_1 S_0 = 000$ — No error

$S_2 S_1 S_0 \neq 000$ — error

- A good polynomial generator need to have:
  - 1 bit error: atleast 2 terms & coefficient of x^0 should be 1.
  - 2 bit error: It should not divide $x^t + 1$
  - odd. no. error: It should have a factor $(x+1)$.

4. Error correctn : Hamming Code.

$$m+r \; C_{d+1} \leq 2^r$$

m = datawordsize
r = redundant bits
d = no. of bit error need to corrected

eg. ① m = 4, d = 1

$n = m + r$
$= 4 + 3$

m = 4, d = 2

$n = m + r$
$= 4 + 6$

overhead is large

- Hamming distance : $d(x,y)$:
  - 2 string → same size
  - no. of diff. btn corresponding bits.

- Min. Hamming dist : In a set of codewords, it is smallest hamming dist. btn all pairs of codewords.

- If Min. hamming dist is d, then we can detect upto $(d-1)$ bit errors.
- To detect 'd' bit error, min. hamming dist req^d = $d+1$.
- To correct 'd' bit errors, Min. hamming dist req^d = $2d+1$.

Steps :

① m = 4 bits $q_3 q_2 q_1 q_0$

r = 3 bits $r_2 r_1 r_0$

q_3	q_2	q_1	r_2	q_0	r_1	r_0
7	6	5	4	3	2	1

Step1: Place r_i at 2^i position.

Step2: Represent all dataword's bit position in sum of redundant bits position.

	r_0	r_1	r_2
	1	2	4
(7) q_3 =	1	1	1
(6) q_2 =	0	1	1
(5) q_1 =	1	0	1
(3) q_0 =	1	1	0

Step3 : calc the value of redundant bit.

$r_0 = q_0 \oplus q_1 \oplus q_3$

$r_1 = q_0 \oplus q_2 \oplus q_3$

$r_2 = q_1 \oplus q_2 \oplus q_3$

Receiver

$S_0 = q_0 \oplus q_1 \oplus q_3 \oplus r_0$

$S_1 = q_0 \oplus q_2 \oplus q_3 \oplus r_1$

$S_2 = q_1 \oplus q_2 \oplus q_3 \oplus r_2$

⇓

If $S_2 S_1 S_0 = 000$ ... No error

$\neq 000$ ... error

4. Media Access Control. logic: Any node can access the common medium at any time.

P(C) ≠ 0 — contention service

P(C) = 0 — contention free service

1. ALOHA, Pure ALOHA. time range where there is changes of collision

(wired) (wireless)

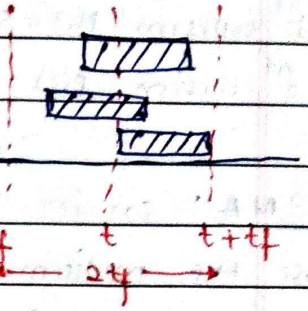
$S = G \times e^{-2G}$

S → throughput of pure aloha. G → no. of frames generated in 1 frame transmission time.

for $S_{max} \rightarrow \frac{dS}{dG} = 0 \Rightarrow \boxed{G = \frac{1}{2}}$

$\therefore S_{max} = \frac{1}{2e} = 0.184 = 18.4\%$

• vulnerable time = $2t_f$



eg. If 1000 frames are generated by n/w in 1 sec. & t_f = 1ms.

→ Is → 1000 frames

1ms → 1 frame

Throughput = $G \times e^{-2G}$

$= 1 \times e^{-2 \times 1} = 13.5\%$

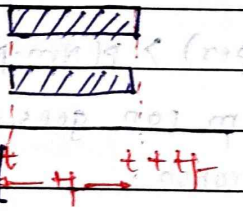
Avg. no. of succ. trans. = $1000 \times 0.135 = 135$

2. Slotted Aloha:

• vulnerable time = t_f

• $S = G \times e^{-G}$

for $S_{max} \rightarrow \frac{dS}{dG} = 0 \Rightarrow G = 1$

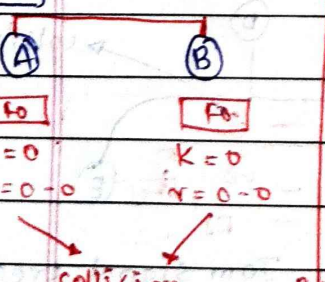


logic: Any node can only transmit at beginning of time slot

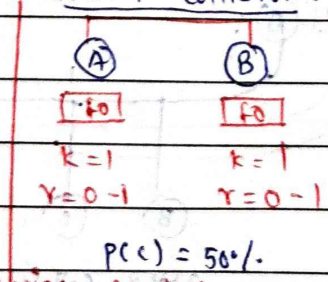
$\therefore S_{max} = \frac{1}{e} = 0.368 = 36.8\%$

3. Exponential Backoff Algo: here we reduce the collisions in pure aloha

Initially:



after 1st collision:



r_A	r_B	
0	0	C
0	1	nc A WT
1	0	nc B WT
1	1	C

k → collision No.

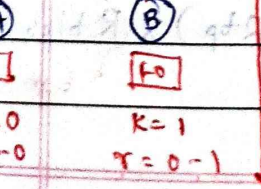
r → random No. generated.

WT → waiting time.

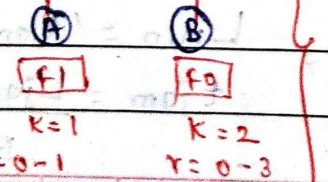
$r \in 0 - (2^k - 1)$

$WT = r \cdot t_f$.

after A-wins:



after 2nd collision:



r_A	r_B	
0	0	C
0	1	nc A
0	2	nc A
0	3	nc A
1	0	nc B
1	1	C
1	2	nc A
1	3	nc B

$P(C) = \frac{2}{8} = 25\%$

$P(A) = \frac{5}{8} =$

$P(B) = \frac{1}{8} =$

• Max^m no. of attempts for a station is 15 in Exponential Backoff Algorithm.

Page No. Date

conclusion:

initially : P(C) = 100 %.
after 1st collisim : P(C) = 50 %.
after 2nd collision : P(C) = 25 %.

prob. of collision reducing.

Disadvantage:

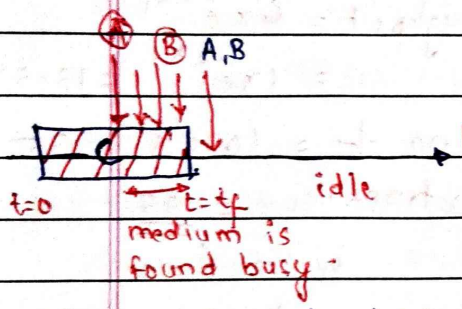
• Suffers from capture effect.
• If any station wins in 1st collision then it have more prob. of winning.

P(A) = 25 %.
P(A) = 62.5 %.
P(A) = 21.25 %.

4. CSMA : (Carrier Sense Multiple Access):

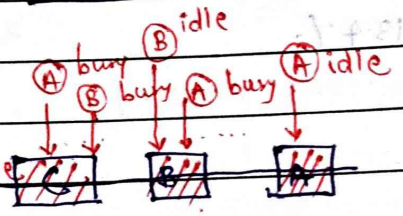
• sense the medium is busy or not before transmitting.
• sensing Technique:
• vulnerable time = tp.

1. 1-persistant :



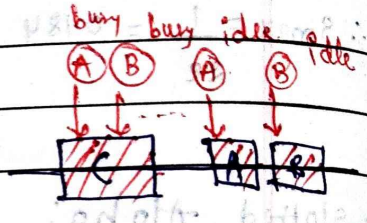
• they sends their data concurrently which leads to collision. bec2, they can't watch each other.

2. Non-persistant



• P (1-per) > P(Nm-per)
• system can goes under starvation.
• randomness is reducing the prob. of collisim.

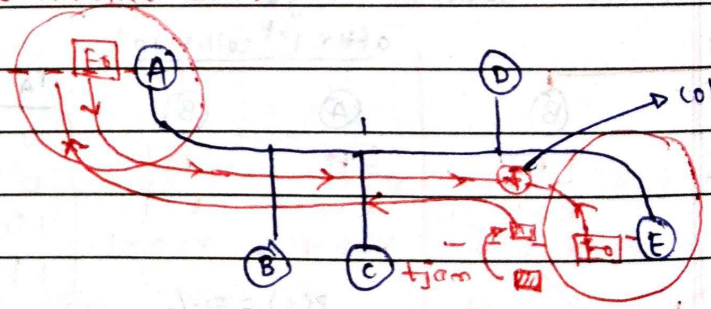
3. p-persistant



• $W_T = r \cdot t_p$
r: random value
t_p: transmission time

5. CSMA/CD : collision detection. (Ethernet LAN):

• No concept to Ack. frame.
• Either collision must detected before complete transmission. OR it must be delivered completely.
• Sender doesn't delete the copy of frame until it transmitted completely.
• Baud rate = 2 x bit rate.



Total time = $t_p + (t_{jam} + t_p)$

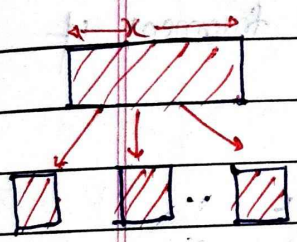
$L_{min} = (t_{jam} + 2t_p) \times R$ bits.
(min^m frame length/size; R: bandwidth)

$t_{jam} = \frac{L_{jam}}{R}$

6. polling protocol (TDM):

- $p(c) = 0\%$
- Total time = $N(t_f + t_p)$ for N-nodes
- $\eta_{N=N} = \frac{N \cdot t_f}{N(t_f + t_p)} = \frac{t_f}{t_f + t_p}$
- $\eta_{N=1} = \frac{t_f}{N(t_f + t_p)}$
- $R_E = \frac{L}{t_f + t_p} = \eta \cdot R$

7. Store forward Technique:



Size of msg = x bits

broken into p no. of packet.

Header size = h bits

∴ size of 1 packet = $(\frac{x}{p} + h)$ bits (data + header)

R = bandwidth

k = No. of hops

∴ optimal value of p = $\sqrt{\frac{x(k-1)}{h}}$

Net-ID : # of nlw can be created by ISP.

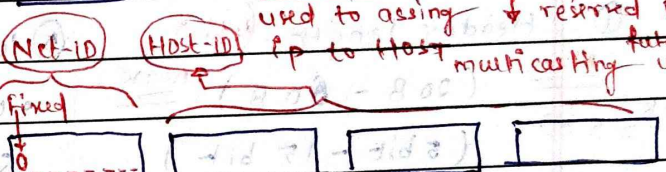
Host-ID : # host possible in given nlw.

Network layer:

- Host to host communication.
- best effort delivery of packet
- No error or flow control
- If error occur then TL will take care of it.

1. IP-addressing (32 bits) — 4 · □ · □ · □ (8 bits or 1 Byte)
   - classful
   - classless.

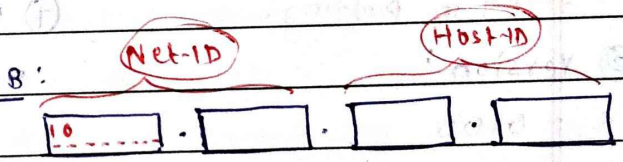
① Classful: A B C D E (used to assign ip to Host; E restricted for multicasting, future use.)



Range : $1 - (2^7 - 1)$: 1 - 127

Networks = $2^7 = 128$

Host within a nlw = $2^{24} - 2$ =

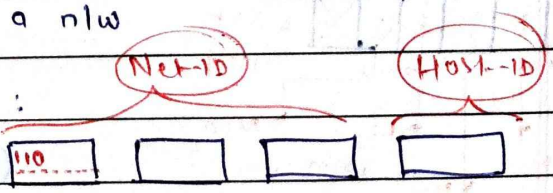


Class B:

Range : 128 - 191

Network = $2^{16-2} = 2^{14}$

Host within a nlw = $2^{16} - 2$



Class C:

Range : 192 - 223

Network = $2^{24-3} = 2^{21}$

Host within a nlw = $2^8 - 2$

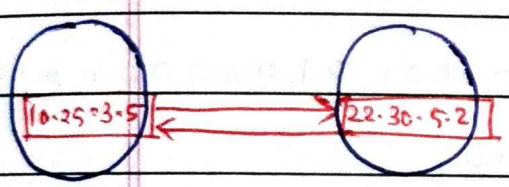
Class D: Range : 224 - 239 (No NID / No HID)

Class E: Range : 240 - 255

Subnetting:
- All 0's in HID = n/w ID of entire n/w
- All 1's in HID = DBA.

- Types of communication:

1. unicast (1:1)
- 1 comp to another comp.

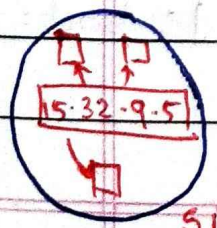


- SIP can be used as DIP
- DIP can be used as SIP

Not change	may be changed	Do change
VER	~~HL~~	TTL
IHL	Total length	checksum
Ser	MF	
Id no	frag. offset	
DF		
protocols		
SIP, DIP		

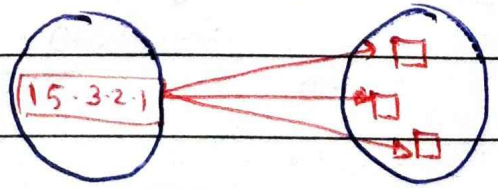
2. Broadcast (1-all)

1. limited: (LBA)
- 1-comp to all other comp in same n/w.
- SIP & DIP can't be interchange.



SIP	DIP
Data | 15.32.9.5	225.225.225.225

2. Direct: (DBA)
- 1-comp to other comp in diff n/w.



SIP	DIP
Data | 15.3.2.1	112.255.255.255

3. multicast (1-many)

Trick:

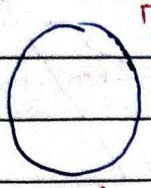
NID	HID	
valid	0's	→ n/w of entire n/w
valid	1's	→ DBA
1's	is	→ LBA

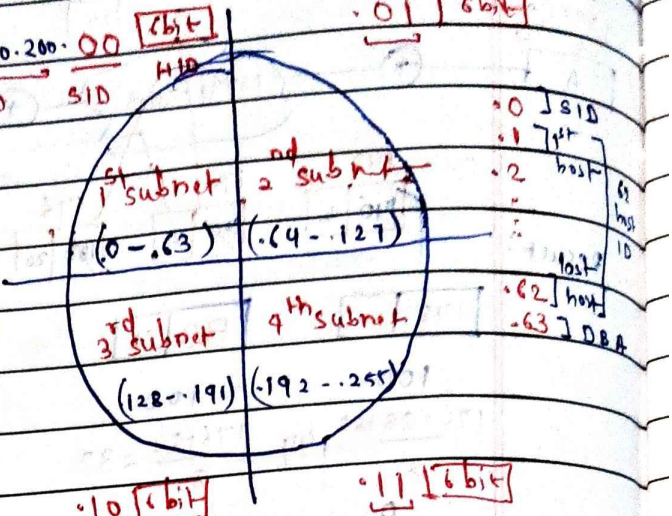
Subnetting in classful: The process of Borrowing bits from HID to generate the Subnet ID.

eg. NID HID
200.200.200.0

Belongs to class-C

NID HID
24 8





Divided into 4 subnet

NID SID HID
24 (2) (6)

No. of Subnets = 2^2 = (4)

No. of Host per subnet = $2^6 - 2$

Subnet Mask:

Rule1: No. of 1's in subnet mask = NID + SID

Rule2: No. of 0's in subnet mask = HID

IP Addr.	IP Addr.
AND	AND
N/w mask	Subnet Mask
NID	SID

eg.① NID = 200.200.200.0

Subnet mask = 255.255.255.192.

→ SM = 255.255.255.11000000

No. of 1's = 26

NID + SID = 26

(26) + SID = 26

SID = 2 bit borrowed

No. of 0's = 6

HID = 6 bit

No. of subnet = 2^2 = (4)

No. of Host/Subset = $2^6 - 2$ = (62)

eg.② IP addr = 200.200.200.126

S.M = 255.255.255.192.

IP: ——.——.——.01111110

SM: ——.——.——.11000000

SID: 200.200.200.01000000 = .64

IP addr = HID + SID

126 = HID + 64

∴ HID = 62

eg.③ SM = 255.255.255.224.

→ SM = 255.255.255.11100000

• Subnet's Id:

NID SID HID
200.200.200.__.[ ]
2bit 6bit

.00 000000 = .0
.01 000000 = .64
.10 000000 = .128
.11 000000 = .192

If class A:

NID + SID = 27

8

∴ SID = 19 bit

No. of 0's = HID = 5 bit

No. of IP addr/subnet = 2^5

No. of host/subnet = $2^5 - 2$

eg① Find SID & subnet No?

→ SM: 11111111.11111111.11111111.11000000

(Class C) NID SID HID

IP addr: 200.200.200.00111100

SID: 200.200.200.00100000

NID HID

3 bit

00 0 → 1st subnet

00 1 → 2nd subnet

01 0 → 3rd subnet

⋮

111 → 8th subnet

decimal: 1 → 2nd subnet

.00100000 → SID

.00100001 → 1st host

.00111110 → last host

.00111111 → DBA

eg⑤ Sums on unequal subnets?

A company has class C n/w addr. of 204.204.204.0. It wishes to have 3 subnets (X, Y, Z)

<100 hosts 50 50 hosts>

NID	SID	HID
24	2	6

2^8 IP

0 → 1 bit (SID, HID)

2^7 IP

X = 100 hosts

10 → 6 bit (SID, HID)

2^6 IP

Y = 50 hosts

2^6 IP

Z = 50 hosts

11 → 6 bit (SID, HID)

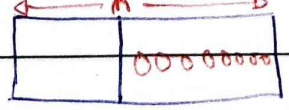
Classless <CIDR> {classless Interdomain Routing}

→ whenever any customer wants a block of IP addr IANA or ISP will create a block assigned to customer.

Rules: ① All IP addr. in the block must be contiguous.

② Block size must be of 2^n. (power of 2)

③ 1st IP addr. of block must be divisible by size of block. and used as block Id.



Note: 127.x.x.x

- Self connectivity
- loop back testing.
- Inter process communication

↓

It will always be used as a D.IP addr.

eg 100.100.100.64

100.100.100.65

⋮

100.100.100.127

64 IP = 2^6 IP

① contiguous ✓

② B.S: $2^6 \div 2$ ✓

③ 1st IP addr: 100.100.100.64 $/2^6$

= 100.100.100.01 000000 ✓

HID.

↓

Valid Block

In this type of Q°, first look at SM convert all HID to 0, if DBA matches ✓ <don't look at NID & SID after that>

eg① SM = 255.255.255.240 [255.255.255.11110000] HID ← No. of 0s.

DBA = ? ▷ All host ID = 1.

→ ✓ (a) 200.56.78.31 [0001 1111]

✗ (b) 200.56.78.15 [0000 1111]

✓ (c) 200.56.78.10 [0000 1010]

✓ (d) 200.56.78.47 [0010 1111]

HID can be max^m 13 bit (HID ≤ 13)

eg⑦ DBA = 144.16.95.225 & class B

SM = ?

→ 144.16.010 11111.11111111

NID SID HID → DBA diff 3 bits

(a) 255.255.(224.0) = 11100000.00000000 13 ≤ 13 ✓

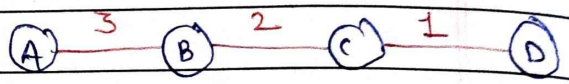
(b) 255.255.(240.0) = 11110000.00000000 12 ≤ 13 ✓

Routing Algorithm.

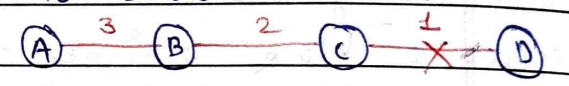
- Static <Non-adaptive Routing>
- Dynamic <Adaptive Routing>
  - DVR
  - LSR
  - PVR

DVR, LSR, PVR } unicast <Intradomain Routing> <Interdomain Routing>

• Count to infinity problem.



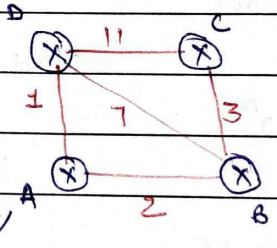
link C-D break. & '>' is used to show ∞ also.



A	B	C	D
6	3	1	Initially
6	3	∞	break
min(3+3) 6	✗ ∞	min(∞+2, 6+3) 5	min(3+2)
min(∞+3) ∞	min(7,9) 7	min(∞+2) ∞	3 steps to determine link breakd
min(7+3) ∞	min(∞,∞) ∞	min(7+2) ✗ ∞	

① DVR:

Step 1: prepare Routing table at every router based on local knowledge.



From A:

eg.

Dest	Dist	NH
A	0	A
B	2	B
C	∞	-
D	1	D

Step 2:

A recieved DV from B, D.

from B	from D
A | 2	1
B | 0	7
C | 3	11
D | 7	0
AB = 2	AD = 1
0+2 ✓	1+1 ✗
3+2 ✓	11+1 ✗
7+2 ✗	0+1 ✓

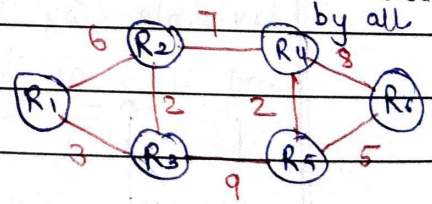
⇒

New Routing table

Dest	Dist	NH
A	0	A
B	2	B
C	5	B
D	1	D

Note: old A ko nhi dekhna h sirf B & D ko dekho.

② LSR:



by all

At ~~After~~ R1:

	R1	R2	R3	R4	R5	R6
R1	6	(3)	∞	∞	∞	
R1 R3	(5)		∞	12	∞	
R1 R2 R3			(12)	12	∞	
R1 R2 R3 R4				(12)	21	
R1 R2 R3 R4 R5					(16)	

IP → Packet stream protocol : i.e, every packet is associated with one seq. no

TCP → Byte stream protocol : i.e, every byte is associated with one seq. no

IPv4 Header:

20B (Fixed)	4 VER	4 IHL	8 Services	16 Total length	32 bit = 4B
	16 Identification No.		3 Flags	13 Frag. offset	32 bit = 4B
	8 TTL	8 Protocols		16 Header checksum	32 bit = 4B
	16 Source IP addr.				32 bit = 4B
	16 Dest IP addr.				32 bit = 4B
40B (Variable)	Option (00-40)				20B
	padding ..				

Flag: // | DF | MF

① Header length : (4 bit)

Min (20B - 60B) Max

20B / 4 = 5 bit (5 bit - 15 bit)

60B / 4 = 15 bit ((0101) - (1111))

if, 30B | 30/4 = 7.5 ✗

then, (30 + 2)/4 = 8 ✓ (1000)

dummy byte = stored in padding.

② Version :

IPv4 = 0100

IPv6 = 0110

③ Service :

0	1	2	3	4	5	6	7
P	P	P	D	T	R	C	✗ → Not used

priority bits

Type of service

D → Min delay

T → high Max throughput

R → high reliability

C → Min cost

④ TL (16 bit) : $2^{16} - 1$

TL = data + header

65,515 B	20B

← 65,535 B →

Max^m data size at NL.

⑤ Identification No :

- used to identify all fragment of same datagram.

⑥ Flags

1. DF: 0 can be frag. 1 can't be frag.
2. MF: 0 last frag. 1 not last frag.

Trick : 0 : Don't frag = false

frag ✓

⑦ Frag. offset :

- No. of databyte ahead of this fragment.
- must be ÷ by 8.

⑧ TTL :

- used to avoid infinite looping
- controls max^m no. of hops visited by datagram.

⑨ Protocol :

ICMP > IGMP > UDP > TCP > OSPF

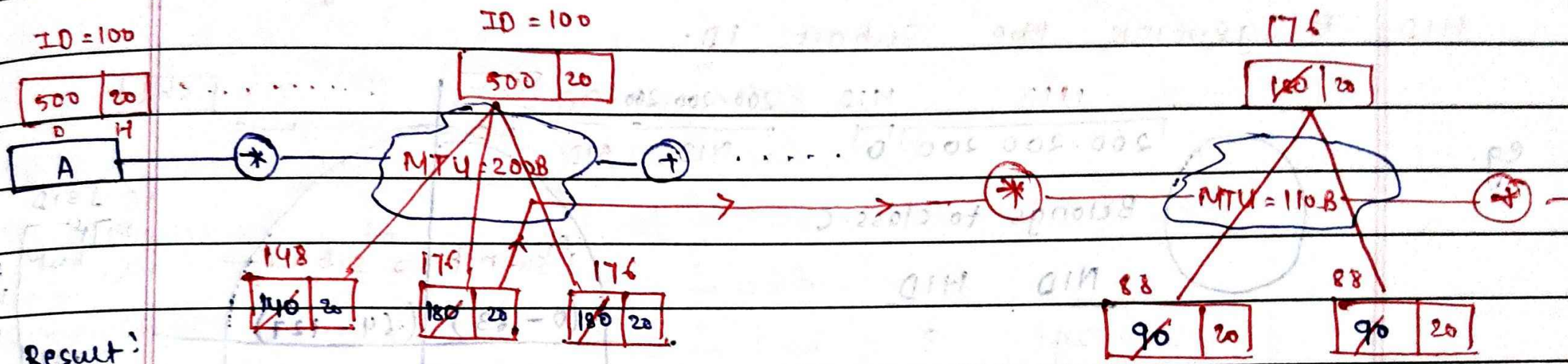
⑩ Header checksum :

- Calc. for Header part only.
- Calc. at each and every router.
- Every router makes some modification.

In qns, check 1st: IP addr is valid or not
2nd: Broadcast / Reserved

Page No.
Date

Fragmentation:



Result:

148 | 20	88 | 20	88 | 20	176 | 20	
100	100	100	100	ID No.
$\frac{176+88+88}{8} = 44$	$\frac{176+88}{8} = 33$	$\frac{176}{8} = 22$	$\frac{0}{8} = 0$	offset
0	1	1	1	MF
20	20	20	20	HL
168	108	108	196	TL

IP Support Protocol:

1) ICMP <Internet control msg protocol>

- companion protocol to IP.
- It defines collect^n of error msgs that are sent back to source host whenever the router is unable to process on IP datagram successfully.

ICMP

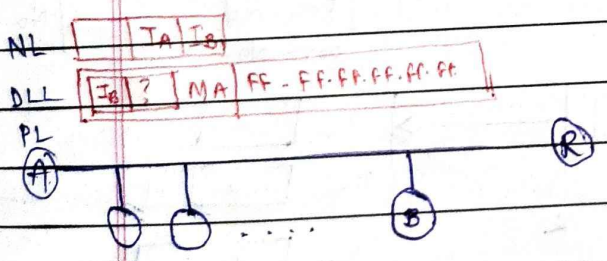
Error msg / feedback
- ~~Dest^n unreachable~~
- Time exceeds
- Redirect^n
- Echo request
- ICMP source quench

Req & Reply / Query
- Echo
- Timestamp
- Addr. mask
- Router Sdication / advertisement

(2) Address Mapping:

a) ARP <Address Resolution protocol>

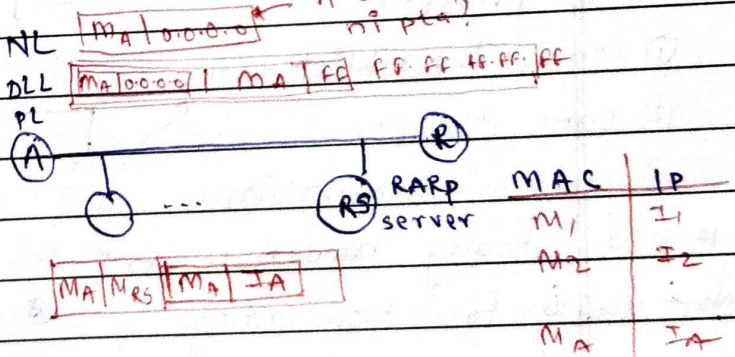
- IP → mac Addr. (given)
- ARP req: broadcasting at DLL.
  ARP response: unicasting containing its MAC addr.
- All broadcasting should never cross network boundaries or it will be blocked or filter all packets.



- ARP req: IP_B pta h app mujhe MAC_B batao.
  → broadcast.

b) RARP <Reverse ARP>

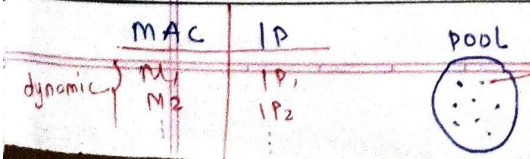
- MAC → IP. (given)
- RARP req. is broadcasting at n/w, so it cannot cross n/w boundaries.
- ~~Every n/w should have 1 RARP~~ Server within the n/w.
- mapping table → static.
  # IP add > # MAC addr.
  <independent of no. of users online at the time>



MAC	IP
M_1	I_1
M_2	I_2
M_A	I_A

c) DHCP <Dynamic host configuration protocol>

- An IP addr. can assign automatically from a pool of IP addr.
- IP addr. are leased for fixed amt of time
- DHCP server need not to be on same LAN as requesting client host.



d) BOOTP

- used to obtain IP addr. from server.
- capable to forward msg over Router
- So we only require 1 BOOTP server across all the LAN.
- Same as RARP and uses UDP.

wrt → with respect to

Routing Algo diff.

	DVR	LSR
1.	Distance vector Routing	1. link state routing.
2.	1980's	2. 1990's
3.	Bandwidth req^{d} is very less becz, we sent only distance vector packet	3. Bandwidth req^{d} is high becz, we sent entire link state packet.
4.	local knowledge	4. Global knowledge.
5.	Bellman ford algo.	5. Dijkstra algo. ~~flooding is used~~
6.	Traffic is very less.	6. Traffic is very high.
7.	Count to infinity problem	7. No count to ∞ problem.
8.	Persistent loops	8. Transient loops / temporary loop experience
9.	RIP → UDP & IGRP (Routing Information protocol)	9. OSPF & IISIS (open shortest path)

Note: ① The max^{m} Hop count allowed for RIP is 15 & Hop Count of 16 is considered as destination unreachable: ↳ unreachable n/w.

② RIP uses UDP as its transport protocol with port Number - 520

10.	Based on local knowledge it updates tables based on info. from neighbours.	10. Based on global knowledge it has knowledge about entire n/w.
11.	It optimizes cost wrt the Hops.	11. It optimizes cost wrt the dist, time, etc

Transport layer

- TCP, UDP
- socket & states of TCP
- congestion control protocol
- ~~leaky & token bucket~~
- Responsible for process to process / end to end communication <port addr = 16 bit>
- It create client-server architecture.

TL

TCP	UDP
<Transmission control protocol>	<User datagram protocol>
↓	
connect^n -oriented	connect^n -less

TCP:

- TCP connect^n are full duplex.
- It has 3 phases:
  1. Connect^n establishment
  2. Data transfer.
  3. Connect^n termination.
- It uses 3-way handshake for
- Not useful for broadcasting & Multicasting.

TCP header:

Source port 16b	Dest^n port 16b	4B
Sequence No. 32		4B
Acknowledgement No. 32		4B
HL | Reserved | U R G | A C K | P S H | R S T | S Y N | F I N	window size or Advertisement window 16	4B
checksum 16	urgent pointer 16	4B
Options (0-40) Byte		20B to 60B

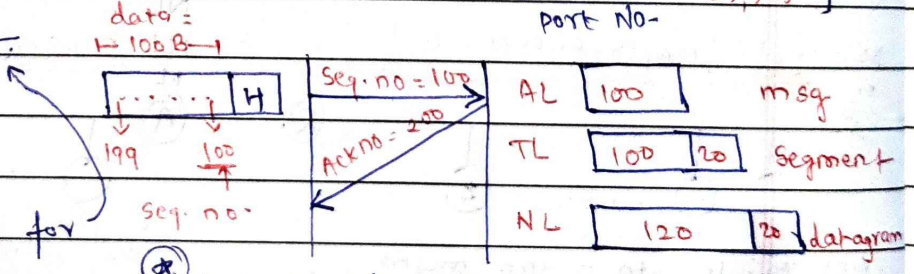
① HL : (4 bit)
- same as IPv4
- Min - 20B
- Max - 60B

② Port No : (16 bit) (0 - 65535)

0 ↓ 1023 — well known port No
1024 ↓ 49,151 — Registered OR Reserved port No.
49,152 ↓ 65,535 — dynamic port No

③ Seq. No : (32 bit)
→ seq. no. of first data byte

④ Ack. No : (32 bit)
→ seq. no. of next data byte



⑤ Data length = Total length − TCP(H) − IP(H)
at TL at NL

⑥ Urgent pointer: from current how far is urgent data.

⑦ Flags:
1. URG: set to 1: If urgent pointer used. → kuch part urgent h sab nahi
2. ACK: 1: Valid Ack
   0: Invalid Ack
3. PSH: data to buffer ma store mat kro seedha bhej do.
4. RST: 1: Reset connect^n due to host crash or some other reason.
5. SYN: 1: connect^n establish krna hai.
6. FIN: Release connect^n.

if SYN == 1 && Ack == 0 :
Piggybacked Ack is not here.

if SYN == 1 && Ack == 1 :
connect^n reply does have Ack

Case < WAT > LT → No problem
WAT < LT → prob <seq. no. repeat>

If seq No = 32 bit
Total seq No = 2^{32} = 4GB

If Data > 4GB
then, seqno repeat.

repeat from 0 again.

 # WRAP AROUND TIME

→ Time taken to wrap around.
→ WAT depends on bandwidth.

eg. ① B = 1MBPS = 10^6 BPS.

→ 10^6 B → 1s
10^6 seq No → 1s
$2^{32} \times 1$ seq No → $\frac{1}{10^6}$ sec $\times 2^{32}$

$$WAT = \frac{2^{32}}{10^6} = 4294.96 \text{ sec}$$

↓ after this time the seq. no. will repeat.

② B = 1GBPS = 10^9 BPS
LT = 180s
min seq no. req^d = ?

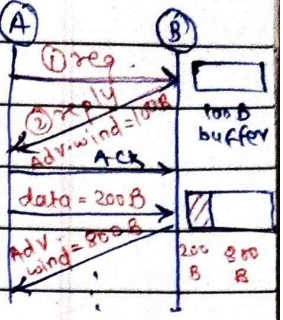
→ 1s → 10^9 B
1s → 10^9 seq No.
180 × 1s → 180×10^9 seq No.

min. seq. no. req^d = 180×10^9 seq. No
= LT × B.

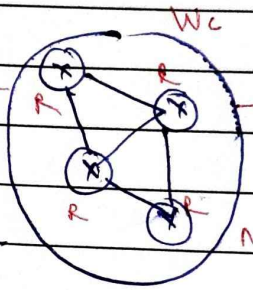
Min. seq. no. = $\lceil \log_2(180 \times 10^9) \rceil$ = 38bit
bits req^d to avoid WAT within LT = $\lceil \log_2(LT \times B) \rceil$

Note: 1> If seq. no. bit not given use 32bit
2> Bandwidth must be in BPs.

① Window size / Adv. window (16 bit)
• used for flow control.
• Wind. size = = 0 <receiver cannot accept any further.



 # TCP congestion control:



(S) Ws = 1MB
No. of segments = $\frac{1MB}{1kB}$ = 1024 seg.

(R) WR = 1MB
No. of segments = 2^{10} = 1024 seg.

→ Ws = min{ Wc, WR }
→ TH = $\frac{1}{2}$ WR

• congestion control Algo:
① slow start (Exponential Increase)
② Congestion Avoidance (Additive Increase)
③ congestion detection (Multiplicative decrease)

★ UDP header:

source port 16b	dest^n port 16b	4B
Total length 16b	checksum 16b	4B
		8B.

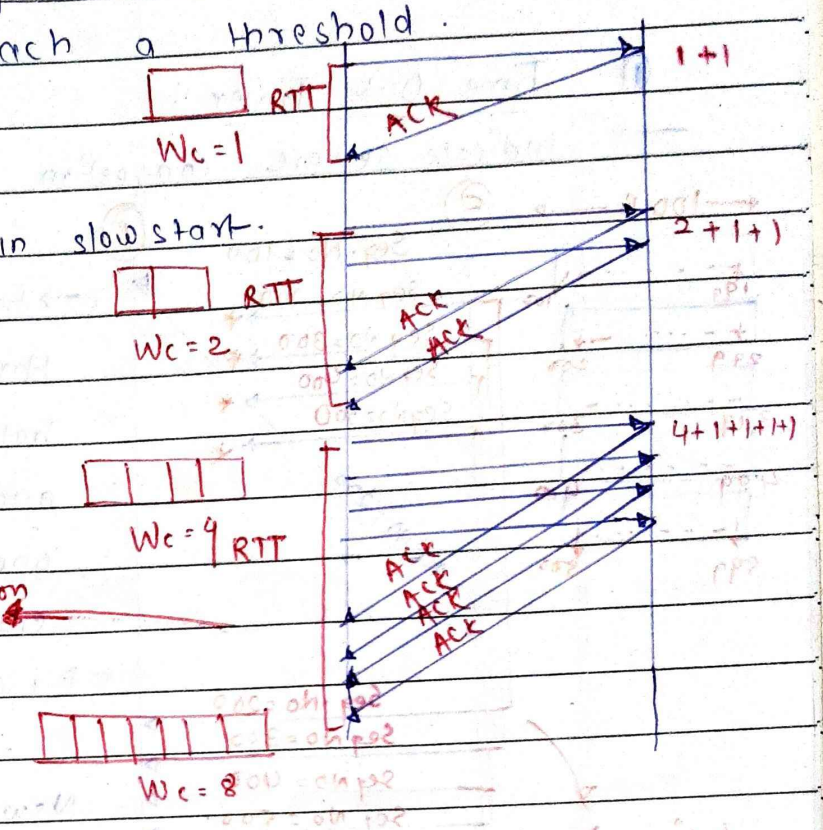
• uses when one req. one reply req^d.
eg. DNS, BOOTP, DHCP
• constant dataflow, fastness rather than reliability, RIP.

① Slow Start Phase:

→ In this, size of congestion window increases exponentially until it reach a threshold.

→ After one RTT congestion window will be double in slow start.

→ If an Ack arrives then $W_c = W_c + 1$.



Considered, transmission time as negligible

② Congestion Avoidance:

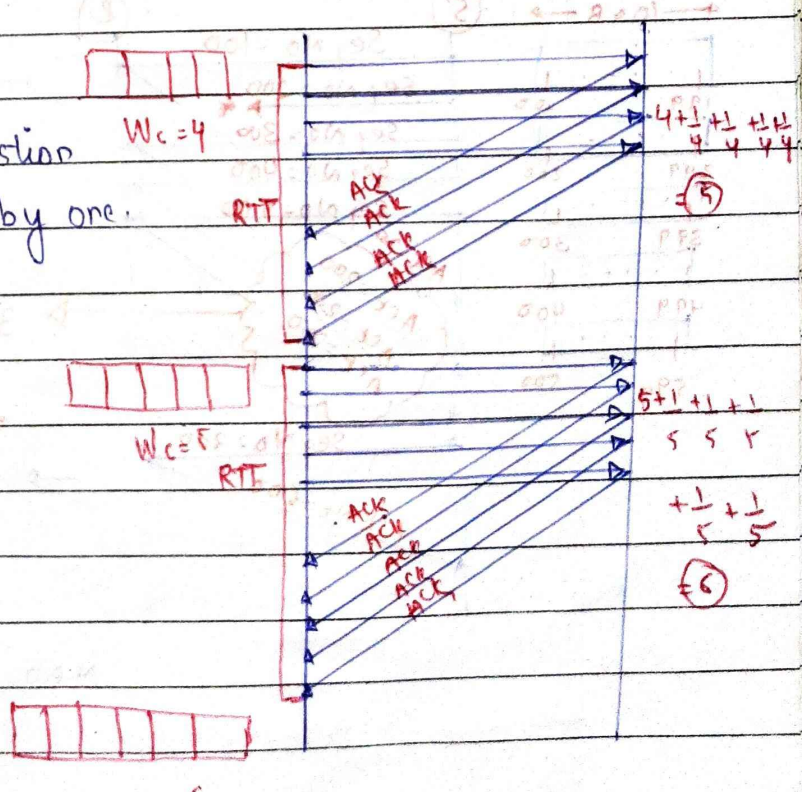
→ To avoid congestion before it happens we must slow down its exponential growth.

→ In congestion avoidance we use additive increase instead of exponential increase

→ After one RTT the congestion window will be increased by one only.

→ If an Ack arrives then

$$W_c = W_c + \frac{1}{W_c}$$

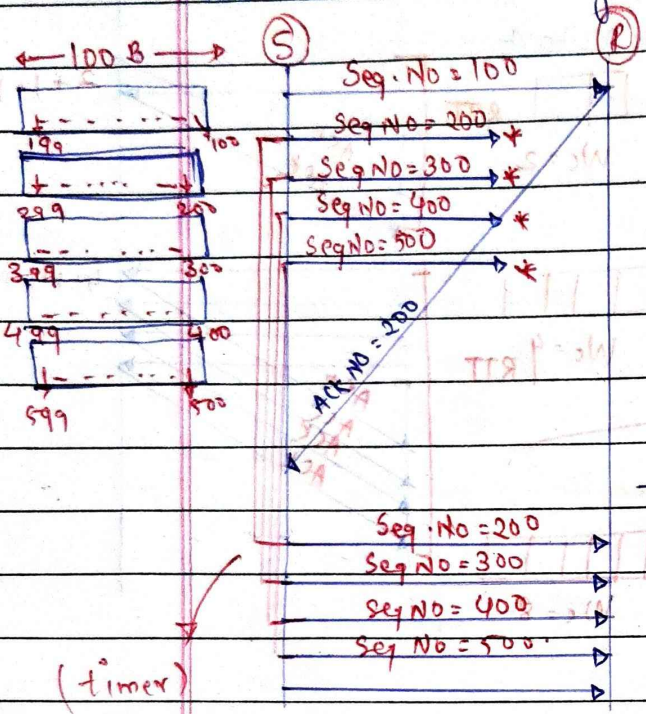


③ Congestion detection:
- ① Time out Timer
- ② 3 Duplicate ACK.

→ Congestion can be detected in 2 ways.

① Time Out Timer:

→ Indicate severe congestion cond^n.



→ In this case, the new threshold value is set to half of current window size and next transmission starts from one segment and algorithm enters in a slow start phase.

→ i.e, if current TH = 512 ↑ T.O.T

∴ New TH = $\frac{CTH}{2}$ = 256

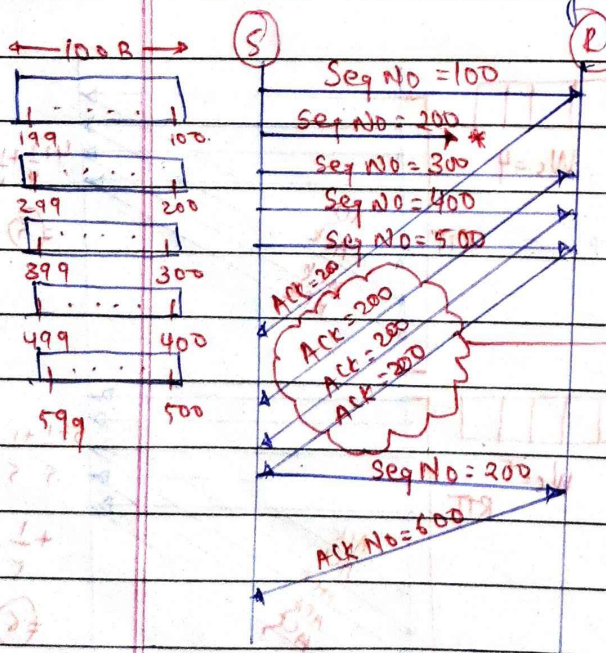
New transmission:

and 1, 2, 4, 8, 16, 32, 64, 128, 256, 257, 258, ... 1024, 1024 ...

→ here, retransmission of lost frame occurs after the timeout

② 3 Duplicate Ack:

→ Indicate mild congestion cond^n.



→ In this case, the new threshold value is set to half of current window size and next transmission starts from new threshold value & algorithm enters in a congestion avoidance phase.

→ 3 duplicate Ack.

→ i.e, current TH = 512 ↑ 3 dup ACK

∴ New TH = $\frac{CTH}{2}$ = 256

New transmission: 256, 257 ... 1024, 1024 ...

→ here, retransmission of lost frame occurs before the timeout.

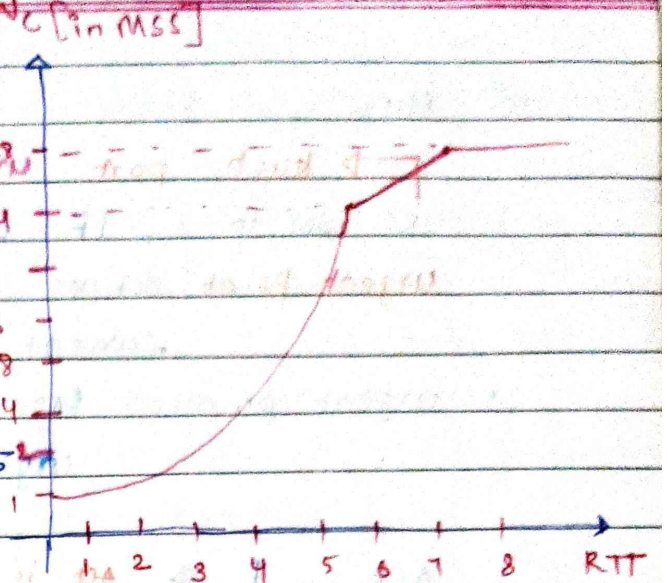
qn ① $W_R = 128$ kb

$MSS = 1$ kb

→ No. of segments $= \frac{128}{1} = 128$

$TH = \frac{1}{2} W_R = 64$ seg.

$\therefore W_c = 1, 2, 4, 8, 16, 32, 64, 65, 66, \ldots 128, 128 \ldots$



(case 2) $\therefore W_c = 1, 2, 4, 8, 16, 32, 64, 65, 66, 67, 68, 34, 35, 36, 37,$

38 1, 2, 4, 8, 16, 19, 20, 10, 11, 12, 1, 2, 4, 6, 7, 8, 9, 10, 128

NTH = 19 (T.O.T)

NTH = 10 (3 dup ACK)

NTH = 6 (T.O.T)

TH

NTH = 34 (3 Dup ACK)

② $W_R = 16000$ B | after how many RTT sender will send full window with 10 ms RTT

$MSS = 1000$ B

→ No. of seg $= 16$ $\Rightarrow$ 11 RTT

$TH = 8$ seg. $= 11 \times 10 = 110$ ms

Slow start: TH

1 | 2 | 4 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ →

③ Window size $= 2$ MSS

$TH = 8$ MSS

Assume T.O.T occurs at 5th transmission. Find C_W at end of 10th transmission.

→ 2 4 ⑧ 9 10 1 2 4 ⑤ 6 7 ✓

TH, T.O.T (5th), NTH, 10th

OR

2 4 ⑧ 9 10 2 4 ⑤ 6 7 8 ✓

TH, T.O.T (5th), NTH, 10th

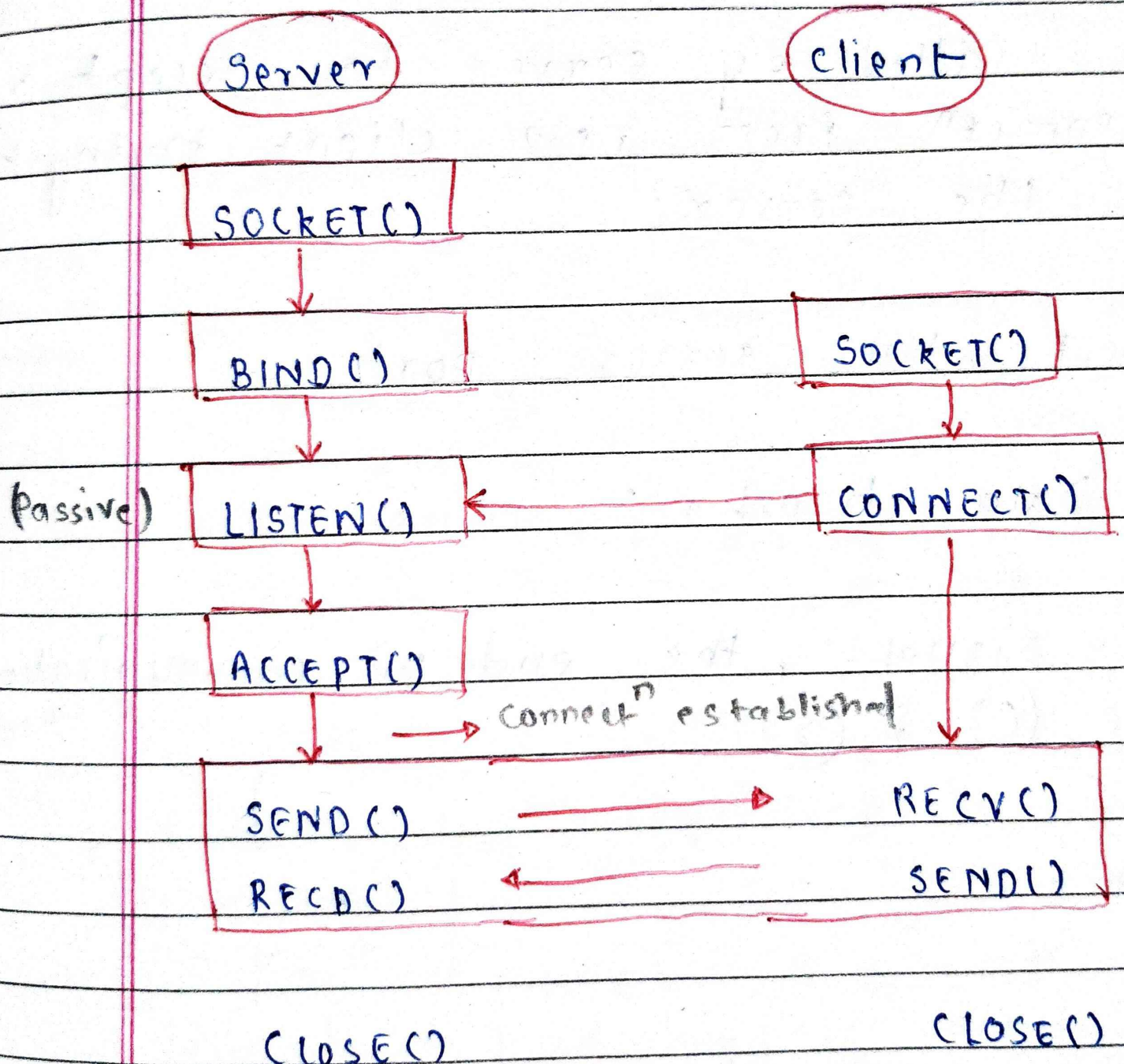
TCP

1. Dynamic header (20-60B)
2. End to End flow control
3. Error control (checksum mandatory)
4. connection-oriented
5. Reliable.
6. Sequence NO.
7. Ack NO.
8. overhead is high
9. keep track of order
10. protocols: HTTP, FTP, SMTP, POP

UDP

1. Fixed header (8B)
2. No flow control.
3. No error control. (checksum optional)
4. connectionless.
5. Not reliable.
6. NO seq no.
7. No ack no.
8. overhead is less.
9. No order.
10. protocols: DNS, SNMP, TFTP, NFS, BOOTP, DHCP, all realtime & multimedia protocols

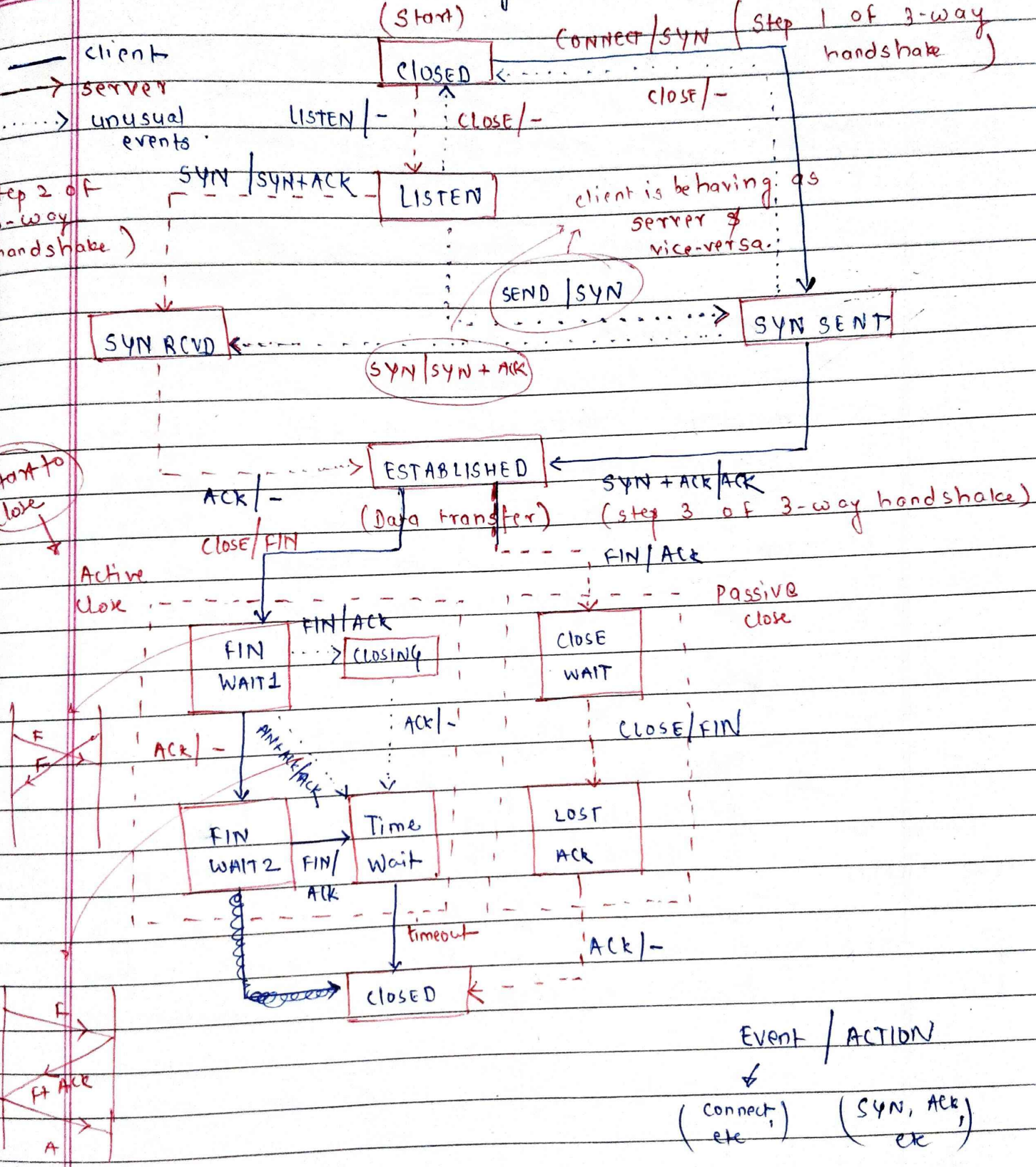
Berkley Sockets.

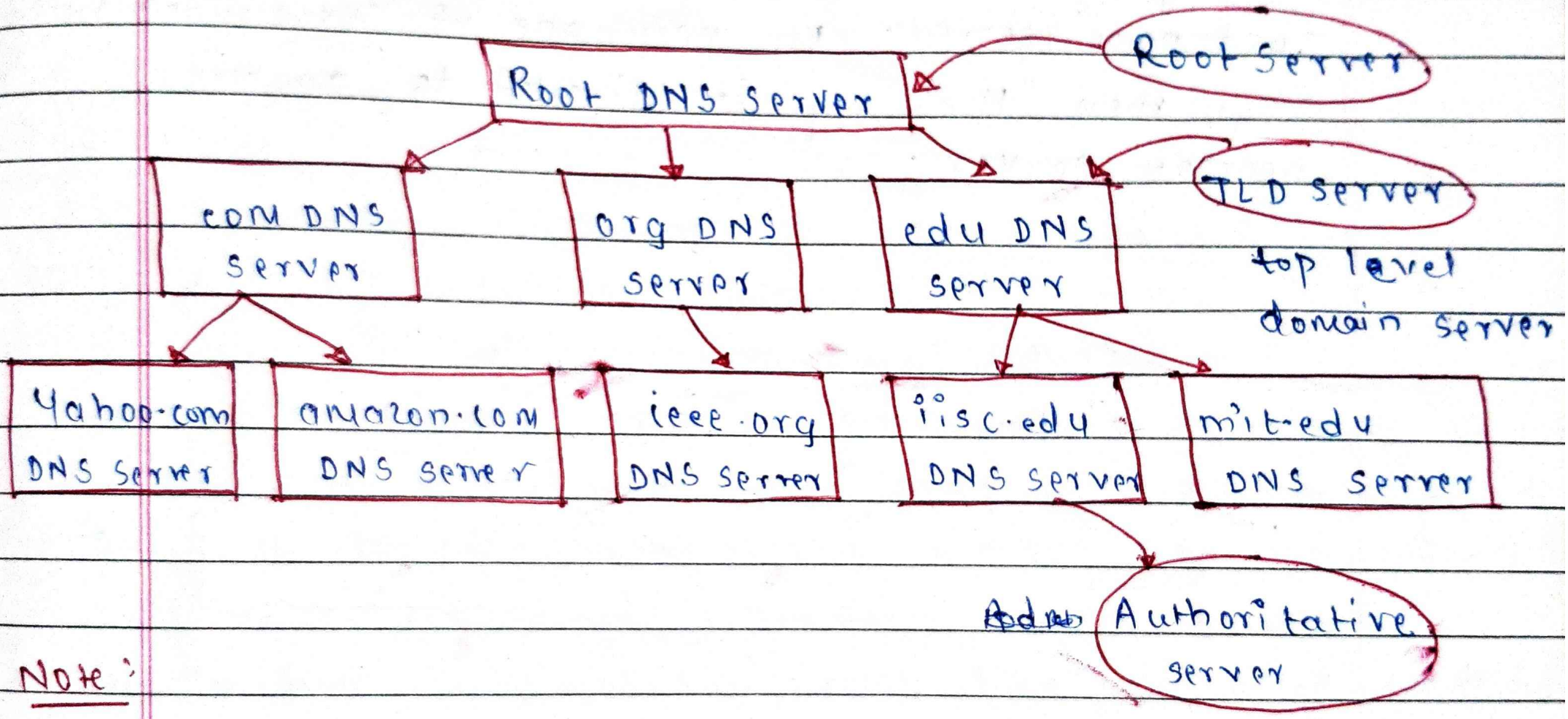


received → A/B → sent /- → kuch nhi bheja.

Page No. | Date

TCP State Transition diagram.





Note:

1. By default DNS uses UDP at transport layer protocol.
2. DNS can use either TCP or UDP.
3. In both cases, the well known port no = 53.
4. DNS query size $\leq$ 512 Byte $\longrightarrow$ UDP
5. DNS query size $>$ 512 Byte $\longrightarrow$ TCP

→ To get IP addr corresponding to domain name.

DNS :

Map domain name into IP addr. dynamically.

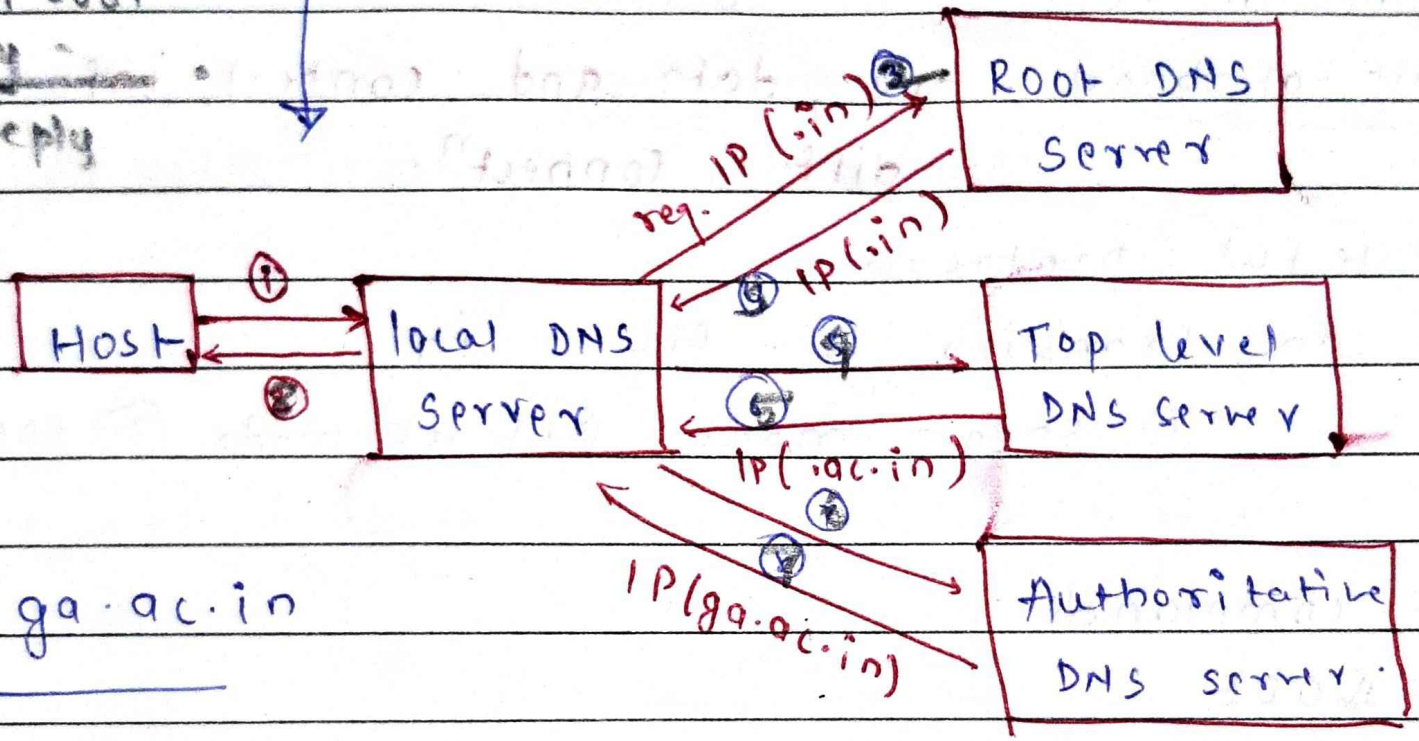
Iterative Method

Recursive method.

one req. & reply

⇕

UDP



www.ga.ac.in ←

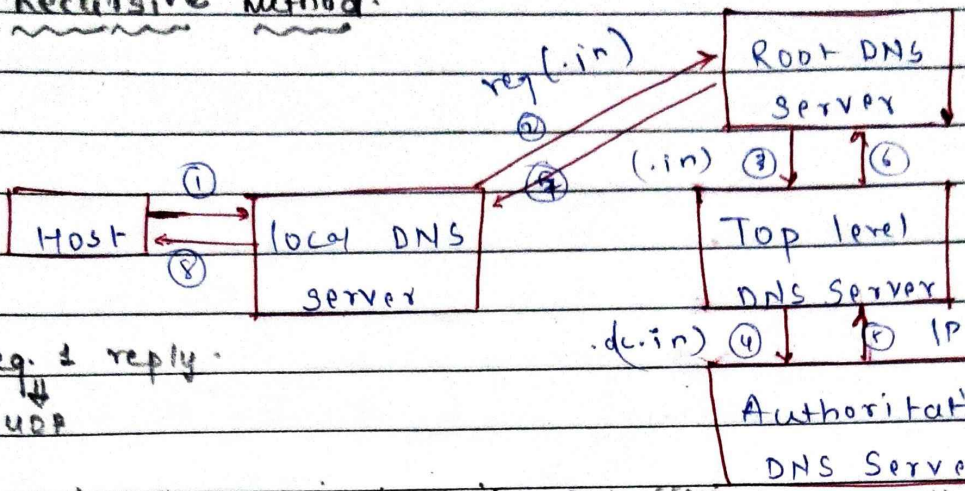
qn. www.ga.ac.in

2 + 2 + 2 = (6)

Total req-res pair

(www.iisc.edu.)

Recursive method:



one req. & reply.
↓
UDP

Domain name system	hypertext transfer protocol	Simple mail transfer protocol	Post office protocol	Internet mail access protocol	version 4	File transfer protocol
DNS	HTTP	SMTP	POP3 ver. 3	IMAP4	FTP	
Stateless	Stateless	Stateless	stateful	Stateful	Stateful	
DNS If query size ≤ 512 B UDP (by default)	TCP If > 512 B ; TCP	TCP	TCP	TCP	TCP	
Connectⁿ less	Connectⁿ less	oriented	oriented	oriented	oriented	
Non persistent	1.0 = non-persistent 1.1 = persistent	persistent	persistent	persistent	control connectⁿ = per. Data connectⁿ = non-per.	
–	–	push	pull	pull	–	
Port No – 53	80	25	110	143	(control) 21 20 (data) 20 - data connectⁿ 21 - control connectⁿ	
In-band	In-band	In-band	In-band	In-band	Out-of-band	

- State less : doesn't maintain any information of user.
- persistant : established for entire session <always open>
- non-persistant : opens/closes several time . established to access only one file.
- In-band : data & control info (request) flow over same connectⁿ.