Computer Networks:

$. IP, Subnetting, Supernetting

- IP (n bits)

NID	HID
K	n-K

2^K Networks, Each n/w size = 2^{n-K}

- Classless addressing:

IP (32b)

NID	HID
8b	24b

- Classful addressing:

1) CA (0-127) → first octet fixed bit → 0

IPs = 2^{31}

Networks = $2^7 - 2$ ⇒ 1-126 (bec. 0 & 127 ✗)

Hosts per N/w = $2^{24} - 2$

IP's per N/w = 2^{24}

8b	24b
0 ← 7b →	← 24b →
NID	HID

2) CB (128-191) → first octet + fixed bit → 10

IP's = 2^{30}

Networks = 2^{14}

Hosts per N/w = $2^{16} - 2$

IP's per N/w = 2^{16}

16b	16b
10 ← 14b →	← 16b →
NID	HID

3) CC (192-223) → first octet fixed bit → 110

IP's = 2^{29}

Networks = 2^{21}

Hosts per N/w = $2^8 - 2$

IP's per N/w = 2^8

24b	8b
110 ← 21b →	← 8b →
NID	HID

4) CD (224-239) → 1110 → # IP's = 2^{28} → used for multicasting

5) CE (240-255) → 1111 → # IP's = 2^{28} → Reserved

- Note:

NID	HID
(valid) 0's → NID	
(valid)	1's → DBA

Ex: 128.1.0.0 → NID

20.255.255.255 → DBA

1

- Casting:

1) Unicast → one host to one host

Data	Source IP	Destination IP

2) Limited Broadcasting → one to all (of same n/w)

Data	Source IP	255.255.255.255

→ LBA (all 1's)

2) Directed Broadcasting → one to all (of other n/w)

Data	Source IP	20.255.255.255

→ DBA (all 1's in HID)

- FLSM:

200.1.2.0 → 2 Subnets

class C

200.1.2 | SID | - - - - - - - -

NID | HID

2^8 → 2^7, 2^7

200.1.2.0 ... 200.1.2.127 (SID = 0) | 200.1.2.128 ... (SID = 1)

A (0-127) | B (128-255)

Total # IP's: 2^7 2^7

Configurable " : $2^7 - 2$ $2^7 - 2$

NID (IP of Subnet): A → 200.1.2.0

B → 200.1.2.128

DBA: A → 200.1.2.127

B → 200.1.2.255

Hosts per Subnet: A → 2^7

B → 2^7

Subnet Mask: A → 255.255.255.128

B → 255.255.255.128

Note:) Subnet mask:-

NID + SID → all 1's

HID → all 0's

Note:) Example: 130.1.2.3 (class B)

NID → 130.1.0.0

DBA → 130.1.255.255

LBA → 255.255.255.255

example: # IP's = 64 (FLSM wali) SM = ?

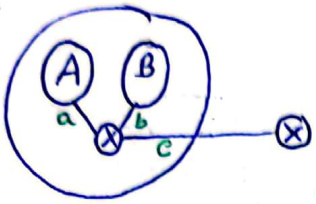
SM: 255.255.255.11 | 00 0000

NID | SID | HID

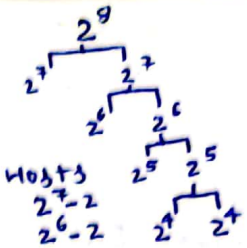
= 255.255.255.192

Note: Routing table (FLSM Example #1)

NID	SM	Interface
200.1.2.0	255.255.255.128	a
200.1.2.64	255.255.255.128	b
0.0.0.0	0.0.0.0	c



200.1.2.0 class C

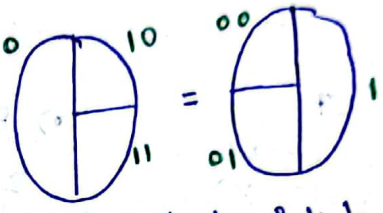


⇒ A

Subnet	#IP's	Range	SM	Hosts
A	2^7	0-127	24+1	2^7-2
B	2^6	128-191	24+2	2^6-2
C	2^5	192-223	24+3	2^5-2
D	2^4	224-239	24+4	2^4-2
E	2^4	240-255	24+4	2^4-2

Note:) Alter to previous

Note:)



•) To find NID of given IP:

→ SM AND Given IP = NID to which this IP belongs

→ A ⟶ B (A wants to msg B)

IP_A & SM_A = NID_{AA} ; IP_B & SM_A = NID_{BA}

If these two match, then A & B are in same n/w acc. to A

B ⟶ A (B wants to msg A)

Same ...

•) To find no. of subnets from SM:

Ex: SM = 255.255.255.128

1111 1111 . 1111 1111 . 1111 1111 . 1000 0000

All 1's → NID + SID = 25 b

All 0's → HID = 7 b

No. of subnets depend on class of IP

class C: 25 = 24 + SID ⇒ 2^1 subnets, SID = 1 b

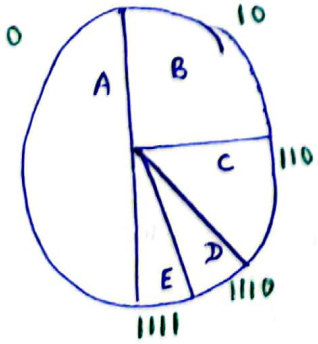
class B: 25 = 16 + SID ⇒ 2^9 subnets, SID = 9 b

Note:) ISP provides 4 things to a m/c

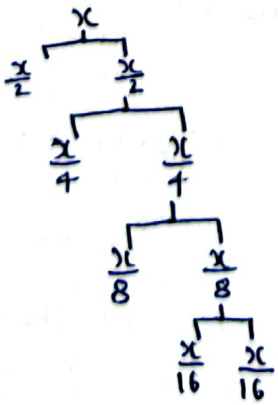
IP, SM, DNS server, Default router

•) VLSM:

200.1.2.0



Divide like this



•) CIDR: classless interdomain routing

→ Example:) 10.1.2.0/20 (Block ID or NID)

10.1.0000 | 0010.0000 0000

NID | HID

#IP's = Block size = $2^{HID} = 2^{12}$

#Hosts = $2^{12} - 2$

→ Rules:)

1. All IP's should be contiguous
2. Block size should be a power of 2
3. If block size = 2^k, then lst k bits of first IP should be all 0's

→ Example:

IP range: 150.10.20.64 to 150.10.20.127

Rule 1 ✓ Rule 2 ✓ Rule 3 ✓

Block size = (127 − 64 + 1) = 2^6

HID = 6

BID / NID = 32 − 6 = 26

150.10.20.01|00 0000

HID

CIDR: 150.10.20.64/26 = BID

150.10.20.127/26 = DBA

#IP's = 2^6

#Hosts = $2^6 - 2$

→ Subnetting in CIDR! Borrow bits from HID

Ex-1 FLSM
20.30.40.10/25 to 4 equal subnets

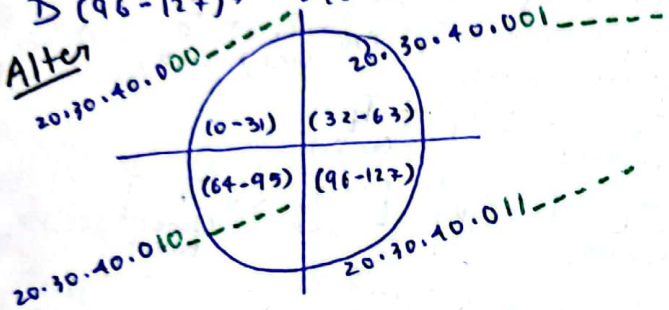
20.30.40.0 | 000 1010 ⇒ 20.30.40.0 to 20.30.40.127
NID | HID

size of N/W = 2^7 ⇒ 2^7 → 2^5 2^5 2^5 2^5

Subnets = 4 = 2^2 ⇒ 2 bits HID → NID

A (0-31) ⇒ 20.30.40.0 to 20.30.40.31 /27
B (32-63) ⇒ .32 to .63 /27
C (64-95) ⇒ .64 to .95 /27
D (96-127) ⇒ .96 to .127 /27

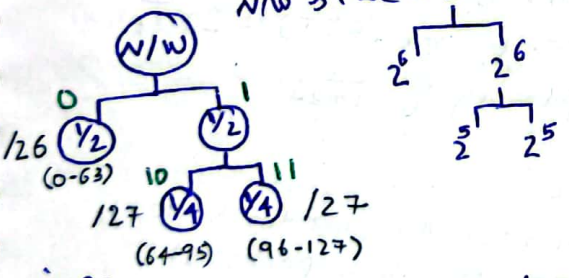
Alter



Ex-2 VLSM
20.30.40.10/25 → V_2 V_2 V_4 V_4

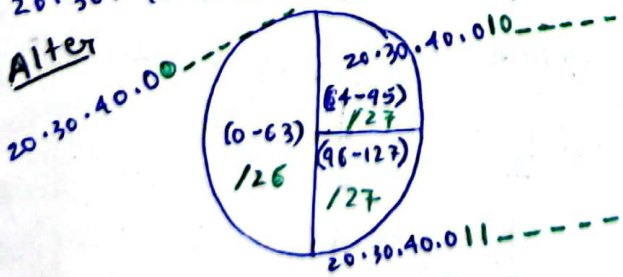
20.30.40.0 | 000 1010 ⇒ 20.30.40.0 to 20.30.40.127

N/W size = 2^7



i.e.
20.30.40.0 to 20.30.40.63 /26
20.30.40.64 to 20.30.40.95 /27
20.30.40.96 to 20.30.40.127 /27

Alter



→ Supernetting in CIDR:

Rules: 1. IP's contiguous
2. Total size should be power of 2
3. If total size = 2^K, then first IP (NID) div. by 2^K
i.e. LSB K bits 0's

Ex-1
200.1.0.0/24 → 0.0 to 0.255
200.1.1.0/24 → 1.0 to 1.255
200.1.2.0/24 → 2.0 to 2.255
200.1.3.0/24 → 3.0 to 3.255
Total block size = 4×2^8 ⇒ Rule 1 ✓
$= 2^{10}$ ⇒ Rule 2 ✓
Now,
200.1.0000 0000.0000 0000 all 0's
⇒ Rule 3 ✓

1) Supernet Mask:
fixed part → 1's
variable " → 0's

200.1.0000 0000.0000 0000
200.1.0000 0001.0000 0000
200.1.0000 0010.0000 0000
200.1.0000 0011.0000 0000
Fixed = 22b
var. = 10b

SM = 255.255.252.0

2) Supernet ID:

SM	255.255.252.0	
& IP	& 200.1.0.0	Supernet
SID	200.1.0.0	NID or ID

3) Private IP addresses:
- Router / DHCP assigns private IP to local m/c in office.
- NAT converts private IP ⇄ Public IP (to access internet)
- Public IP's are unique
  Private IP's are not unique

IP range	class	Default Subnet mask	size
1) 10.0.0.0 to 10.255.255.255	A	255.0.0.0	≃ 16 M
2) 172.16.0.0 to 172.31.255.255	B	255.255.0.0	≃ 1 M
3) 192.168.0.0 to 192.165.255.255	C	255.255.255.0	≃ 64 K

$. Flow control methods

#) Delays :-

1) Transmission delay (T_t)

$T_t = \frac{L}{B}$; L = Length of packet (bits), B = BW (bps)

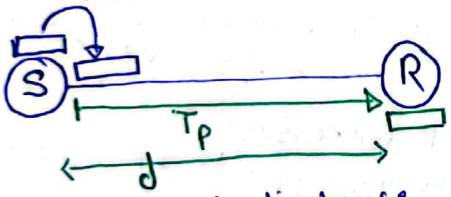
Ex1) $L = 1kb$, $B = 1Kbps$

$T_t = \frac{1024b}{1000 bps} = 1.024 s$

Note:

	Data	BW
K	2^{10}	10^3
M	2^{20}	10^6
G	2^{30}	10^9

2) Propagation delay (T_p)

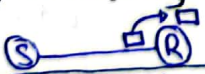


$T_p = \frac{d}{v}$; d = distance, v = velocity

Generally, $v \simeq 70\%$ of 3×10^8
$\simeq 2.1 \times 10^8$ m/s
(in optical fibre)

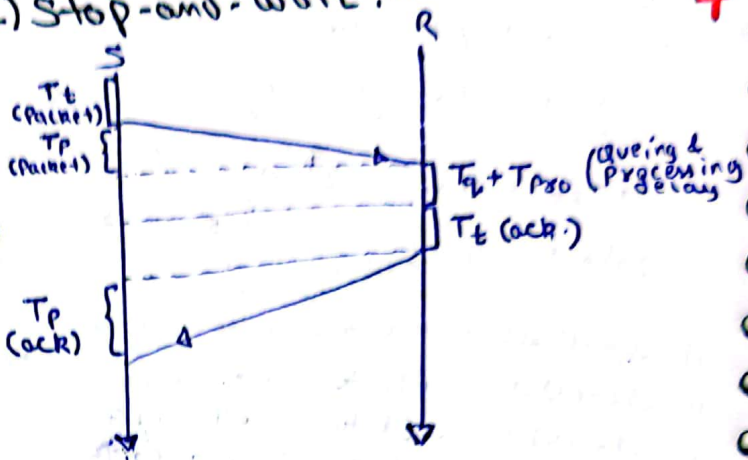
3) Queuing delay: $\simeq 0$

4) Processing delay: $\simeq 0$



#) Methods of flow control :

1) Stop-and-wait :-



i) Total time = T

$T = T_t + T_p + T_q + T_{pro} + T_t + T_p$
(Packet) (Packet) (ack) (ack)

$= T_t + 2T_p + T_t$
(packet) (ack.)

$$\boxed{T = T_t + 2T_p} \quad \left[\because T_t \text{ (ack)} << T_t \text{ (data) packet}\right]$$

ii) Efficiency : (link utilization)

$\eta = \frac{\text{useful time}}{\text{Total cycle time}} = \frac{T_t}{T_t + 2T_p} = \frac{1}{1+2a}$ [Here, $a = \frac{T_p}{T_t}$]

iii) Throughput / Effective BW / BW utilization / max. Data rate possible :-

$Th. = \eta \times BW$

Note: Round Trip Time (RTT) $= 2 \times T_p$

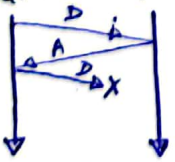
Note: On expanding $\eta = \frac{1}{1+2a}$

$\eta = \frac{1}{1 + 2 \cdot \frac{d}{v} \cdot \frac{B}{L}}$

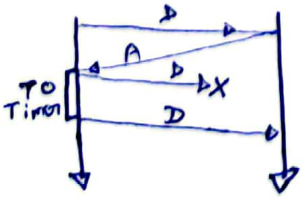
Stop & wait is
$d \uparrow \eta \downarrow$ (Better for LANs, not for WANs)
$L \uparrow \eta \uparrow$ (Better for big packets)

Note: Problems with S&W

i) Data lost (Deadlock):

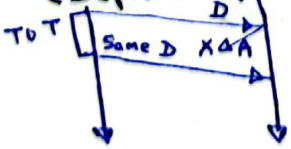


Solution: Timeout Timer

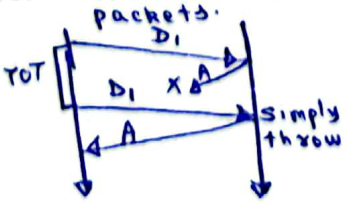


S&W + TOT = S&W ARQ

ii) Acknowledgement lost (Duplicate packet)

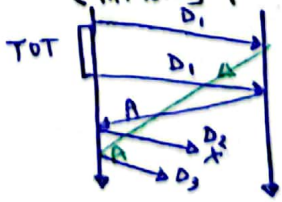


Sol^n: add numbering on packets.

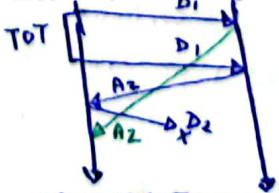


S&W + TOT + Seq. no. on Data

iii) Ack. delayed (missing packet)



Sol^n: numbering Ack. also



S&W + TOT + Seq. no. of Data + Seq. no. on ack.

⇒ All problems solved

- Questions on S&W!

1) S&W; 10 packets sent, every 4th packet lost, #packets Transmitted = ?

1 2 3 4 + 5 6 7 + 8 9 10 10 ⇒ 13

2) (S) ——error prob. = 0.2——▶ (R) ; 400 packets sent. #Packets Transmitted = ?

400 × 0.2 → 80 packets lost
80 × 0.2 → 16 " "
...

⇒ $400 + 400(0.2) + 400(0.2)^2 + \cdots$

$= \frac{400}{1-0.2} = 500$

3) n packets sent; Error prob. = p

$\#\text{packets sent} = \frac{n}{1-p}$

$(\because n + np + np^2 + \cdots)$.

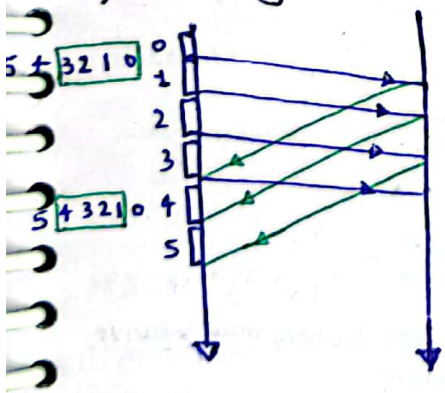
- Capacity of channel/pipe:

Capacity = BW × Tp (Half Duplex)
= 2 × BW × Tp (full Duplex)

Note: we use stopdwait just to send one packet over a channel/pipe.

⇒ Thick pipe (High capacity) → Don't use S&W
Thin pipe (Low capacity) → use S&W

2) Sliding window protocol:



$W_S = 1 + 2a$

$W_S \le \min\{1+2a, 2^n\}$

Seq. nos = $\min\{1+2a, 2^n\}$

n = # bits for seq. no = $\lceil \log_2(1+2a) \rceil$

Q) T_t = 1 ms, T_p = 49.5 ms; SWP
If we are restricted to only 6 bits in the seq. no. field. η_{max} = ?

$W_S = 1 + 2a = 100$

we've seq. nos = $2^6 = 64$

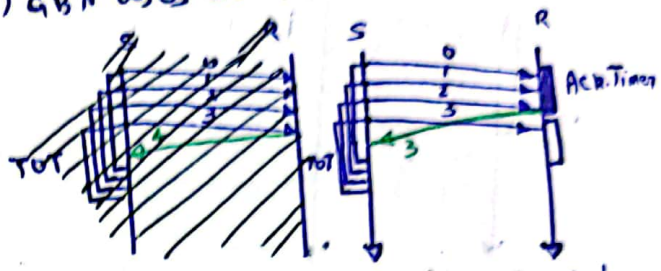
$\eta_{max} = \frac{64}{100} = 64\%$

Note: SWP ◁ Go Back N / Selective repeat

1) Go Back - N

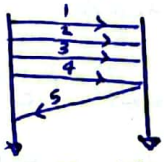
a) Sender w.s = $W_S = N$
b) Receiver w.s = $W_R = 1$
c) GBN uses cumulative ack.

N = no. of packets



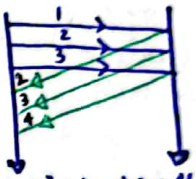
- If Ack. Timer too large then sender's timeout may occur
- If ack. Timer too small, then it becomes independent ack.

Note:
cumulative ack.



Less Traffic
Less reliable

Independent ack.



High traffic
High reliability

Q:) GB3; 10 packets transmit; every 5th packet lost.

1 2 3 4 5 6 7 5 6 7 8 9 7 8 9 10 9 10 ⇒ 18 packets sent

Q:) GB 4; 10 packets transmit; every 6th packet lost,

1 2 3 4 5 6 7 8 9 6 7 8 9 10 8 9 10 ⇒ 17 packets sent

- Rel.ⁿship b/w window sizes & Seq. nos in GBN

$W_S = N$
$W_R = 1$
$\#(\text{Seq. nos}) = N + 1$
(Required to detect duplicates)

N = no. of packets at sender side

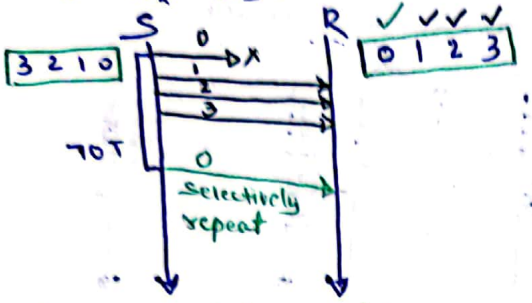
$\text{Bits} = \lceil \log_2(N+1) \rceil$

Alter

Bits = k
# seq. nos = 2^k
$W_S = 2^k - 1$
$W_R = 1$

ii) Selective repeat (SR):

1. $W_S > 1$; $W_S \in$ Integer
2. $W_R = W_S = N$



3. Acknowledgement:
   - Missing packet = discard silently
   - corrupted packet = Neg. Ack.

Note:
- for missing packet both GBN & SR discard silently
- for corrupted packet GBN discard silently but SR sends Neg. ack.

Q) $W_S = 3$; 10 packets send; SR used
Every 5th packet lost

1 2 3 4 5 6 5 7 8 9 9 10

⇒ #(Packets Sent) = 12

Q) 10 packets send; Every 4th lost; SR used

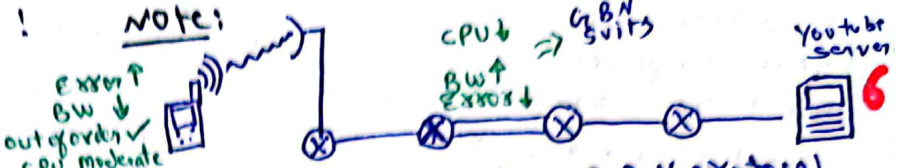
1 2 3 4 5 6 4 7 7 8 9 10 10

⇒ 13 packets transmitted

Note: flow control methods comparison

	S&W	GBN	SR
η	$\frac{1}{1+2a}$	$\frac{N}{1+2a}$	$\frac{N}{1+2a}$
Buffers	1+1 = 2	N+1	N+N
Seq. No.s	1+1 = 2	N+1	N+N
Retransmission (1 packet Lost)	1	N	1
BW	Low	High	Moderate
CPU (req.)	Low	Moderate	High
Implem.n complexity	Low	Moderate	complex

Note:



i.e. Link to Link → GBN protocol
end to end → SR protocol

Note: Two types of links:

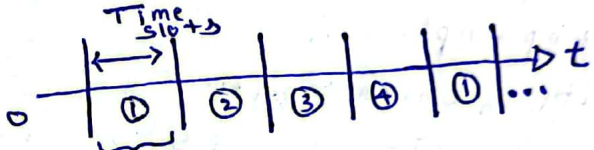
① Broadcast link: (needs access control method)

② Point to Point link: (S)—(R) (no need for access control method)

#) Methods of Access control:

1) Time division multiplexing (TDM):

- Lets say 4 stations on n/w link



1 cycle = $T_t + T_p$

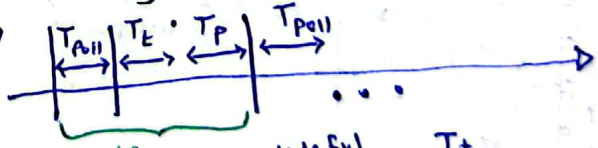
- $\eta = \frac{\text{useful}}{\text{Total}} = \frac{T_t}{T_t + T_p} = \frac{1}{1+a}$

- Disadv.: Time slot wasted if someone have no data to transmit.

Q:) N stations; each station requires 2Kbps BW; max.m no. of stations connected = ?
(The channel that we have ⇒ BW = 2 mbps)

$N = \frac{2 \text{ mbps}}{2 \text{ kbps}} = 1000$ stations

2) Polling:

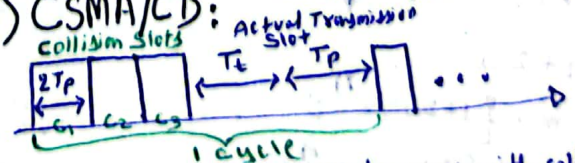


- $\eta = \frac{\text{useful}}{\text{Total}} = \frac{T_t}{T_{Poll} + T_t + T_p}$

Ethernet (or)

3) CSMA/CD:

- Disadv: Some stations may starve.



- Carrier sense multiple access with collision detection
- No acknowledgements
- For collision detection:-

$T_t \geq 2 T_p$

i.e. $L \geq 2 \cdot T_p \cdot B$

↳ min.m length of packet to detect collision

- $\eta = \frac{T_t}{c \cdot 2T_p + T_t + T_p} = \frac{1}{1 + 6.44a}$

(c = no. of collisions)

•) $\eta = \frac{1}{1 + 6.44 \times \frac{d}{v} \times \frac{B}{L}}$

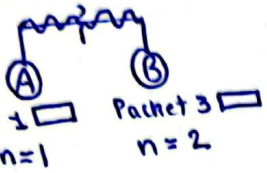
↓ η ↓ (Good for LANs not for WANs)

↑ η ↑ (Bigger packets suitable)

•) Back-Off Algo for CSMA/CD:

→ Defines that if collision has occured b/w A & B then how much time to wait before retransmission.

→ Assume,



n = collision no. of the packet

Waiting Time (for A) = Rand $(0, 2^n - 1) * T_{slot}$

Waiting Time (for B) = Rand $(0, 2^n - 1) \times T_{slot}$

Now, Possibilities:-

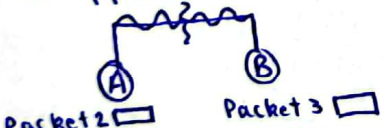
A (0-1)	B (0-3)	
0	0	
0	1	A won
0	2	
0	3	
1	0	B won
1	1	collision again
1	2	
1	3	

P(B won) = 1/8

P(A won) = 5/8

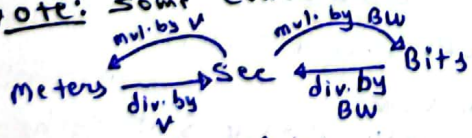
P(collision again) = 2/8

Suppose A won, then:-

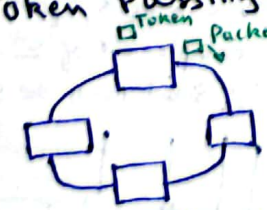


→ Capture effect: If a station 'x' won, then prob. of 'x' winning again ↑ & this continues.

Note: some conversions:



4) Token Passing:



• Ring Latency: Time taken by single bit to start from a location on ring & come back to it.

$RL = \frac{d}{v} + \frac{N \cdot b}{B}$ sec.

d → Length of ring
v → velocity of packet
N → no. of stations
b → bit delays by which every station delays a packet.

• Cycle Time: Time taken by token to start from a location & come back to it.

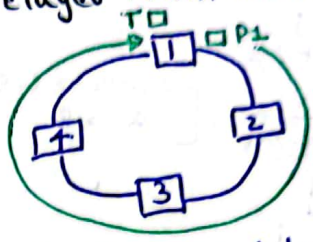
• Token Holding Time (THT): Time for which every station holds a packet.

• cycle Time $= \frac{d}{v} + N \times THT$

$= T_p + N \times THT$

• Scenarios for THT:-

i) Delayed Token Reinsertion (DTR)

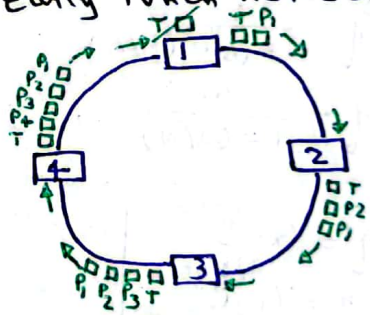


$THT = T_t + \text{Ring latency}$

$= T_t + T_p + N \times b \;(\to 0) = T_t + T_p$

$\eta_{DTR} = \frac{1}{1 + (N+1)(a/N)}$ → Efficiency ↓ → Reliability ↑

ii) Early Token Reinsertion (ETR)

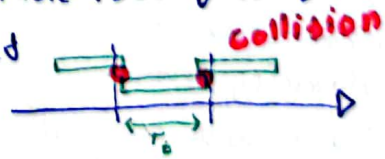


$THT = T_t$

$\eta_{ETR} = \frac{1}{1 + (a/N)}$ → Efficiency ↑ → Reliability ↓

5) Aloha (Additive Links Online Hawaii Area)

• No carrier sensing (send anytime)
• collisions possible
• Ack. there (No need of collision detection)
• Retransmission there (Bec. of Ack.)
• Aloha < Pure / Slotted



i) Pure Aloha:

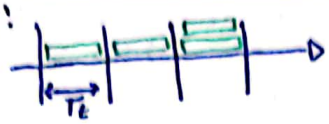
→ vulnerable Time $= 2T_t$

→ $\eta = G \cdot e^{-2G}$; G = No. of stations who wants to transmit in T_t slot

→ $\eta_{max} = 18.4\%$ at $G = 1/2$

[i.e. when 1 station transmits in 2 time slots]

ii) Slotted Aloha:



→ Vulnerable time = T_t

→ $\eta = G \cdot e^{-G}$

→ $\eta_{max} = 36.8\%$ at $G = 1$ [i.e. 1 station transmits in T_t slot]

Note: Efficiencies chart

Flow control

1) S&W: $\eta = \frac{1}{1+2a}$

2) SWP: $\eta = \frac{N}{1+2a}$; [For Both GBN & SR]

Access control

1) TDM: $\eta = \frac{1}{1+a}$

2) Polling: $\eta = \frac{T_t}{T_{poll} + T_t + T_p}$

3) Ethernet (or) CSMA/CD: $\eta = \frac{1}{1+6.44a}$

4) Token Ring:

i) DTR: $\eta = \frac{1}{1+(N+1)(a/N)}$

ii) ETR: $\eta = \frac{1}{1+(a/N)}$

5) Aloha:

		maxim
i) Pure:	$\eta = G \cdot e^{-2G}$	18.4%
ii) Slotted:	$\eta = G \cdot e^{-G}$	36.8%

#) Error Control Methods:

*) Error handling:

1) Error Detection:

i) Twice data transmission

ii) Parity checking

iii) CRC (Cyclic Redundancy Check)

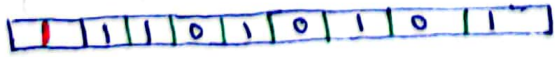
iv) Checksum

2) Error correction:

i) Hamming code

*) Twice data transmission: Send the data twice, if different, ask for retransmission.

*) Parity checking:
- Add 1 extra bit
- 0 ⇒ Even parity ⇒ Even no. of 1's
- 1 ⇒ odd parity ⇒ Odd no. of 1's
- Ex:



Odd Parity

*) CRC:

- Ex) Sequence of Data: 1011011

CRC Generator: 1101

At sender side

Pad (4-1) zero's to Data

```
1101 ) 10110 11000 (
     ⊕ 1101
       ----
       01100 11000
         1101
         ----
         000 111000
             1101
             ----
             001100
              1101
              ----
              0001
```

CRC

i.e. sender → Reciever

At Reciever side

```
1101 ) 10110 11001 (
     ⊕ 1101
       ----
       01100 11001
         1101
         ----
         000 111001
             1101
             ----
             0D110 1
              1101
              ----
              0000
```

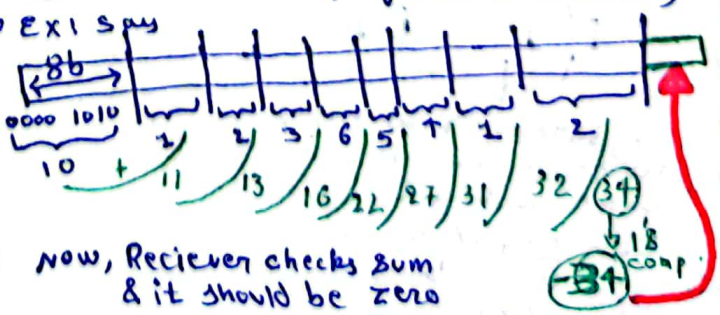
All zeros ⇒ correct Data

- If CRC Generator Polynomial $= x^3 + x + 1$

⇒ $1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 1 \cdot x^0$

⇒ CRC Generator = 1011

*) Checksum:
- Break data into slots of 8 bit each
- If last slot is less than 8 bits, append 0's
- At the end, add 1's complement of Σ Num. (Num is decimal rep.^{n} of 8 bit no. in the slot)
- Ex: Spy



Now, Reciever checks sum & it should be zero

Note: 100% Error detection not possible.

ISO-OSI (International Standards Organization-Open System Interconn.) Stack

*) Mandatory functions:
1) Error control
2) Flow control
3) Access control
4) Multiplexing/Demul.ng
5) Addressing
e.t.c

*) Optional Functions:
1) Encr.n & Decr.n
2) Checkpointing
3) Routing (optional bec. you can do Flooding)
e.t.c

*) Grouping of mandatory & optional functions is done by models like ISO-OSI, TCP/IP, ATM e.t.c

*) ISO-OSI Model:
→ Layers:

[user convinience]
7. Application Layer Alligator
6. Presentation Layer Pet
5. Session Layer Steve's
(Thick) 4. Transport Layer Touch
(complex) 3. Network Layer Not
(H/w & S/w) 2. Data link Layer Do
(H/w) 1. Physical Layer Please

→ Benefits of Layers:
- Div. & Conq.
- Encapsulation
- Abstraction
- Testing

*) Physical Layer:

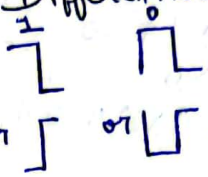
Physical Link	Signal
Copper	Electrical
Optical fiber	Light
wireless	EM waves

→ Transmission modes:
i) Simplex: S → R
ii) Half duplex: S ⇄ R (one at a time)
iii) Full duplex: S ⇄ R (Both at same time)

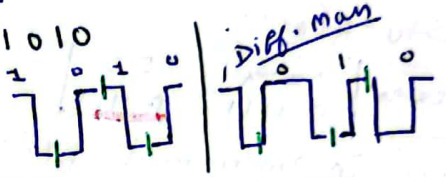
→ Topologies:
i) Bus
ii) Star (Central Hub)
iii) Ring
iv) Mesh
v) Hybrid

→ Encoding:
i) Manchester Encoding:

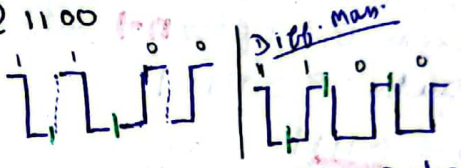
ii) Differential man. enc.



Ex-1) 1010
Man / Diff. Man



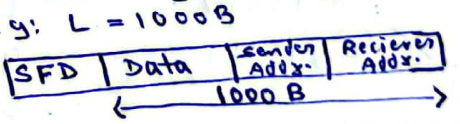
Ex-2) 1100
Man / Diff. Man



Note: Baud Rate = 2 × Bit-Rate
(Voltage/s) (bits/s)

*) DLL:
→ Responsibilities:
1) Flow control (S&W, GBN, SR)
2) Error control (CRC, checksum)
3) Access control (CSMA/CD, Token passing)
4) Framing
5) Physical addressing

→ Framing at DLL:
•) Types of frames
I) Fixed Length: Only SFD required
Eg: L = 1000B



II) Variable Length: SFD & ED required
a) Length field:
- Add length of to frame
- Used in ethernets
- Problem: If length bit gets corrupted

b) End Delimeter (ED):
- 

- Used in token rings
- Problem: ED matches with a data segment
- To resolve this problem:—
  i) Character Stuffing
  ii) Bit Stuffing

*) Character Stuffing:

SFD	Data	ED
SFD	10111010001111	$

let ED = '$'

- Problem: Data contained '$'
  Soln: Append \0 for $
  Ex) 10$11 → 10$\011
- Problem: Data contained '$\0'
  Soln: Append \0 for \0
  Ex) 10$\011 → 10$\010\011

*) Bit Stuffing:

SFD	Header	Data	ED
		01111	01111

add a zero in Data to differ Data from ED after n-1 bits
i.e. Data → 011101

Ex1) ED = 01111

Sender
D: 01111 → 011101
D: 011101 → 0111001

Receiver
01111 → 01111
011101 → 011101

ED = 011111

D: 011110 → 0111100
D: 011111 → 0111110

01111 0 → 01111
011111 → 011111

ED = 01111

D: 011 00011110 → 01111000111110

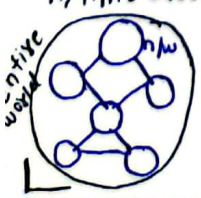
01111000111110 → 011110001111

→ Physical Addressing:

Note: Types of Addresses / Addressing
i) Physical = Unique in one n/w
ii) Logical = Unique in entire world

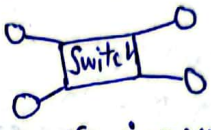
i) IP address:
- 32 bit s/w no.
- Globally unique
- Used as logical address

ii) MAC address:
- 48 bit H/w no.
- Hard coded on the ROM of NIC
- Globally unique
- Used as Physical address (in ethernet & Token ring)



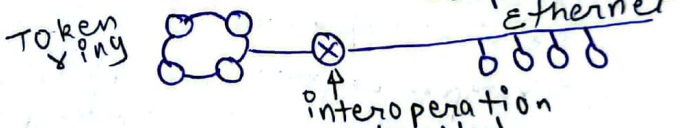
Note:
DLL
- LLC (Logical Link ctrl) → Flow ctrl, Error ctrl
- MAC (Medium access ctrl) → Framing, Access ctrl, Addressing

*) Network Layer:
→ Responsibilities:
1) Host to Host connectivity
2) Logical Addressing
3) Switching (connecting various n/w together)

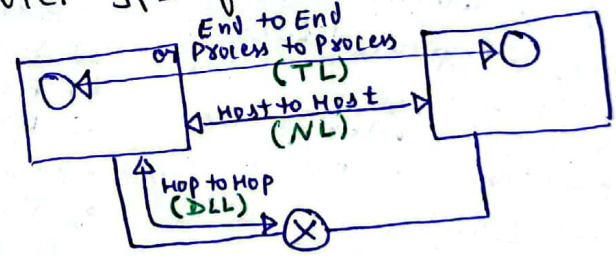


4) Routing (using routing table is called switching)

5) Congestion control
6) Fragmentation (Divide large packets into smaller ones for compactibility)

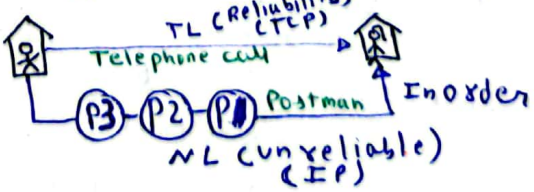


Note: Types of connections:-



*) Transport Layer:
→ Responsibilities:
1) End to End connection
2) Flow control (SR)
3) Error control (Checksum)
4) Segmentation
5) Multiplexing & Demultiplexing
6) congestion control

NOTE:



*) Session Layer:

→ Responsibilities:-

1) Authorization & Authentication
2) Checkpointing
3) Synchr.^{n}
4) Dialog ctrl
5) Logical grouping

*) Presentation Layer:

→ Responsibilities:

1) character translation
   (ASCii ↔ Unicode or so)
2) Encr.^{n} / Decr.^{n}
3) Compression

*) How all the layers work together:

#) How all the layers work together:

I_A I_G (got from DNS).

or

I_A	I_G
& SA	SA
NID_{AA}	NID_{GA}

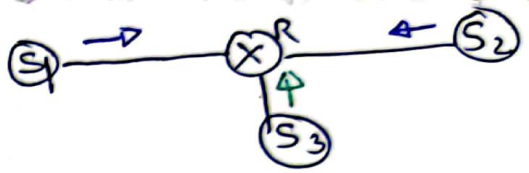
Case-1: $NID_{AA} == NID_{GA}$ ⇒ Send directly

Case-2: $NID_{AA} != NID_{GA}$ ⇒ Send like this :—

NID	SM	If

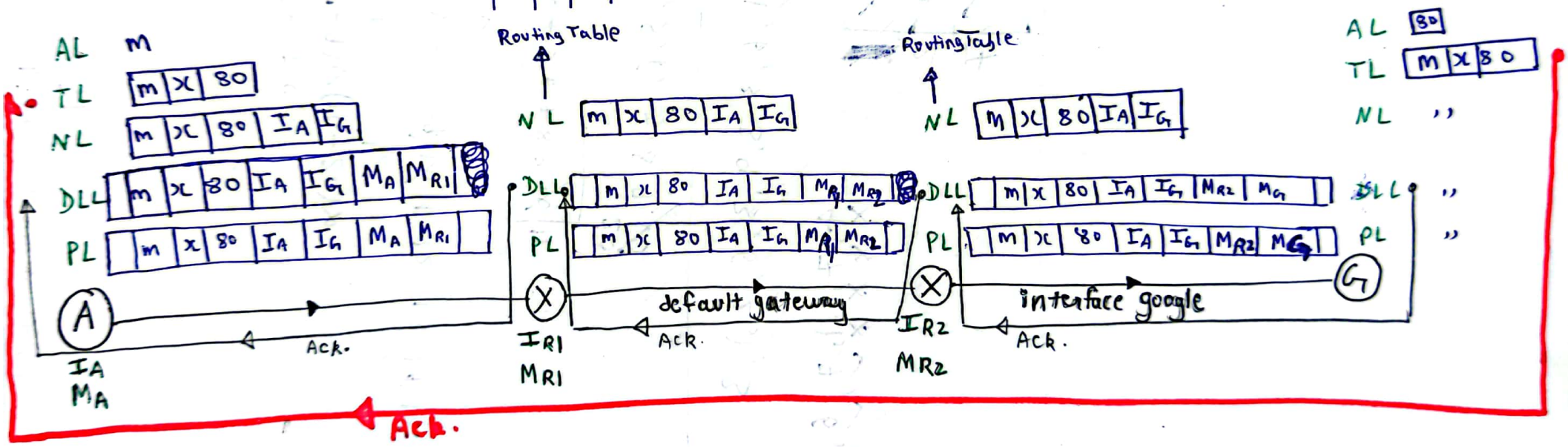
Routing Table

Note:)



at router R

NL		
DLL	DLL	DLL
PL	PL	PL



x → Port no of source

80 → Port no of destination

I_A → IP address of source.

I_G → " " " Destination

M_A → MAC address of source

M_{R1} → MAC address of R1

[Note: To get someone's MAC
ARP (I_{R1}) → M_{R1}
Address Resolution Protocol]

#) LAN Technologies: 1. Ethernet 2. Token ring

*) Ethernet (IEEE 802.3):

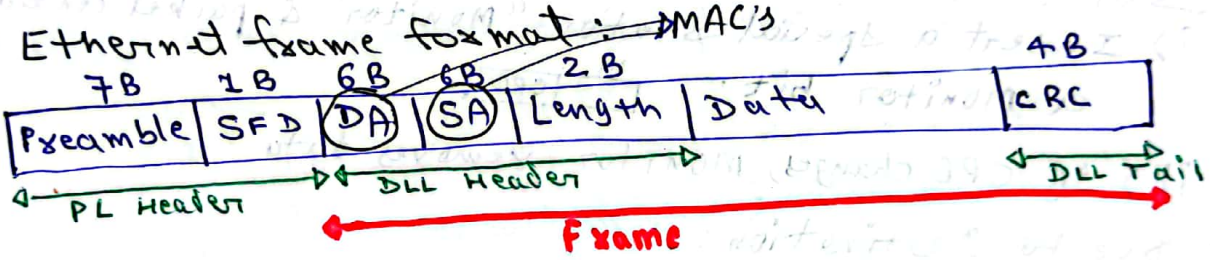
→ Specifications:

1) Topology = Bus
2) Access control method = CSMA/CD
3) Acknowledgements = No
4) Band width = 10 Mbps, 100 Mbps (Fast), 1 Gbps (Gigabit)
5) Encoding Technique = Manchester encoding

Note:
AL - Message
TL - Segment
NL - Datagram
DLL - Frame
PL - Single Protocol Data Unit (I-PDU)

message
sc
Data
fairig

→ Ethernet frame format: → MAC's



i) Preamble:
- 1010...10
- Used to dictate starting of frame & Synchroniz.n

ii) SFD:
- 10101011

iii) MAC Address:
- Physical address (6B)
- Types of MAC address
  - Unicast — LSB of 1st Byte is 0
  - Multicast — " " " " " 1
  - Broadcast — All 1's
- Ex: 1A : 2B : 34 : 48 : 56
  0001 1010 — LSB of 1st Byte
  ⇒ Unicast
- SA must always be unicast

iv) Frame size:
Bec. of CSMA/CD; $L \geq 2 \cdot Tp \cdot BW$
$L \geq 64B$ (Frame size)
for collision detection

v) Data

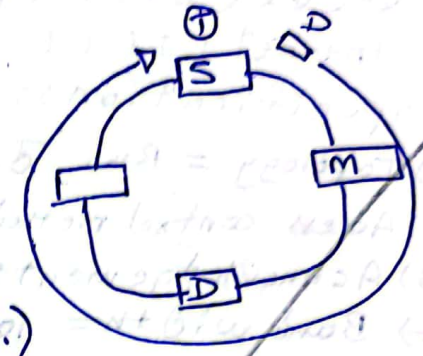
	min	max
Data	46 B	1500 B
frame	64B	1518B

→ Ethernet disadvantages:
- Not useful for:-
  i) Real time app.n (Bec. of collisi...)
  ii) Interative app.ns (Bec. at leas... 46B data needed so padding useless data)
  iii) Client-server app.ns (Bec. ne... priorities)

*) Token Ring (IEEE 802.5)

→ Specifications:

1) Topology = Ring
2) Access control method = Token passing
3) Direction = Unidirectional (Simplex)
4) BW = 4 mbps, 16 mbps
5) Ack. = Yes (Piggy backing ack.)
6) Encoding = Diff. Man. Enc



→ Problems with token ring

1) Due to source:
   i) Orphan packet problem (Sender gets down after sending data, so packet will rotate forever)
   ii) Stray packet problem (Data packet got corrupted)

Soln:-
i) Insert a special station "Monitor" & packet contains a monitor bit.

ii) If CRC changes, Monitor removes data

2) Due to Destination:
   i) Destination Down
   ii) Destination Busy (Buffers full)
   iii) Error in Data
   iv) Data copied successfully

Problem at source & destination

Soln:-
Header contains two bits

Data Available → Copied

	A	C
initial	0	0
copied	1	1
Error	1	0
not possible	0	1
Data not available	0	0

Dest. busy → 0
Dest. Error → 1

final

3) Due to Token:

i) Captured token: Someone using token & not releasing it cuz it has large data to send. (Monopolization)

Soln:-
max THT (10ms by default)

Q:) max. THT = 10ms
BW = 4mbps
max. frame size (in ETR) = ?

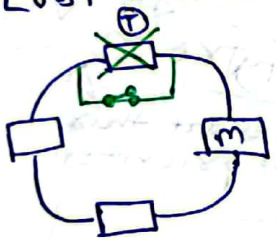
$= THT_{max} \times BW$

= 10ms x 4mbps
= 40,000 bits

Q:) THT_{max} = 10ms
RL = 9ms
BW = 4mbps
max. frame size (in DTR) = ?

$= (THT - RL) \times BW$
$= 1ms \times 4mbps$
$= 4000$ bits

ii) Lost token: After getting token, station gets down



- TRT (min.) = RL
- TRT (max.) = RL + N * THT

If within this ↑ time ⏳ token doesn't get to monitor, token is lost.
(TRT = Token Return Time)

Sol.n:-
Monitor generates the token

iii) Corrupted token: Token is corrupted

Sol.n:-
monitor regenerates the token after waiting TRT (max) time.

4) Due to monitor

i) Monitor Down: monitor is down

Sol.n:) Periodically monitor sends a heart beat message saying I'm alive (AMP frame i.e. Active Monitor Presence)
If not alive then form new monitor (Polling)

ii) Monitor Hacked (Malfunctioning): Monitor is just sending AMP frame, but not doing any work.

Sol.n:) Human intervention is required.

→ Advantages:
It has lot of problems, but still in use bec.
i) Real time app.ns ✓ (No collision)
ii) Interactive app.ns ✓ (Any data size allowed)
iii) Client-Server app.ns ✓ (Priorities allowed)

→ Token Ring frame format :-

Data frame

1B	1B	1B	6B	6B		4B	1B	1B
SD	AC	FC	DA	SA	Data	CRC	ED	FS

(DA, SA → MAC's)

SD → Start delimiter
AC → Access control
FC → Frame control
ED → End delimiter
FS → Frame status

Token

1B	1B	1B
SD	AC	ED

i) SD:

J	K	0	0	J	K	0	0

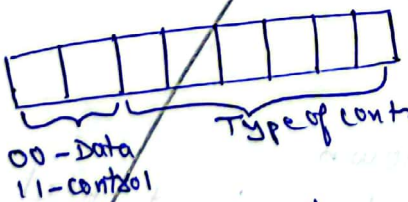
0 ⎍ or ⎍ | J
1 ⎍ or ⎍ | K

ii) AC:

P	P	P	T	M	R	R	R

R → Reservation
M → monitor bit (1 means stamped)
T → TOken bit (1 means I have token)
P → Priority Note:

iii) FC:



00 - Data
11 - control
Type of control

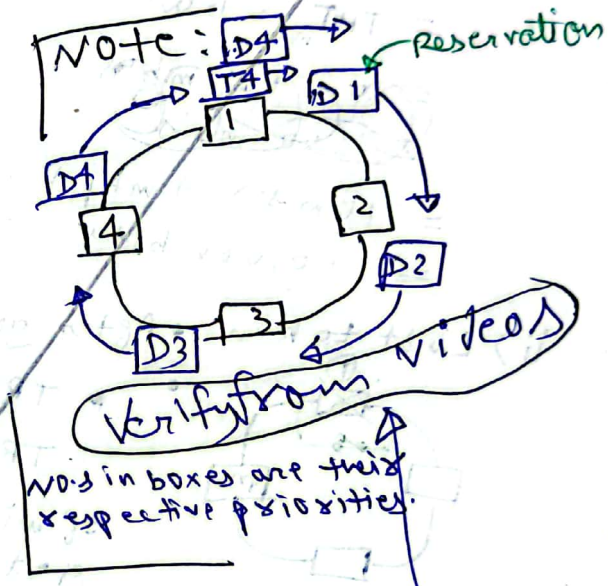
Here, Type of control maybe:-
AMP (Active monitor Presence)
Polling (new monitor)
Purge (Everyone Stop)
Beckon (cut in ring)

iv) ED:

J	K	1	1	J	K	I	E

I → Information (i.e. more information is following or not)
E → Error (i.e. Data not accepted if E=1)

Note:



v) FS:

A	C	0	0	A	C	0	0

Available → A, copied → C

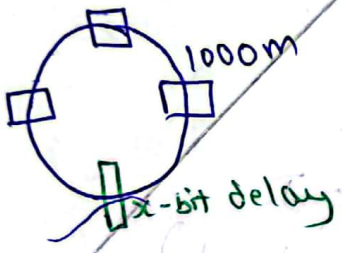
- Taking 2 copies bec. CRC not taking care of FS.
- CRC not taking care of FS so that receiver need not change CRC when destination A or C bit changes

→ Min.m Length of token ring: (d)

$d \geq \frac{2 \times 4 \times V}{BW}$

at BW = 4 mbps, V = 2×10^8 m/s | $d \geq 1200$ m

But if you've only 1000 m wire:-



1200 − 1000 ⇒ 200 m

200 m → bits?

meters → (/V) Sec → (*BW) bits

$\Rightarrow \frac{200 \times 4 \times 10^6}{2 \times 10^8} = 4$ bit time.

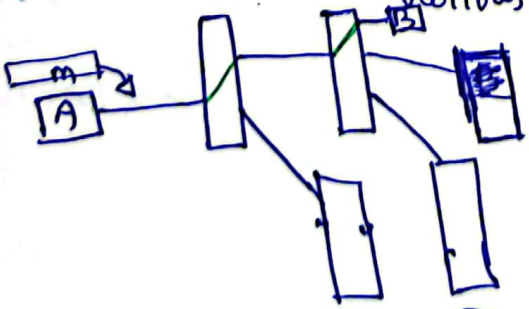
⇒ 4 bit delay is required for 1000 m wire

#) Switching:

*) Classification:

Switching
- Circuit switching
- Packet switching → Virtual Circuits / Datagram
- Message switching

*) Circuit switching: Manual / Physical intervention to connect various Jacks / Stations



m → message size
B → BW
x → # (Hops)
d → Length of a hop
V → velocity

- Total Time = Setup Time + $\frac{M}{B}$ (T_t) + $\frac{x \times d}{V}$ (T_p) + Tear down time

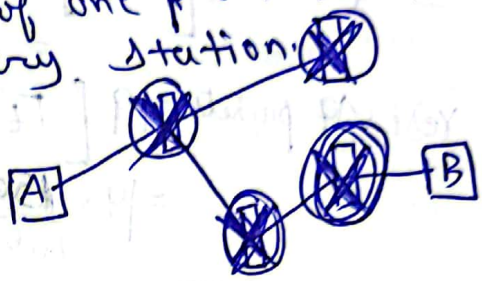
- No header required
- No reordering there
- Applied at PL
- It's obsolete

C
Array
Stack
Queue
Recursion
12-18

*) Packet Switching:

- Automatically multiplexing (selection of one path from many paths) is done at every station.
- Total time = $x \times \frac{m}{B}$ + $x \times \frac{d}{v}$

T_t T_p



note:) circuit switching VS Packet Switching

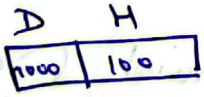
Extra {Setup + Teardown} $(x-1) \times \frac{m}{B}$

if M ↑ ⇒ CS good
if M ↓ ⇒ PS good

note:) Pipelining (Packetization) in Packet Switching:

Ex:) Data = 1000 B
Bw = 1 mBps = 10^6 Bps
Header = 100 B

case-I)

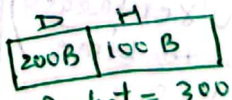


⇒ Packet = 1100 B

$T_t = \frac{L}{B} = \frac{1100}{10^6} = 1.1$ ms

Total time = $3 \times T_t = 3.3$ ms

case-II) (Pipelining)
#Packets = 5
Data (in each Packet) = $\frac{1000}{5} = 200$ B



⇒ Packet = 300 B

$T_t = \frac{L}{B} = \frac{300}{10^6} = 0.3$ ms

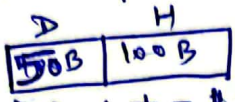
⇒ For 1st packet = $[3 \times T_t] = 0.9$ ms

For Remaining 4 packets = $4[T_t] = 1.2$ ms

TT = 2.1 ms

case-III :) (Pipelining) # Packets = 20

Data (in each packet) = $\frac{1000}{20}$ = 50 B



⇒ Packet = 150 B

I^{st} packet = $[3 \times T_t] = 3 \times \frac{150}{106}$

$= 0.45$ ms

Rest of 19 packets = $19 [T_t]$

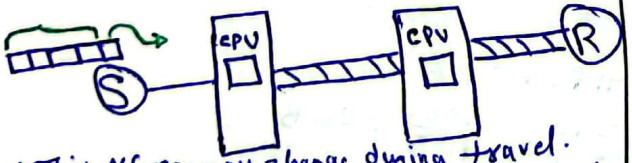
$= 19 \times \frac{150}{106} = 2.85$ ms

TT = 3.3 ms

Conclusion: Packetization helps reducing TT but you've to choose #Packets wisely.

Virtual Circuits

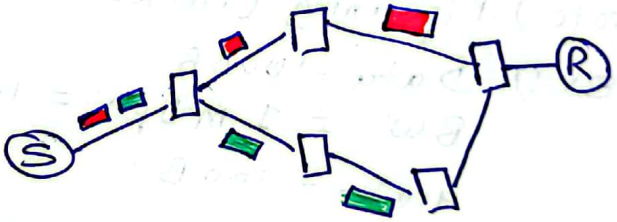
- All packets follow I^{st} packet
- Reservations are made (By I^{st} packet)
- Here, header of a packet carry a VC number



- This VC no. may change during travel.
- Ex: calling (You're charged acc. to time)
- Header required only for I^{st} packet (Global header) Rest packets only have local header.
- Connection oriented (Resource reservation there)
- Insure inorder arrival (cuz. entire data follow same path)
- Reliable
- cost High
- Ex: ATM (Asynchronous Transfer mode)

Datagram

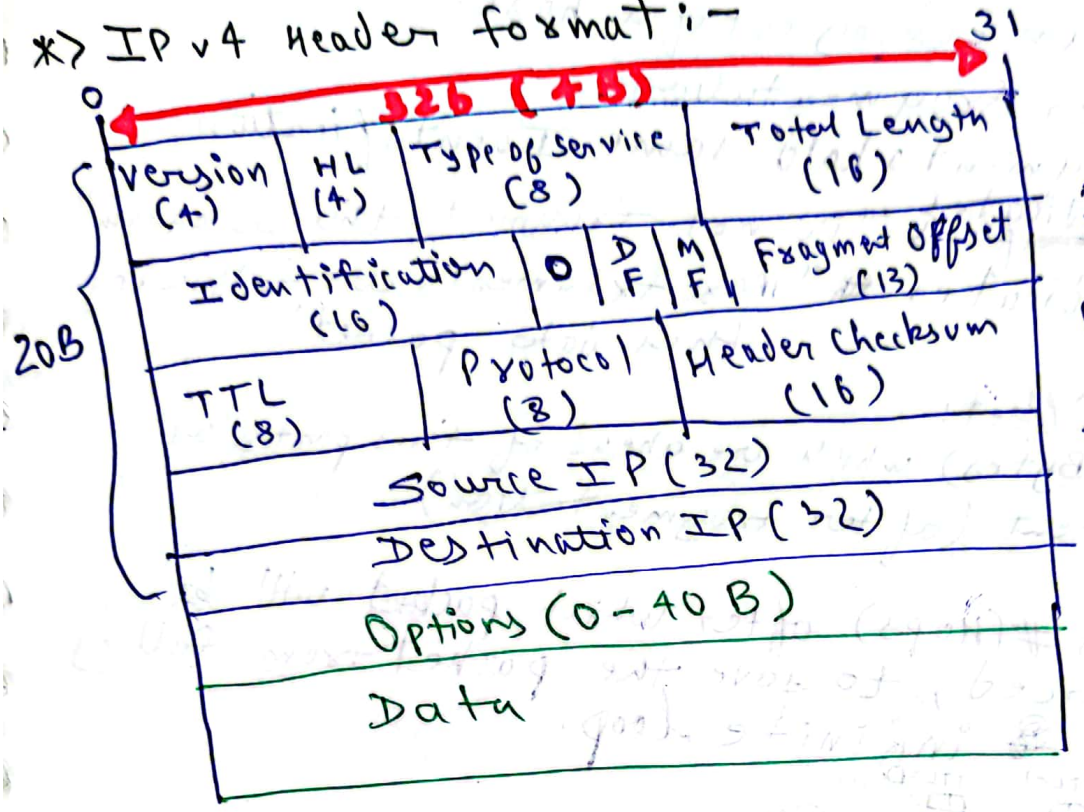
- All packets go independently
- NO reservation



- Ex: E-mail (You're charged acc. to data transfer)
- Header required for everyone
- connection less (Resource reservation not there)
- Out of order packets may arrive
- Not Reliable
- cost low
- Ex: IP (Internet Protocol)

#) IP (Internet Protocol)

*) IP v4 Header format:-



HL → Header Length
DF → Do not fragment
MF → More fragments
TTL → Time to live

→ HL:
- Min.^{m} = 20 B (options = 0 B)
  Max.^{m} = 60 B (options = 40 B)
- Scaling factor:
  HL is 4b ⇒ it can hold a value 15 B,
  But HL_{max} = 60 B ; so, scaling factor = $\frac{60}{15}$ = 4
  i.e. Divide HL by 4 & then put the resultant no. in HL field

  i.e. $20 \le HL \le 60$ (actual) , $5 \le HL \le 15$ (field)

- If HL is not exact multiple of 4, then we dummy data.
  Ex: HL = 30B, add 2B dummy (padding) ⇒ HL = 32 B ⇒ HLF = 8
- In WC you might need to pad 3B data, ध्यान.

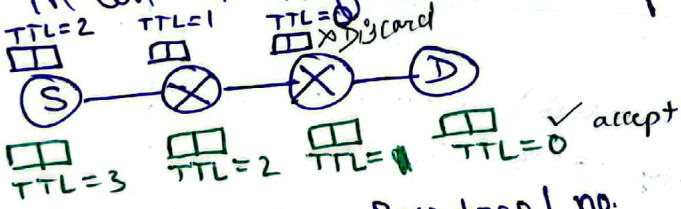
→ version:
V1, V2, V3, V4, V5, V6
0100 0110

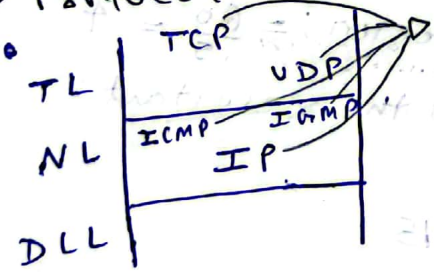
→ Identification:
- Contains unique identification number of the datagram coming out of a host.
- Used for fragmentation
- All fragments hold same Identification.

→ DF: Indicates "Do not fragment this Datagram".

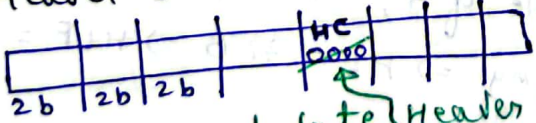
→ MF: Indicates "More fragments are following this data packet".

→ Fragment Offset:
(Data Bytes) which are ahead of this particular fragment (after fragmentation)

→ TTL:
- Defines #(Hops) after which packet will be discarded, to save the packet from falling in an infinite loop.
- 

→ Protocol: containing Protocol no.
- 

- If a router's buffer is full, see the protocol of incoming packet
  if TCP ⇒ Discard one of buffer packet & make place for TCP packet
- Priority: ICMP < IGMP < UDP < TCP

→ Header Checksum:


calculate Header checksum & put it here!

Note: Fields of IP which can change in routers :-
- TTL (always)
- FO, MF, Total Length (maybe)
- options (maybe)
- checksum (maybe)
- Header length (maybe)

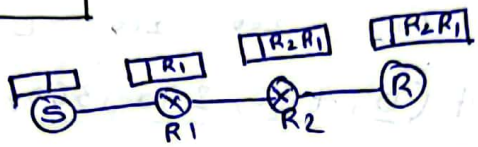
Note:	NID	HID
Valid IP	✓	✓
NID	✓	0's
DBA	✓	1's
LBA	1's	1's
SM	1's	0's
Host within a n/w	0's	✓
I don't have IP	0's	0's
Loopback address	127	✓

→ Type of Service:

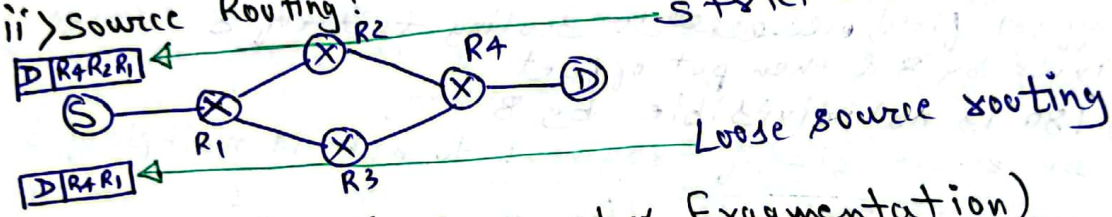
(Priority of Packet) (send my packet via the route where cost is least) (... where throughput is highest)

→ Options:

i) Record route:



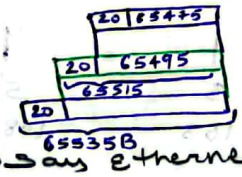
ii) Source Routing:



→ Total length: (and concept of Fragmentation)

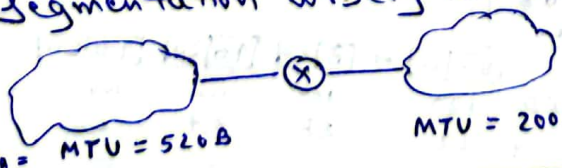
- Indicates total size of datagram
- Max ⇒ 16b = $2^{16}-1$ = 65535

Message AL
Segment TL
Datagram NL
Frame DLL ⇒ as ethernet i.e. MTU = 1500B
IPDU PL



H | 1500B | T

- AL can give any amt. of data to TL. Its responsibility of TL to divide that packet into parts so that each part is at most 65495 B i.e. "Segmentation" done at TL.
- Now, if DLL contains ethernet with restriction of data to be atmost 1500B (say), then NL has to divide datagram into parts so that each part is at most 1500 B. i.e. "Fragmentation" done at NL.
- If you do the segmentation widely, you do not require fragmentation.
- Fragmentation is not done at source, bec. at source we do segmentation widely so that there's no need of fragmentation.



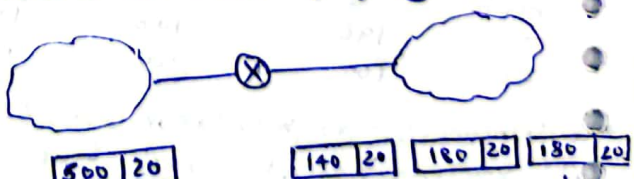
Max.^m data = MTU = 520B

DLL Frame: T | 520B | H

Max.^m # bytes Datagram: 520 - 20 (Header) = 500B

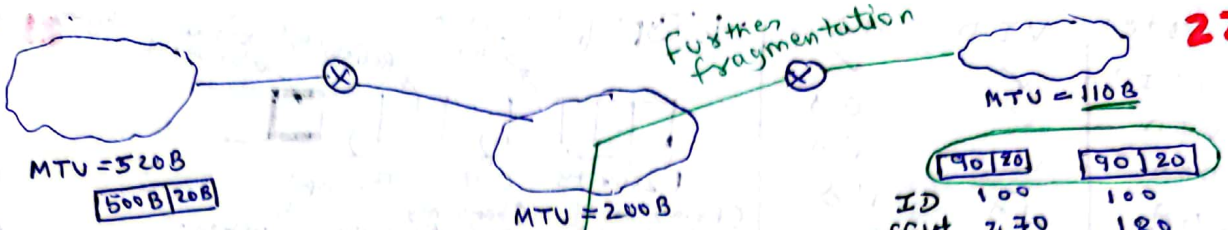
H | 200B | T

200 - 20 (Header) = 180B



Q:) How to identify that all the fragments are coming from same datagram?

A:) Identification Number



Offset: No. of data bytes ahead of this datagram.

140 20	180 20	180 20	
100 (save)	100	100	ID
360	180	0	offset
0	1	1	MF
5	5	20/4 = 5	HLF
160	200	200	TL

⇒ # fragments = 4 (0, 180, 270, 360) ← offsets

worst case fragment offset is 65514, but we've only 13b in fragment offset field, so scaling factor of 8 is used.
i.e. Divide by 8 & then put offset.

Problem:) 180 is not divisible by 8
Soln:) Decrease size of fragment to make it multiple of 8
i.e.

148 20	176 20	176 20		88 20	88 20
100	100	100	ID	100	100
(176+176)/8 = 44	176/8 = 22	0/8 = 0	offset	22+88/8 = 33	22
0	1	1	MF	1	1
5	5	5	HLF	5	5
168	196	196	TL	108	108

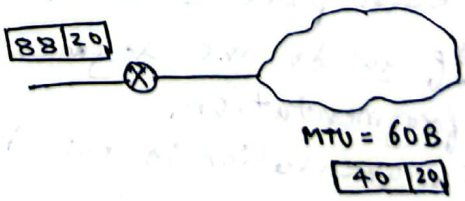
η = Useful bytes / Total bytes = 500 / (500 + 4×20)

"W utilz" = η × bottleneck BW
or
Throughput

i.e. 1 packet → 4 fragments
500B + 20B 500B + 4×20B

Overhead = 3×20B

Q:) Suppose a router only gets fragment II & it has to transmit it through a n/w with MTU 60B



Fragment:

	I	II	III	IV
	20 176	20 88	20 88	20 148
offset	0	22	33	44
MF	1	1	1	0
HLF	5	5	5	5
TL	196	108	108	168
Id	100	100	100	100

88 20 →	8 20	40 20	40 20
ID 100	100	100	100
offset 22	28	27	22
HLF 5	5	5	5
TL 108	28	60	60
MF 1	1	1	1

Q:) Where is fragmentation done?
At routers, not at source bec. source does segmentation.
Q:) Where are segments reassembled?
At destination not at routers bec.
→ All fragments may follow different route
→ All fragments may not meet at a router.
→ There might be a need of further fragmentation.

• Reassembly algorithm:

	20|176	20|88	20|88	20|48
Offset	0	22	33	44
MF	1	1	1	0
HLF	5	5	5	5
TL	196	108	108	168
Id	100	100	100	100

To reassemble these fragments at destination we use two fields offset & MF :-

MF	Off	
1	0	→ 1st Fragment
1	≠0	→ Some middle fragment
0	≠0	→ Last Fragment
0	0	→ Not Fragmented

i) Destination should identify that datagram is fragmented (MF + off.)
ii) Destination should identify all fragments, belonging to the datagram (Id)
iii) Identify 1st fragment (offset=0, MF=1)
iv) Identify subsequent fragments

$$\text{Offset of next fragment} = \text{off} + \frac{TL - HLF \times 4}{8}$$

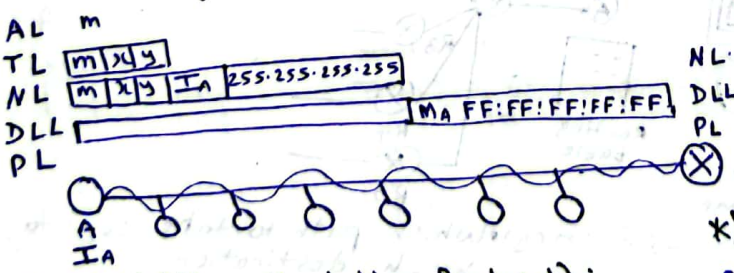
like $= 0 + \frac{196 - 5\times4}{8}$

$= 22$ ✓

v) Repeat (iv) until MF != 0

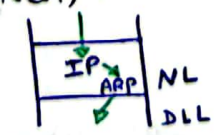
#) Protocols & concepts at NL:

*) Implem.n of Broadcasting:



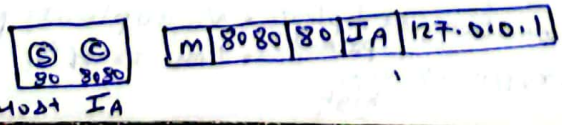
*) ARP (Address Resolution Protocol):

- works at NL
- ARP Request → Broadcast
  ARP Reply → Unicast
- IP ≠ MAC (एकसारखी नहीं)



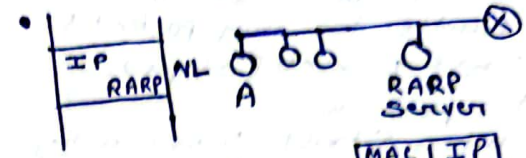
*) Special address 127: (Loopback address)

- For testing self connectivity it is used.
- 127.0.0.1 to 127.255.255.254
- Also used for client-server connectivity check when both present in same host



*) RARP (Reverse ARP):

- MAC से IP निकालना है



- RARP Request

M_A	0.0.0.0	FF:FF:FF:FF

MAC	IP
M_1	I_1
M_2	I_2
...	...

- In every n/w there is one RARP server.
- Disadv.:- Every n/w should have an RARPS static mapping tables (IP ≥ Hosts online)

*) BOOTP (Bootstrap Protocol)

- Runs on AL
- Same as RARP, only difference is that each n/w need not have a BOOTP server. One BOOTP server there & each other n/w have Relay Agent who knows IP of BOOTP server.
- Adv. → Only 1 BOOTP server required
  Dis. → static BOOTP tables.

*) DHCP (Dynamic Host Config." Protocol)

- Same as BOOTP, only diff. is that table is dynamic
- DHCPS Table
- IPs = online Hosts

Static	MAC	IP
	M_1	I_1
	M_2	I_2
	M_3	I_3

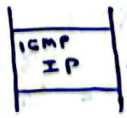
Dynamic	MAC	IP	LeaseTime
	M_A	I_A	10 min.

Note: DHCP is backward compatible

RARP → BOOTP → DHCP

Same Port No.s

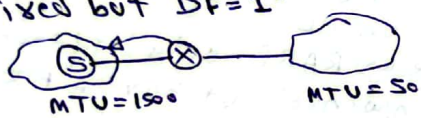
*) ICMP (Internet Control Message Protocol)



- ICMP
  - Error handling or Feedback messaging
    - → TTL exceed
    - → Parameter Problem
    - → Source Quench
    - → Destination Unreachable
  - Request & Reply
    - → Echo req. & rep.
    - → Timestamp req. & reply
    - → N/w mask req. & reply
    - → Router Solicitation & Advertisement

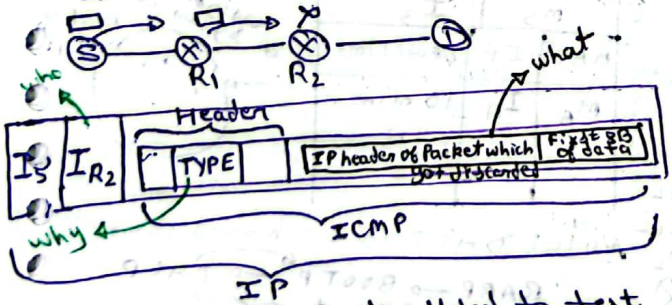
- No ICMP is sent for a lost ICMP packet.
- ICMP is unreliable.
- ICMP is sent for TCP or UDP

- TTL exceed: when TTL of packet is insufficient to reach destination
- Source Quench: when receiver not able to handle too many packets, then source quenches (stops).
- Parameter problem:
  - case-1: when Strict source routing is used but that path is down
  - case-2: when checksum problem there
- Destination unreachable (Host unreachable / Port ")
  - Host unreachable ⇒ Host down
  - Port " ⇒ Port not available
- DHU may also arise if fragmentation is required but DF=1
  - like:

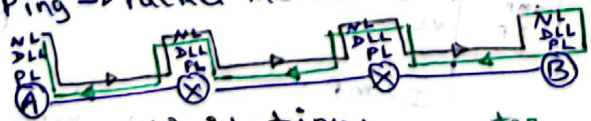


ICMP = DHU (FR, DF=1, MTU=500)

- Source redirect: Warning only, no error message & packet is not discarded actually.
  - here, an ICMP packet is sent saying that a better path is so & so, therefore follow this path next time.
- In error handling or feed back messaging we need to answer 3 things:-
  1) who discarded?
  2) why discarded?
  3) what data discarded?



- Echo Request & Reply: Used to test whether NL of routers & destination are working or not.
- Ping → Packet internet gopher

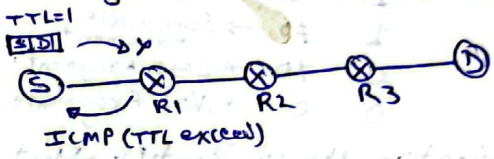


- Router Solicitation:

- Router advertisement: new router sending ICMP to every host saying I am new router, you may use me as default router.
- N/w mask req. & reply:

- Timestamp req. & reply: for synch.n blw hosts in different time zones.
- Trace route: (An app.n of ICMP)

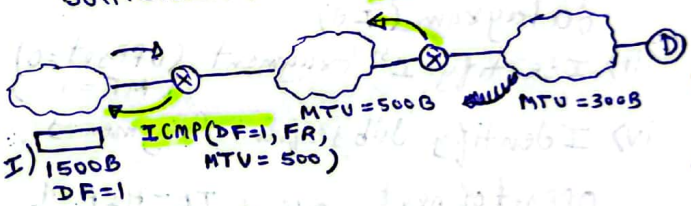
To find source to Destination route, we send a packet with TTL=1 & increase TTL gradually AND instead of actual data, we send UDP packet with dummy port no.s so that when it reaches destination we get an ICMP (Port unreachable)



TTL = 2 & so on...

- PMTDU: (An app.n of ICMP)

Path MTU discovery, used to find bottleneck MTU



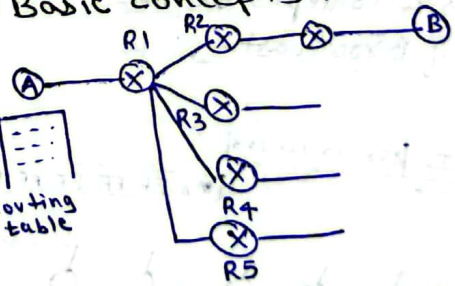
I) 1500B, DF=1

II) 500B, DF=1 & so on..

And when it reaches (D), bec. of dummy port no.s we get ICMP (port unreachable)

Routing:

*) Basic concepts:-



→ Routing: • which path to take, so as to reach destination.
- Process of preparing routing table so that we can do switching better.

→ Switching: • taking incoming packet & sending it to one of the outgoing path.

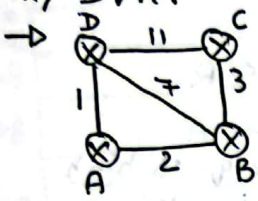
→ Flooding: • send packets to all possible n/w's

→ Flooding	vs	Routing
• No routing		• Routing table required
• Shortest path guaranteed		• No shortest path guaranteed
• Highly reliable		• Not much reliable
• Destination gets duplicate packets		• No duplicate packets
• Traffic is very high		• Low traffic

*) Types of routing algorithm:

- **Static** (manually change करना पड़ता)
- **Dynamic** (changes with change in traffic & topology)
  - **DVR** — Distance Vector Routing
  - **LSR** — Link State Routing

*) DVR:



Routing table

Destination	Distance	Next Hop

A node gets routing table from its neighbors only

Step-1:

RT of A

Des	Dis	N+1
A	0	A
B	2	B
C	∞	-
D	1	D

RT of B

Des	Dis	N+1
A	2	A
B	0	B
C	3	C
D	7	D

RT of C

Des	Dis	N+1
A	∞	-
B	3	B
C	0	C
D	11	D

RT of D

Des	Dis	N+1
A	1	A
B	7	B
C	11	C
D	0	D

Step-2: Exchange of Distance vectors

- **At A:-**

From B: 2, 0, 3, 7 — AB = 2

From D: 1, 7, 11, 0 — AD = 1

New RT of A

Des	Dis	N+1
A	0	A
B	2	B
C	5	B
D	1	D

- **At B:-**

From A: 0, 2, ∞, 1 — AB = 2

From C: ∞, 3, 0, 11 — CB = 3

From D: 1, 7, 11, 0 — DB = 7

New RT of B

Des	Dis	N+1
A	2	A
B	0	B
C	3	C
D	3	A

- **At C:-**

From B: 2, 0, 3, 7 — BC = 3

From D: 1, 7, 11, 0 — DC = 11

New RT of C

Des	Dis	N+1
A	5	B
B	3	B
C	0	C
D	10	B

- **At D:-**

From A: 0, 2, ∞, 1 — DA = 1

From B: 2, 0, 3, 7 — DB = 7

From C: ∞, 3, 0, 11 — DC = 11

New RT of D

Des	Dis	N+1
A	1	A
B	3	A
C	10	B
D	0	D

Repeat this one more time to get final result (Total n-1 times)

RT of A

Des	Dis	N+1
A	0	A
B	2	B
C	5	B
D	1	D

RT of B

Des	Dis	N+1
A	2	A
B	0	B
C	3	C
D	3	A

RT of C

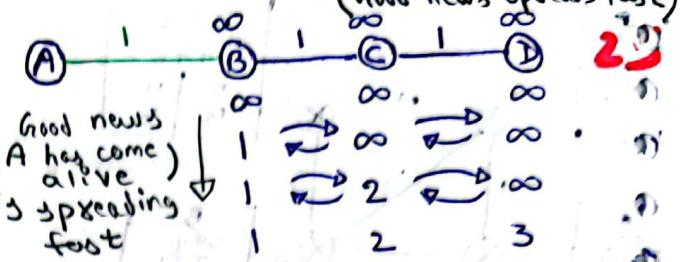
Des	Dis	N+1
A	5	B
B	3	B
C	0	C
D	6	B

RT of D

Des	Dis	N+1
A	1	A
B	3	A
C	6	A
D	0	D

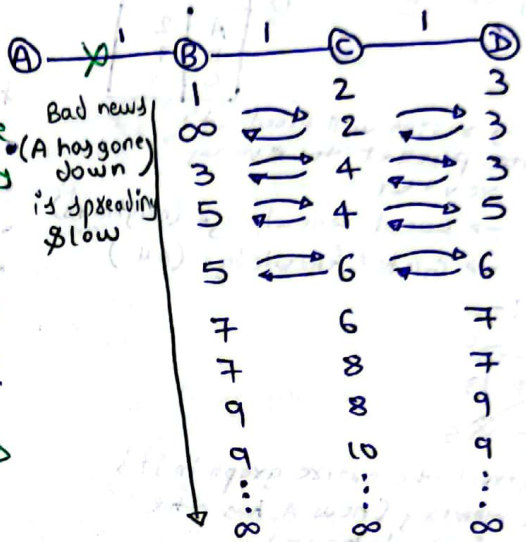
CD & BD are never used ⇒ Remove them

*) Count to infinity (Bad news spreads slow, Good news spreads fast)



Good news (A has come alive) is spreading fast

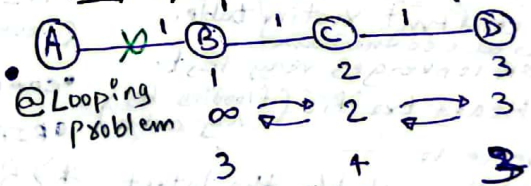
A	B	C	D
∞	∞	∞	∞
1	∞	∞	∞
1	2	∞	∞
1	2	3	



Bad news (A has gone down) is spreading slow

A	B	C	D
1	2	3	
∞	2	3	
3	4	3	
5	4	5	
5	6	6	
7	6	7	
7	8	7	
9	8	9	
9	10	9	
∞	∞	∞	

*) Split horizon (Sol.ⁿ for count to ∞)



- @ Looping problem

A	B	C	D
1	2	3	
∞	2	3	
3	4	3	

RT of C

A	B
4	D

RT of B

A	C
3	-

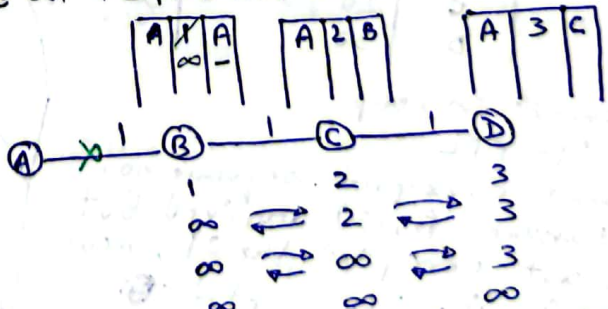
RT of D

A	C
3	C

If B wants to send a packet to A then B → C → D

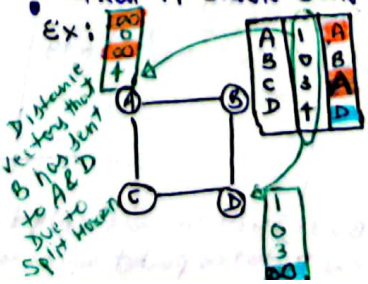
Loop

- @ Sol.ⁿ: Split horizon



A	B	C	D
1	2	3	
∞	2	3	
∞	∞	3	
∞	∞	∞	

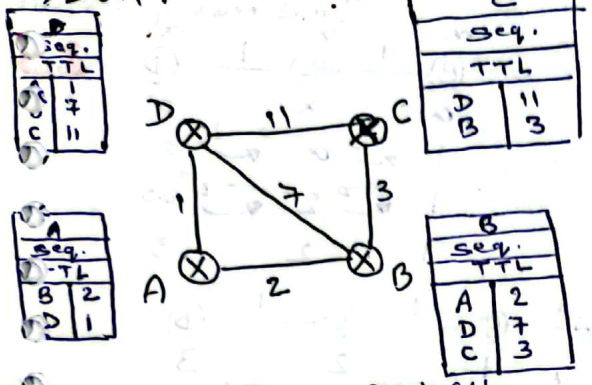
- **Split horizon:-**
  If a node is dependent on some other node, then it should send ∞ to that node (to avoid loops)
- Ex:



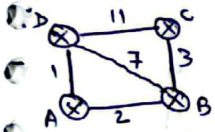
- Disadv. of DVR w/o split horizon
  - † Count to ∞
  - † Slow convergence
  - † Loops

All solved using split horizon

LSR:



- Now, every router will flood it's link state packet to every other router.
- DVR → Local knowledge (neighbors)
  LSR → Global knowledge (AN)
- At A:-



→ A constructs the entire graph in it's local memory (now A has the global database)
→ A will apply SSSP (Dijkstra)
→ and will construct routing table.
→ other nodes also do the same.
- Therefore, LSR converges very fast.
- Problem-1: Heavy traffic (flooding by every router)

Sol.ⁿ - Sequence No.
To distinguish b/w the latest & old packets.
(Difference caused due to delay of arrival of packets bec. of flooding)
i.e.

At A:-

Router	Latest Seq. no.
B	~~10~~ 15
C	20
D	30

(B, 8) rejected
(B, 15) accepted

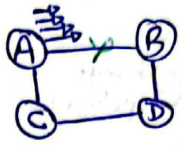
Problem-2: ∞ loop
Sol.ⁿ :- TTL

Problem-3: corrupted sequence no.
Lets assume (B, 15) arrived but 15 got corrupted to 31, now (B, 31) will be accepted.
And now, (B, 16), (B, 17) ... (B, 30) all will be rejected.
Sol.ⁿ :- Lifetime / validity

Router	Latest	Lifetime/validity
B	31	~
C	20	~
D	30	~

particular packet (say B, 31) will be used only till lifetime, after that new updated packet will be used (say B, 21)

Problem-4: Black hole problem



A will take time to know that B is down, and packets sent by A will be lost (as if gone to black hole)

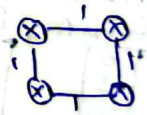
- Transient problems (i.e. not persisting forever)
  + looping
  + Black hole

DVR vs LSR

DVR	LSR
• 1980's	• 1990's
• BW req. is less	• BW req. is high
• Local knowledge	• Global knowledge
• Bellman ford	• Dijkstra
• Traffic is less	• Traffic is very high
• Periodic updates are done	•
• Converges slow	• Converges fast
• Count to ∞	• No count to ∞
• Persistent looping	• Transient looping
• RIP (Routing Info. Protocol)	• OSPF (Open shortest path first)
• computation simple	• Computation complex
• RIP ⇒ UDP	• OSPF ⇒ IP

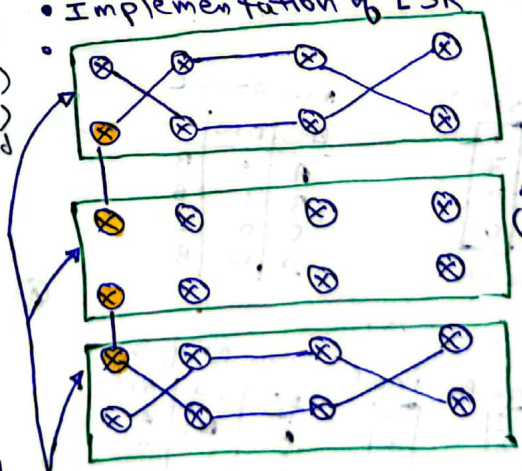
RIP (Routing Inform. protocol)

- implementation of DVR
- Metric is hop count
- ∞ represented by 16



OSPF (Open Shortest Path First)

- Implementation of LSR



Border Routers

Areas / Regions

- Flooding is restricted to only one region at a time.
- Here, hop count is not used as metric, system admin can set the weights of edges.

EIGRP (Enhanced interior gateway routing protocol)

- Combination of RIP & OSPF

Transmission Control Protocol (TCP):

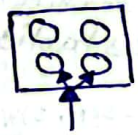
Basics: TCP is
- Transport layer protocol
- End-to-end protocol
- Connection oriented (Reserved resources)
- Byte stream protocol
- Full duplex connection

TCP header:

4B	Source Port (16)								Destination Port (16)
4B	Sequence no. (32)								
4B	Acknowledgement no. (32)								
4B	Header Length (4)	Reserved (6)	URG	ACK	PSH	RST	SYN	FIN	window size (or) adv. window (16)
4B	Checksum (16)								Urgent Pointer (16)
0B to 40B	Options (0 - 40 B)								

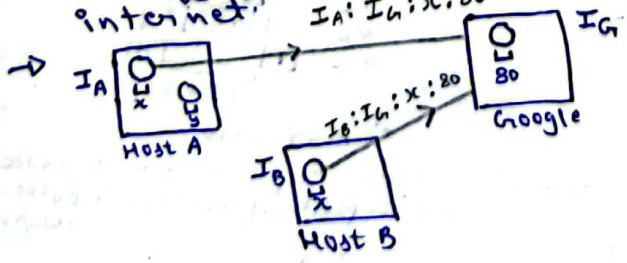
- Min. header length = 20 B
  max. " " = 60 B

Source Port & Destination Port:
- SP or DP → 16b ⇒ 0 to 65535 Port nos
- It enables us to do multiplexing & demultiplexing

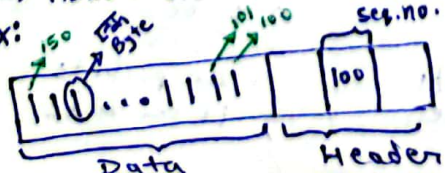


- HTTP → 80 Port no.
  FTP → 20 & 21 → Data & control
  Telnet → 23 (remote login)
  SMTP → 25 DNS → 53
- 0 to 1023 ⇒ Well known Port nos
  1024 to 49151 ⇒ Reserved
  49152 to 65535 ⇒ General Public
- Socket = (IP + Port no.) = 48b
  (32b, 16b)

→ Used to solve problem of uniquely identifying a process on the internet.

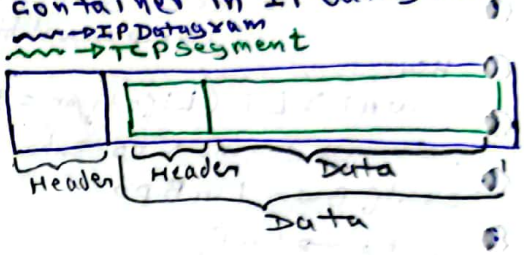


Seq. no.:
- → TCP = Byte stream protocol (TL)
  → IP = Packet " " (NL)
  → HDLC = bit " " (DLL)
- Ex:

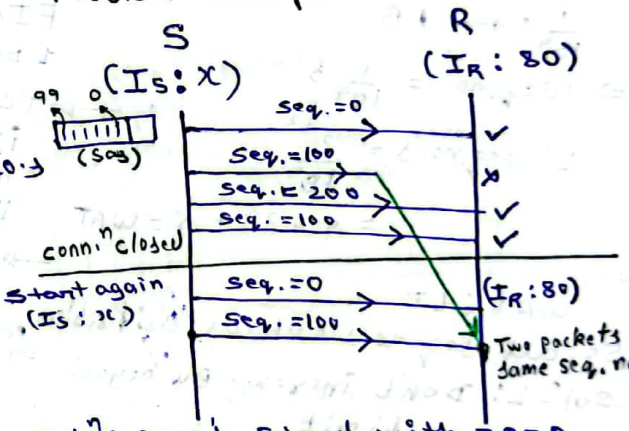


Acknowledgement no.:
- Seq. no. of segment expected next
- As in previous example,
  Ack No. = 151
- Problem: How to calculate last byte's seq. no., so that I can find Ack No.

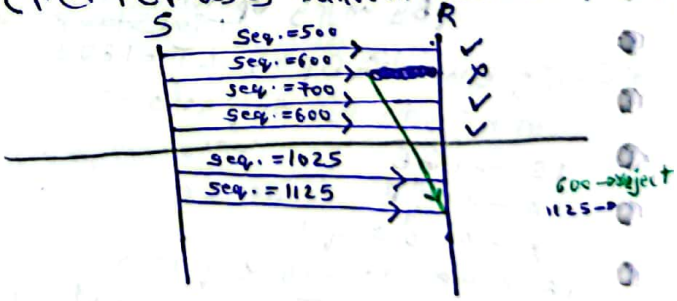
Sol.ⁿ: Calculate from IP header, as every TCP segment is contained in IP datagram



- Problem: Two packets with same seq. no.



Sol.ⁿ: Don't start with zero
Start with a random seq. no.
(i.e. TCP uses random initial seq. nos)



@ Wrap around in seq. no.s:
→ Seq. no. ⇒ 32b ⇒ 2^{32} ⇒ 4G seq nos
now, 4GB + 1B
↳ After this much data, wrap around & use the same seq. no (0)

i.e.

→ For random seq. no.s also this concept holds.
→ wrap around time: Time taken to finish using (or eat up) all seq. nos.

→ Ex: If BW = 1 MBps

2^{32} seq. nos

1 s → 1 MB

10^6 s → 10^6 B

$\frac{1}{10^6}$ s ← 1 B

⇒ 1 seq. no. = $\frac{1}{10^6}$ s

2^{32} seq. nos = $\frac{2^{32}}{10^6}$ s

= 4291.96 s = WAT (wrap around time)

Here,

WAT > LT (Life time = 3 min)

So, all seq. nos not exhausted

→ Ex: If BW = 1 GBps

1 s → 1 GB

1 s → 10^9 B

$\frac{1}{10^9}$ s ← 1 B

⇒ 1 seq. no. = $\frac{1}{10^9}$ s

2^{32} seq. nos = $\frac{2^{32}}{10^9}$

= 4.294 s = WAT

Here,

WAT < LT

So, all seq. nos exhausted: Problem

Soln 1: Don't increase BW beyond a limit

Soln-2: Increase seq. no. bits (By using Options field) TimeStamp

→ Ex: If BW = 1 GBps, LT = 180 s

min.m bits for seq. nos = ?

1 s → 1 GB

1 s → 10^9 B

1 s → 10^9 seq. nos

180 s → 180×10^9 seq. nos

⇒ min.m bits req. = $\lceil \lg_2(180 \times 10^9) \rceil$

= 42 b

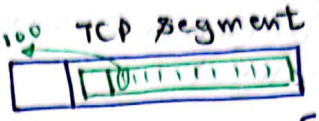
i.e. 32 b (seq. no. field) + 10 b (options field (TimeStamp))

Q: calculate ending seq. no. of segment & Ack no.

If: IP datagram → TL = 1000, HLF = 5

TCP segment → HLF = 5

Seq. no. (starting) = 100



Seq. no. (ending) = 100 + {1000 − 5×4 − 5×4} − 1

= 1059

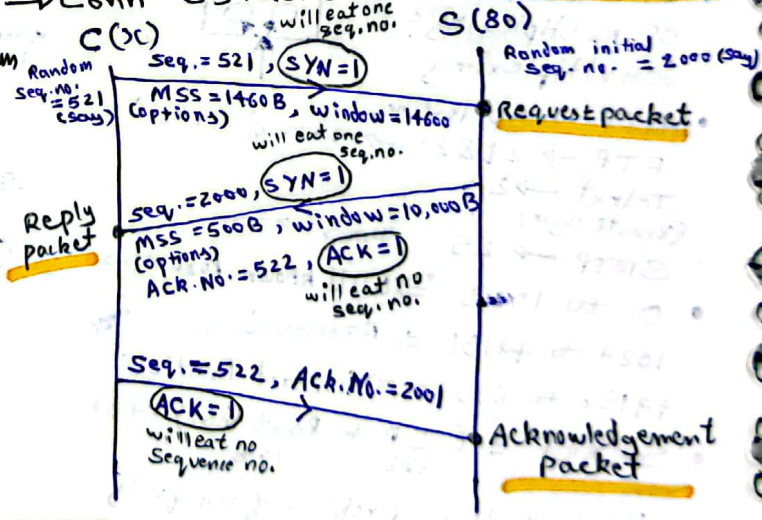
Ack no. = 1059 + 1 = 1060

*) SYN (Synchronization Flag)
ACK (Acknowledgement Flag)
FIN (Finish flag)

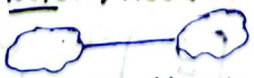
→ SYN = 1 ⇒ 1 seq. no. consumed (eats up)
ACK = 1 ⇒ 0 " " " "
FIN = 1 ⇒ 1 " " " "
1 Data Byte ⇒ 1 " " " "

→ TCP is conn.n oriented protocol having 3 phases:
i) Conn.n establishment
ii) Data transfer
iii) Conn.n termination

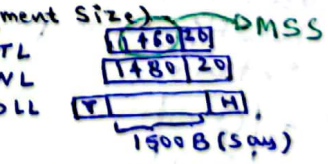
→ Conn.n establishment: (3 way handshake)



Note: • MSS (max.m segment size)

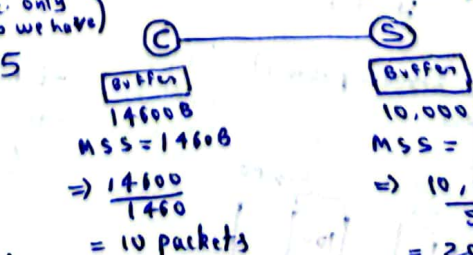


i.e. MTU = HF Data
MSS = Data



⇒ MSS = 1460

• Window size: capacity of host as a recieverer i.e. capacity of reciever's buffer reserved.



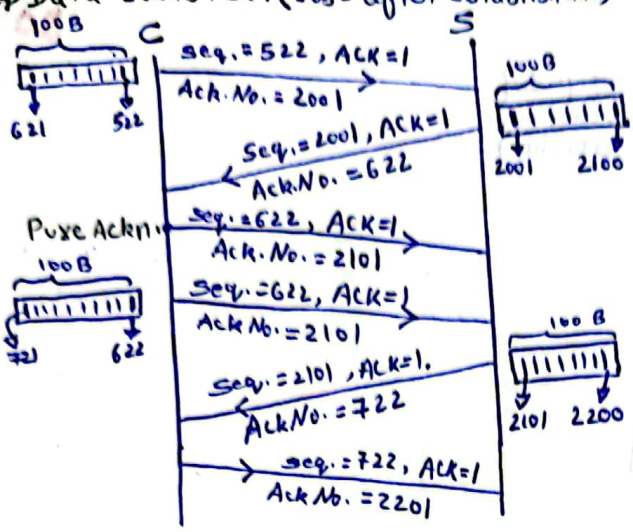
---

*) Header Length:

→ Scaling factor of 4 is used (sec. only 4b we have)

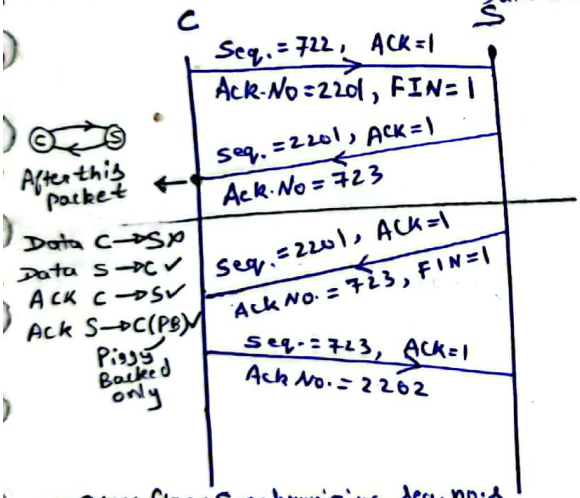
i.e. if HL = 60 ⇒ HLF = $\frac{60}{4}$ = 15

→ Data transfer: (Just after establishment)



SYN	ACK	
1	0	→ I^st segment / request packet
1	1	→ II^nd segment / reply packet
0	1	→ Acknowledgement pkt
0	0	→ NOT Possible

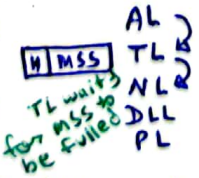
→ Connection termination: (Just after Data transfer)



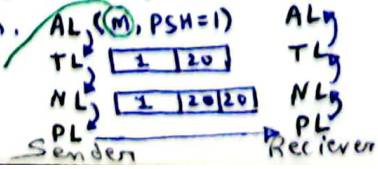
→ SYN Flag: Synchronizing Seq. no.s
ACK Flag: Indicate whether I'm using acknowledgements or not. (i.e. Ack. No. valid or not)
FIN Flag: Request for conn.^n term.^n
(SYN+ACK+FIN) ⇒ Conn.^n mngmt.

*) PSH (Push Flag):

- TL does not send the data immediately after getting it from AL: Buffering

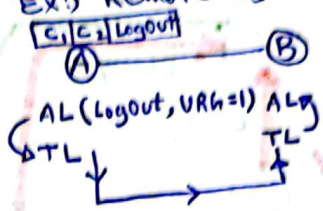


- But, interactive app.^n's (like chat app.^n's) do not want TL to wait for MSS size of data to be sent.
- So, if AL sends PSH=1, Buffering will not happen.



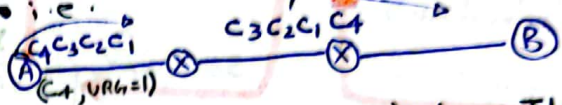
*) URG flag & Urgent pointer:

- If some command is urgent than other commands, Set URG=1.
- Ex:) Remote login (Telnet)



malicious activity detected so we want Logout to be executed before C1, C2

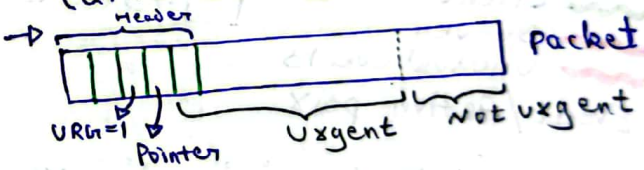
- i.e.



- Problem: Routers don't have TL
  Sol.^n: Set priority to 7, so that in NL (in type of service field) priority becomes 7 & at router, packet with max.^m priority will be sent first.

- Urgent pointer: (16b)

→ It specifies from where to where my packet's data is urgent. (after setting URG=1)



Seq.no. = 1000 (say)
URG = 1
Urg. Ptr. = 100
Seq.no + Urg.ptr. = Seq.no of last urgent byte = 1100

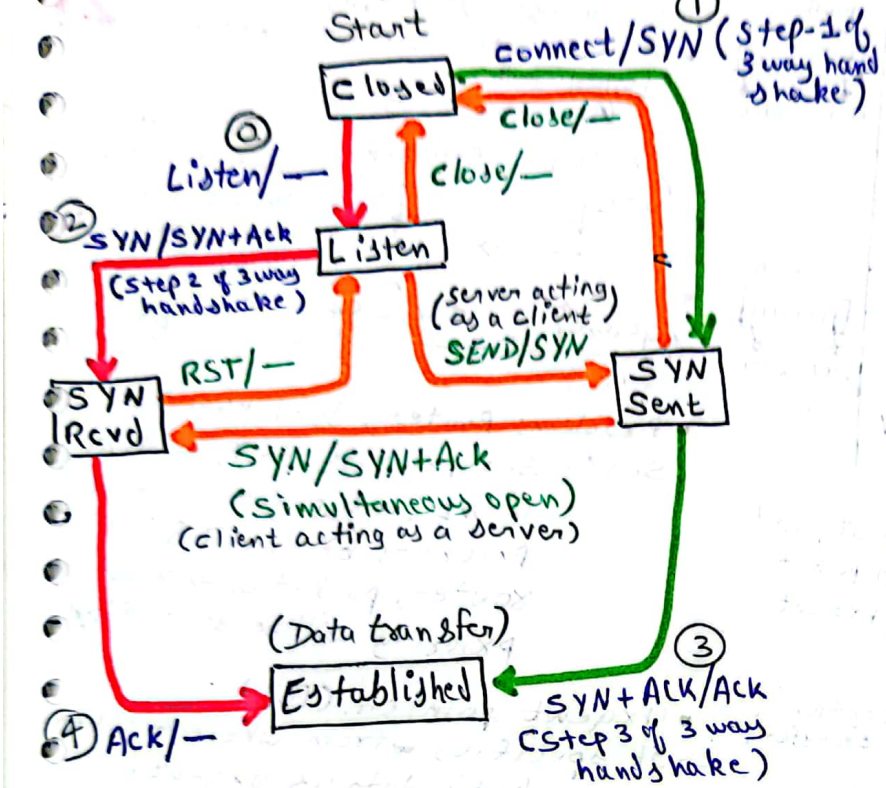
1000 to 1100 → Urgent data = 101 B

if Urg.Ptr. = 100, then # Bytes Urgent = 101
i.e. # Bytes urgent = Urg. Ptr. + 1

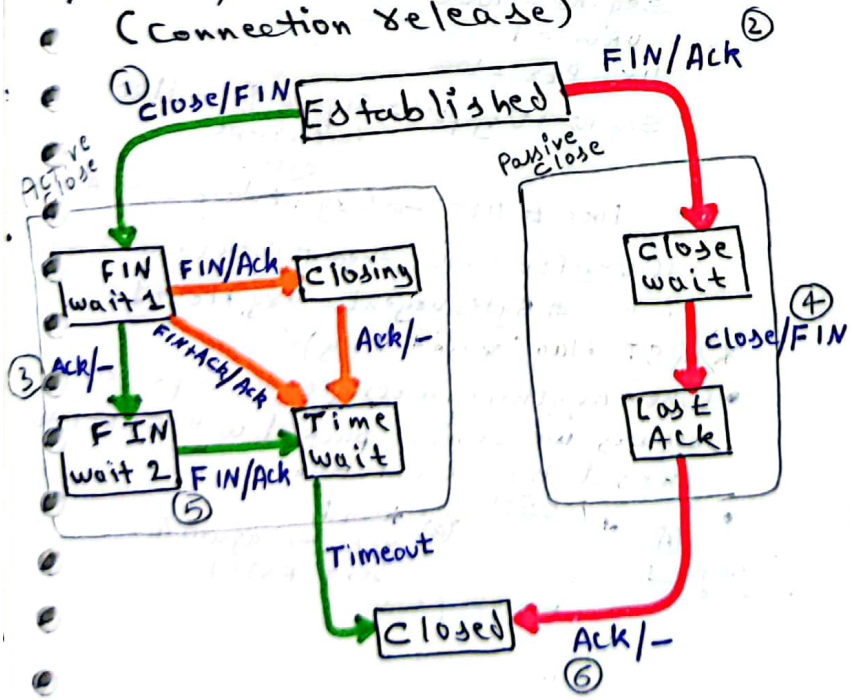
*) RST Flag (Reset flag):

- When anything wrong or unexpected occurs we send a packet with RST=1 to reset the connection.
-

*) TCP State transition diagram
(Connection establishment)

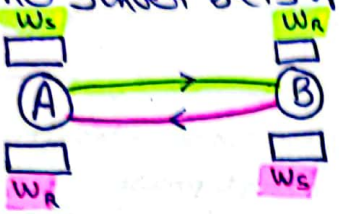


*) TCP state transition diagram:
(Connection release)

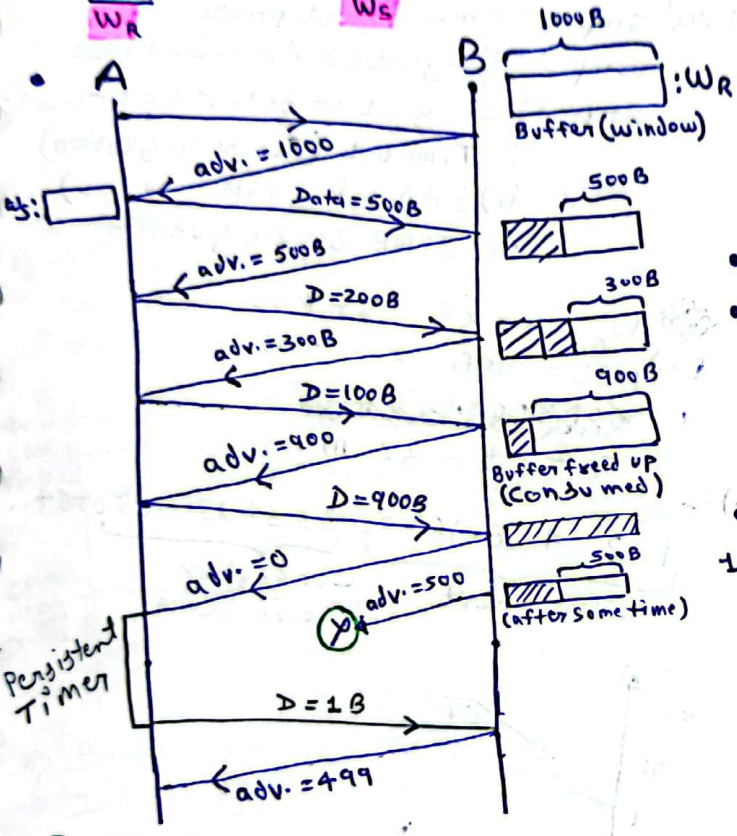


*) Window size (or) Advertisement window : (16b)

- It specifies that this much amt of buffer I'm having free.
- Acc. to adv. window of reciever, the sender sets up its W_S size.



i.e. flow control



Problem:
A thinks B is full
B thinks A has no data to send
⇒ Deadlock

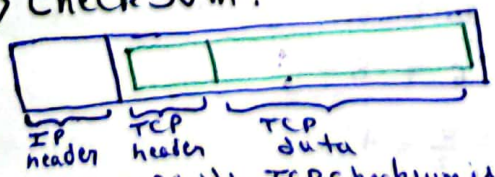
Sol.ⁿ: Persistent Timer
Send 1B data after this time

- Problem: window size ⇒ 16b
  ⇒ max.ᵐ buffer size = 65535B ≅ 64KB

Sol.ⁿ: Use bits from options

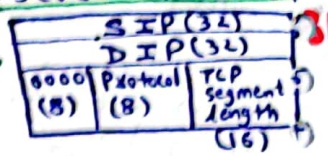
Ex:) we want to achieve 1GB buffer size
16b window size + 14b options
⇒ 30b
⇒ 1GB buffer

*) Checksum:



On these 3 fields, TCP checksum is calculated.

- IP header might change, so we take only pseudo IP header



- i.e.
  TCP CS = pIPh + TCPh + TCPdata

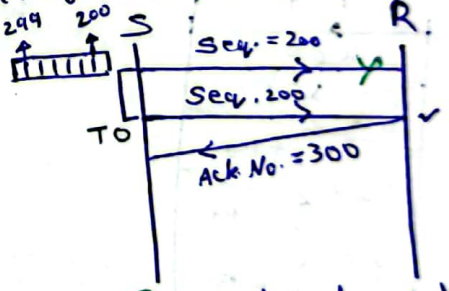
*) Options:

1) Timestamp → used when WAT < LT
2) window size (buffer size) extension
3) Parameter negotiation
4) Padding → To make header length a multiple of 4

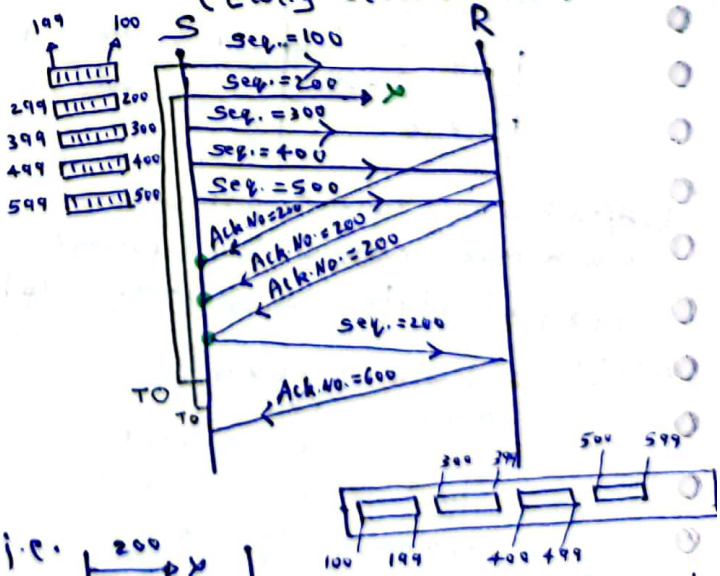
*) Retransmissions in TCP:

- TCP uses SR (75%) + GBN (25%)
- i.e.,
  → $W_S = W_R$
  → Out of order packets will be accepted
  → Acknowledgements used are cumulative ack.
- There are 2 types of ReTransmissions

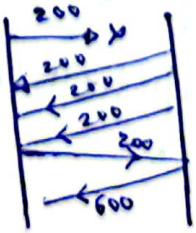
1) RT after timeout:-



2) RT after (3) duplicate acknowledgements (Early retransmission)



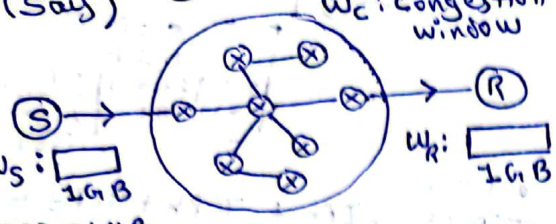
i.e.



Note: • Timeout indicates the congestion is severe.
• 3 dup ack. indicates the congestion is mild

TCP congestion control:

(Say)

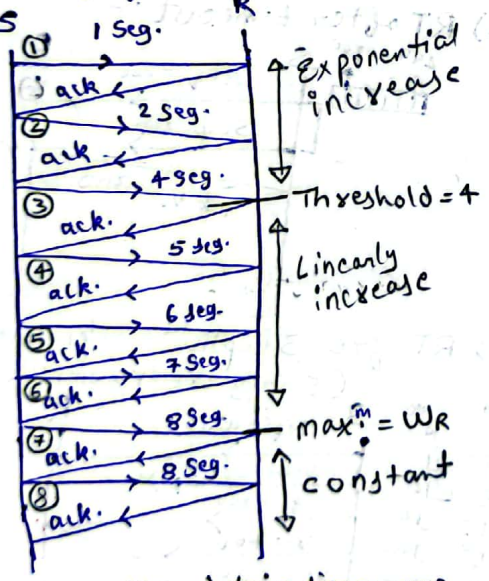


MSS = 1MB

$\frac{1GB}{1MB}$ = 1024 packets at a time it can send

Now, S should send min(W_c, W_R) sized packets only.

Ex: adv. win = 8KB
MSS = 1KB
W_R = 8KB or 8 segments
W_c = 1 segment
W_S = min (W_R, W_c) = 1
(take min^m for I^st time)
Threshold = $\frac{W_R}{2} = \frac{8}{2} = 4$
(after 4 increase linearly)

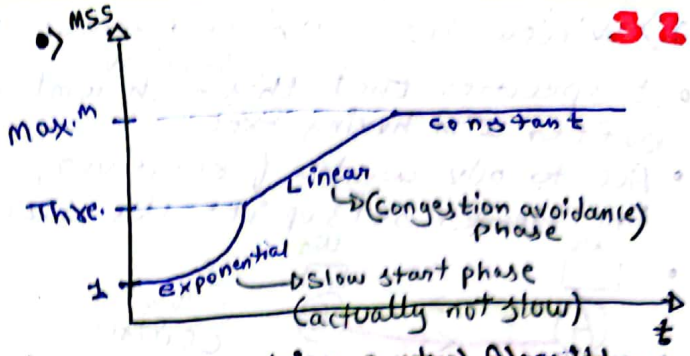


Q:) After how many round trip times, we reach max^m sender window capacity?

1 | 2 | 4 | 5 | 6 | 7 | 8 ⇒ 6 RTTs

Q:) If adv. win. = 16KB
MSS = 1KB
W_R = 16 segments
W_c = 1 segment
Threshold = $\frac{16}{2} = 8$

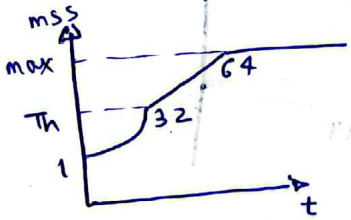
1 | 2 | 4 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16
⇒ 11 RTTs



TCP congestion control Algorithm:
Step-1: Slow start phase
Step-2: Congestion Avoidance Phase
Step-3: Congestion detection:-
i) Time out (Severe congestion)
ii) 3 dup. ack. (mild ,,)
~~iii) ICMP source quench~~

Ex:) W_R = 64KB = 64 mss
MSS = 1KB = 1 mss
Th = $\frac{64}{2}$ = 32 mss

1 2 4 8 16 32 33 34 35 ... 63 64 64 ...
Slow Start | congestion avoidance



Ex:) Congestion detection:-(TO = Time Out)
1 2 4 8 16 32 33 34 (TO)

New Th = $\frac{1}{2}$ W_c = $\frac{34}{2}$ = 17 mss

Start from Slow Start phase when TO has occured

34 | 1 2 4 8 16 17 18 19 20 (3 Dup. Ack.)
TO

New Th = $\frac{1}{2}$ W_c = $\frac{20}{2}$ = 10 mss

Start from cond.^n avoidance phase when 3 Dup. Ack. has occured.

20 (3DA) 10 11 12 (TO)
New Th = $\frac{1}{2} \times 12 = 6$

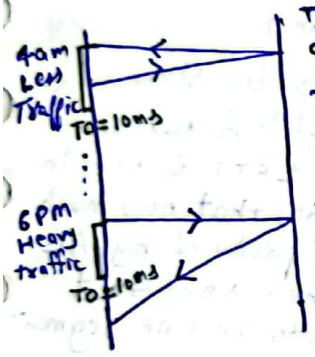
12 (TO) 1 2 4 6 7 8 (3DA)
New Th. = $\frac{8}{2} = 4$

8 (3DA) 4 5 6 7 ... 64 64 64 ...

TCP timer management:

i) Time-wait timer (for late packets)
ii) Keep alive timer (close idle conn^{n}s)
iii) Persistent timer (adv. win = 0, Probe)
iv) Acknowledgement timer (cumulative ack + Piggy backing)
v) Time-out timer

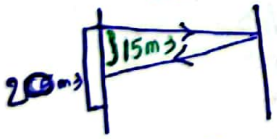
- Time-wait timer = 2 × LifeTime (LT), so that delayed packet may arrive safely.
- Keep-alive = Sending many probes to clients to know who are idle conn^{n}s so that I could close them & revoke resources.
- Time-out timer: If I don't get an ack., then after how much time do I retransmit.



Therefore, time-out timer can't be static in TCP

Therefore, Two Algorithms proposed (to compute TO timer):
1) Basic Algo
2) Jacobson's Algo → Dynamic TO timer

- Basic algorithm:-
  - Initial RTT = 10 ms (Guess)
  - TO = 2 RTT = 20 ms



Actual RTT = 15 ms (Luckily)

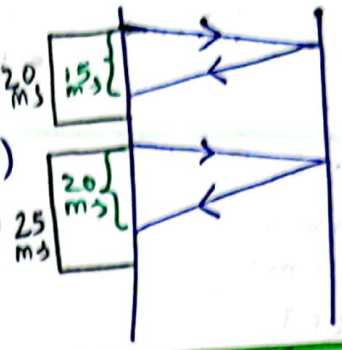
- Next RTT = $\alpha \cdot IRTT + (1-\alpha)\, ARTT$

α → Smoothening factor; $0 \le \alpha \le 1$
$\alpha = 1$ ⇒ Static TOT
$\alpha = 0$ ⇒ Purely Dynamic TOT

- let, $\alpha = 0.5$ (Guess)

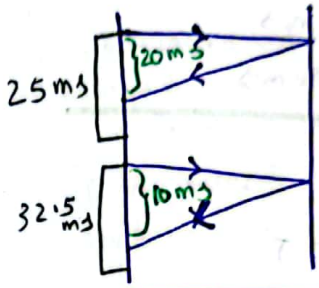
NRTT = 0.5 × 10 + 0.5 × 15
= 12.5 ms

- For next packet:-
  IRTT = 12.5 ms
  TO = 2 × RTT = 25 ms



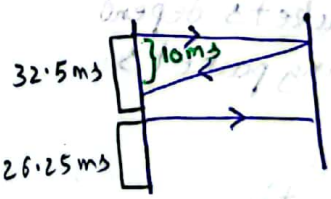
ARTT = 20 ms
NRTT = α IRTT + (1−α) ARTT
= 16.25 ms

- For next packed:-
  IRTT = 16.25 ms
  TO = 2 × RTT = 32.5 ms



ARTT = 10 ms
NRTT = $\alpha \cdot$ IRTT + (1−α) ARTT
= 13.125 ms

- For next packet:-
  IRTT = 13.125 ms
  TO = 2 × RTT = 2 × 13.125 ms = 26.25 ms



& so on...

- Disadv.: Always TO = (2) × RTT (No logic behind it) — Improved by Jacobson

- Jacobson's algorithm:-
  - IRTT = 10 ms (Guess)
  - ID = 5 ms (Initial deviation Guess)
  - TO = 4 × D + RTT
    = 4 × 5 + 10 = 30 ms
  - ARTT = 20 ms (Guess)
  - AD = 10 ms = |IRTT − ARTT| (Actual Deviation)
  - NRTT = $\alpha \cdot$ IRTT + (1−α) · ARTT
    let, $\alpha = 0.5$
    NRTT = 15 ms
  - ND = $\alpha \cdot$ ID + (1−α) · AD
    (next deviation) = 7.5 ms

- For next packet
  IRTT = 15 ms
  ID = 7.5 ms
  TO = 4 × D + RTT = 45 ms
  ARTT = 30 ms (Guess)
  AD = 15 ms

NRTT = 0.5×15 + 0.5 × 30
= 22.5 ms
ND = 11.25 ms

For next packet :-
IRTT = 22.5 ms
ID = 11.25 ms
TO = 4×D + RTT
= 67.5 ms
ARTT = 10 ms
AD = 12.5 ms

NRTT = 16.25 ms
ND = 11.875 ms

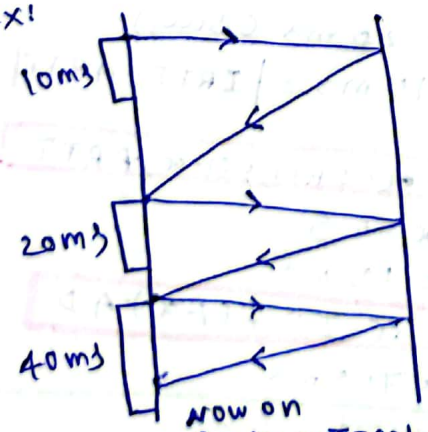
IRTT = 16.25
ID = 11.875
TO = 4×D + RTT
= 63.75 ms

& so on...

Disadv.: Next packets depend upon all preceeding packets. (when TO occurs)
Improved by Karn

Karn's modification:
Whenever a time-out occurs, for the next packet's TO timer don't apply Basic or Jacobsons, Just keep on doubling the TO timer value.

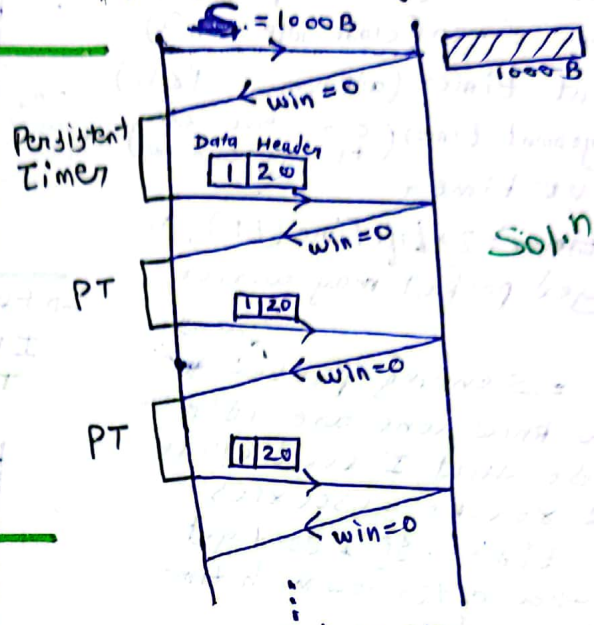
Ex:



Now on Basic or Jacobson's Continued

*) Silly window syndrome: 34

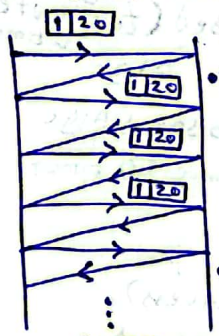
•) Problem-1: If receiver is too slow



and so on (window size = 0 for long time)

Sol.n: Use Reset flag

•) Problem-2: If Sender is too slow

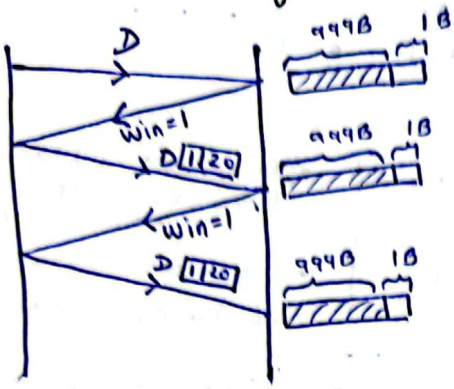


Sol.n: Nagle's Algorithm
- wait for 1 RTT & see in this time, that how much data is produced by AL and then send that much data in one segment.
- If 1 mss is filled before 1 RTT or less, send 1 mss immediately

Ex:) AL Speed = 1 B/ms
RTT = 100 ms
mss = 200 B
wait for 1 RTT
⇒ 100 ms
⇒ 100 B इकट्ठा होगा
100 B < 1 mss, so send 100 B each time

Ex:) ...
mss = 50 B
wait for 1 RTT
⇒ 100 ms
⇒ 100 B
but 100 B > 1 mss
i.e. 100 B > 50 B
so, Send 50 B each time.

*) Problem-3: Reciever consumes 1B only after every trip



Soln: Clark's solution

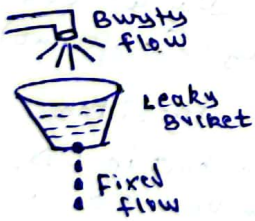
Receiver should wait for at least 1/2 of the buffer (to be freed up) or 1 MSS of the buffer (to be freed up), then only advertise it's window size.

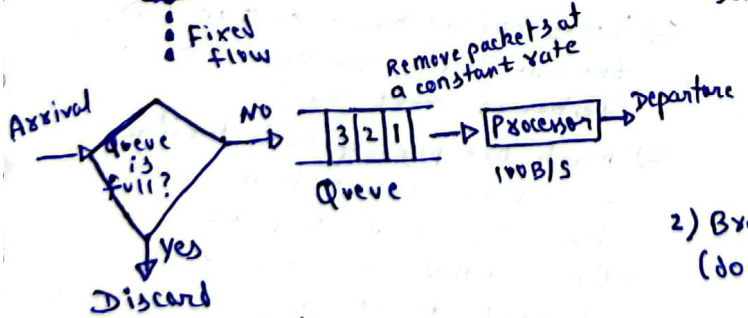
*) Traffic Shaping:

→ A method of congestion control where we shape the traffic (rate at which packets are sent) before it enters the n/w.

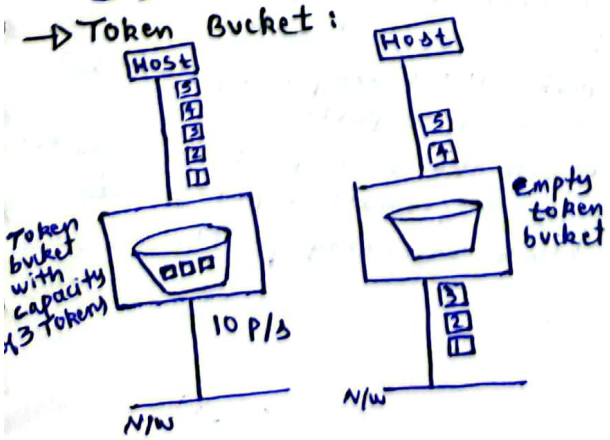
→ During conn.n establishment, the sender & the carrier negotiate a traffic pattern (shaping).

→ Leaky Bucket:-





→ Token Bucket:



Let, the capacity of token bucket be 'c' tokens & the tokens enter the bucket at the rate of 'r' tokens per sec.

Now, The max.m no. of packets that can enter the n/w during time interval 't' is:-

max.m no. of packets = $c + r \times t$

max.m avg. rate = $\frac{c + r \times t}{t}$ packets/sec

Q:) consider a token bucket of capacity 250 KB and the tokens arrive at a rate of 2 MB/s. If the max.m o/p rate is 25 MB/s, what is the burst time?

$$\frac{250\,KB + 2\,MB/s \times t}{t} = 25\,MB/s$$

$$250 + 2 \times 10^3 t = 25 \times 10^3 t$$

$$250 = 23 \times 10^3 t$$

$$t = \frac{250}{23 \times 10^3}\ s$$

$$= 10.86\ ms$$

#) UDP (User Datagram Protocol):

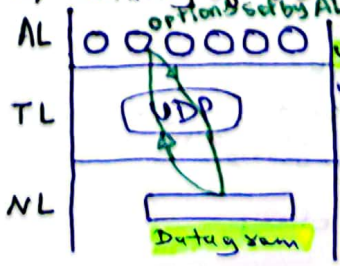
*) Need for UDP:

1) for app.ns which needs 1 req./1 reply only:
   like:- i) DNS
   ii) BOOTP / DHCP
   iii) NTP (N/w time Protocol) (Find time)
   iv) NNP (N/w News Protocol)
   v) Quote of the day Protocol

2) Broadcasting/Multicasting app.ns (don't establish conn.n & reserve resources, just 1 req/1 reply)

3) App.ns where speed is priority, not reliability
   like:- i) Online games
   ii) Multimedia app.ns

*) working of UDP:



UDP is called user datagram protocol bec. it's almost like Null protocol, i.e. giving datagram to user directly.

1) Trace Route
2) Record Route
3) Time Stamp

ICMP error:
1) Destination unreachable
2) Source quench

○ ⇒ An app.n

*) UDP header:

a)

SP (16)	DP (16)
Length (16)	Checksum (16)

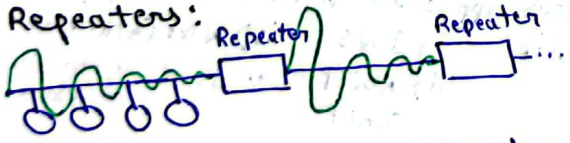
SP = Sender Port
DP = Destination Port
Length = UDP header + UDP data [8B fixed]
Checksum = pIPH+UDPH+UDPD

b) In UDP :-
- → No conn.n establishment/term.n
- → No acknowledgements
- → No sequence no.s
- → No window size (flow ctrl)
- → No flags
- → No options

#) H/w & various devices used in networking :-

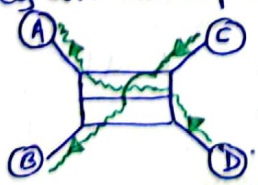
1) Cables:
- → Ethernet cables :-
  - 10 BaseT (10 Mbps, No multiplex, 100 m)
  - 10 Base 2 (" " 200 m)
  - 10 Base 5 (" " 500 m)
  - 100 BaseT (100 Mbps " 100 m)
- → operates at PL
- → Attenuation
- → collisions there (collision domain = n)

2) Repeaters:



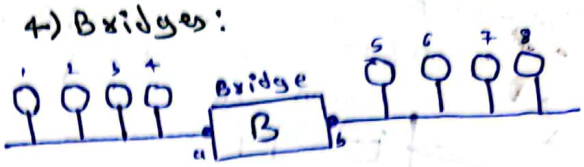
- → used to connect two LAN segment
- → works at PL
- → collisions possible (coll. domain = n)
- → Range of LAN is increased.
- → Un-necessarily increase traffic.

3) Hubs:
- → They are multiport repeaters



- → Traffic is very high
- → They have PL only
- → collisions possible (coll. domain = n)
- → Flooding
- → They are very cheap

4) Bridges:



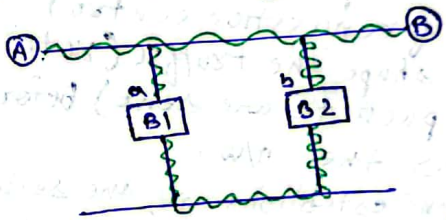
- → They have both PL & DLL

→ mapping table

MAC	Port
1	a
2	a
3	a
4	a
5	b
6	b
7	b
8	b

- If this table is static (N/w admin types manually)
- If this table dynamic (it learns by itself)

- → Here msg can be filtered (bec. u can block a mac)
- → It's a forwarding device
- → Flooding (when destination not in mapping tables)
- → Store & forward
  - ⇒ No collisions inside
  - ⇒ Collision domain is reduced
- → Problem: Message in a loop



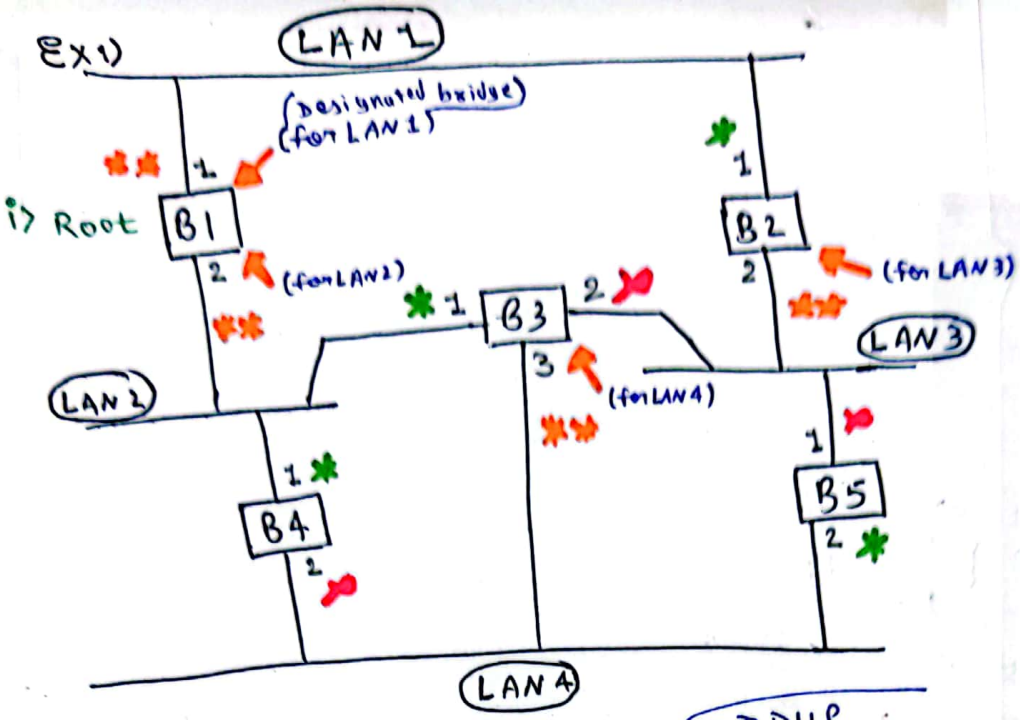
Sol.n: Spanning tree algorithm

- → Theoritically bridges should be able to connect LANs of different types, practically we don't do it (MTU's may be different)

→ Spanning tree algorithm:

1) Every bridge has a built in ID, the one with smallest ID is taken as root bridge
2) Mark one port of each bridge, which is closest to the root bridge, as root port.
3) Every LAN chooses a bridge, closest to it, as a designated bridge for that LAN. Make the corresponding port as designated port.
4) Mark the root port & designated port as forwarding ports and block remaining ports.

Ex1)



i) Root

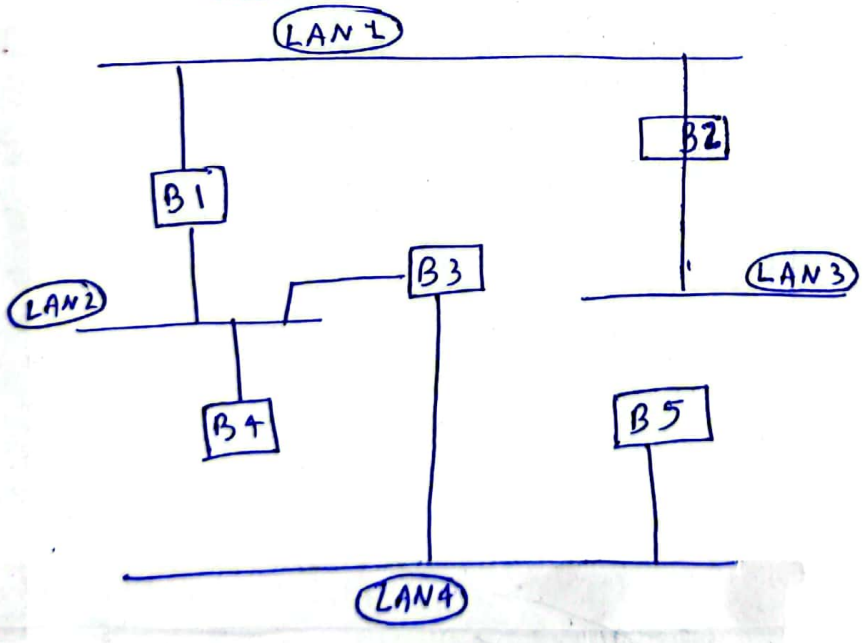
ii) * (root port)

iii) , ** (Designated port)

(Designated bridge)

BDUP
Bridge
Data Unit
Protocol

Now, remove ports without * or **



5) Switch:
- → Sh has both PL & DLL
- → No collisions (collision = 0)
- → Traffic is reduced
- → Its costly

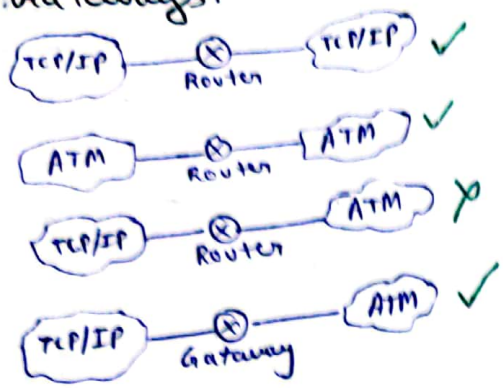
6) Routers:

- Forwarding
- Filtering
- Flooding
- Routing
- No collision
- Broadcast domain ↓
- costly

Note:
37 wires+ Hub+ Repeater + Bridge + switches
LAN Components
Any of these devices should not decrease the broadcast domain

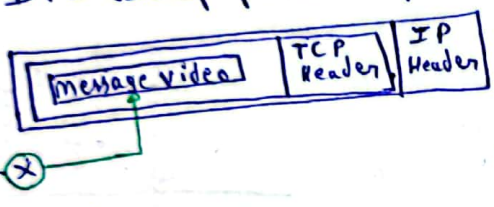
Device	Broadcast Domain	Collision Domain
Repeater	Same	Same
Hub	Same	Same
Bridge	Same	Reduces
Switch	Same	Reduces (to 0)
Router	Reduces	Reduces
Gateway	Reduces	Reduces

7). Gateways:

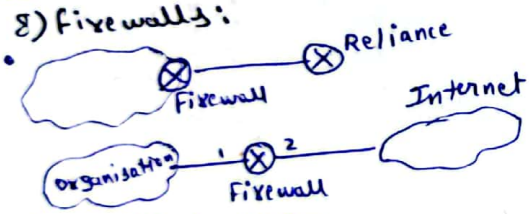


- Protocol converter
- used for proxy
- used as NAT (N/w address transl.n)
-

- used for firewall
- DPI (Deep packet inspection)



8) Firewalls:



- Firewall types:-
  1) Layer 3 FW (Packet filtering FW)
  2) Layer 4 FW
  3) Layer 5 FW (Proxy firewall)
- Layer 3 FW: (PL, DLL, NL)
  - i → SIP & DIP protocols used to block a host completely
  - ii → You can also block a protocol completely (TCP, UDP, ICMP, IGMP)
  - iii → You can also block a protocol from a particular host.
- Layer 4 FW: (PL, DLL, NL, TL)
  - → Layer 3 capabilities +
  - iv → Port No. to block a particular service completely
  - v → A particular service from a particular host can also be blocked.
- Layer 5 FW: (Most powerful FW)
  - → Layer 4 capabilities +
  - vi → Using app^n layer data (like Username & Password)

  You can block a particular user on a particular host.

Application protocols:

- DNS: Domain name Service (Name → IP)
- HTTP: Hypertext Transfer Protocol (People see pages through this)
- FTP: File transfer protocol (web page at home → server)
- SMTP: Simple mail transfer protocol (email direction)
- POP: Post-Office Protocol (Retrieving email)

DNS (Domain name → IP)

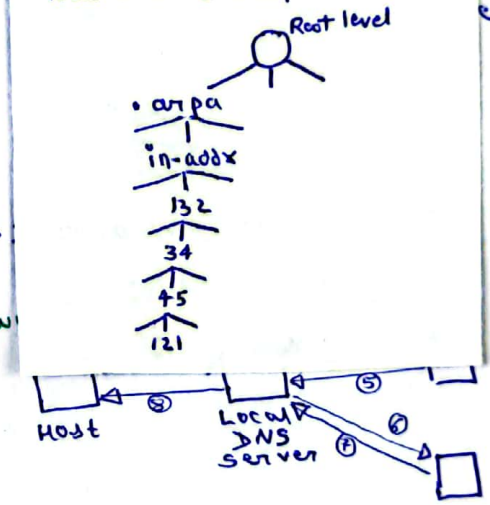
- → Generic domain:-
  - .com, .edu, .mil, .org, .net
- → Country domain:-
  - .in, .us, .uk
- → Inverse domain:- (IP → Domain name)

Ex: Inverse Domain

132.34.45.121

→ 121.45.34.132.in-addr.arpa

Now use nslookup command



→ Root Server

DNS → Top level Domain server

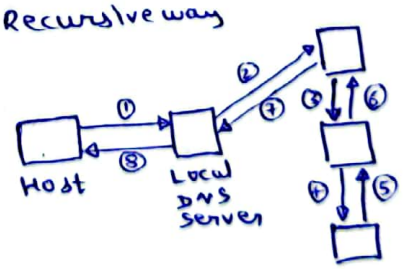
→ Authoritative server

Root level server

Top level DNS server (.in)

Authoritative server (iisc.in)

→ Recursive way

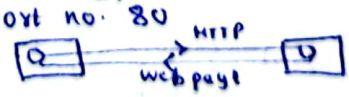


→ DNS uses UDP at TL

→ DNS user Port no. 53

HTTP (Hypertext Transfer Protocol)

→ Port no. 80



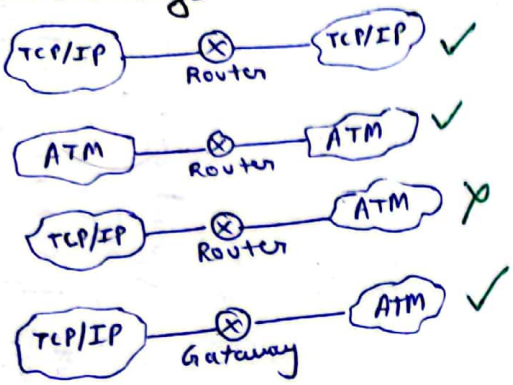
→ Requires reliability

→ Uses TCP at TL

→ Its an Inband protocol (No distinction b/w command & data)

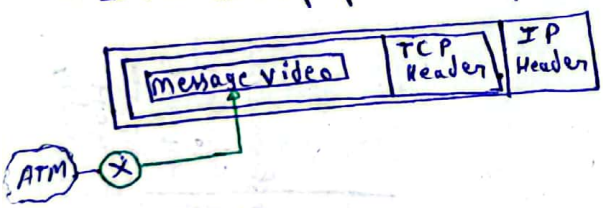
→ Its Stateless protocol (Doesn't remember anything about client)

7). Gateways:

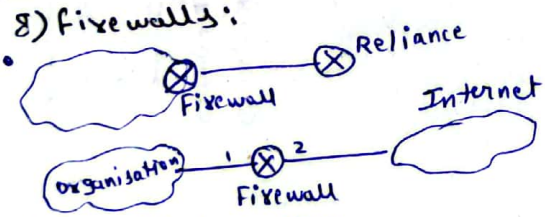


- Protocol converter
- used for proxy
- Used as NAT (N/w address transl.")
- 

- Used for firewall
- DPI (Deep packet inspection)



8) Firewalls:



- Firewall types:-
  1) Layer 3 FW (Packet filtering FW)
  2) Layer 4 FW
  3) Layer 5 FW (Proxy firewall)
- Layer 3 FW: (PL, DLL, NL)
  - i → SIP & DIP protocols used to block a host completely
  - ii → You can also block a protocol completely (TCP, UDP, ICMP, IGMP)
  - iii → You can also block a protocol from a particular host.
- Layer 4 FW: (PL, DLL, NL, TL)
  - → Layer 3 capabilities +
  - iv → Port No. to block a particular service completely
  - v → A particular service from a particular host can also be blocked.
- Layer 5 FW: (Most powerful FW)
  - → Layer 4 capabilities +
  - vi → Using app. layer data (like Username & Password)

  You can block a particular user on a particular host.

#) Application protocols:

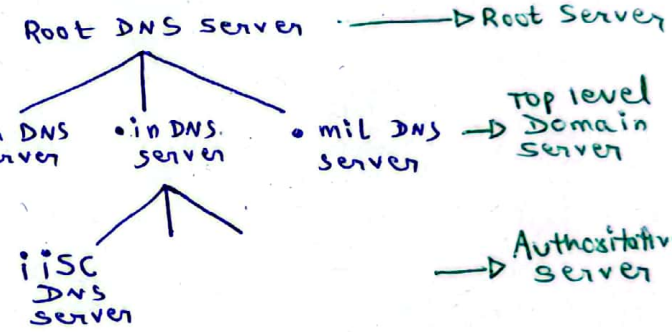
38

*) DN
HT
FT
SM
P

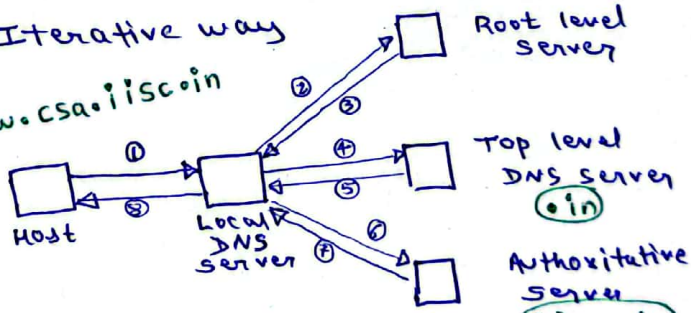
(Name → IP)
people see pages through this)
page → server)
mail direction)
eving email)

*) D
→ Ge
→ C
→ I

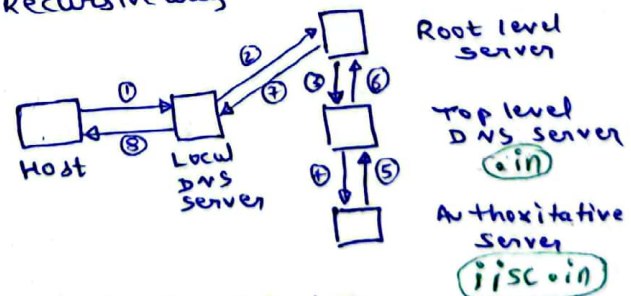
Domain name)



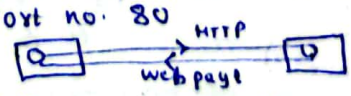
→ Iterative way



→ Recursive way



→ DNS uses UDP at TL
→ DNS uses Port no. 53

*) HTTP (Hypertext Transfer Protocol)
→ Port no. 80
→ 

→ Requires reliability
→ Uses TCP at TL
→ Its an Inband protocol (No distinction b/w command & data)
→ Its stateless protocol (Doesn't remember anything about client)

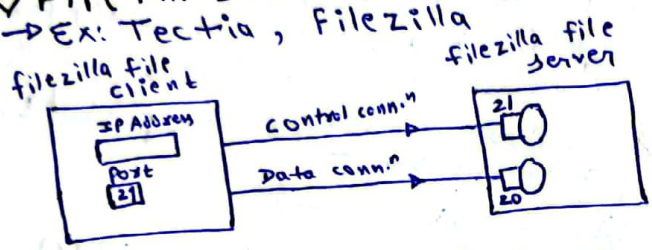
→ HTTP 1.0 , non-persistent conn.n (Server need not hold the conn.n for long time)
→ HTTP 1.1 , persistent conn.n (Server holds the conn.n for a long time) (BW high)

→ Methods supported by HTTP:-
1. Head - contains metadata of webpage
2. Get - Get the web page
3. Post - Fill form
4. Put - Upload object
5. Delete - Delete object
6. Trace - Trace from where data is coming
7. Options - Check if server is providing 1 to 6 facilities.
8. Connect - HTTPS:// security (Authenticated)

To Remember: connect head, put options, get post, delete trace

FTP (File transfer protocol) (Active FTP)

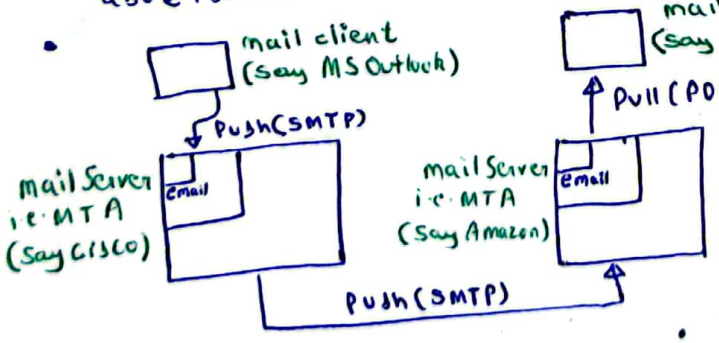
→ Ex: Tectia, Filezilla



- Control conn.n: Persistent (as long as connected)
- Data conn.n: non-Persistent (Lasts only till file is transferred)
- Data conn.n is out of band conn.n, i.e. commands & data are in different conn.ns
- Requires reliability (uses TCP at TL)
- Its stateful protocol (Log every activity)

SMTP & POP: [Simple mail transfer Protocol / Post office protocol]

- G-mail is web based mail
- IBM was not, it was as follows:-



- MIME (Multipurpose internet mail extension) → { Non-text data → Text data; Text data → Non-text data }
- SMTP & POP
  → Requires reliability (uses TCP at TL)
  → In band connectivity
  → are stateful

Note: gn CRC
- x is a bad generator
- If x+1 is the generator, then CRC can detect odd no. of errors

Note: POP vs IMAP (Internet Access Protocol messaging)
- POP downloads the email from a server to a single computer, then deletes the email from the server
- IMAP stores the message on a server and synchronizes the message across multiple devices.

Note: Router can forward or block data by observing IP destination address of packet.

Note: In Time wait timer
→ It stops new conn.ns during this time
→ It stops malicious requests" " ".

Note: Efficiency or Link or line or channel or sender utilization are same.

Notes:
*) CSMA/CD:
- Contention: Any computer can send data any time (FCFS) (Like in TDM & CSMA/CD)
- TDM & CSMA/CD both have contention.

*) UDP:
- ICMP packet can be sent by both TCP & UDP.

*) Subnetting:
- If NID matches with two row, take the one with more no. of 1's in SM

like:

SM
IP
NID

Destination	SM	Interface
128.75.43.0	255.255.255.0	a
128.75.43.0	255.255.255.128	b
192.12.17.5	255.255.255.255	c
Default	—	d

Packet IP → 128.75.43.16
⇒ NID ⇒ 128.75.43.0
So, go with 'b' not 'a'

- Firstly, start with longest mask route SM और NID को AND करते हैं तो
- CIDR में भी यही होता है, if two are matching then go with bigger 'n'.
like: matching with 131.16.0.0/12 & 131.22.0.0/15
→ So, go with IInd one

SWP:

- If n-bit for seq.no
  then: max^m window size
  for SR $\Rightarrow \frac{2^n}{2}$ [S $\frac{2^n}{2}$ — R $\frac{2^n}{2}$]
  for GBN $\Rightarrow 2^n - 1$ [S 2^n-1 — R 1]

- For GBN; If $W_S = N$, $W_R = 1$
  then, we know that # Seq. no.s (max.) $= 1 + 2a = 41$ (say)
  then, $W_S = 40$, $W_R = 1$

Network layering:

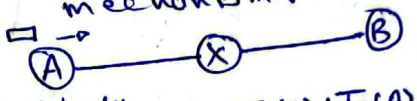
- PL does the job of encoding/decoding of data for physical transmission.
- Chart of protocols in Layers:-
  AL → SMTP, POP, FTP, HTTP, DNS
  TL → TCP, UDP,
  NL → BGP; ARP, RARP, BOOTP, DHCP, ICMP, IGMP, IP
  DLL → PPP
  PL →
- SMTP ⇒ Only 1 TCP conn^n
  Telnet ⇒ Only 1 TCP conn.^{n}
  HTTP ⇒ Multiple conn.^{n}s for each resource
  FTP ⇒ [1 TCP conn.^{n} for Telnet; 1 TCP conn^n for data exchange]
- max.^{m} data rate possible = Effective BW

TCP:

- Seq. no.s & WAT: Say we have 32b for seq.no. & LT = 64s
  then, seq.no. ranges 0 ... $2^{32}-1$; time ranges 1 ... 64
  i.e. Seq.no. rate change = 1/64
- If a fragment gets lost, the actual source is needed to retransmit the entire packet, bec. source doesn't know about fragmentation, only routers know about fragmentation.
- Only from one fragment of a packet you can't determine destination port no., we can determine destination port no. after reassembling at the destination side.
- If in between, a particular fragment is lost, it is not a guarentee that the original source of the packet will retransmit that packet with same identification no., it may change.
- TCP ensures in-order delivery (bec. seq. no.s are there)

Random Notes:-

- RIP uses port no. 520 in UDP
- ARP is NL protocol, but ARP is encapsulated in DLL frame, not in IP datagram
- In passive FTP, instead of 20&21 port no. you use some random port no.s
- (X) uses Store & forward (or) packet switch mechanism:-



Time taken by a frame to completely recieved at B $= T_t(A) + T_p(A) + T_t(X) + T_p(X)$

- FTP uses persistent control connection & non-persistent data connection.

60 packets Send
Here, for first packet :-
Total time $= 3[T_t + T_p]$
, for rest 59 packets :-
Total time $= 59 \times T_t$

1. Hamming Code
2. Combinatory
3. Eigenvalue
4. Cards
5. Aptitude
6. Test solutions

*) Detection & retransmission is better than correction

Error

not in syllabus

*) Logic of error detection

Actual Data → Divided into Blocks (Datawords) → Add redundant bit to each dataword → codeword

Now, send this codeword instead of actual data

If Actual data is divided into data blocks (or) datawords where each block is 'k' bits i.e. dataword

If we add 'r' redundant bits to each dataword then we get a codeword of size n

where $n = k + r$

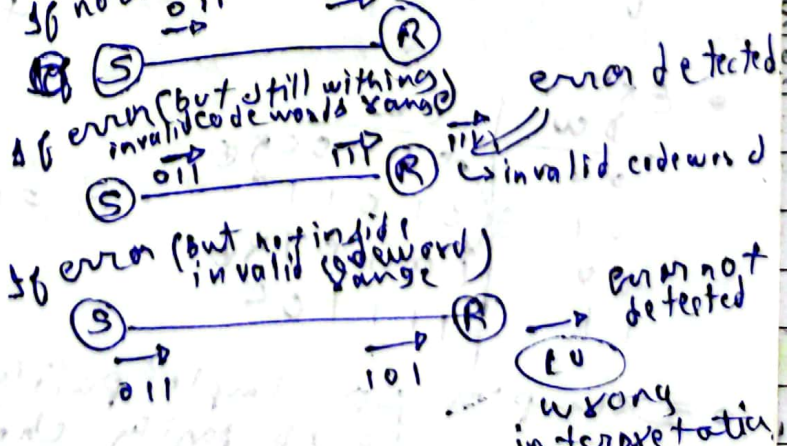
No. of datawords $= 2^k =$ No. of valid codewords

No. of invalid codewords $= 2^n - 2^k$

Ex:

Dataword	codeword
0 0	0 0 0
0 1	0 1 1
1 0	1 0 1
1 1	1 1 0

Invalid codewords $= 2^3 - 2^2$
$= 4$

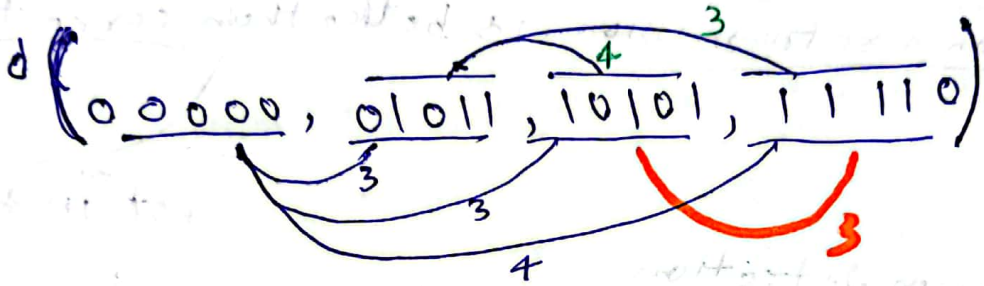


*) Hamming distance $d(x, y)$

- Ex:) $d(100, 011) = 3$

$d(101, 100) = 1$

$d(101101, 110) \Rightarrow \begin{matrix} 101101 \\ 000110 \end{matrix} \Rightarrow 4$



$\min(3, 3, 4, 4, 3, 3) = 3 = \text{Ans}$

- Minimum hamming distance to detect 'x' bit errors with guarentee:

If min.m hamming distance is 'd'

then we can detect '$d-1$' bit error with guarentee & we can correct $\lfloor \frac{d-1}{2} \rfloor$ bit er

i.e. to detect 'x' bit error

you need $d = x+1$

*) Linear block codes:—

- If xor of two valid codewords is a valid codeword then the codewords are linear block codes
- Min.m hamming distance of linear block codes = min.m no. of 1's among all the non-zero code words

- Ex:

dw	cw
00	00000
01	01011
10	10101
11	11110

$d = \min.^{m}$ no. of 1's in $(01011, 10101, 11110)$:

$= 3$

- Examples → Simple parity check code
  → 2D parity check code
  → cyclic codes
  → Checksum

Simple parity check code:

- Stuff a bit in dataword to make codeword containing either even no. of 1's (even parity) or odd no. of 1's (odd parity)

Ex: (even parity)

DW		CW
10101101	⇒	1 10101101
	or	10101101 1

- Here, we can only detect odd no. of errors (1 bit, 3 bit, ...) we can't detect even no. of errors (2 bit, 4 bit, ...)
- Here min^m hamming distance will be 2.

2D parity check code:

- Organise datawords in a table.
- for each row & column, one parity bit is calculated.
- whole table is then sent to receiver which again finds parity for each row & column.
- It guarentees to correct one bit error & detect upto 3 bit errors.
- Ex: 7 bit for 4 datawords

1	0	1	1	0	1	0
1	1	0	0	0	1	0
1	0	1	1	0	1	1
1	0	1	1	0	1	1
0	1	0	1	1	0	

so, send 10110100 11 ...

i.e. 7 × 4 = 28 bits Data

8 × 5 = 40 bits code

- This scheme fails if parity bits have error

Cyclic codes:

- It is a type of linear block code in which if you cyclically rotate (left or right) a codeword, you get a valid codeword

Ex:

DW	CW
00	000
01	001
10	010
11	100

001 ⇒ 100 ✓

- CRC is a type of cyclic code.

Checksum:

- Send 1's complement of sum of all data bit packages along with data
- Ex: (7, 11, 12, 0, 6) ← 5 data packets of 4-bit each

⇒ checksum = −36

but 36 needs 6 bits (100100)

so do folding

```
  0100
+ 0010
------
  0110
```

⇒ checksum

9 ⇒ 1001

1's(9) ⇒ 0110

i.e. 36 ⇒ 0110

1's(36) ⇒ 1001

⇒ Checksum = 9

Send (7, 11, 12, 0, 6, 9)

At receiver: 9 ⇒ 1001

1's(9) ⇒ 0110

At receiver: sum ⇒ 45 ⇒ 101101

```
  1101
+ 0010
------
  1111
```

1's (1111) = 0000 ✓

- If we have 1 bit error in 2 different data packets at same bit position, then we can't detect errors.

7, 11, 12, 0, 6

```
1011   1100
1111   1000
```

*) Hamming codes:

Q1:) In conversion of Dw to CW, how many redundancy bits to add?

A:) $2^k \geq n+k+1$; k = no. of redundancy bits
n = Data bits
$n+k$ = codeword bits

Ex:) If Dw = 4b

then $2^k \geq 4+k+1$

at $k=2$; $4 \geq 7$ ✗

at $k=3$; $8 \geq 8$ ✓

$\Rightarrow$ $k=3$

CW = 7b

Q2:) What are positions of redundancy bits? what is CW?

A:) They are at powers of 2

Ex:) If Dw = 1011

$\Rightarrow n=3$

$\Rightarrow 2^k \geq 3+k+1$

at $k=3$ ✓

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	_	1	_	_

$D_7 = P_4 + P_2 + P_1$

$D_6 = P_4 + P_2$

$D_5 = P_4 + P_1$

$D_3 = P_2 + P_1$

$P_1 = \underline{1}\ D_7\ D_5\ D_3$ (1 1 1) even parity

$P_2 = \underline{0}\ D_7\ D_6\ D_3$ (1 0 1)

$P_4 = \underline{0}\ D_7\ D_6\ D_5$ (1 0 1)

$\Rightarrow$ CW = 1 0 1 0 1 0 1

Q3:) Given CW is valid or not?

CW = 1 0 1 0 1 0 1

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	0	1	0	1

$D_7 = P_4 + P_2 + P_1$

$D_6 = P_4 + P_2$

$D_5 = P_4 + P_1$

$D_3 = P_2 + P_1$

$P_1 = \underline{1}\ D_7\ D_5\ D_3$ (1 1 1) ✓

$P_2 = \underline{0}\ D_7\ D_6\ D_3$ (1 0 1) ✓

$P_4 = \underline{0}\ D_7\ D_6\ D_5$ (1 0 1) ✓

correction bits: $C_1 = 0$, $C_2 = 0$, $C_4 = 0$

$\Rightarrow$ No correction needed

$\Rightarrow$ Codeword valid ✓

Q: CW = 1 0 0 0 1 0 1

given cw valid?

If not, then what's correct cw?

1 0 0 0 1 0 1

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

$D_7 = P_4 + P_2 + P_1$

$D_6 = P_4 + P_2$

$D_5 = P_4 + P_1$

$D_3 = P_2 + P_1$

$P_1 = \underline{0}\ D_7\ D_5\ D_3$ (1 0 1) $\Rightarrow C_1 = 1$

$P_2 = \underline{0}\ D_7\ D_6\ D_3$ (1 0 1) $\Rightarrow C_2 = 0$

$P_4 = \underline{1}\ D_7\ D_6\ D_5$ (1 0 0) $\Rightarrow C_4 = 1$

$C_1 = 1$

$C_2 = 0$

$C_4 = 1$

$C_4\ C_2\ C_1$

1 0 1

$\Rightarrow 5^{th}$ bit corrupt

7 6 5 4 3 2 1

1 0 0 0 1 0 1

1

$\Rightarrow$ correct CW = 1 0 1 0 1 0 1

Note: while calculating C_1, C_2, C_4, C_8 ; shortcut :—

$C_8\ C_4\ C_2\ C_1 \equiv P_8\ P_4\ P_2\ P_1$

$\oplus\ P_8\ P_4\ P_2\ P_1$

$C_8\ C_4\ C_2\ C_1$

~~~ $\Rightarrow$ given

~~~ $\Rightarrow$ calculated

Ex:

1 1 0 1

1 0 0 1

0 1 0 0 $\Rightarrow 4^{th}$ bit

2:) $P_1 P_2 P_4 P_8$ वाली Example

CW: 1 1 1 0 0 1 0 0 1 1 1 1

$P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7\ P_8\ D_9\ D_{10}\ D_{11}\ D_{12}$

$D_{12} = P_8 + P_4$

$D_{11} = P_8 + P_2 + P_1$

$D_{10} = P_8 + P_2$

$D_9 = P_8 + P_1$

$D_7 = P_4 + P_2 + P_1$

$D_6 = P_4 + P_2$

$D_5 = P_4 + P_1$

$D_3 = P_2 + P_1$

$P_8 = 0$ — $D_{12}\ D_{11}\ D_{10}\ D_9$ → 1 1 1 1

$P_4 = 0$ — $D_{12}\ D_7\ D_6\ D_5$ → 1 0 1 0

$P_2 = 0$ — $D_{11}\ D_{10}\ D_7\ D_6\ D_3$ → 1 1 0 1 1

$P_1 = 1$ — $D_{11}\ D_9\ D_7\ D_5\ D_3$ → 1 1 0 0 1

$C_8 C_4 C_2 C_1 \equiv P_8\ P_4\ P_2\ P_1 \equiv$ 0 0 1 1

$P_8\ P_4\ P_2\ P_1$ 0 0 0 1

0 0 1 0 ⇒ 2nd bit corrupt

⇒ correct CW: 1 0 1 0 0 1 0 0 1 1 1 1

Q:) • For 8 bit data word 0 0 1 1 1 0 0 1, check bits (or) parity bits (or) redundancy bits would be 0 1 1 1.

• Suppose when dw is read from memory, the check bits are found to be 1 1 0 1.

• what is dw that was read from memory?

Sol:) $D_{12}\ D_{11}\ D_{10}\ D_9\ D_8\ D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

0 0 1 1 0 0 0 1 1 1 1

$P_8 = 1$ ~~~

$P_4 = 1$ ~~~

$P_2 = 0$ ~~~

$P_1 = 1$ ~~~

$C_8 C_4 C_2 C_1$: 1 1 0 1

⊕ 0 1 1 1

1 0 1 0

⇒ 10th bit

DW read: 0 0 0 1 0 1 0 0 1 1 1 1

Q:) If Delimiter: 0 111 1110

And after bit stuffing o/p is 0 11111 00101 then before, bit stuffing?

ED: 0 1111 110

Data: 0 11111 0 0 1 0 1

before bit stuffing: 0 11111 0 1 0 1

Frame/Header
1. Ethernet
2. IP V4
3. TCP
4. VDP