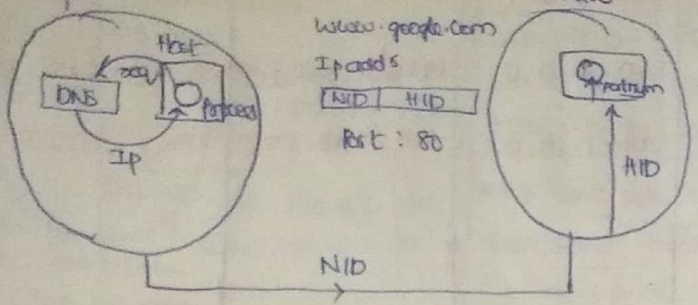
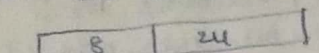
Computer Networks

*) N/W



*) When CN introduced in early 1980's



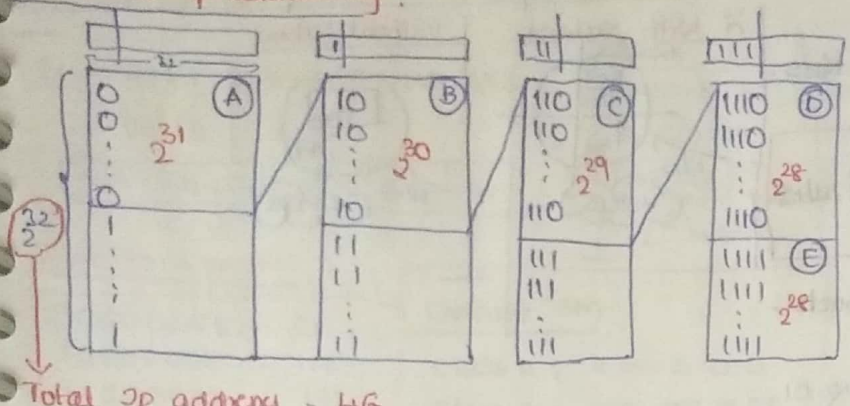
Networks = 2^8 = 256 — Not sufficient due to Internet revolution

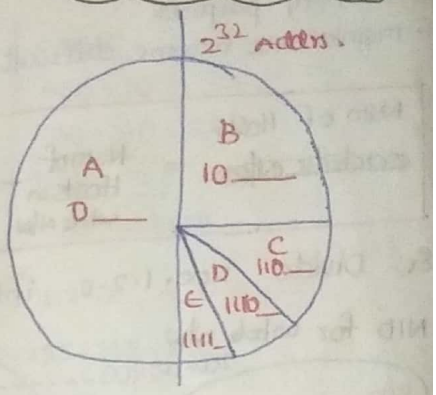
Hosts = 2^{24} = 16M Hosts.

↓ sol

Classfull Ip addressing

Classfull Ip addressing :





Representation of Ip addr : Dotted decimal representation

Points to remember

Class	NID	HID	Range	Num of Nws	Num of Hosts	structure
A	8	24	1 – 126	126	2^{24} = 16M/Nw − 2	0 ..... (8, 24)
B	16	16	128 – 191	16,384	65K H/Nw − 2	10 ---- (16, 16)
C	24	8	192 – 223	2M	256 H/Nw − 2	110 ---- (24, 8)
D	–	–	224 – 239	2^{28} Ip addresses		1110 ----
E	–	–	240 – 255	2^{28} Ip addrs		1111 ----

*) Class D :- used for multicasting

*) Class E :- used for Military Govt Apps

*) 127.0.0.1 – 127.255.255.255

Loop back Ip addresses.

→ Used to identify device, OSPF protocols uses loopback addressing

→ An Address that sends outgoing signals back to the same Computer for testing (Ping)

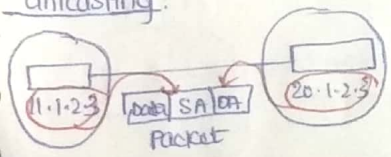
* 0.0.0.0 :- non routable meta address.

Uses :- Invalid, Unknown, Inapplicable } targets.

So, 0 is not used in Class A

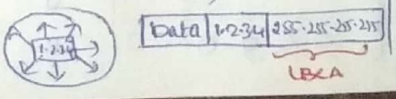
Casting :

unicasting :

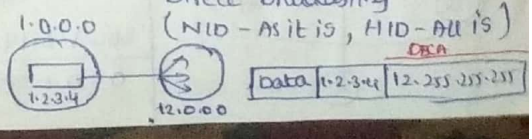


Broadcasting

Limited Broadcasting (Done in same N/w)



Direct Broadcasting (NID – As it is, HID – All 1's)



Note :

LBCA : 255.255.255.255

DBCA : NID + All Host's 1's

NID : NID + All Host's 0's

So, for Every N/w we remove 2 Host id addresses

	IP	NID	DBCA	LBCA
A	10.15.20.60	10.0.0.0	10.255.255.255	255.255.255.255
B	130.1.2.3	130.1.0.0	130.1.255.255	255.255.255.255
C	200.1.10.100	200.1.10.0	200.1.10.255	255.255.255.255
D	230.1.2.3	—	—	—
E	245.2.6.8	—	—	—
K	300.1.2.3	—	—	—

Subnetting :

- Dividing a N/w into smaller N/w

Why subnetting ?

- Security purposes
- maintenance become difficult for large N/w's.

Num of Hosts = Num of Hosts in whole N/w − 2 * Num of subnets

∴ With subnet: from host → router → N/w → 4 steps

Without subnet: from host → router → N/w → 3 steps

Ex:- Divide 200.1.2.0 into 4 subnets.

NID for whole N/w

200.1.2.0

200.1.2.00 ------ 200.1.2.01 ------

SID - 200.1.2.0 (00000000)

DBC - 200.1.2.63 (00111111)

SID - 200.1.2.64 (01000000)

DBC - 200.1.2.127 (01111111)

200.1.2.10 ------ 200.1.2.11 ------

SID - 200.1.2.128 (10000000)

DBC - 200.1.2.191 (10111111)

SID - 200.1.2.192 (11000000)

DBC - 200.1.2.255 (11111111)

DBCA for whole N/w

*) To identify which subnet is used for an ip addr we use subnet mask

Subnet Mask : 32 bit → 1's ⇒ NID & SID; 0's ⇒ Hosts

Ex:- 200.1.2.00------ / 01------ / 10------ / 11------ → Subnet mask → 255.255.255.192

If we 'AND' the subnet mask with the ip address, we find the subnet that particular ip addr belongs to

Ex:- 200.1.2.130

SM : 11111111.11111111.11111111.11000000

Ip : 11001000.00000001.00000010.10000010

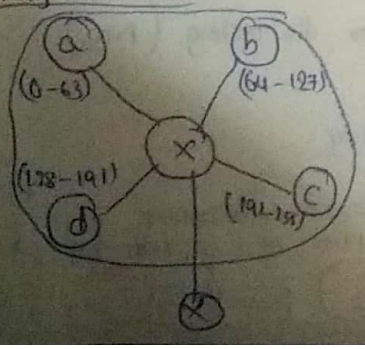
11001000.00000001.00000010.10000000

200 . 1 . 2 . 128 ⇒ SID

*) If a subnet contains fixed num of hosts then subnet mask is same for all subnets.

*) If ip address matches with two subnets

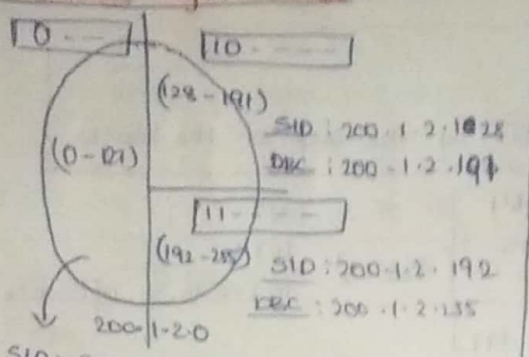
Physical view



Routing table :

NID	SM	Interface
200.1.2.0	255.255.255.192	a
200.1.2.64	" "	b
200.1.2.128	" "	c
200.1.2.192	" "	d
0.0.0.0	0.0.0.0	e → packet has to be sent out

Variable length Subnet :



SID : 200.1.2.0

DBC : 200.1.2.127.

Subnet mask	
SID	SM
200.1.2.0	200.1.2.128
200.1.2.128	200.1.2.192
200.1.2.192	200.1.2.192

*) If there are two matches then select highest SM

ie 200.1.2.192 > 200.1.2.128

Routing table (2)

SID	SM	Interface
200.1.2.0	200.1.2.128	a
200.1.2.128	200.1.2.192	b
200.1.2.192	200.1.2.192	c
0.0.0.0	0.0.0.0	d

Ex:- SM : 255.255.255.192

- 26 1's 6 0's

NID + SID = 26 | 2^6 Hosts | N/w

→

Class A	Class B	Class C
NID = 8	NID = 16	NID = 24
8 + SID = 26	16 + SID = 26	24 + SID = 26
SID = 18	SID = 10	SID = 2
2^{18} subnets	2^{10} subnets	2^2 subnets

Points to remember :

00000000	0
10000000	128
11000000	192
11100000	224
11110000	240
11111000	248
11111100	252
11111110	254
11111111	255

Default SM

Class A : 255.0.0.0

Class B : 255.255.0.0

Class C : 255.255.255.0

Ip : 200.1.2.3 (Class C) → NID = 24

SM : 255.255.255.192

NID + SID = 26

24 + SID = 26

SID = 2 → Subnets = $2^2 = 4$,

Hosts = $2^6 - 2 = 64$ (56)

Subnet mask	Num of Hosts	Subnets CA	Subnets CB	Subnets CC	NID /16
255.0.0.0	$2^{24} - 2 = 16M$	$2^0 = 1$	×	×	×
255.128.0.0	$2^{23} - 2 = 8M$	$2^1 = 2$	×	×	×
255.192.0.0	$2^{22} - 2 = 4M$	$2^2 = 4$	×	×	$2^0 = 1$
255.240.0.0	$2^{20} - 2 = 1M$	$2^4 = 16$	×	×	$2^2 = 4$
255.255.0.0	$2^{16} - 2 = 65k$	$2^8 = 256$	$2^0 = 1$	×	$2^6 = 64$
255.255.254.0	$2^9 - 2 = 510$	2^{15}	$2^7 = 128$	×	2^{13}
255.255.255.240	$2^4 - 2 = 14$	2^{20}	2^{12}	$2^4 = 16$	2^{18}
255.255.255.0	$2^8 - 2 = 254$	2^{16}	2^8	$2^0 = 1$	2^{14}

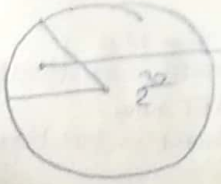
Classless Inter domain routing :

∴ Class full Ip addressing :

If we want 300 Ip addresses then we have to buy Class B N/w which contains 65k Ip addresses from IANA (Internet Assigned Number Authority)

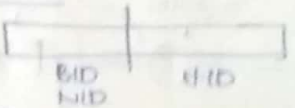
So, 65k - 300 Ip addresses will go waste

Classless :



→ Block As per our requirement

format :



Eg :

a.b.c.d/(n) → num of bits for NID

Rules for forming CIDR :

- All IP addresses should be Contiguous
- Block size should always be power of 2 (2^n)
- First IP address in the block should be evenly divisible by the size of the block.

Ex:- 150.10.20.64
150.10.20.65
⋮
150.10.20.127
$\Rightarrow 64 \Rightarrow 2^6$

(i) ✓
(ii) ✓
(iii) 150.10.20.01 000000
$\div 2^6$

∴ CIDR

150.10.20.64
150.10.20.65
⋮
150.10.20.191
$\Rightarrow 128 \Rightarrow 2^7$

(i) ✓
(ii) ✓
(iii) 150.10.20.0 0000000
$\div 2^7$ ×

∴ Not CIDR

Ex:- 20.10.30.35/27
20.10.30.001 | 00011
NID | HID
First IP ← 00000
⋮
Last IP ← 11111
$\Rightarrow 2^5 \Rightarrow 32$ Hosts

BID : 200.10.30.32
DBC : 200.10.30.63

Ex:- 100.1.2.35/28
100.1.2.0010 | 0011
BID | HID
0000
⋮
1111
$2^4 = 16$ Hosts

BID : 100.1.2.32
DBC : 100.1.2.47

Ex:- 100.1.2.32
⋮
100.1.2.47
$\Rightarrow$ 16 hosts.
NID = 28 HID = 4
∴ 100.1.2.32/28 (or)
100.1.2.40/28

Subnetting in CIDR :

10.20.30.0 | 010 1000
HID → 128 Hosts.
10.20.30.40/25

00 | 01
10 | 11

BID : 10.20.30.0
DBC : 10.20.30.31
10.20.30.0/27

BID : 10.20.30.32
DBC : 10.20.30.63
10.20.30.32/27

BID : 10.20.30.64
DBC : 10.20.30.95
10.20.30.64/27

BID : 10.20.30.96
DBC : 10.20.30.127
10.20.30.96/27

BID	SM /27
10.20.30.0	255.255.255.224
10.20.30.32	〃 〃
10.20.30.64	〃 〃
10.20.30.96	〃 〃

Variable length subnet mask in CIDR :

20.30.40.f0/25
20.30.40.00001010
(0-127) → 128 hosts.

0 | 10
11

BID : 20.30.40.0
DBC : 20.30.40.63
20.30.40.0/26

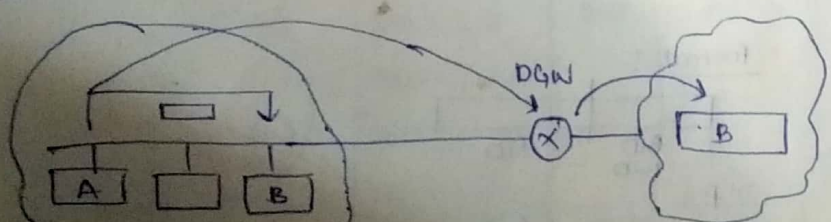
BID : 20.30.40.64
DBC : 20.30.40.95
20.30.40.64/27

BID : 20.30.40.96
DBC : 20.30.40.127.
20.30.40.96/27

BID	SM
20.30.40.0/26	255.255.255.192
20.30.40.64/27	255.255.255.224
20.30.40.96/27	255.255.255.224

*) ISP provides
- IPV4 address
- DGW (Default gateway)
- SM
- DNS

*)



Host A: IP_A / SM_A / NID_{AA}

Host B: IP_B / SM_A / NID_{BA}

$\Rightarrow$ = → Same N/w
Packet is sent directly
≠ → diff N/w
Packet is sent to router

Ex:- A: I_A : 200.1.2.10
S_A : 255.255.255.128
NID_{AA} : 200.1.2.0

B: I_B : 200.1.2.69
S_A : 255.255.255.128
NID_{BA} : 200.1.2.0

A/c to A both are in same N/w

I_A : 200.1.2.10
S_B : 255.255.255.192
NID_{AB} : 200.1.2.0

I_B : 200.1.2.69
S_B : 255.255.255.192
NID_{BB} : 200.1.2.64

A/c to B both are in different N/w.

Note:

- If SM is 255.255.255.255 then point to point Connection is present. ie,

I_A : - . — . — . — I_B : — . — . — . —

$I_A \neq I_B$ → point to point Connection

- If DBCA : 200.1.15.255 then Num of hosts has many possibilities

200.1.0000 1111.11111111 HOST HID

Supernetting / Aggregation:

- Combining small N/w's into a whole N/w

Rules:

- All the N/w should be Contiguss
- All N/w size should be same and present in powers of 2 (2^n)
- The first NID should be divisible by size of the block.

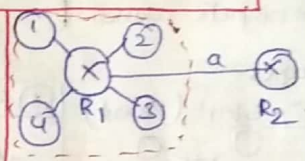
Ex:- ① 200.1.0.0
② 200.1.1.0
③ 200.1.2.0
④ 200.1.3.0

(i) ✓
(ii) 2^8 ✓
(iii) 200.1.00000000.00000000 ✓

Size of N/w ⇒ $4 * 2^8$ ⇒ 2^{10}

$\div 2^{10}$

∴ supernetting possible.



At R_2

NID	Interface	SM
200.1.0.0	a	255.255
200.1.1.0	a	.255.0
200.1.2.0	a	
200.1.3.0	a	

↓ minimized.

At R_2

SNID	Interface	Supernet mask
200.1.0.0	a	255.255.252.0

Supernet mask: (32 bit)

fixed part with is and variable part with 0's

200.1.00000000.00000000
200.1.000000 01.00000000
200.1.000000 10.00000000
200.1.000000 11.00000000

fixed: 11111111.11111111.111111 00.00000000 (var)

255.255.252.0 → supernet mask

Shortcut:

To find supernet id :- First ip address in N/w

To find supernet mask :- $2^8+2^8+2^6+2^8 \Rightarrow 2^{10}$ ⇒ HID = 10, NID = 22

/22 ⇒ 255.255.252.0

Ex:- 100.1.2.0 /25 ①

100.1.2.128 /26
100.1.2.192 /26
→ ② Not Equal size → They can be supernetted.

SNID :- 100.1.2.128
SMask :- $2^6+2^6 \Rightarrow 2^7$ ⇒ HID :- 7, SNID :- 25 /25

vs

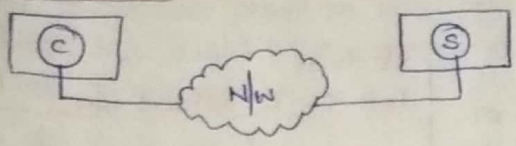
100.1.2.128 /25
100.1.2.0 /25
→ They can be supernetted ⇒

SNID :- 100.1.2.0 /24
SMask :- $2^7+2^7 \Rightarrow 2^8$ ⇒ HID :- 8, SNID :- 24

∴ SMask :- /24
:- 255.255.255.0

∴ ID : 100.1.2.0 /24

ISO-OSI layer :



*) Functions (7o)

- mandatory
  - Flow Control
  - Error Control
  - Access Control
  - Mux & Demux
  - Addressing
- optional
  - Encryption
  - decryption
  - checkpointing
  - Routing (Imp)

Implemented models: ISO-OSI, Tcp/IP, ATM, X.25, IEEE

ISO-OSI Model :

- International standard organization
- open standard interface.

User Interactability

App layer
Presentation layer
Session layer
Transport layer
Network layer
Datalink layer
Physical layer.

Thick layer

Complex (routing Algos)

H/w & S/w

H/w

Imp

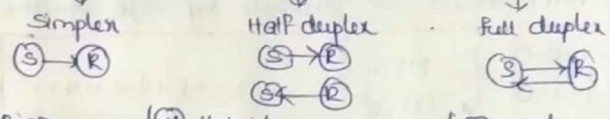
3 DLC strategy
Encapsulation
abstraction
Testing is easy

① **Physical layer** : It deals with electrical, mechanical, functional and procedural characteristics of physical links

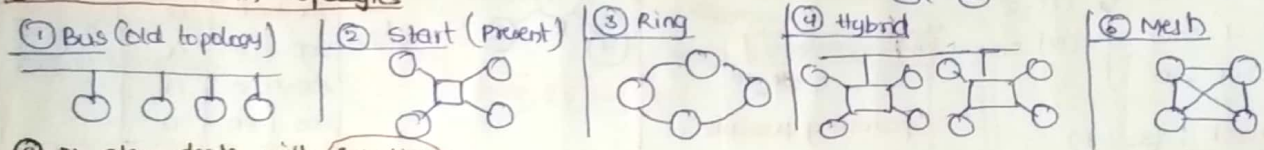
Ex:- Copper wires – Electrical signals.
optical – light
wireless – Electromagnetic waves.

ⓐ Physical layer deals with functional properties.

Transmission modes

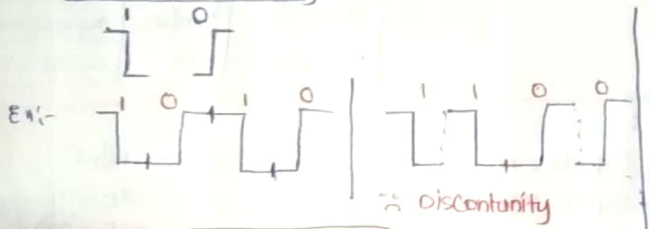


ⓑ PL deals with topologies

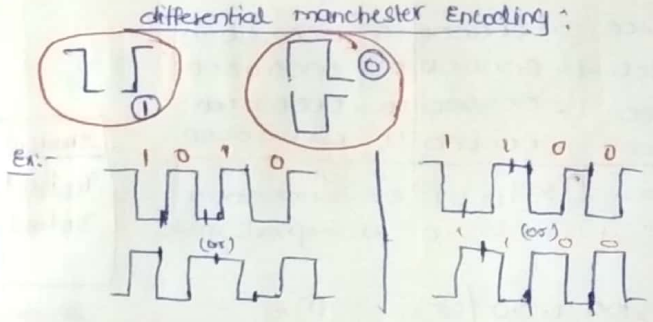


ⓒ PL also deals with Encoding

Manchester Encoding



differential manchester Encoding :



Baud rate = 2 * bit rate

↓

Num of voltages changing in Encoding

② Data link layer : Responsible for

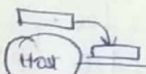
1. Flow control (stop & wait, Goback N, SR)
2. Error control (CRC, Checksum) → Tcp/Ip.
3. Access control (csma/cd, Token passing)
4. Framing
5. physical addressing

Total delay = $T_t + T_p + T_q + T_{proc}$ (0 = negligible)

Imp ∴ Total delay = $T_t + T_p = \frac{L}{B} + \frac{d}{v}$

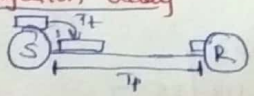
Delays in CN :

① Transmission delay :



$T_t = \frac{data}{B/s} \Rightarrow \frac{L}{B}$

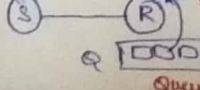
② Propagation delay



$T_p = \frac{d}{v}$

For optical fibres $v = 3 * 10^8$ m/sec

③ Queue delay & processing delay (negligible)

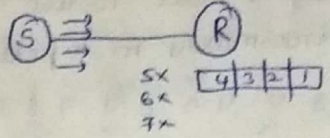


Earth satellite propagation time = 270 msec

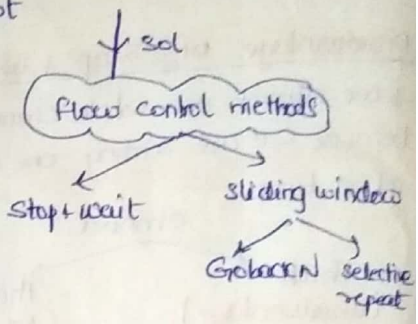
Flow control methods :

Points to remember :

Property	Stop & wait	GBN	SR
Efficiency	$\frac{1}{1+2a}$	$\frac{N}{1+2a}$	$\frac{N}{1+2a}$
Buffers	$1_S + 1_R$	$N_S + 1_R$	$N_S + N_R$
Seq Numbers	2	N+1	2N
retransmissons	1	N	1
Band width	low	high	moderate
CPU	low	moderate	high
Implementation	low	moderate	Complex
Acknowledgement	Independent	Cumulative	Independent

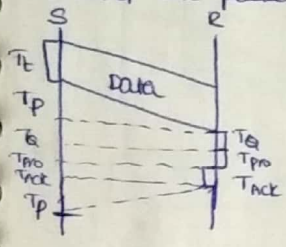


∴ Sender should send packets only when reciever is ready to accept



① Stop & wait protocol :

To send one packet



∴ Total time = $T_t(data) + T_p + T_Q + T_{pro} + T_t(ack) + T_p$

Negligible

$T = T_t + 2T_p$ | $a = \frac{T_p}{T_t}$

Efficiency (η) = $\frac{\text{Useful time}}{\text{Total time}} = \frac{T_t}{T_t + 2T_p} = \frac{1}{1+2a}$

Throughput :

num of bits able to send per sec.

Throughput = $\frac{L}{T_t + 2T_p} = \frac{\frac{L}{B} \cdot B}{T_t + 2T_p}$

Throughput = $\eta * B$

effective BW

Bw utilization

Analysis of Efficiency :

$\eta = \frac{1}{1 + 2\left(\frac{d}{v}\right)\left(\frac{B}{L}\right)}$ → Constant

If $d \uparrow \Rightarrow \eta \downarrow$ → favourable → LAN

If $L \uparrow \Rightarrow \eta \uparrow$ → favourable → Big packet

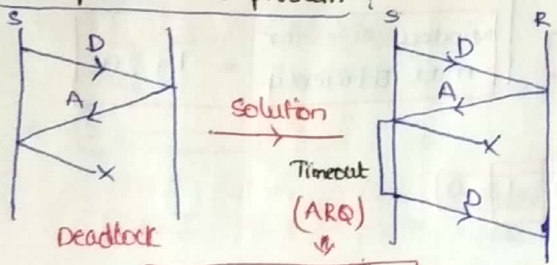
unfavourable → WAN, Small packet

If BER(p) is Included

$\eta = \frac{1-p}{1+2a}$

Problems in Stop & wait protocol :

① Data packet lost problem :

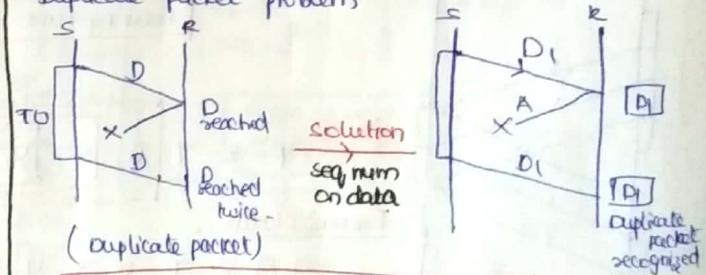


Stop & wait + Timeout → SLW ARQ

(Automatic repeat request)

② Acknowledgement lost problem :

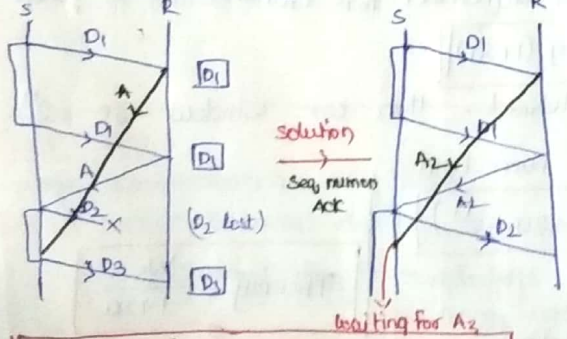
– Duplicate packet problem



(Duplicate packet)

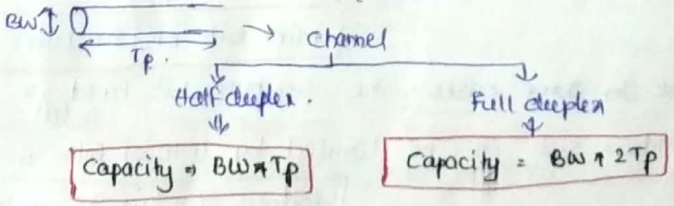
SLW + Timeout + seq num on data → SLW protocol

③ Delayed Ack problem :



Stop & wait + TO + seq num on data + seq num on Ack

Capacity of channel / wire / link :



Capacity = BW * T_p

Capacity = BW * $2T_p$

Q) S&W protocol
out of every 4th packet is lost
Num of transmisons for 10 packets

1 2 3 4 4 5 6 7 7 8 9 10 10
(4 and 7 and 10 are crossed)

Transmissons = 13
retransmitted packets → 4, 7, 10

Q) S → R (400 packets)
Error prob = 0.2
Total num of packets to be sent?

Total packets ⇒ $400 + \frac{400(0.2)}{80} + \frac{80(0.2)}{16} + \frac{16(0.2)}{3.2} + \frac{3.2(0.2)}{0.64} + \dots$

$\Rightarrow \frac{a}{1-r} \Rightarrow \frac{400}{1-0.2} \Rightarrow 500$

Note: n - packets, p error prob.

Total packets $= n + np + np^2 + np^3 + \dots$
$= n[1 + p + p^2 + \dots]$
$= n\left[\frac{1}{1-p}\right] \Rightarrow \frac{n}{n-p}$

Disadvantage of stop & wait:

→ We cannot use whole channel capacity because we are sending one data packet at a time

channel
Thick (disadvantage) (η low) ⟷ Thin (Advantage) (η high)

$\Rightarrow \eta = \frac{1}{1+2a} \Rightarrow \frac{1}{1+2\frac{T_p}{T_t}} \Rightarrow \frac{1}{1+\frac{2T_p * B}{L}}$ Capacity.

(Thick) Capacity ↑ η (low)
(Thin) Capacity ↓ η (high)

To Improve Efficiency we use pipelining

Pipelining:

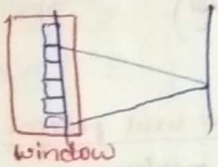
Normally T_t sec → 1 packet
1 sec → $\frac{1}{T_t}$ packet

$T_t + 2T_p$ sec → $\frac{T_t + 2T_p}{T_t}$ packets ⇒ $1+2a$ packets can be sent

but we are sending 1 packet ∴ Efficiency $(\eta) = \frac{1}{1+2a}$

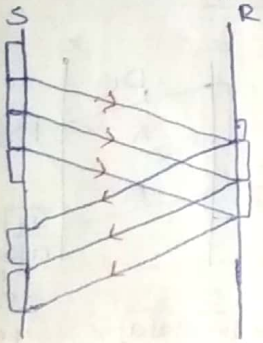
To Improve Efficiency we should send more packets instead of 1 packet ie

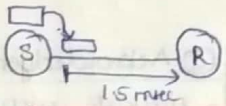
$\eta = \frac{1}{1+2a}(1+2a) \Rightarrow 100\%$ (window size) ⇒



Sliding window protocol:

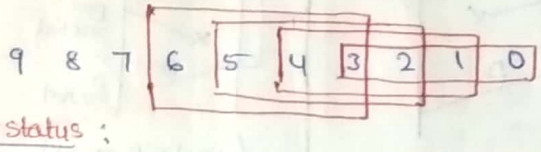
Let $T_t = 1$ msec $T_p = 1.5$ msec.





Routing time = 2 * Tp

Window size for max Efficiency = 1 + 2a



Present status:

9 8 7 | 6 5 4 3 | 2 1 0
Yet to send | waiting for ACK | Acknowledged packets

→ We can use infinite seq num for packet but min seq number for transmission of packet

Min bits of seq num = $\lceil \log_2(1+2a) \rceil$

But in some cases, the seq num bit field (n) is predefined. then our window size = 2^n

- window size can be limited by num of bits in seq num field.

Window size $(W_s) = \min(1+2a, 2^n)$

Efficiency = $\frac{W_s}{1+2a}$

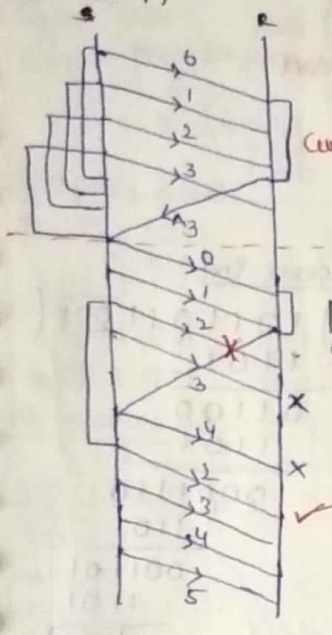
- Sliding window protocol (Theoretical)
  - GBN (Go Back N)
  - SR (Selective repeat)

⇒ Practical Implementations.

GO BACK N ($N>1$)

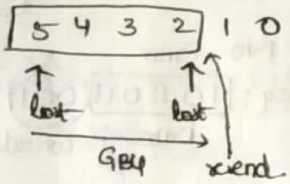
① Sender window size in GBN is N
② Reciever window size is Always '1'.
③ Acknowledgement is cumulative

Eg:- GB 4;



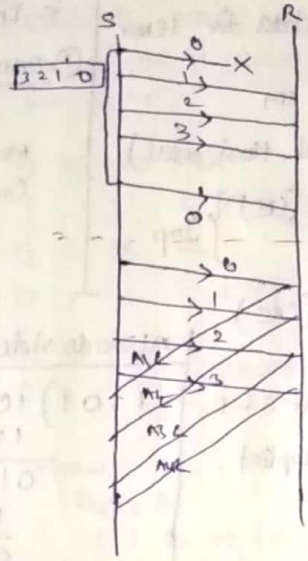
∵ less traffic
∴ poor reliability

*)



Selective repeat (SR)

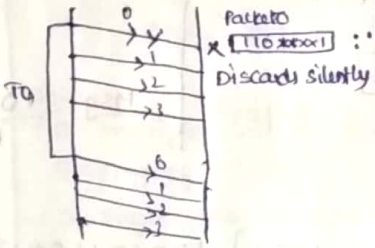
① Sender window size is greater than 1 ($W_S > 1$)
② Sender W_S is equal to reciever W_R ie ($W_S = W_R$)
③ Acknowledgement is Independent.



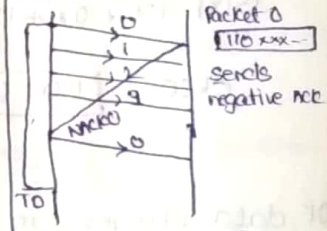
∵ reliability
∴ more traffic

If packet is corrupted :

In GBN



In SR



Packet is sent faster before timeout

Note :

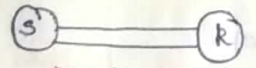
- In sliding window protocols. $W_S + W_R \leq$ num of seq numbers
- If sequence numbers are fixed as N then

GBN	SR
$W_S = N-1$	$W_S = N/2$
$W_R = 1$	$W_R = N/2$

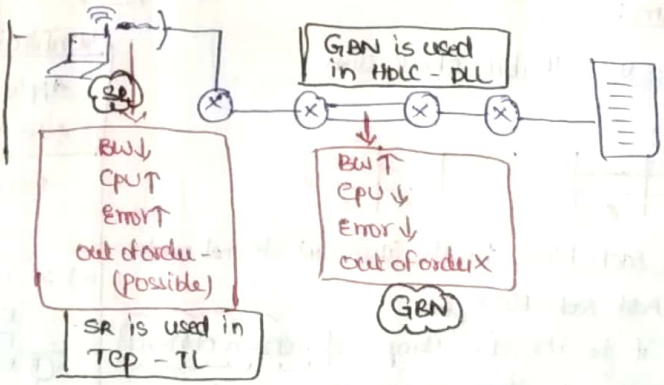
- If K bits are present in seq num field then

GBN	SR	Possible seq num
$W_S = 2^K - 1$	$W_S = 2^{K-1}$	$= 2^K$.
$W_R = 1$	$W_R = 2^{K-1}$	

*



BW ↑ CPU ↓ Buffer ↓ = GBN
BW ↓ CPU ↑ Buffer ↑ = SR



*) TCP - SR + GBN
uses combo of 75% 25%.

Error Control :

- mainly Concentrates on bit losses
Errors: packet discarded due to Congestion
bits changed due to thunderbolts
freq change - etc

GBN N(4) Total packet = 10 5th packet lost.

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

SR

1 2 3 4 5 5 6 7 8 9 9 10 ⇒ 12 transmissions.

Error handling:

Error detection:

retransmisson is reqd

① Sent Data 1M + Sent Data 1M = 2M data for 1Error

If Error Cond is 1 in 1M

② parity checking (vertical, horizontal)

③ cyclic redundancy check (CRC) } Imp

④ Checksum

Error Correction

No retransmisson because the locality of Error bit is known

① Hamming Code (1KB + 0.5KB redundant)

More redundancy

Computationally infeasible.

Cyclic redundancy check : (CRC)

⑤ → CRC Generator 1101 → R

1011011 Known to both S & R

→ Exor/Mod 2 operation is applied.

CRC Generator G(x):

If G(x) is x^3+x+1 then

$G(x) = 1x^3 + 0x^2 + 1x + 1$

CRCG : 1011

At sender side

```
1101) 1011011 000 (
      1101
      ----
      01100
       1101
       ----
       0001110
          1101
          ----
          001100
           1101
           ----
           0001   CRC Code
```

Msg: 1011011 001 (Data, CRC code)

At Reciever End

```
1101) 1011011 001 (
      1101
      ----
      01100
       1101
       ----
       0001110
          1101
          ----
          001101
           1101
           ----
           0000
```

No Error

Msg: 1011011 001

Data is considered.

If data changes. ie

1011011 → 1001011

```
1101) 1001011 001 (
      1101
      ----
      01000
       1101
       ----
       01011
        1101
        ----
        01101
         1101
         ----
         0000 001   Not 000
```

Error detected

requested for retransmission.

Checksum:

TCP, UDP → uses 16 bit check sum

Algorithm:

① 

② Convert Each block to its equivalent decimal number

③ Now Add Each Block

④ Convert it to its 1's Comp → Checksum obtained

⑤ Add checksum to data in Checksum field

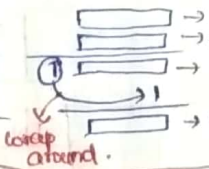
Verification: (At reciever End)

Apply check sum Algo upto step ③

Add checksum value in Checksum field

Result → 1's comp < If zero → No Error; Not zero → Error occured → req for retransmission

*) If we use 8 bit Checksum



Note:

- No Error detecting mechanisms are reliable bcoz of data and CRC/checksum Corrupted such that it agrees with the Corrupted data then reliability is lost

Such meaningful Errors are Impossible to handle but the probability of occuring of such Errors is very less. ~~ie If CRC/checksum is k bits then prob is~~

Error Control :

Hamming Code : (Error correction Strategy)

2m :- P_0 - check 1 bit, skip 1 bit - - -
P_2 - check 2 bit, skip 2 bit - - -
P_4 - check 4 bit, skip 4 bit - - -

Error detection :

- 7 bit hamming code.

4 bits → data
3 bits → Parity bits (Even / odd)

D_7	D_6	D_5	P_4	D_3	P_2	P_1
7	6	5	4	3	2	1

2^n $(n=0,1,2\ldots)$ → P_1 P_2 P_3

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

Even parity is used

Ex:- D : 1011

	P_3	P_2	P_1
P_1 (1) (Even Parity)	1	1	1
P_2 (0)	1	0	1
P_4 (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

↓ sent to reciever

At Sender End

At reciever end.

No change in code

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$: 1 0 1 0 1 0 1

Calculate

P_1 (1)	1	1	1
P_2 (0)	1	0	1
P_4 (0)	1	0	1

$P_1 P_2 P_4$

Compare with

If Equal → No Error

6th bit Error → Change in code

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$: 1 1 1 0 1 0 1

Calculate

P_1 (1)	1	1	1
P_2 (1)	1	1	1
P_4 (1)	1	1	1

Compare with

Not Equal → Error occured

Go for Error Correction

Error Correction

- calculate parity of matches put parity bit as '0', if it doesn't match then put parity bit as '1'.

Consider P_1 : $P_1\ D_3\ D_5\ D_7$: 1 1 1 1 → Even ($P_1 = 0$)

Consider P_2 : $P_2\ D_3\ D_6\ D_7$: 0 1 1 1 → odd ($P_2 = 1$)

Consider P_4 : $P_4\ D_5\ D_6\ D_7$: 0 1 1 1 → odd ($P_4 = 1$)

Now $P_4\ P_2\ P_1$: 1 1 0 ⇒ $(6)_{10}$ ⇒ 6th is Error bit

Change it

∴ 1 0 1 0 1 0 1

↓

Apply Error detection hamming code

No Error ✓

Note :

→ If multiple parity bits / data bits changes then Error correction gives wrong results.

→ If single parity bit / data bit changes then Error correction will give Correct result.

c) Access Control :

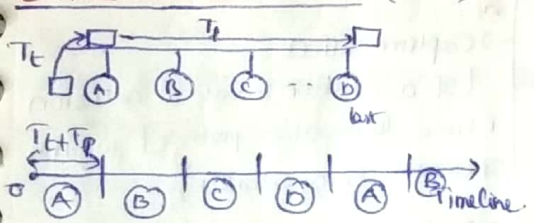
links

- Broadcast link → Collision
- Point to point link (S — R)

*) The Access to the link has to be controlled such that no collision occurs in the link and this is done by (Access Control methods)

Access Control methods :

① Time division method (TDM)

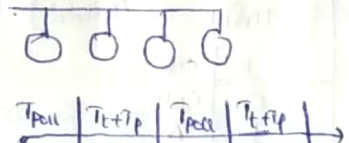


$$\eta = \frac{\text{useful time}}{\text{cycle time}} = \frac{T_t}{T_t+T_p} \Rightarrow \frac{1}{1+a}$$

*) The slot which is recieved for a station might not use it Completely

- No Ack in TDM

② polling :

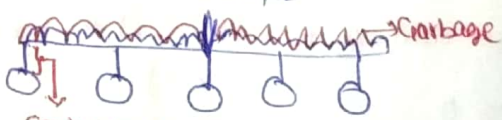


$$\eta = \frac{T_t}{T_{poll}+T_t+T_p}$$

- No Ack in polling

③ CSMA/CD :

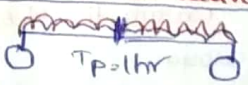
- Carrier sensing multiple access / collision detect
- No Ack in CSMA/CD



Carrier sensing that collision occured

How to detect collision occured

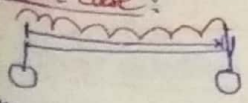
Best case :



10:00 Am - A & B started
10:30 Am - Collision
11:00 Am - Collision detected at A & B

The collision detection condition : $T_t > T_p$

Worst case:



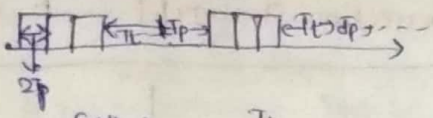
10:00 AM - A started transmitting
10:59:59 - B started transmitting
11:00 AM - Collision, B will see Collision
12:00 AM - A will see Collision.

Here

Collision detection Condition : $T_t \geq 2T_p$

$L \geq 2T_p B$

Efficiency of CSMA/CD:

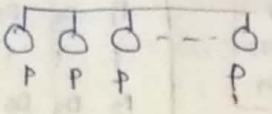


$\eta = \frac{\text{useful time}}{\text{cycle time}} = \frac{T_t}{C*2T_p + T_t + T_p}$

C = num of collision slots (not known)

To find C:

① n stations send data with prob p



② Probability for success (when only one station transmit the data)

$P_{succ} = {}^nC_1 \cdot p(1-p)^{n-1}$

For max success

$\frac{dP_{succ}}{dp} = 0$

↓ solve

$P = \frac{1}{n} \Rightarrow n = \frac{1}{p}$

$\therefore P_{max} = n \cdot \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \rightarrow \left(1-\frac{1}{n}\right)^{n-1}$

$P_{max} = \left(1-\frac{1}{n}\right)^{n-1}$

when num of stations are not known $n \to \infty$

Apply limit

$\lim_{n\to\infty} P_{max} = \lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n-1}$

$\lim_{n\to\infty} P_{max} = \frac{1}{e}$

③ According to poisson distribution

How many times we should send before getting first success $= \frac{1}{P_{max}} = e$

∴ Num of Collisions = e

$\text{Efficiency}(\eta) = \frac{T_t}{e*2T_p + T_t + T_p}$

$\eta = \frac{1}{1+6.44a}$

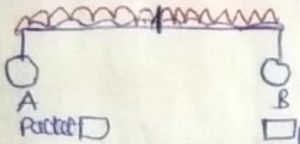
Analysis:

$\eta = \frac{1}{1+6.44\left(\frac{d}{b}\right)\left(\frac{B}{L}\right)}$

d↑ η↓ LAN ✓ WAN ✗

L↑ η↑

Back off Algo for CSMA/CD:



Coll num (n): 1 1

Choose randomly from $(0, 2^n - 1)$ for A & B

(0,1)

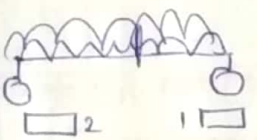
A	B	
0	0	→ Collision
0	1	→ waiting time of A = 0 * Tslot, waiting time of B = 1 * Tslot
1	0	
1	1	→ Collision.

Prob A wins = $\frac{1}{4}$

B wins = $\frac{1}{4}$

Collision = $\frac{1}{2}$

→ Tslot is set in such a way that if B station sends data then it senses that carrier is busy.



Coll num: 1 (0,1) 2 (0,1,2,3)

A	B	o/p
0	0	Coll
0	1	A
0	2	A
0	3	A
1	0	B
1	1	Coll
1	2	A
1	3	A

Prob A wins = $\frac{5}{8}$

B wins = $\frac{1}{8}$

Collision = $\frac{2}{8}$

Advantages

→ It gives waiting time

$WT = K * T_{slot}$

$K \in (0, 2^{\text{Collision num}} - 1)$

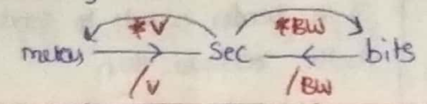
∴

→ Capture effect (If a station wins a collision 1 time then it's prob of winning increases exponentially)

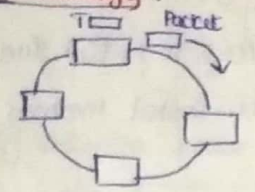
→ Applicable only to two stations : Binary Exponential Backoff Algo

Token passing Access Control : (Ring Topology)

Note : Delay may be given in sec, meters, bits.

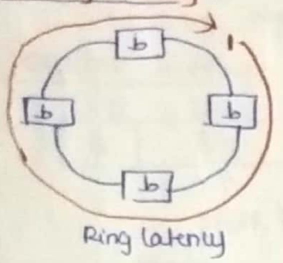


Terminology :



⇒ unidirectional
⇒ only one token ie one station can send data
∴ No collision occurs.

(1) Ring latency :



$$\text{Ring latency} = \frac{d}{v}\,(\text{sec}) + N*b\,(\text{bits})$$

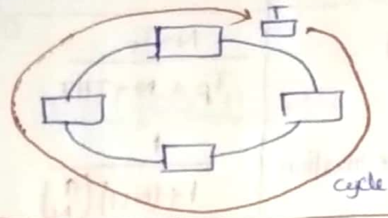
$$RL = T_p + \frac{N*b}{BW}\ \text{sec}$$

(or)

$$RL = T_p * BW + N*b\ \text{bits}$$

(2) Cycle time :

Token holding time : Time for which a station holds the token.



$$\text{Cycle time} = \frac{d}{v} + N*THT \Rightarrow T_p + N*THT$$

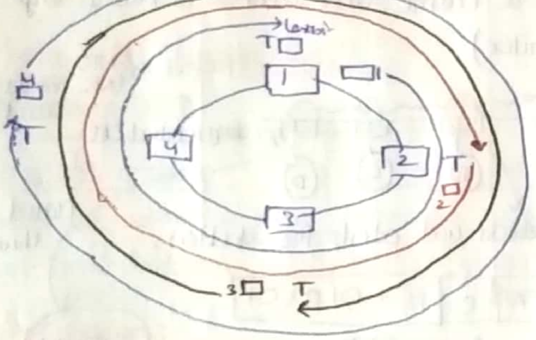
$$\text{Efficiency} = \frac{\text{Usefultime}}{\text{Cycletime}} = \frac{N*T_t}{T_p + N*THT}$$

Finding THT :

– Token passing depends on THT

Delayed Token reinsertion | (default) Early token reinsertion

Delayed Token reinsertion

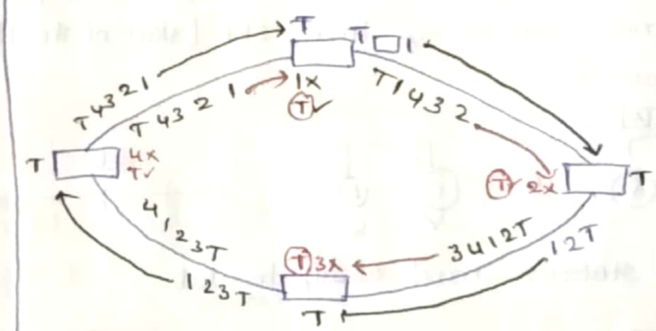


$$\therefore THT = T_t + T_p$$

$$\eta = \frac{1}{1+(N+1)\left(\frac{a}{N}\right)}$$

→ only one packet in Entire time
→ more reliable

Early token reinsertion (default)



$$THT = T_t$$

$$\eta = \frac{1}{1+\left(\frac{a}{N}\right)}$$

→ There can be many packets in Entire time
→ less reliable

ALOHA (Not in Gate)

→ Any station could transmit data at any time
→ No carrier sensing
→ Collisions are possible
→ No need of collision detection because of ACK
→ Every station pickups some random amount of time for retransmissons.
Back off time.

G = num of stations want to transmit data in T_t time

ALOHA
- Pure Aloha
- Slotted Aloha

Pure Aloha :

$$\eta = G*e^{-2G}$$

$$\eta_{max} = 18.4\%$$

Slotted Aloha

$$\eta = G*e^{-G}$$

$$\eta_{max} = 36.78\%$$

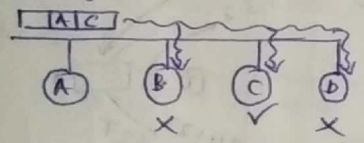
Points to remember:

Access Control methods	Efficiency	Acknowledgement
① Time division method	$\frac{1}{1+a}$	–
② polling	$\frac{T_t}{T_{poll}+T_t+T_p}$	–
③ CSMA/CD	$\frac{1}{1+6.44a}$	×
④ Token passing	$\frac{N*T_t}{T_p+N*THT}$	×
a) Delayed Token reinsertion	$\frac{1}{1+(N+1)(\frac{a}{N})}$	
b) Early token reinsertion	$\frac{1}{1+\frac{a}{N}}$	
⑤ ALOHA		
a) Pure ALOHA	$G*e^{-2G}$; max 18.4%	✓
b) slotted ALOHA	$G*e^{-G}$; 36.78%	

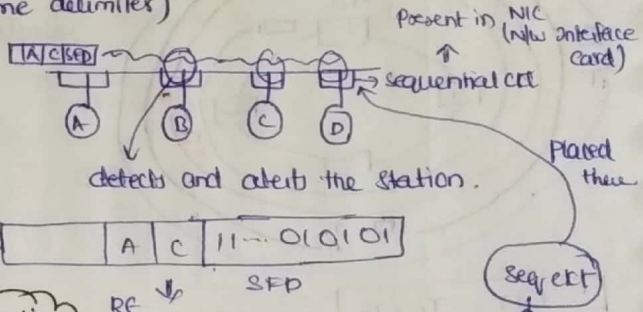
Note :-
If a sender wants to send data to the receiver then
(i) Apply Access Control method
↓
Get Access to the link then
(ii) Apply Flow Control methods

a) Framing (Bitstuffing)
- Taking the data from network layer and putting it in a frame and send it to below layer.
- Framing will always have SFD (start of the frame delimiter)

Generally



Sol



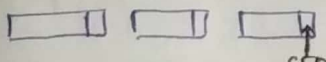
∵ All stations should always be alert

RE
(1010---11) $(1+0)^*$ → NFA → DFA

Framing

Like paging

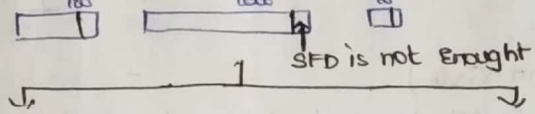
Fixed length framing :
→ length of msg is fixed
Eg:- L=1000B



∵ suffers from IF
If L=10B then 990B has to be padded

Like segmentation

Variable length framing :
→ length of msg is var



Length field
→ used in Ethernet
∵ If len field is corrupted then reciever reads wrong data
can't be rectified using CRC because len field is present before CRC
↓
Error probability is less in Ethernet. So, we use length field

End of frame delimiter
→ used in Token ring
∵ If End of frame delimiter pattern matches with the data then reciever reads wrongly

Implementation
→ character stuffing (Not used)
→ Bit stuffing (Present)

Bit stuffing: (done bcoz of data contains the pattern like EFD

At sender:

If EFD pattern is 011111 then

Data: 011110 → 0111100
011111 → 0111101

(stuff 0)

At recieved

Data: 0111110 ⇒ 011110
0111111 → 011111

Q Delimiter pattern 01111, Data: 0111000111110

O/P: 0111000011110 (sender) → N/w → 0111000011110 ⇒ Data

Character stuffing:

EFD | Data | SFD

$ | $/o → stuff /o ⇒ $/o (sender) → N/w → $ | $/o ✓

If data it self contain /o

$ | $/o $ /o /o → N/w → $ | $/o /o ✓

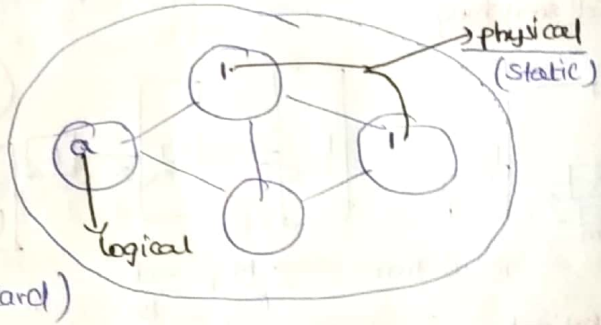
$/o/o/o

e) physical addressing:

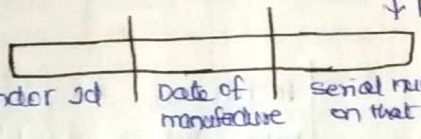
Addresses

Physical address:
→ unique within a N/w
Ex:- MAC Addrs (48)

Logical Address
→ unique within Entire world
Ex:- Ip (32) ⇒ s/w Address present in memory



MAC Address:
– H/w Address printed in NIC (Network Interface card)
↓ present in ROM



⇒ It is Globally unique but used as physical address bcoz IP [NID | HID] is more convinient to route packets

⇒ **Note:**
The Network called Appletalk doesn't use MAC Id as physical address ∴ It generates a random id for physical address

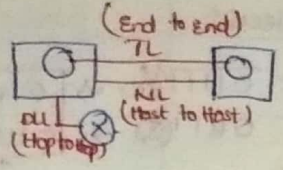
Note:

Data link layer
- logical link Control → Flow control, Error Control
- Medium Access Control → Framing, Access control, Physical Addressing

Network layer:

Responsibilities

1. Host to Host Connectivity
2. logical Addressing
3. Switching (Using Routing table)
4. Routing (Building Routing table)
5. Congestion Control × (Not in syllabus)
   Ex:- AIM
6. Fragmentation



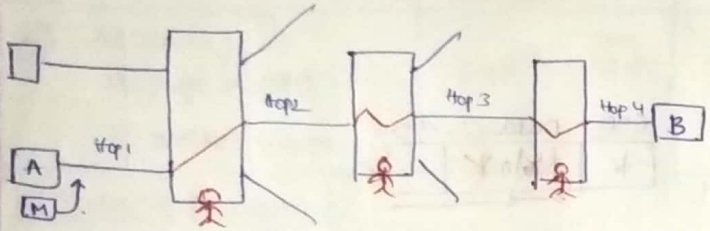
Protocols

- ICMP
- ARP, RARP, BootP, DHCP
- Routing protocols
  - Interior Gateway
    - RIP
    - OSPF
  - Exterior Gateway
    - BGP

a) Switching:

- Circuit switching (Telephone Networks)
- Packet switching
  - Virtual circuits
  - Datagrams
- message switching

Circuit switching:



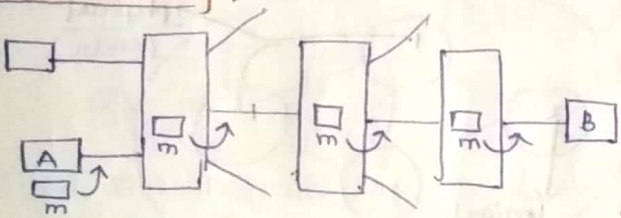
Message - M
Bandwidth - B
distance - d
Num of Hops - x
Velocity - v

$$\text{Total time} = \text{setuptime} + \frac{M}{B} + \frac{xd}{v} + \text{Teardown time}$$

It is applied at physical layer

(i) No Headers read
(ii) No reordering

Packet switching:



Store & Forward

(i) Headers read
(ii) re ordering done

$$\text{Total time} = \frac{xM}{B} + \frac{xd}{v}$$

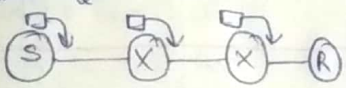
(CS) Circuit switching	(PS) Packet switching
Extra time: setup + Teardown	$(x-1)\frac{M}{B}$

Bursty data – CS is best
Small data – packet switching is best

Packetization in packet switching

Pipelining is applied

Generally
Data = 1000B
BW = 1MBps
Header = 100B
Packet size = 1100B

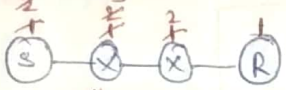


Assume $T_p = 0$.

$$TT = 3 * T_t \Rightarrow 3 * \frac{1100}{10^6}$$
$$\Rightarrow 3.3 \text{ msec}$$

Packets = 5
Data = $\frac{1000}{5} = 200B$
Header = 100B
PS = 200 + 100 = 300B

$$T_t = \frac{300}{10^6} \Rightarrow 0.3\text{ms}$$

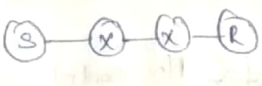


1st packet — rem 4 packets

$$TT = 3 * T_t + 4 * \bar{T}_t$$
$$\Rightarrow 2.1 \text{ msec}$$

Packets = 10
Data = $\frac{1000}{10} = 100B$.
Header = 100B
ps = 100 + 100 → 200B

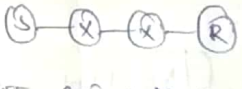
$$T_t = \frac{200}{10^6} \Rightarrow 0.2\text{ms}$$



$$TT = 3 * T_t + 9 * \bar{T}_t$$
$$\Rightarrow 2.4 \text{ msec}$$

Packets = 20
Data = $\frac{1000}{20} = 50B$
Header = 100B
PS = 150B

$$T_t = \frac{150}{10^6} \Rightarrow 0.15\text{ms}$$



$$TT = 3 * T_t + 19 * \bar{T}_t$$
$$\Rightarrow 3.3 \text{ msec}$$

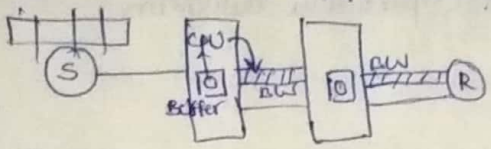
∴ Packetization should be done optimally

Packet switching

- virtual circuits
- Datagrams

Virtual circuits (Ex:- ATM)

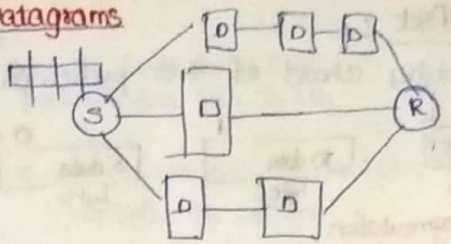
- Reservations of Buffer, CPU, BW are made



- phone call
- Packet - 1 ⇒ Global header
  Other ⇒ local header
- Connection oriented (we do some reservations)
- Same path is followed ⇒ inorder
- more reliable
- Costly

Datagrams (Ex: IP)

Internet protocol

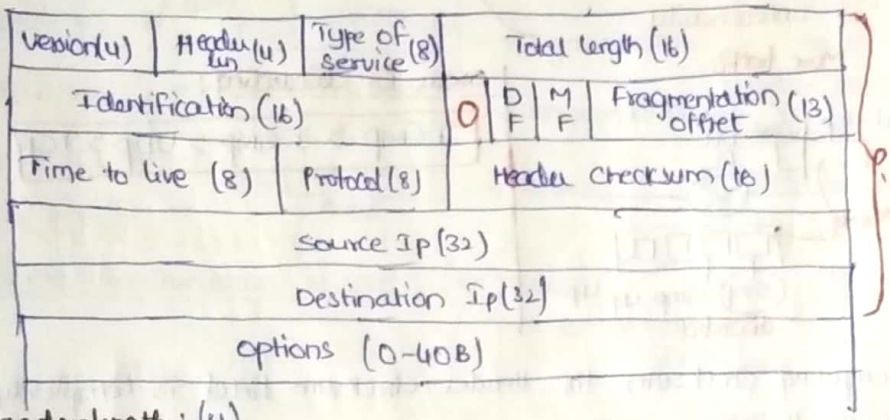


- Email
- packet (Every) - Headers read
- Connection less (No reservations)
- ⇒ may follow different paths ⇒ out of order is possible
- not reliable
- Cost efficient

b)

Internet protocol :

IPV4 Header :



DF - Do not fragment
MF - More fragment

Header len Freq

Min HL = 20B	5
Max HL = 60B	15

Header length : (4)

⇒ Here Header length (4 bits) ⇒ 1111 ← max value
⇓
15

If Header length = 60B then we perform $\frac{60}{4} = 15 \Rightarrow 1111$ (HL)

Scaling factor.

Header length	Header len field
20B	5
28B	7
60B	15
~~30B~~ 32B	~~7~~ 8

Padding

Borrowed from options

⇒ max num of Bytes padded = 3

Version : (4)

Parsed by

IpV1	0001 → V_1 parser	outdated
IpV2	0010 → V_2 parser	outdated
IpV3	0011 → V3 parser	outdated
IpV4	0100 → V4 parser	present
IpV5	0101 → V_5 parser	limited functionalities
IpV6	0110 → V6 parser	Advanced
		future use.

Total length (16)

Header + data

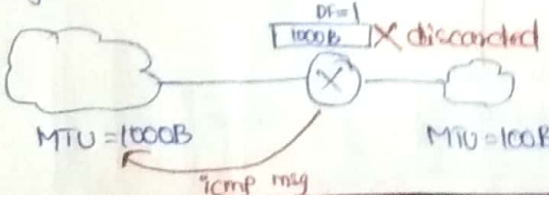
Max length of packet = $2^{16} - 1$ ⇒ 65,535 B

Identification : (16)

→ The datagram/packet coming out of host will get a unique identification num
→ Used for reordering of data packets
→ Also used in fragmentations, The fragments of a packet will have same identification num

DF (Dont fragment)

0 → Fragmentation can be done
1 → No fragmentation



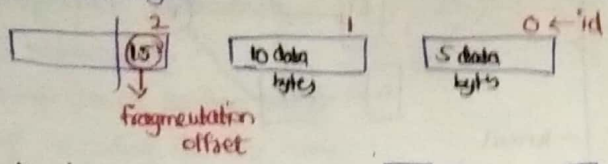
MF (more fragments)

0 → End of fragments
1 → More fragments are following

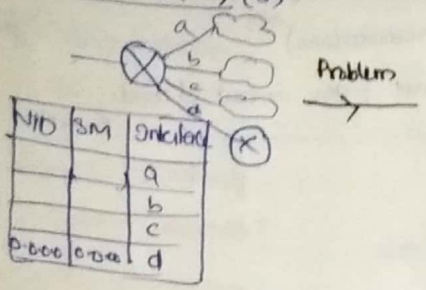
3) Fragmentation offset :

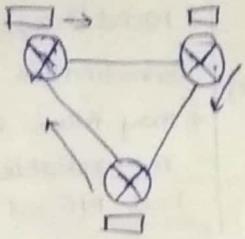
- Num of data bytes ahead of this particular fragment in this particular datagram

Ex:



Time to live (TTL) (8)

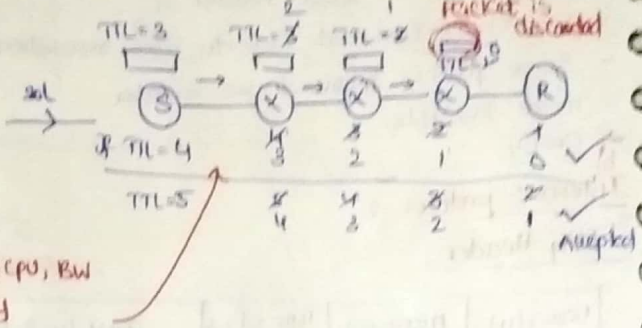




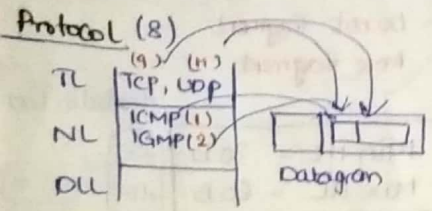
Infinite loop

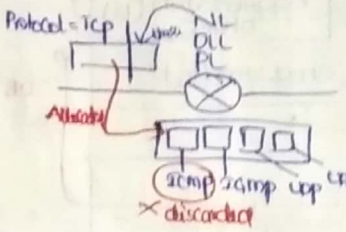
→ ∴ uses Buffers, CPU, BW unnecessarily

More traffic



Protocol (8)

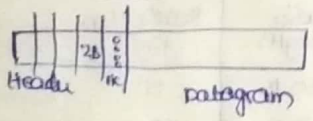




Order for discarding :

ICMP > IGMP > UDP > TCP

Header checksum (HC) (16)



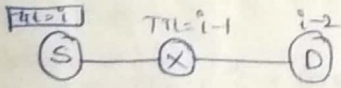
While Computing checksum the Header checksum field is completely Filled with 16 zeros

*) How to Compute Checksum

- Divide the Header into 2B (Generally the Header length will be in multiples of 4. So odd num Header lengths not possible)
- Convert it to decimal
- Add them
- Convert it to 1's Comp and place it in Header

*) Why to calculate checksum only to header :

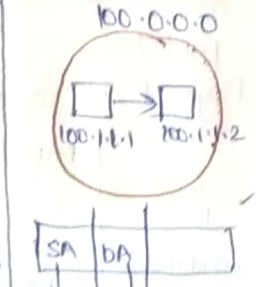
- because for every hop the fields like (i) TTL ← Always change
  (ii) FO, MF, TL
  (iii) options
  (iv) HL
  } may / may not change → So for every time calculating checksum for data and header is d/a so we perform HC



Source Ip and destination Ip :

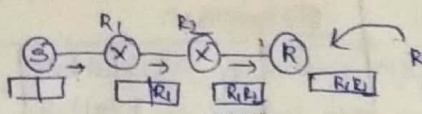
NID	HID	Result
✓	✓	valid Ip
✓	0's	NID
✓	1's	DBCA
1's	1's	LBCA
1's	0's	~~Host~~ Network mask
0's	✓	Host within a N/w
0's	0's	"I don't know Ip"
127	✓	"loopback address"

Host within a N/w



SIP	DIP	Results
✓	✓	Valid Ip, Host within N/w
✓	×	I don't know Ip
×	✓	DBCA, LBCA, loopback add
×	×	NID, N/w mask

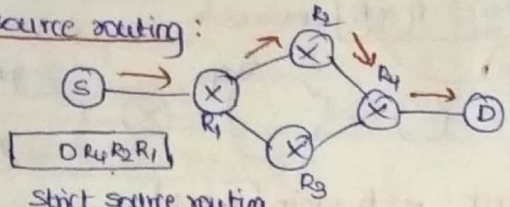
Options : (0-40B)

① Record route → 

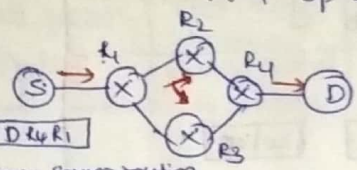
Reciever understands that packet come from R_1 & R_2

Max num of Ip addr recorded = $\frac{40B}{4B}$ → 10 (options) ⇒ But in practice some bits are wasted to indicate the option
(Ip)
∴ 9 Ip addresses are recorded

→ Source routing :



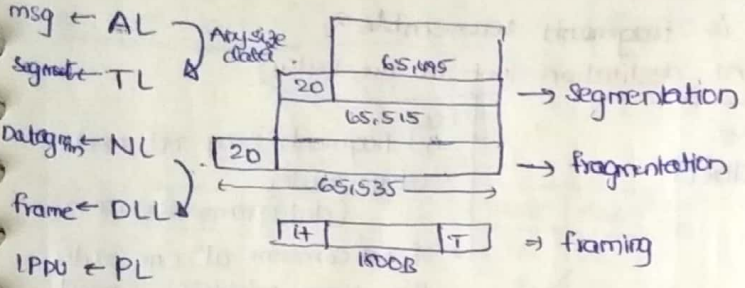
Strict source routing



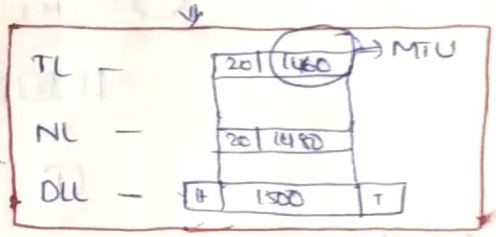
loose source routing

→ Padding : To make Ip header as multiple of 4.

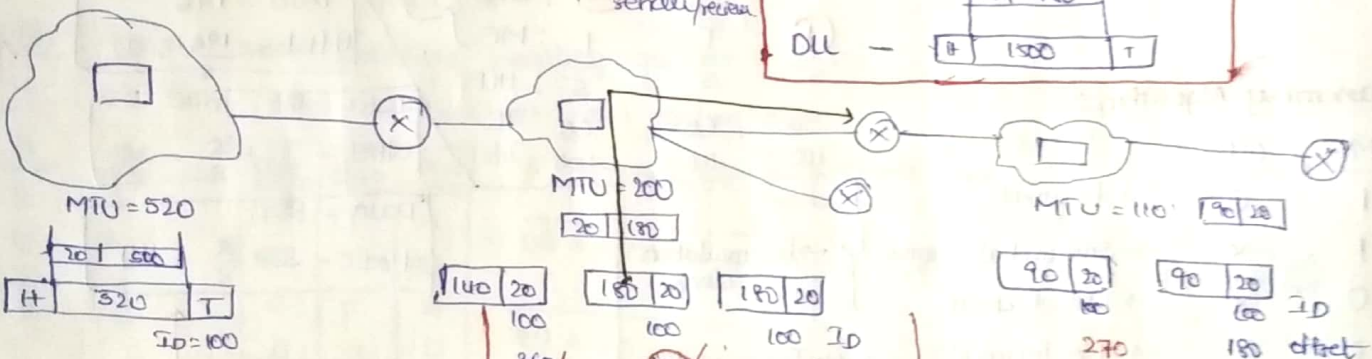
c) Fragmentation :



*) Doing segmentation and then fragmentation is a o/H
so At TL it intelligently calculates the Bottle neck ie MTU by $\min(MTU_{NL}, MTU_{DLL})$ and do segmentation. so that Fragmentation can be avoided at NL of sender.

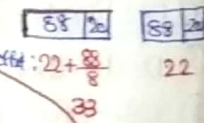


Where fragmentation is used : (At router)
Not at the (Host) sender/reciever

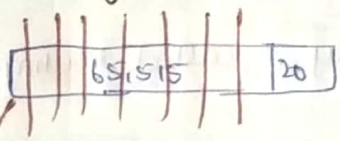


Num of fragments = 4 (0, 180, 270, 360)

How to assemble → Sorting ⇒ Some fragments may miss (Prob)
→ Take one fragment
Count data bytes
How?? (Prob) ADD fragmentation offset & DB ⇒ Next fragment offset
In Ip → TL − HL = data bytes



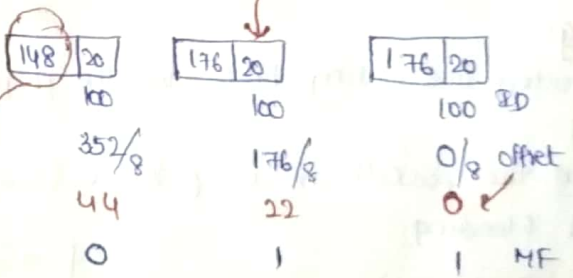
Worst case fragmentation offset :



FO = 65,514 ≅ 2^{16} → 16 bits reqd but FO is 13 bits

↓sol

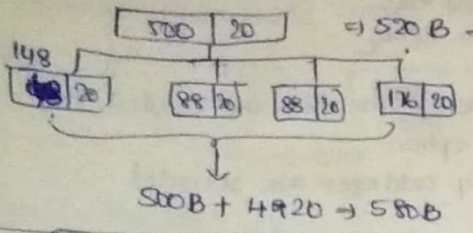
Scaling factor = $\frac{2^{16}}{2^{13}} = 8$ → Divide Every FO with 8



⇒ So data bytes that are sending in fragments should be multiples of 8

The last fragment need not be a multiple of 8.

Fragmentation overhead:



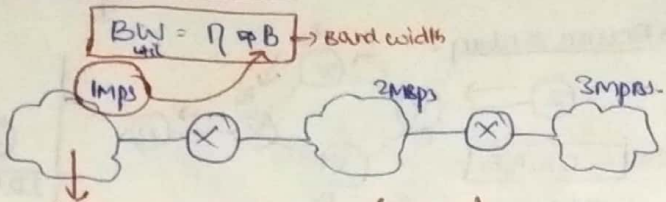
O/H = 580 − 520 = 60B

Efficiency:

$$\eta = \frac{\text{Useful bits}}{\text{Total bits}} = \frac{500}{500+4920} \Rightarrow \frac{500}{580} \Rightarrow 86.2\%$$

Bandwidth utilization:

$BW_{util} = \eta * B$ → Bandwidth



$B = \min(1, 2, 3)$

B = 1 MBps

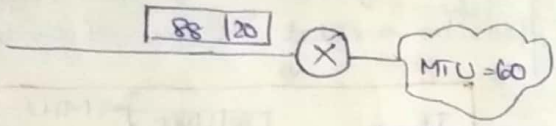
Q

	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

What info is needed by router? → 3 tuple: MF, offset, ID

① where is fragmentation done?
- At Routers, not at the source

② where is fragments reassemble?
- At destination (not at the router)
  - bcoz → All fragments may not meet at the router (datagram service)
  - If we combine at one router then there might be a need for further fragmentations



8 | 20	40 | 20	40 | 20	
32	27	22	: offset
1	1	1	: MF
5	5	5	: HLF
28	60	60	TL
100	100	100	ID

Data = 176
offset = 196/8 ⇒ 22

Data = 88
offset = 22 + 88/8 = 33

Data = 88
offset = 33 + 88/8 = 44

Reassembly Algorithm:

MF	offset	
1	0	→ fragment ①
1	X	→ intermediate fragment → fragmentation done
0	X	→ last fragment → fragmentation done
0	0	→ No fragment → No fragmentation

Algorithm:

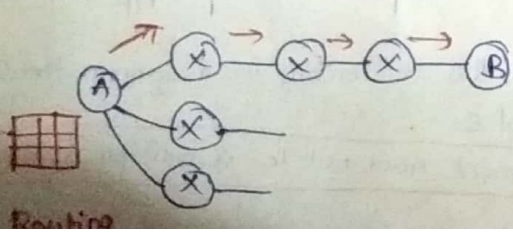
① Destination should identify that datagram is fragmented (MF, offset)
② Destination should identify all fragments belonging to the datagram (ID)
③ Identify first fragment (offset = 0)
④ Identify subsequent fragments (TL, HL, offset)
⑤ Repeat step ④ until MF = 0

d) Routing:

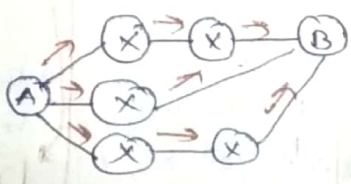
- Constructing the routing table for the purpose of switching between is called Routing

Flooding:

Sending the packets in all possible directions then definetly 1 packet reach the destination is called Flooding



Routing



Flooding

Routing	Flooding
- Routing Algo reqd	- No routing Algo
- Shortest path may/maynot be guranteed	- shortest path is always guranteed
- less reliability	- More reliable
- No duplicates	- Duplicate packets
- less Traffic	- More Traffic

Routing Algo

- Static (Manual) → ⇒ very difficult to manage
- Dynamic (depends on Topology & Traffic)
  - distance vector routing
  - Link state routing

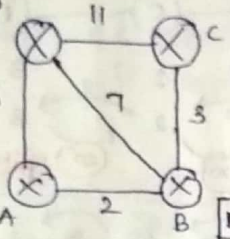
Distance vector routing :

Dest	dist	Hop
A	1	A
B	7	B
C	11	C
D	0	D

D

Dest	dist	Hop
A	∞	—
B	3	B
C	0	C
D	11	D

C



Dst	dist	Hop
A	0	A
B	2	B
C	∞	—
D	1	D

Dest	dist	Next hop
A	2	A
B	0	B
C	3	C
D	7	D

Constructed based on local neighbours Info

step ① : Construct local routing table based on Info present in neighbours.

step ② ; send distance vectors to the neighbour routers at timestamps

step ③ : At A :
Distance vector from B, D

At B
DV from A, C, D

At C
DV from B, D

At D
DV from A, B, C

step ④ : Do the same procedure parallely at Each router

At A : DV from B, D → Construct new routing table.

B	D
2	1
0	7
3	11
7	0

A

Dest	dist	Next hop
A	0	A
B	2	B
C	5	B
D	1	D

(i) $A \to B = \min\{ A \to D + D \rightsquigarrow B,\ A \to B' + B \rightsquigarrow B \}$ (B)
(1+7) (2+0)

(ii) $A \to C = \min\{ A \to D + D \rightsquigarrow C,\ A \to B + B \rightsquigarrow C \}$ (B)
(1+11) (2+3)

(iii) $A \to D = \min\{ A \to D + D \rightsquigarrow D,\ A \to B + B \rightsquigarrow D \}$ (D)
(1+0) (2+7)

Next

Method (I)

Method II (short cut)

At B : DV from A, C, D

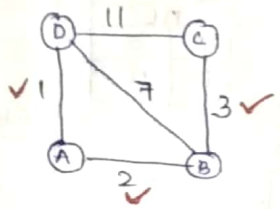
A	C	D
0	∞	1
2	3	7
∞	0	11
1	11	0
BA→2	3←BC	7←BD

⇒

B

Dest	dist	Hop
A	2	A
B	0	B
C	3	C
D	3	A

The edges that are not used after construction of routing table.

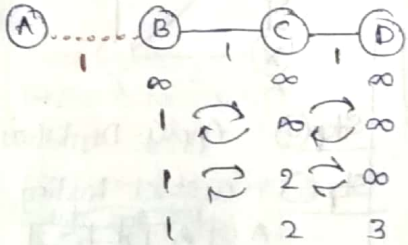


(11), (7) Edges are not used

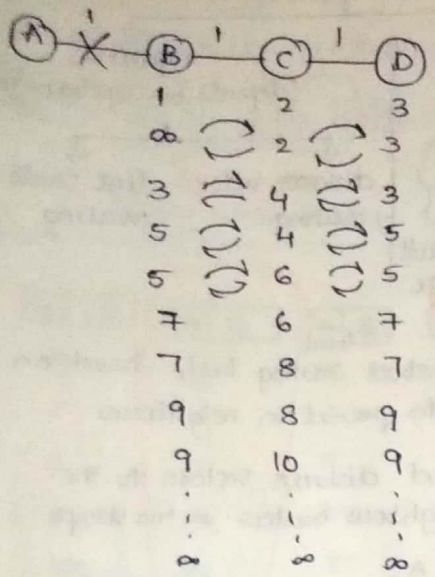
Problem with DVR (Count to Infinity problem)

- Bad news spread slow
- Good news spread fast

From A



Good news on link creation



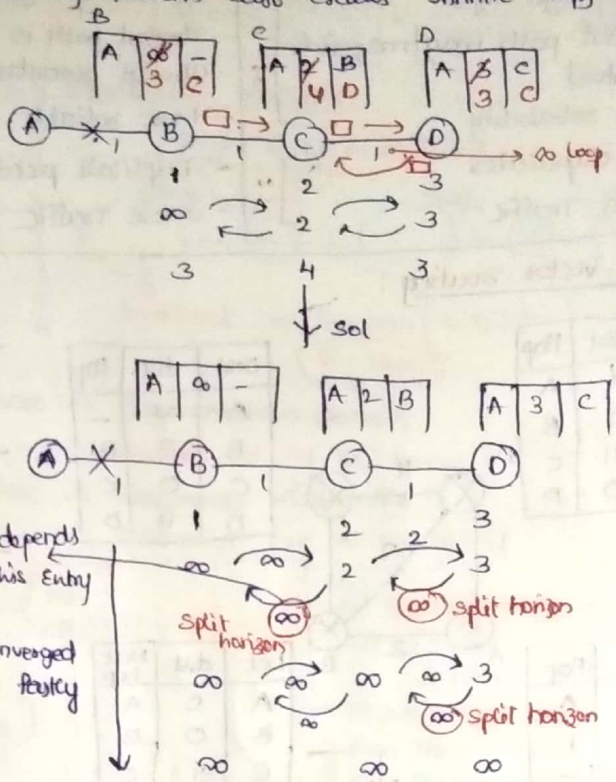
Count to Infinity prob.
(Bad news on link down)

↓

If Next hop also passed along with distance vectors then Count to ∞ prob can be avoided

Split horizon

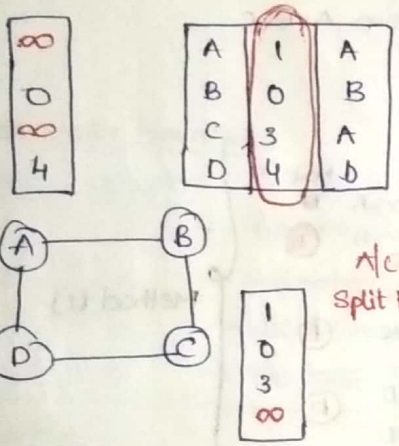
- Count to infinity problem also creates Infinite loops



bcoz 'c' depends on 'B' on this Entry

Converged stage

*)



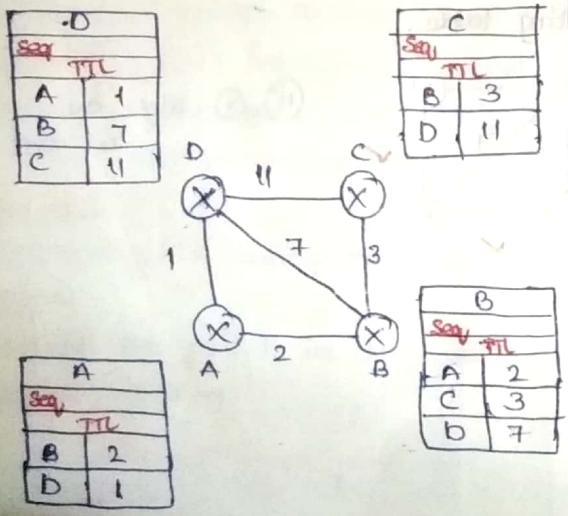
A/c to Split Horizon

DVR

- → Counto to Infinity problem
- → Convergence slow
- → Infinite loops

→ solved by split horizon

Link state routing: (superior to DVR)

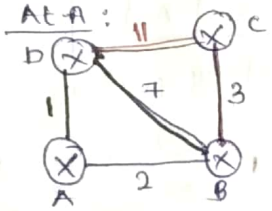


Step ① : Construct link state table based on Info present at the neighbour routers

Step ② : Flood all link state tables

DVR - local knowledge
LSR - Global knowledge

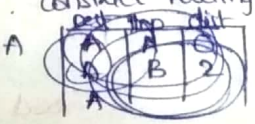
Step ③ : Construct topology of the N/w at every router



B ← Blue
C ← Red
D ← Black.

Step ④ : Apply Dijikstra's Single Source Shortest path Algo

Step ⑤ : Construct Routing table.



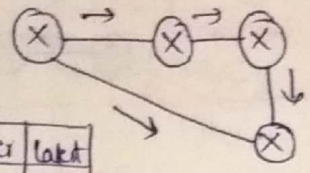
(A)

dest	dist	hop
A	0	A
B	2	B
C	5	B
D	1	D

LSR:
- Converges faster

Problems in LSR:
- More traffic due to flooding
  - ↓sol
  - Seq. number
  - ↓Prob
  - may fall into a loop
  - ↓sol
  - Time to live

Explicit problems



At A	
Router	Latest
B	~~10~~ 15
C	20
D	30

(B,8) ✗
(B,15) ✓
(B,11) ✗
(B,15) ✗

After 10 mins B packet is discarded and accept latest one.

Route	Latest	TTL
B	15 / 31	10 min

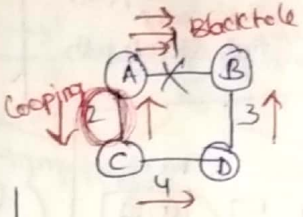
16 17 18 ⋮ 31 → All are discarded

0008 1111 Corrupted 31 ⇒ 31

→ 1 more prob.

Even though they are latest bcoz of Error
- ↓sol
- Lifetime valid

Implicit problem:
Transient problem → (i) Black hole
(ii) looping
- Temporary

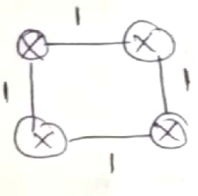


Differences between DVR & LSR:

DVR	LSR
- 1980's	- 1990's
- Bw req is less	- Bw req is high
- local knowledge	- Global knowledge
- Kind of Bellman ford	- Dijikstra's Algo
- less Traffic	- More traffic
- Periodic updates	- periodic updates
- Converges slowly	- Converges faster
- Count to ∞	- No Count to ∞
- persistent looping	- Transient looping
- RIP protocol	- OSPF protocol

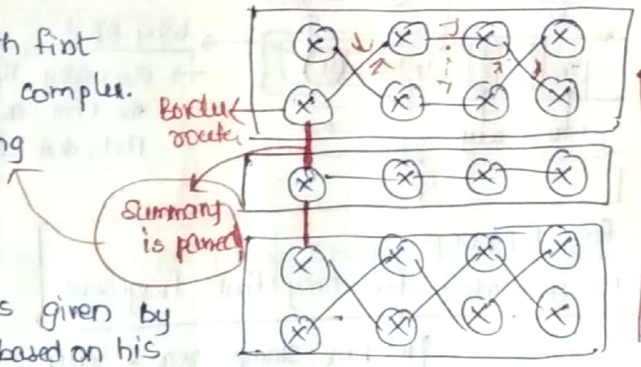
RIP
- Routing Information protocol
- practical implementation of DVR

Metrics:



Hopcount
∞ is rep by '16'.
- Same as DVR (computationally simple)

OSPF:
- open shortest path first
- computationally complex.
- reduces flooding



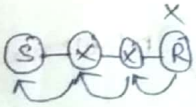
- Divide into Areas/regions.

Note:
RIP uses UDP for transport with Port num 520
OSPF Encapsulates data directly into IP packets
So it doesn't use TCP/UDP

Metrics:
Edge weight is given by System Admin based on his requirement.

ICMP Protocol:
- Internet Control message protocol
- present at Nlw layer

ICMP
- Error handling or feedback messaging
  - TTL Exceed
  - parameter problem
  - Source quench
  - Source redirect
  - Destination Unreachable
- Req & Reply
  - Echo req & reply
  - Timestamp req & reply
  - Nlw mask req & reply
  - Router solicitation and advertisement.



- one direction

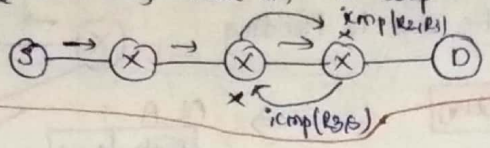
- Both direction

Note :

Whenever Ip packet is discarded Icmp packet is generated and whenever an Icmp packet is discarded No Icmp packet is generated. bcoz it may fall in oo loop

Ip x → Icmp

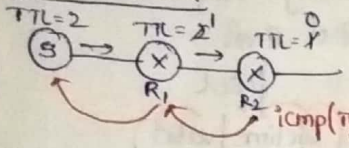
Icmp x → No



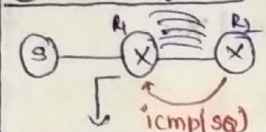
- Ip ⇒ unreliable
- Ip + Icmp → unreliable

Icmp Error handling or feedback messaging :

① TTL Exceed :



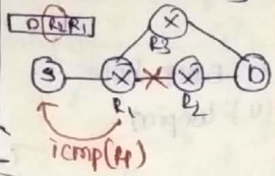
② Source quench :



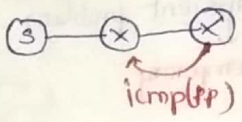
Slows down / finds alternate path

③ Parameter problem :

Strict Source routing

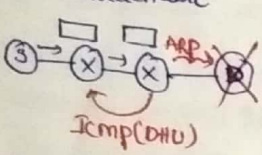


Header fields Corrupted → Not valid checksum

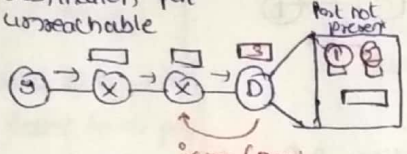


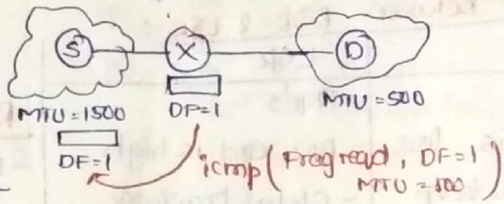
④ Destination unreachable :

Destination Host unreachable

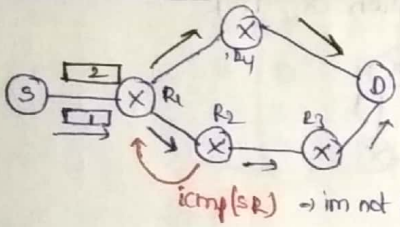


Destination port unreachable



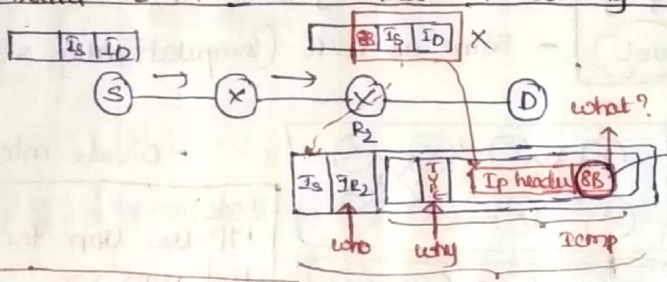


⑤ Source redirect



What things should be known to Sender on recieving icmp packet?

1) Who?
2) Why?
3) What?



Why SB?

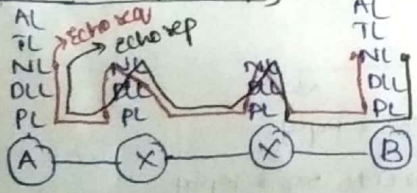
→ Generally TCP packets are sent the First SB are generally source Port, destination port, seq num..

Note :

- Icmp packets are generated for TCP, UDP
- In case of fragments Icmp is generated for only first fragment

Icmp Req and Reply :

(i) Echo req & reply :



→ used to find out whether station is present or not

→ PING (Packet Internet groper)

→ Not client level packet

→ Denial of service attack possible

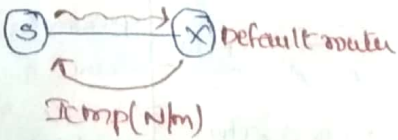
(ii) Time stamp req & reply :

→ used for synchronize clocks in two stations.

→ Highly unreliable

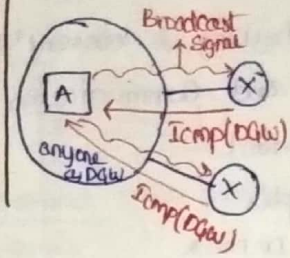
→ presently not in use

(iii) Network mask req & reply

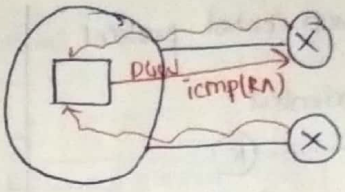


To have Internet Connection
1. Ip req'd - DHCP
2. Default gateway (DGW) - Route solicitation
3. N/w mask - N/w mask req & reply

Route solicitation :



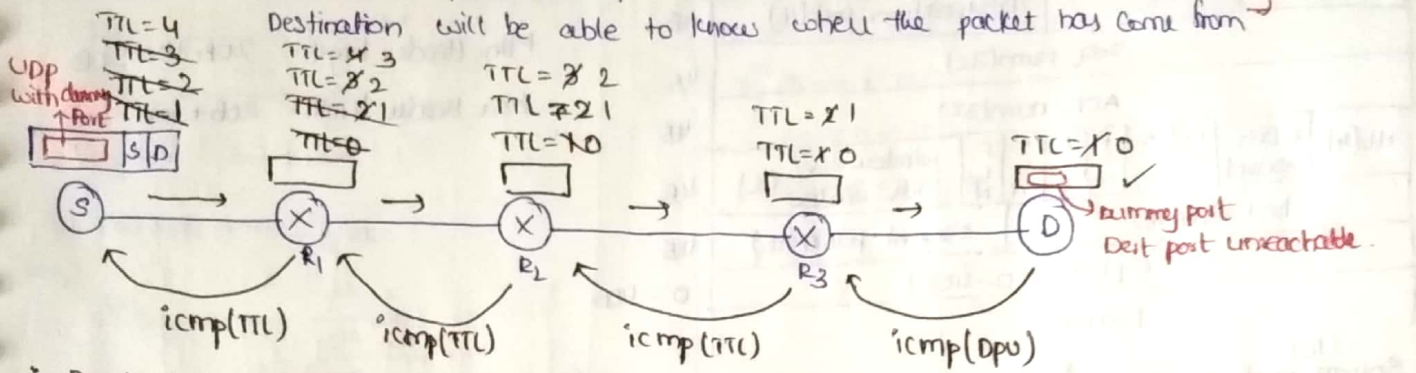
Routers Advertisement



Trace route Application of ICmp

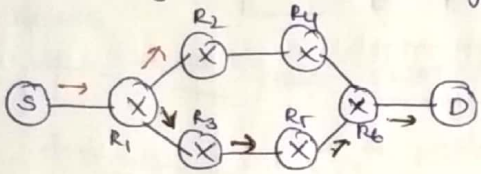
Trace route : sender finds the path in which the packet is going
Record route : present in options of Ip
} differences

Destination will be able to know when the packet has come from



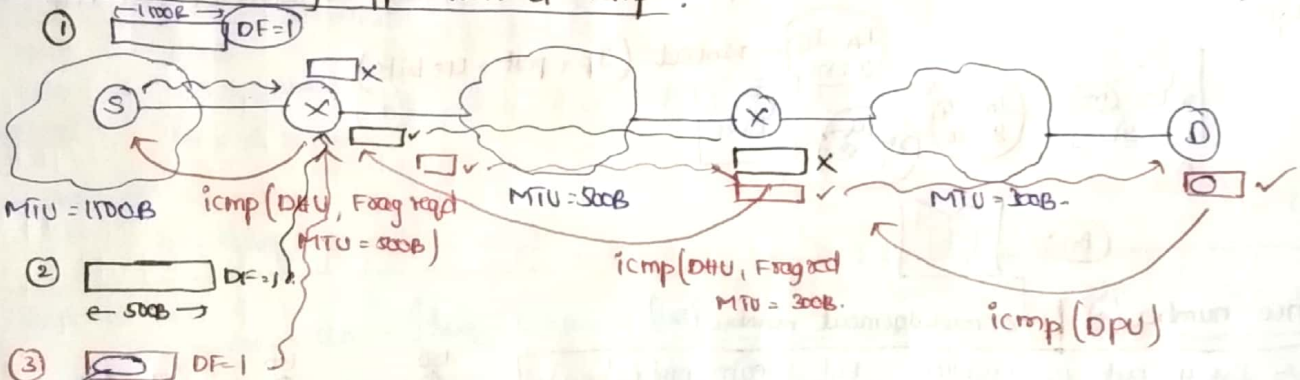
∴ Route : R_1 R_2 R_3 D

→ It is not guaranteed that always same path is ensured ie



Route : R_1, R_2, R_5, R_6, D → but occurence is rare.
No such path bcoz of TTL

Path MTU discovery Application of icmp :



∴ MTU = 300B for this path.

Problem :-
- path may change so that MTU may change

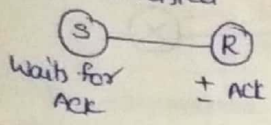
4) Transport layer :

Responsibilities :
- End to End Connectivity (service point Addressing)
- Flow Control (SR) ×
- Error Control (Checksum) ×
- Segmentation
- Mul & demux
- Congestion Control

Protocols
- TCP
- UDP

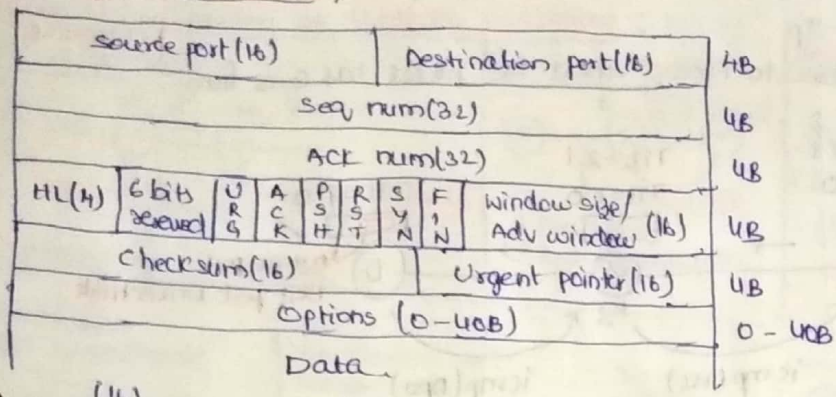
TCP (Byte stream protocol) ⇒ Every byte is counted

- Transmission Control protocol
- reliable
- Conn oriented



- Error checking & recovery mech
- End to End Communication
- Flow Control
- full duplex
- Mux & De mux

Tcp segment structure:

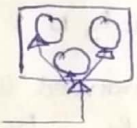


Min Header length = 20B + 0 = 20B

Max Header length = 20B + 40B = 60B

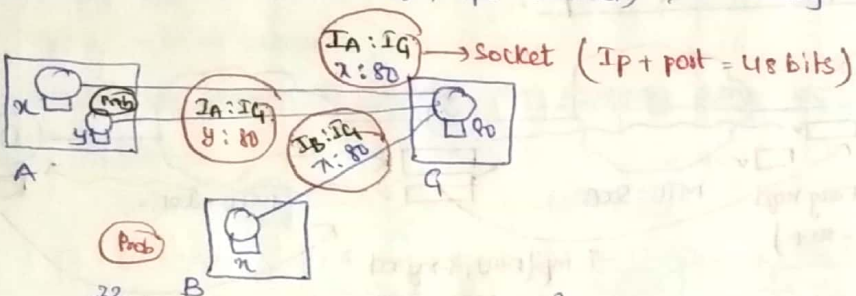
① Source port (16) & Destination port (16): $[0, 2^{16}-1)$, $(0, 65,535)$

- port nums are mainly used for multiplexing and demultiplexing



HTTP - 80	0 — 1023 — Well known
FTP - 21	1024 — 49,151 — Reserved
Telnet - 23	49152 — 65,535 — General public
SMTP - 25	

- Tcp is Connection oriented (Buffers, ptrs, CPU reserved). To identify the connection we use sockets

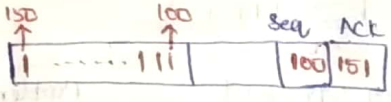


② Sequence number (32) | Acknowledgement number (32)

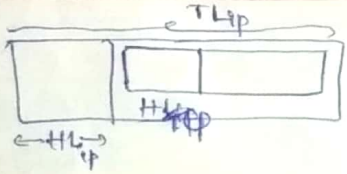
Tcp - Every byte is Counted - Byte stream protocol

Ip - Every packet is Counted - packet stream protocol

DLL - HDLC - Every bit is Counted - bit stream protocol



*)



$TL_{TCP} = TL_{ip} - HL_{ip}$

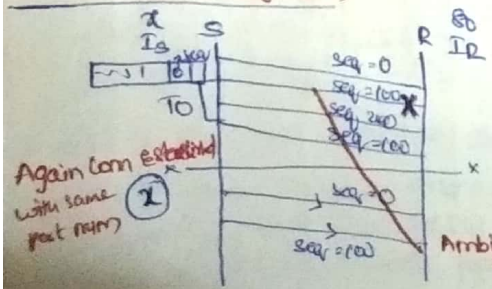
$\text{last Data byte} = TL_{TCP} - HL_{TCP}$

Ack = last byte + 1

To identify a segment uniquely in the N/w

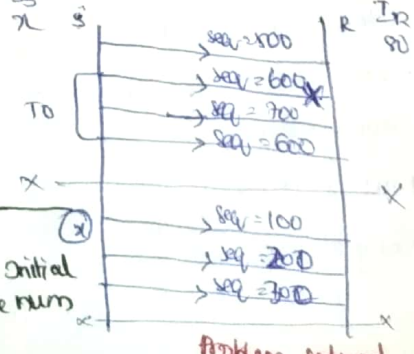
Ip + host + seq num

Problem with seq num



The packet with seq = 100 of the prev connection enters into present Conn if we always start sequencing the segments with zero

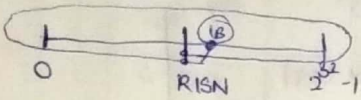
Sol → Choose Random initial sequence num



Wrap around problem :

Num of seq num's = 2^{32} = 4G seq → 4GB can be transfered.

If data = 4GB + 1B



Wrap around time (WAT)

*) The time for which this wrap around takes place is called **wrap around time** taken to finish all seq num's

- Wrap around time depends on bandwidth

WAT for 2^{32} Bytes of BW = 1MBps

1sec → 1MB

2^{20} B → 1sec

2^{20} seq num → 1sec

1 seq num → $\frac{1}{2^{20}}$ sec

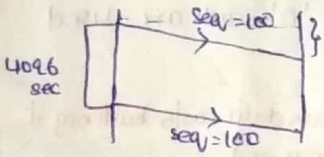
Now,

2^{32} seq num → $\frac{2^{32}}{2^{20}}$ sec

→ 4096 sec

WAT

Generally on internet the packet will alive for 180 sec



WAT > lifetime

↓

No problem with wrap around

WAT for 2^{32} B for BW = 1GBps :

1sec → 1GB

2^{30} B → 1sec

2^{30} seq num → 1sec

1 seq num → $\frac{1}{2^{30}}$ sec

2^{32} seq num → $\frac{2^{32}}{2^{30}}$ sec

→ 4 sec

WAT

WAT << LT

⇒ ordering is difficult bcoz occurence of same seq num is more frequent

↓ sol (i) Decrease BW - Not scalable

(ii) Increase seq num

How to increase seq numbers :

1sec — 1GB

1sec — 1G seq num's

$\frac{180sec}{LT}$ — 180 × 1G seq num's.

Min num of seq num ↑

bits = $\lceil \log_2(180 \times 1G) \rceil$ → 38 bit

∴ Extra bits = 38 − 32

6 bits

↓

Taken from options in Tcp header as Timestamp

Num of bits req to avoid WAT = $\lceil \log_2(\text{Lifetime} * \text{BW}) \rceil$

③ Header length : (4)

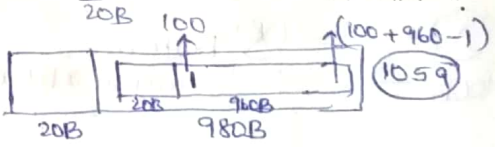
scaling factor = $\frac{60}{15} = 4$ (60 → max header size; 15 → max num rep in HL)

→ Tcp header will be choosen as multiple of 4 or not some bits are padded to make it so.

Q TL = 1000B, HL = 5 } IP

HL = 5 → 20B → Tcp, seq = 100

Ack num = ?

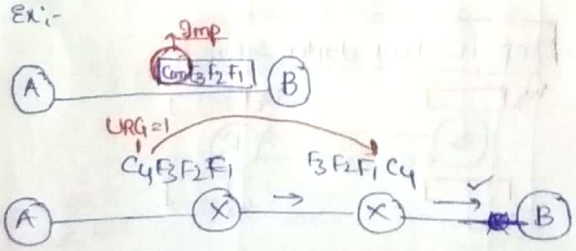


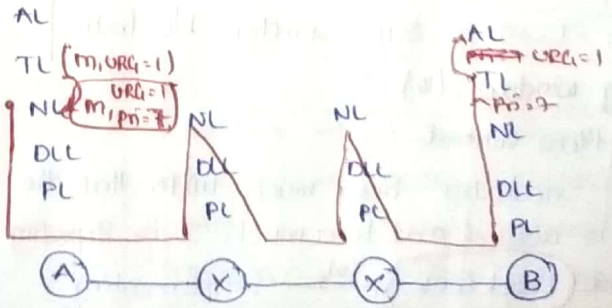
last byte = 1059

Ack num = last byte + 1 → 1060

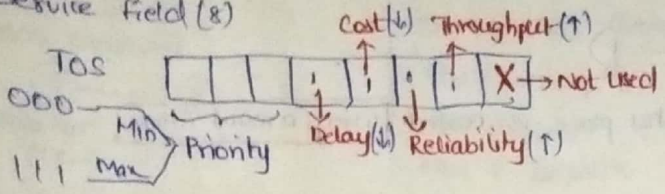
④ URG flag and urgent pointer :

For Ex:-





→ In between Sender and reciever, routers are present. These routers doesn't have Transport layer. So inorder to understand the routers that the data is URG we use Type of service field (8)

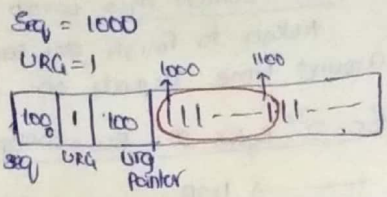


Urgent pointer (16)

If URG = 1 → till what part the data is urgent?

↓

determined by urgent pointer



Now, Seq = 1000, URGptr = 100

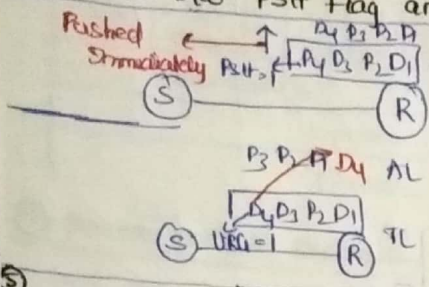
Urgent data = Seq + Urg ptr → 1000 + 100 = 1100

∴ 1000 − 1100 Bytes are URG

101 B are URG

Num of bytes Urgent = Urgent ptr + 1

Difference blw PSH flag and URG flag:



*) PSH guarantees Inorder of data. URG causes out of order of data.

(5) ACK flag:

If ACK = 1 → It Expects next packet

ACK = 0 → End of packet.

6 bits reserved

All these bits are set to '0' and are used for future use

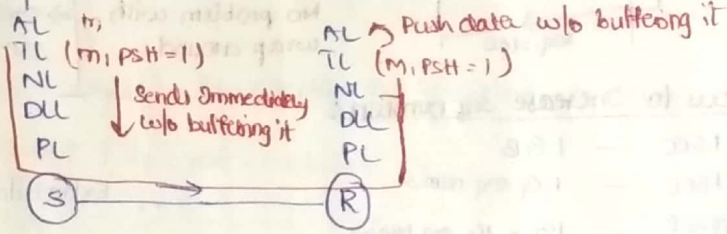
(6) PSH flag:

- Used for Interactive applications

Ex:- Chatting, Telnet

Hi → 2B

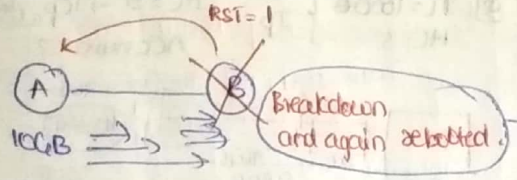
will not wait until it fills Mss



(7) RST flag:

- If any unExpected things happen in the N/w then RST is set to '1'.

Ex:-



After rebooting If packets arrived the station B doesn't know whether it has asked for 100B file or not.

Used to:

refuse Incoming Connection

reject a segment from unknown host

restart a Connection

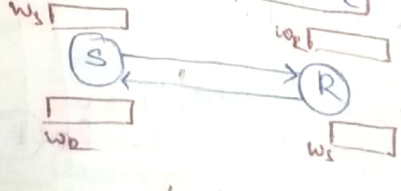
(8) SYN flag:

When syn = 1 → It sets Connection b/w hosts

(9) FIN flag:

When FIN = 1 used to release a Connection

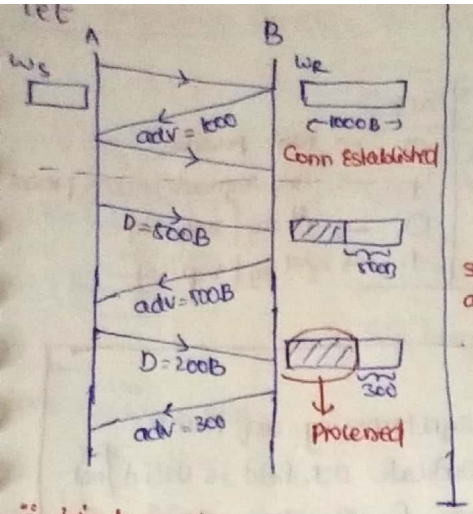
As TCP is Full duplex service

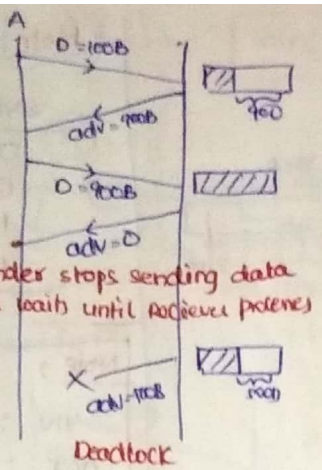


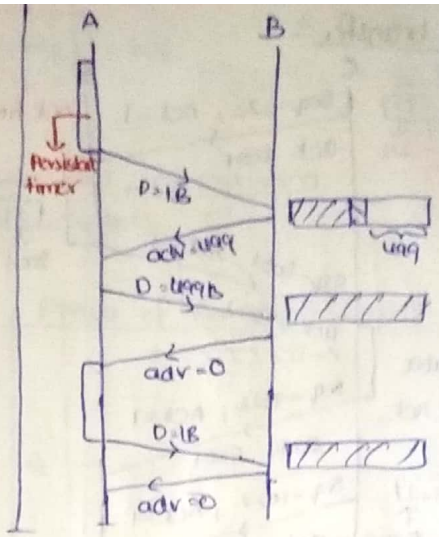
4 windows

(10) Advertising window: (16)

- Used for Flow Control
- It also indicates how much buffer that the reciever is allotted and how much it is Expecting
- May cause deadlock → sol → Implementing Persistent timer







(6)

(10) Window Size :

- 16 bits

Max buffer size = $2^{16} - 1 \Rightarrow 65,535$ B → If our Receiver Buffer is 1GB → (30 bit req'd)

We add 16 + 14 (options)
Addn option field.

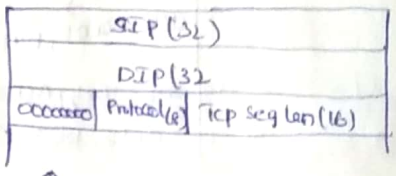
(11) Checksum : (16)

- Checksum is computed on 3 fields → Tcp Header, Tcp data, Ip header (2 times checksum is applied for reliability)

The Ip header may change often at each receiver. So, few fields are used to compute checksum

- Source Ip
- Dest Ip
- protocol
- Tcp seg length

→ Pseudo Ip header



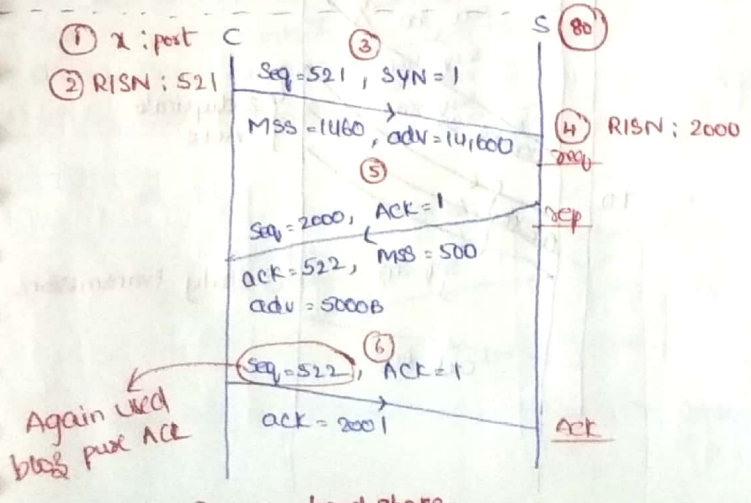
(12) Options (0-40 B)

- Time stamp (WAT << Lifetime) ⇒ Increase seq num (some bits are borrowed)
- Window size Extension
- parameter negotiation (MSS, ---) (used during Connection establishment)
- padding (To make header multiple of 4)

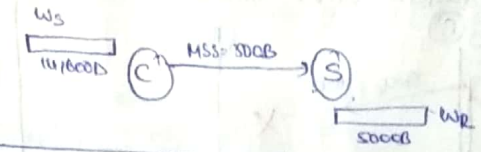
Tcp Connection management : → Connection establishment (SYN, ACK), → Data transfer, → Connection termination (ACK | FIN) } Connection oriented.

(1) Connection Establishment

- Tcp uses pure ACK or piggybacked ACK depending upon requirement



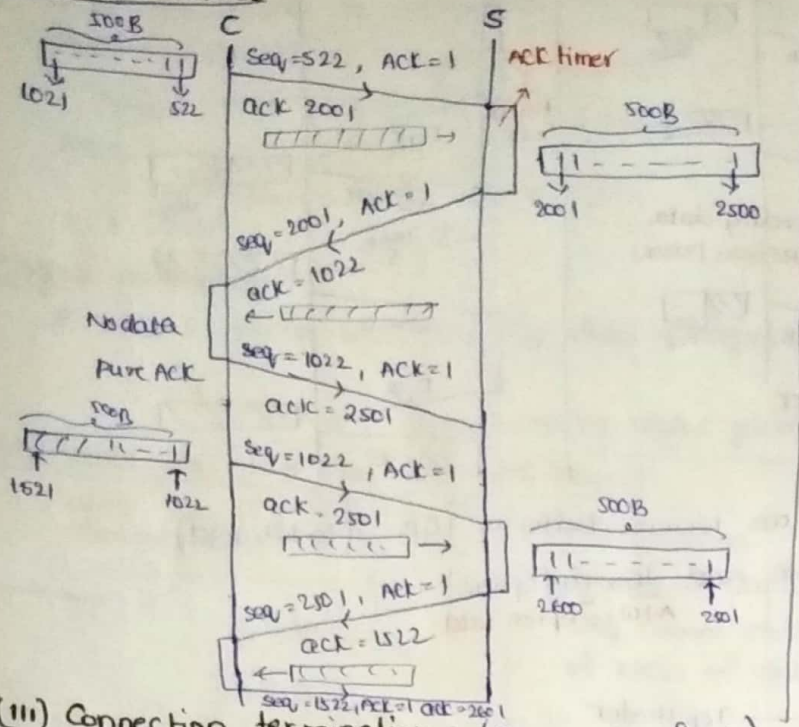
MSS = min (1460, 500)
= 500 B



Note :

SYN = 1	→	1 seq Consumed
ACK = 1	→	0 seq Consumed
FIN = 1	→	1 seq Consumed
1 Data B	→	1 seq Consumed

(II) Data transfer:



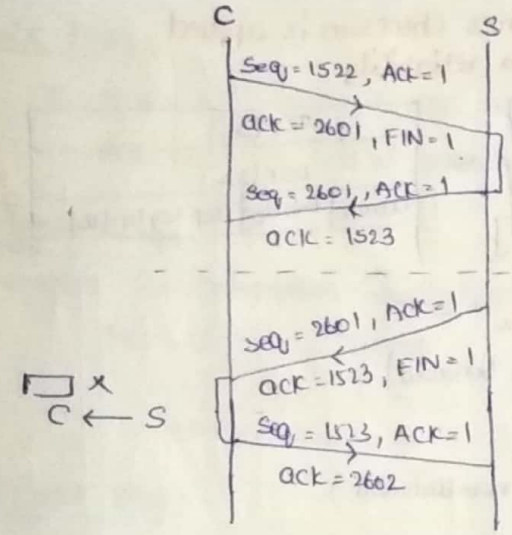
* Note:

SYN	ACK	
0	0	→ Not possible
0	1	→ Any segment, ACK is present
1	0	→ 1st seg / req seg
1	1	→ 2nd seg / rep seg

Note:

SYN : synchronizing seq nums
ACK : Indicate ack field is valid / not
FIN : seq for Conn termination

(III) Connection termination: (Resources are freed)



C → S ☐ X

- Data : C → S ✗
- Data : S → C ✓
- ACK : C → S ✓ (Pure)
- ACK : S → C ✓ (Piggybacked.)

☐ X
C ← S

In 3-handshakes we can close. (Best case)

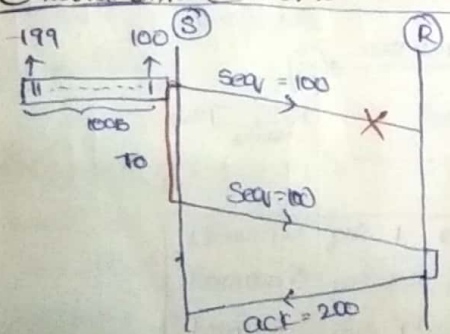
4-way hand shake

Why 3? — Experimentally proven that it is best

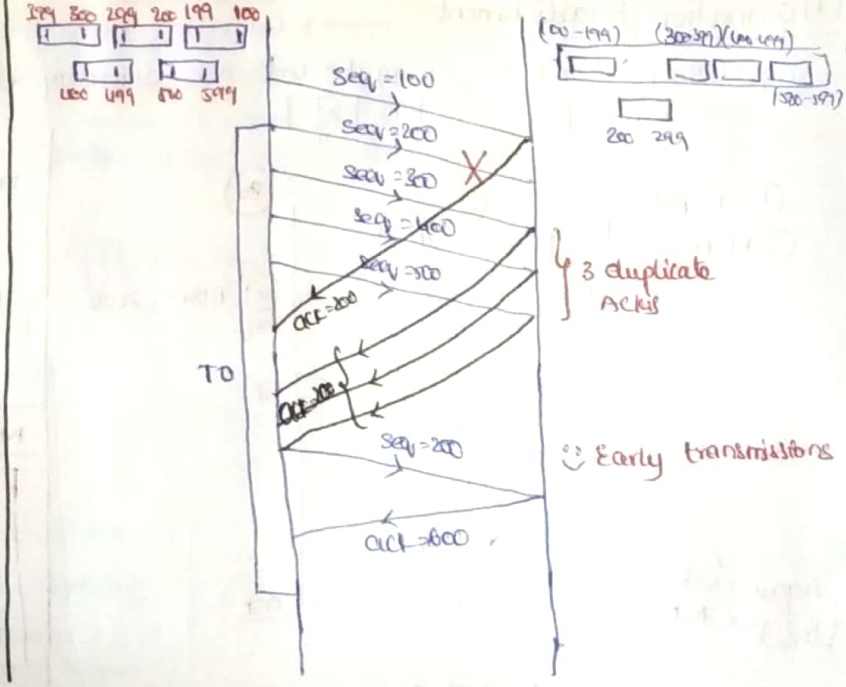
Retransmissions in TCP:

- TCP uses SR + GBN
  - (75%) → $W_S = W_R$ → out of order accepted
  - (25%) → Ack are cumulative

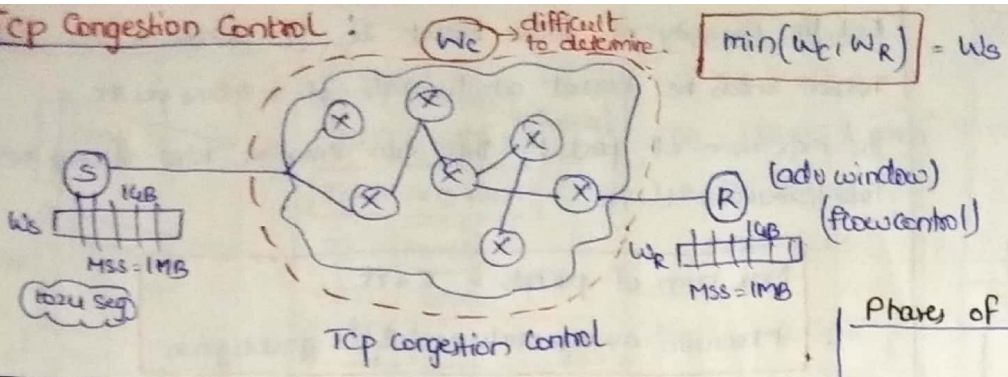
① Retransmissions after Timeout:



② Retransmission after (3) dup Acks



Tcp Congestion Control :



adv window = 8KB (WR)

MSS = 1KB

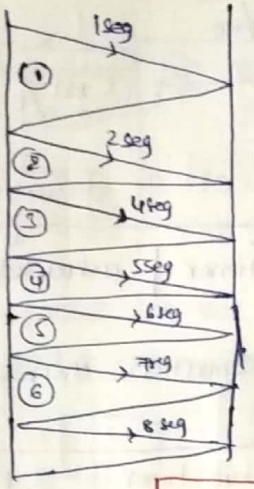
W_R = 8 seg $\Rightarrow$ Th = $\frac{W_R}{2}$ = 4 seg

Wc = 1 2 4 5 6 7 8

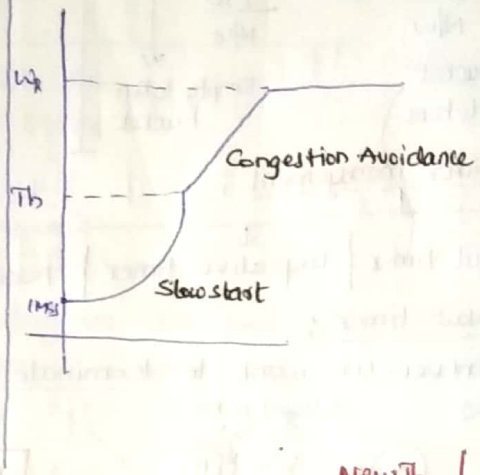
Num of RTT's taken to reach max seg size = 6

(or)

1	2	4	5	6	7	8



Phases of Congestion Control :



Congestion Control Algorithm :

Step ① : Slow start phase (exponential)

Step ② : Congestion Avoidance phase (linear)

Step ③ : Congestion detection →

How to detect Congestion.	New Th	Action
(a) Timeout (severe Congestion)	$\frac{1}{2}$ Wc	SS phase
(b) 3 dup acks (mild Congestion)	$\frac{1}{2}$ Wc	CA phase
(c) icmp Source quench		

Ex:- W_R = 64KB

MSS = 1KB

W_R = 64 seg Th = $\frac{64}{2}$ = 32

Algo :

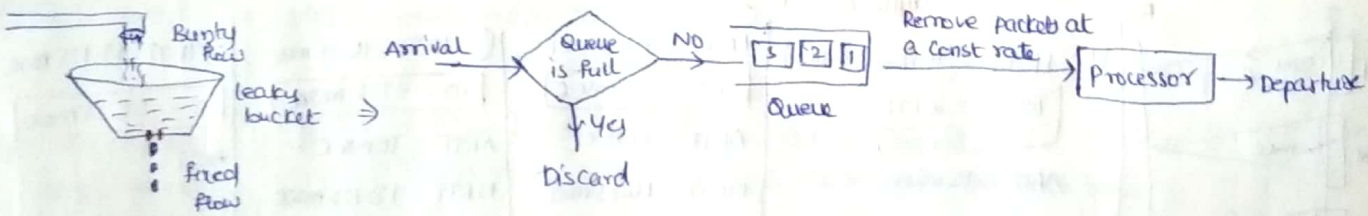
1 2 4 8 16 32 33 34 (Th=17, TO) 1 2 4 8 16 17 18 19 20 (Th=10, 3 dup) 10 11 12 (Th=6, TO)

1 2 4 6 7 8 (Th=4, 3 dup) 4 5 6 - - - - - 64 64 64 (Th=32, TO) 1 2 4 8 16 32 (Th=16, 3dup) 16 17 (Th=8, TO) 1 2 4 8 9 - -

Traffic shaping :

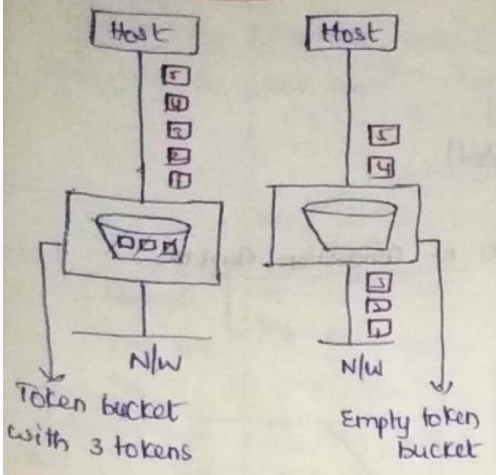
- Another method of Congestion Control is to shape the traffic before it enters into network.
- Traffic Control Controls the rate at which packets are sent
- During conn establishment, the sender and the carrier negotiate a traffic pattern (shaping)

(1) Leaky bucket Algorithm :



∴ It will always transmit the data with const bit rate

(ii) Token Bucket:

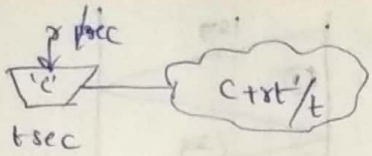


Let the capacity of token bucket be 'C' tokens

Token Enters the bucket at the rate of γ tokens per sec.

The max num of packets that can Enter the N/w during the time interval 't' is

> Max num of packets = $C + \gamma t$
>
> Maximum average rate = $\frac{C+\gamma t}{t}$ packets per sec



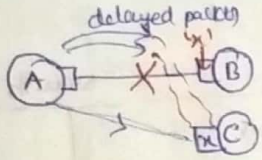
Tcp timer management:

Time wait timer | Keep alive timer | Persistent timer | acknowledgment timer | Time out timer

Time wait timer:

- Whenever we want to terminate the Connection then we have to wait for some time

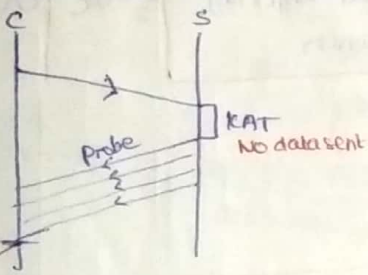
bcoz



> Time wait timer = 2 * life time

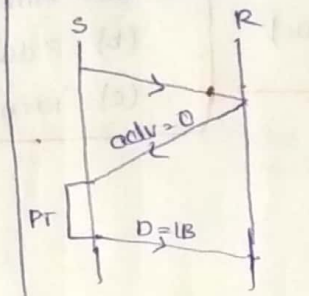
Keep alive timer:

☺ Closes the idle Connection



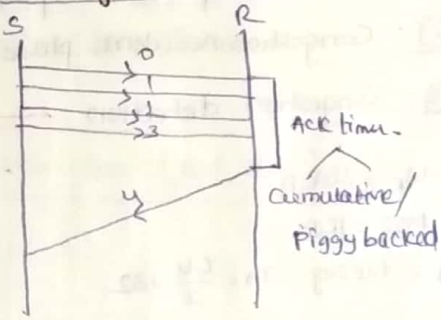
still no response then client got down and close conn

Persistent timer:



used when adv=0

Acknowledgment timer:



Time out timer:

> Time out timer at DLL = 2 * RTT = 4 * Tp

*) Time out timer is set dynamically bcoz many factors influence time out → (i) Traffic
(ii) Chosen path

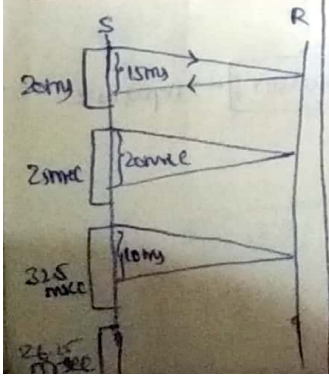
*) If TO Timer

- Large → waste of time if packet is lost
- Small → unnecessary retransmissions

Basic Algorithm:

> Next RTT = α (Initial RTT) + $(1-\alpha)$ (Actual RTT)

static ↑ $0 \le \alpha \le 1$ ↑ dynamic



① IRTT = 10msec (Guess)

TO = 2 * RTT = 20msec

ARTT = 15 msec

NRTT = 12.5 msec

(α = 0.5)

② IRTT = 12.5 msec

TO = 25 msec

ARTT = 20 msec

NRTT = 16.25 msec

③ IRTT = 16.25 msec

TO = 32.5 msec

ARTT = 10 msec

NRTT = 13.125 msec

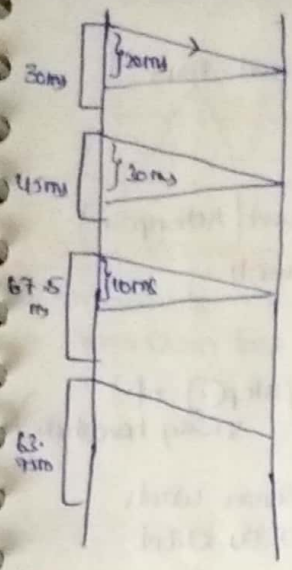
④ IRTT = 13.125 msec

TO = 26.25 msec

ARTT =

∴ TO = (2) * RTT → why ?? (don't know)

Jacobson Algorithm :



① IRTT = 10ms, ID = 5ms (Guess)

$TO = 4*D + RTT$

TO = 30ms

ARTT = 20ms

$AD = |IRTT - ARTT|$

= 10ms

$NRTT = \alpha IRTT + (1-\alpha) ARTT$, $\alpha = 0.5$

NRTT = 15ms

$ND = \alpha ID + (1-\alpha) AD$, $\alpha = 0.5$

ND = 7.5ms

② IRTT = 15 ms

ID = 7.5ms

TO = 30+15 = 45ms

ARTT = 30ms

AD = 15ms

NRTT = 22.5ms

ND = 11.25ms

③ IRTT = 22.5 ms

ID = 11.25 ms

TO = 45 + 22.5 = 67.5ms

ARTT = 10 ms

AD = 12.5 ms

NRTT = 16.25 ms

ND = 11.875 ms

④ IRTT = 16.25 ms

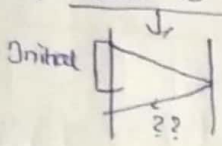
ID = 11.875 ms

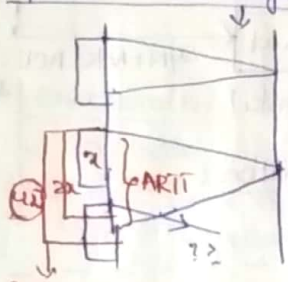
TO = 4*D + RTT

TO = 63.75 ms

∴ - For guessing the TO of next packet we are depending on all prev packets

- If Guess is wrong or the packet acknowledges after time out



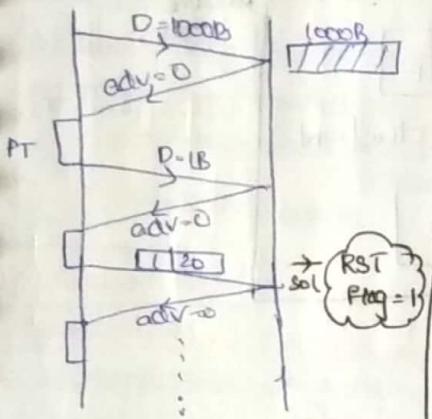


Sol → Karn's modification (double the timeout timer)

Silly window syndrome :

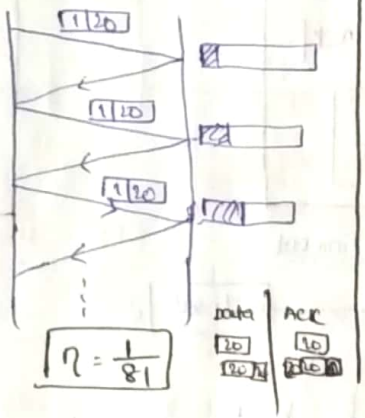
Reasons

(I) when adv = 0



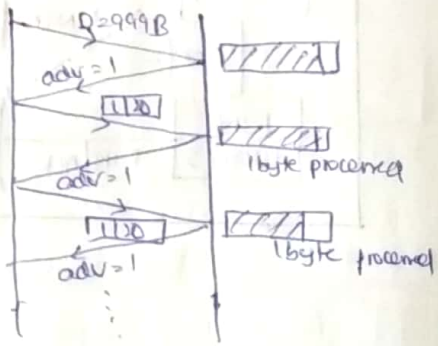
Transient problem (Temporary)

(II) when sender is sending few bytes



$\eta = \frac{1}{81}$

(III) when Reciever is Consuming few bytes



$\eta = \frac{1}{81}$

Persient (Permanent) problem.

Nagles Algorithm : (II)

- Deals persient problem of silly window syndrome.

Algo : wait for 1RTT ⟶ Buffer the data at sender. (of data)

>1MSS	<1MSS say 50B
Send 1MSS	send 50B instead of 1B Each time

Ex:- S = 1B/msec

RTT = 100 ms

MSS = 200 B

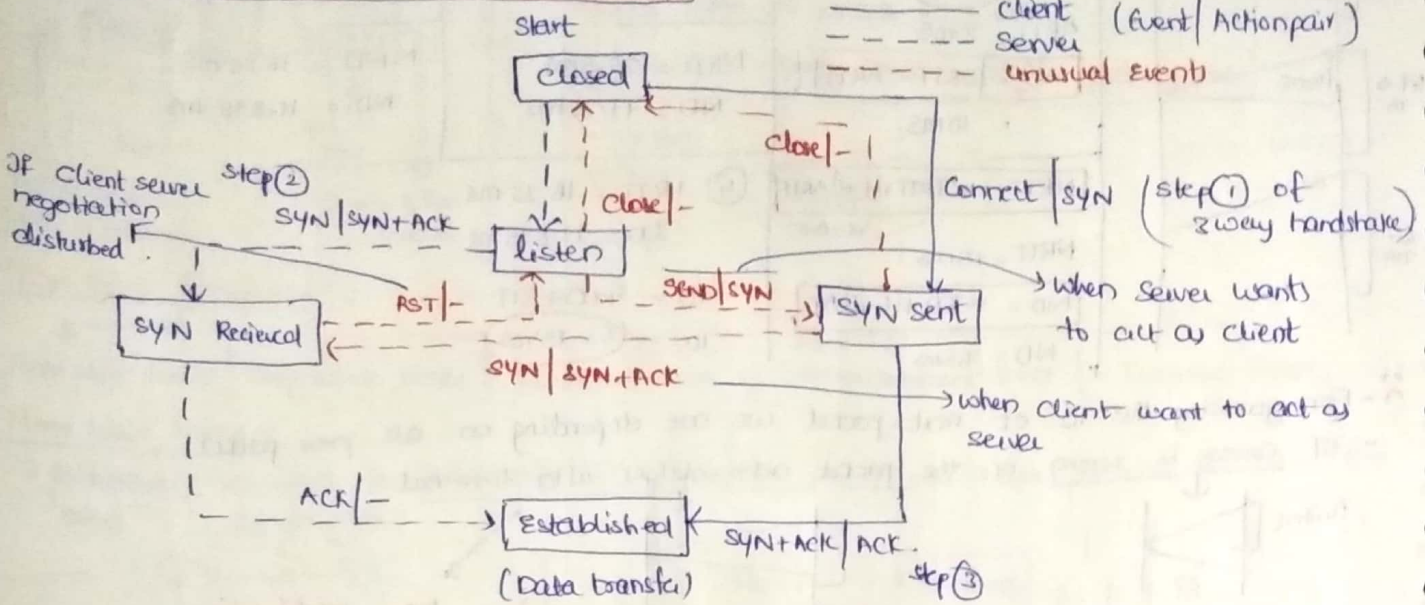
1RTT → (100B)

wait for 100B and Send it

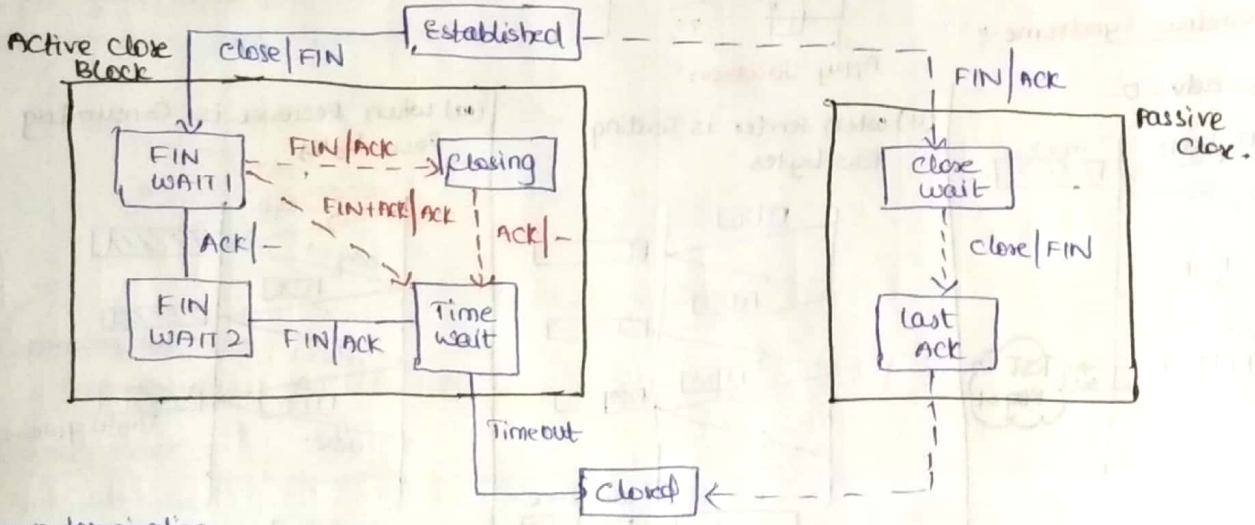
Clarck's Algo :

- Deals with persient problem (iii) of silly window syndrom
- The adv window should wait until it ½ of Buffer is Empty and then start advertising it (or 1MSS is free)

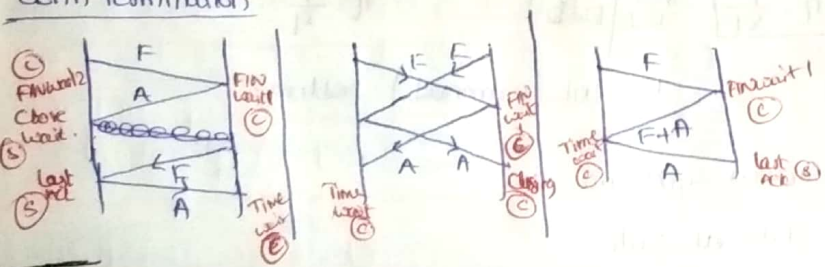
State transition table – Conn establishment :



state transition diagram – Conn termination :



Conn termination



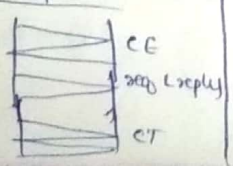
Need of UDP ie disadvantage of Tcp :

① If Application needs 1 req & 1 rep

Ex:- DNS
BootP
DHCP
N/w time protocol

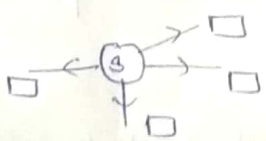
UDP

TCP



② Broadcasting or multicasting Apps

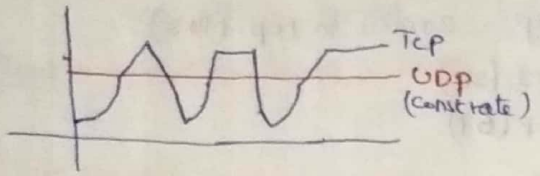
TCP is Conn oriented



If N/w is Class A
2^{24} Buffers are received

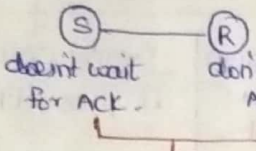
③ Fastness rather than reliability

Ex:- online games, multimedia Applications



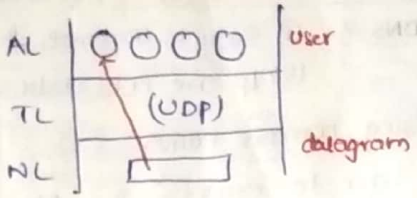
User datagram protocol (UDP)

- Unreliable
- Connection less

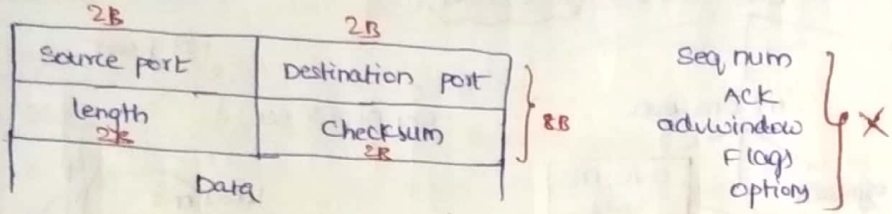


↓

unreliable but easy to process



UDp segment structure :



Seq num
ACK
advwindow
Flags
option
} X

length : (16) → $(2^{16}-1)$ → (65,535 B)

UDp Header + data.

8B — 65,527 B

Checksum(16) :

- UDp Header + UDp data + pseudo Ip header
- In most cases checksum is disabled by setting cs field as '0' or '1's bcoz of unreliable service.

Functionalities of UDp :

1) Trace route
2) Record route
3) Time stamp (intime, outtime) at NL
4) ICMP Error realization
   - destination unreachable — Host, port
   - Source quench

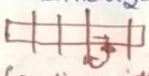
Imp

Protocols that uses UDp			Protocol that uses Tcp
DNS	RIP	TFTp	FTp
Bootp	OSPF		Http
DHCp			

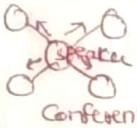
⑤ Session layer :

Responsibilities

- Authentication or Authorization (Digital signature)
- Check pointing



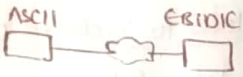
- Synchronization (Audio + video) → diff files
- Dialogue Control (like Flow Control) → use as half duplex
- Logical grouping of operations
  - Atomicity



⑥ presentation layer :

Responsibilities :

- Character translation



- Encryption & decryption. (cryptography)
- Compression / Expansion

↓

There responsibilities are not need always by Application. so, the Application which requires those functionalities will Implement session & presentation layer

Application layer protocols :

	Port	
① DNS ⇒ Domain name → Ip	53	UDP
② HTTP → webpages	80	TCP
③ FTP ⇒ files	21, 20	TCP
④ SMTP ⇒ mails	25	TCP
⑤ POP ⇒ retreiving mails.	110	TCP

MIME - Base 64 Encoding
IMAP - similar to pop. (143)
Telnet (23)
TFTP (69)

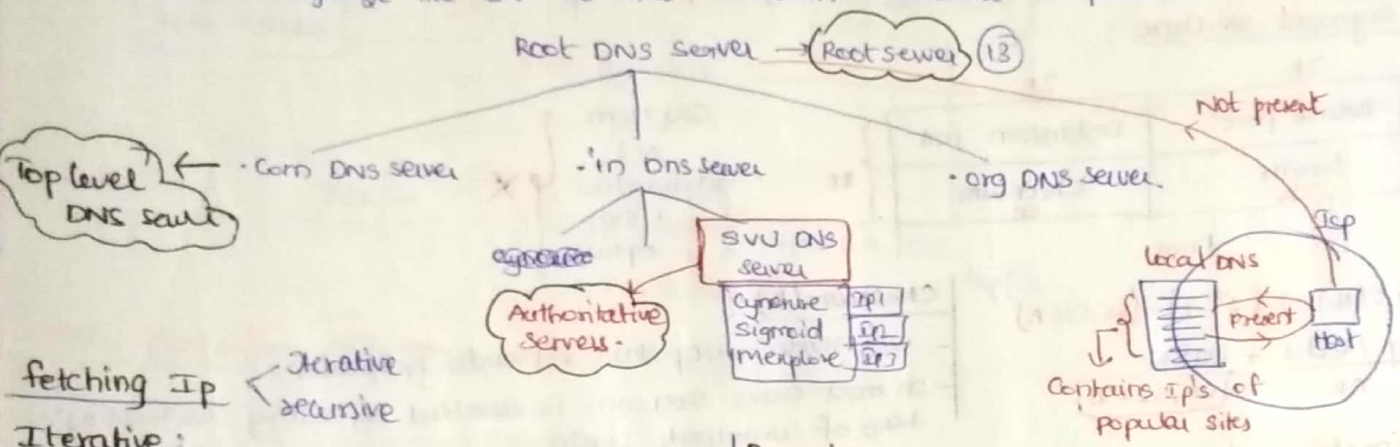
Domain Name Service (DNS) : (Port : 53)

Why DNS ? (i) Easy to remember the domain
(ii) Ip are not static.

Organizing domain names :
- We have to organize the DN so that efficient retreival is possible.

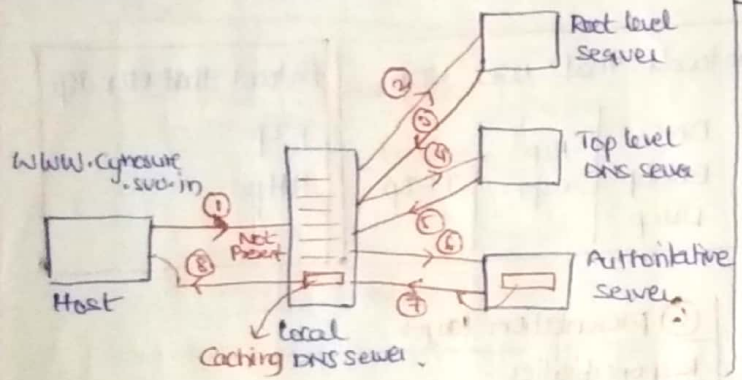
DNS (DN → Ip)

Generic domain	Country domain	Inverse domain
.com, .edu .org, .net ..	.in, .us, .uk ..	(Ip → DN)

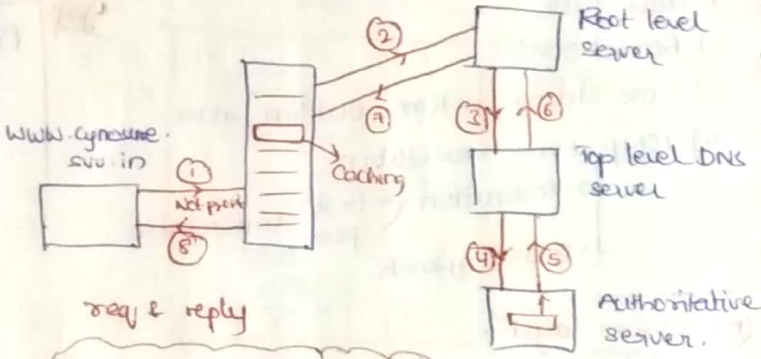


fetching Ip ← Iterative, recursive

Iterative :



Recursive :

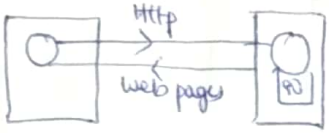


DNS uses UDP protocol

How to reduce DNS traffic :
- powerful caching at local DNS server
- Contacting the nearby root level DNS server.

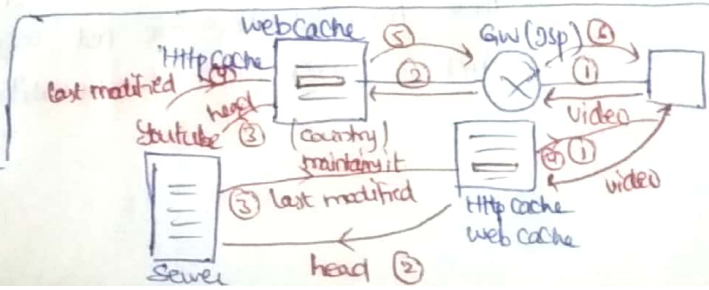
Hyper text transfer protocol (Http) : (Port : 80)
- in band protocol (Commands + Data go in same conn)
- stateless protocol (Recording the vistors of website)
  ↓ to keep track on users
  Cookies are used
- HTTP 1.0 → non persistent connection
  HTTP 1.1 → persistent connection

HTTP uses TCP at TL
→ bcoz it Expects reliability and it is provided by TCP ~~and~~ not by its own



Methods in HTTP :

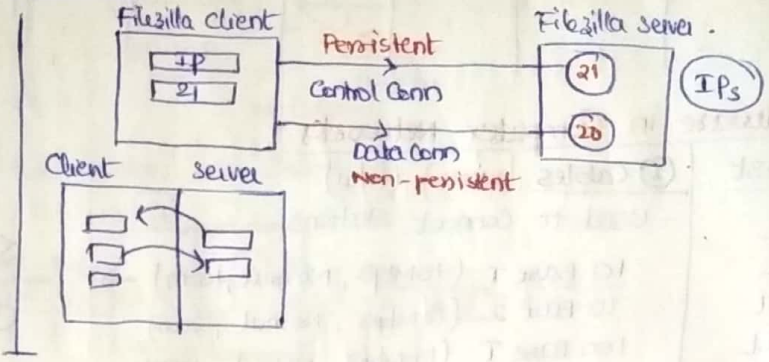
① Head : To retreive meta data of a webpage



(2) Get : Get the webpage / resource
(3) post : Used in forms
Send data to server
(4) put : Upload an object to server
(5) Delete : Delete a object
(6) Trace : Tracing the servers in which the data is coming from.
(7) options : provides the options that the server is providing.
(8) Connect : Used by Https
Authenticated Acces is present

(3) File transfer protocol (FTP) (Port : 21) (Port : 20)
Control Conn Data Conn

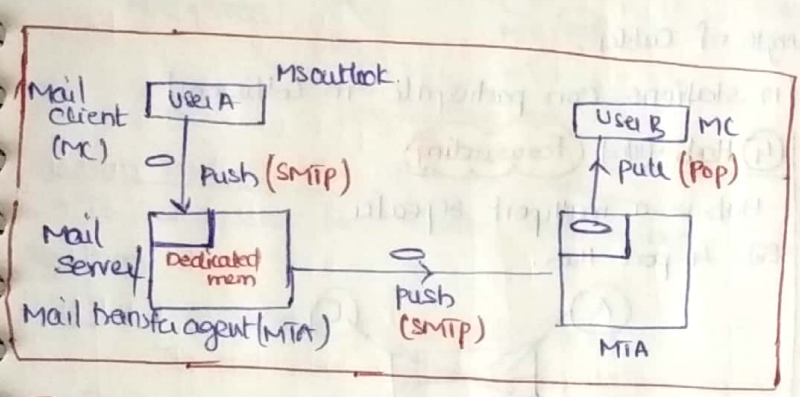
- Tectia, Filezilla
- out of band
- persistent connection for Control Conn
  Non persistent Conn for Data Conn
- FTP uses TCP
- Statefull protocol



(4) Simple Mail transfer protocol (SMTP) (Port :

Why SMTP if FTP supports file transfer, mails ?

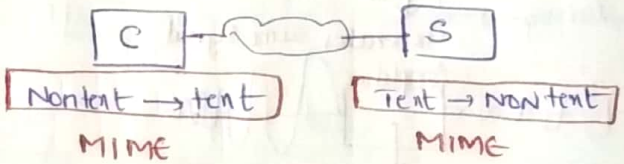
- In Ftp both client and server should be online
- In SMTP it is not need always



Internet msg acces protocol

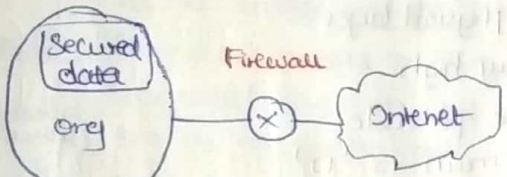
Text transfer : SMTP, POP3 / IMAP4

Non text : video, audio.



(Multipurpose Internet mail Extension)

Firewalls :



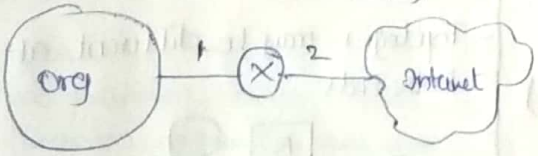
Types of Firewalls
- → layer 3 FW / packet filtered FW
- → layer 4 FW
- → layer 5 FW / Proxy Firewall

layer 3 FW / Packet filtered FW :
- It can see NL, DLL, PL
(i) Block hosts (SIP, DIP)
(ii) protocol Blocking (TCP, UDP, ICMP, IGmp)
(iii) particular protocol from particular host
× Dos attack

layer 4 FW :
- It can see TL, NL, DLL, PL
(i) Block hosts (SIP, DIP)
(ii) protocol (Block a protocol)
(iii) Block a service (HTTP, SMTP, FTP) (Port)
(iv) A particular service on a particular host

Blocking
(i) Ip (Hosts)
(ii) ports (services)
(iii) service on a host
(iv) Application layer (Authentication) by username & password data

layer 5 FW / Proxy Firewall (Present)
(AL, TL, NL, DLL, PL)



How proxy firewall works :

SIP	DIP	SP	DP	Interface	User	
Ics	–	21	–	1	–	→ Block file sending from org
–	Ics	–	23	2	NE	→ Block Non Emp who are trying for remote login (telnet)
–	–	80	–	1	–	→ Block using webpage from org

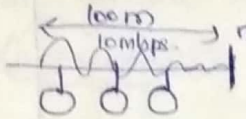
Hardware in Computer Networks :

① Host

AL
TL
NL
DLL
PL
A

② Cables (wires) (H/W)

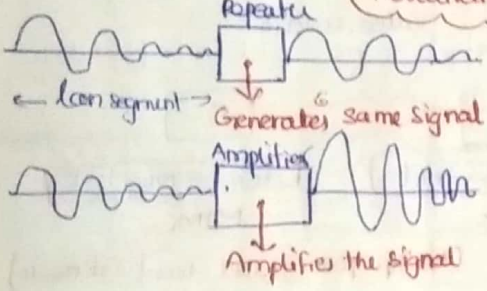
- Used to connect stations
  - 10 Base T (10Mbps, Normal, 100m) →
  - 10 Base 2 (10Mbps, Normal, 200m)
  - 100 Base T (100Mbps, Normal, 100m)
  - 100 Broad 5 (100Mbps, Mul, 500m)



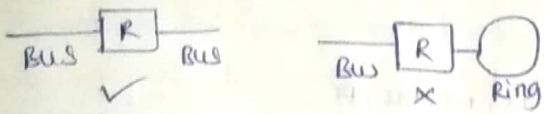
- Operates on physical layer
- Authentication prob (losing of Energy)
- Collisions are possible (Base)
- length of LAN is limited by length of Cables.
- Collision domain is 'n' => (n stations can participate in collisions)

LAN < 10km
MAN (10km – 100km)
WAN > 100km

③ Repeater : (H/W) (Forwarding)



- Repeater is used to connect two lan segment
- operates at physical layer
- Topology should be same at both End

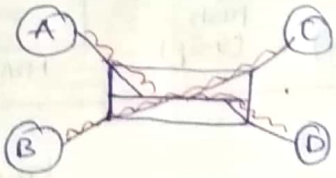


- Collisions are possible inside repeater
- collision domain is 'n'
- Range of LAN is increased

④ Hub : (H/W) (Forwarding)

- Hub is a multiport repeater.

Ex:- 4 port Hub.



- operates at physical layer
- Traffic is very high
- Collisions are possible
- Collision Domain is 'n'

∵ Cost is less.

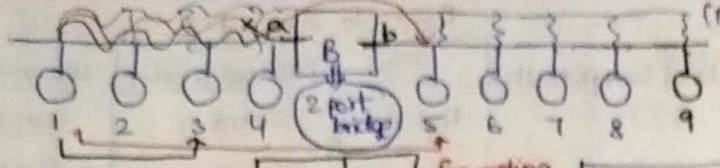
⑤ Bridge :

- operates at PL, DLL

Capabilities

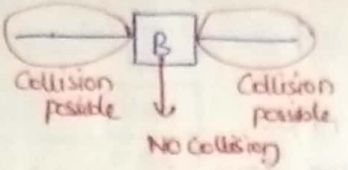
- Filtering (1 to 3) (If two stations present in same LAN)
- Forwarding (If two stations are present in different LAN's) (1 to 5)
- Flooding (If destination is not known by bridge) (1 to 9)

- Topologies may be different at both Ends

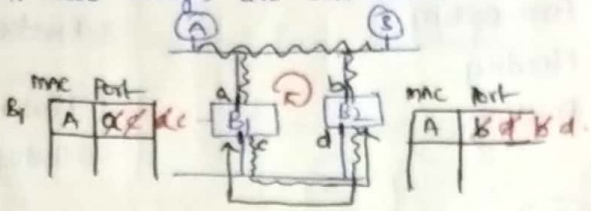
*) 

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

- Store and forwarding is also provided so,
- No collisions inside a Bridge
- Collision domain is reduced

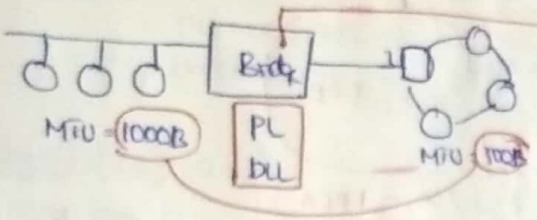


∴ Bridge :
- packet may fall in infinite loop if two bridges are introduced
- The port num will always changes



sol → Spanning Tree Algo

-



→ Don't support fragmentation

So, Although it supports diff topology we generally use same topology at both Ends

Spanning tree Algorithm : ← sol
- To save packet from falling into a loop → sol TTL Time to live (not implemented in frame)

Algorithm :

Step(1) : Every bridge has a built in id,
The one with smallest id is taken as root bridge (R)

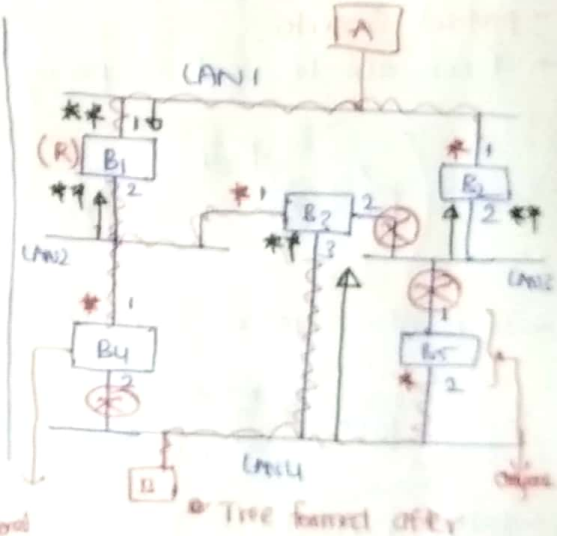
Step(2) : Mark one port of each bridge which is closest to root bridge as root port (*)

Step(3) : Every LAN chooses a bridge closest to it as a designated for that LAN and make the corresponding port as designated port (**)
(Lowest id forward)

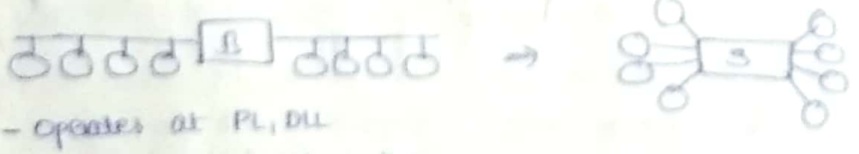
Step(4) : Mark the root port and designated ports as forwarding port and block remaining

↓

(Bridge Data unit protocol)

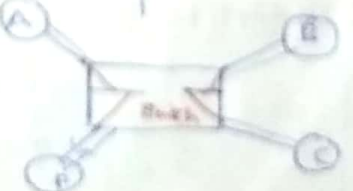


(6) Switch :
- Similar to bridge but main difference is interms of interfaces



- Operates at PL, DLL
- No collisions inside switch
- Collision domain is reduced to '0'
- Traffic is very less
- Connections are full duplex

∴ Switch → Costly

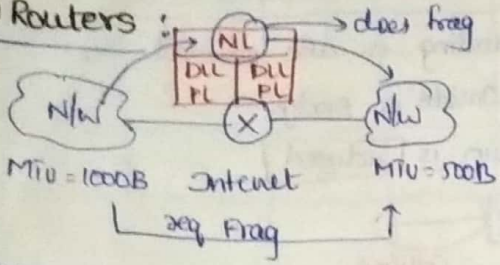


*) Wire + Repeater + Hub + Bridge + switch (cable) → LAN Components

Points to remember

H/w	Broadcast domain	Collision domain
Repeater	Same	Same
Hubs	Same	Same
Bridges	Same	reduces
Switch	Same	reduces (0)
Routers	reduce	reduce
Gateways	reduce	reduce

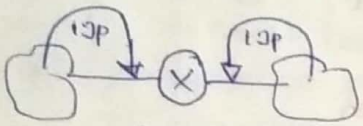
(7) Routers :

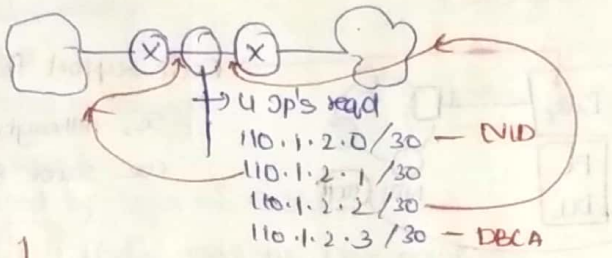


- Operates on NL, DLL, PL
- Capabilities (i) Filtering (ARP, RARP, Bootp)
  (ii) Forwarding
  (iii) Flooding
  (iv) Routing

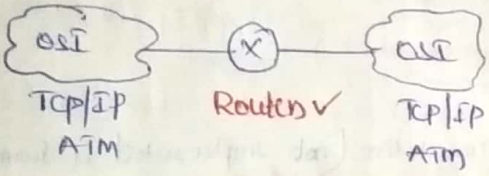
→ No Collision inside router
→ Collision domain reduced
→ Broadcasting domain reduced.
∴ Router → Costly

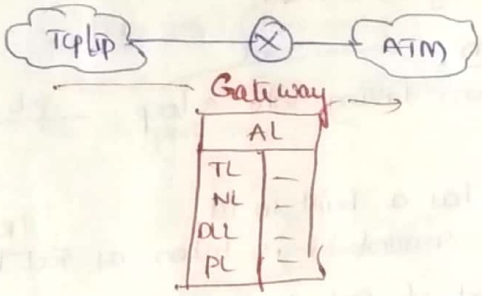
*)



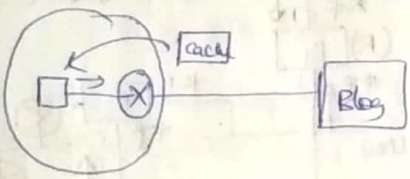


(8) Gateways :

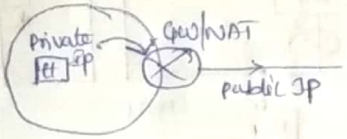




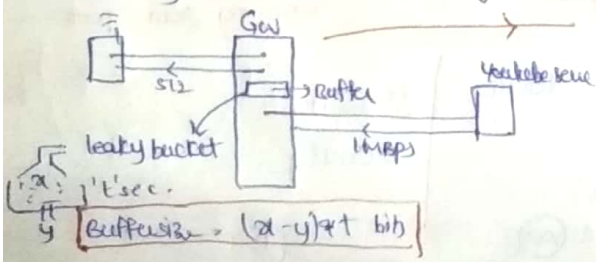
- protocol Converter.
- It can also be used as proxy



- It is also capable of Network address translation



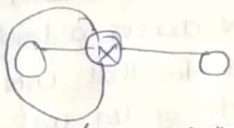
- Buffering present in gateways



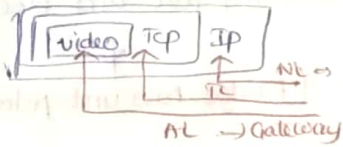
blog to make mobiles power Efficient

Buffer size = (x-y)*t bits

→ Also used in Firewall



- DPI (Deep packet inspection) because of presence of App layer



- DHCP are configured on gateways

LAN Technologies :

IEEE standards	Technology
IEEE 802.3	Ethernet
IEEE 802.4	Token bus
IEEE 802.5	Token ring
IEEE 802.6	MAN
IEEE 802.11	WiFi
IEEE 802.15.1	bluetooth
IEEE 802.15.4	Zigbee

Ethernet : (IEEE 802.3)

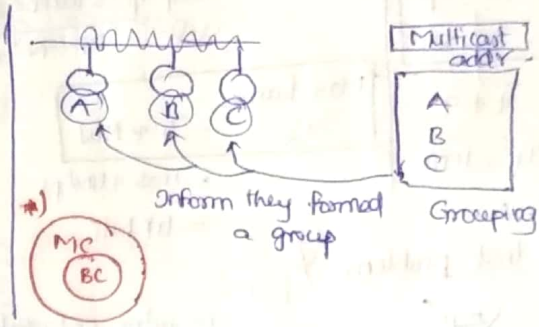
Topology : Bus (ddd--d)
Access Control : CSMA/CD
Acknowledgment : NO
Data rate : 10Mbps, 100Mbps, 1Gbps
Encoding : Manchester (1 ⊓⊔, 0 ⊔⊓)
Mode : half duplex

Ethernet frame format :

7B	1B	6B	6B	2B		4B
PREAMBLE	SFD	DA	SA	LENGTH	-- DATA --	CRC

PL header | DLL header | DLL trail
64B

1) Preamble (7B) : 1010---10 (7B) — used to alert all stations that frame has started
2) SFD (1B) : 10101011 (11 = End of SFD) — used for synchronization.



3) Destination & Source Address (6B)
- MAC Address (Physical addr) — SA is always unicast addr
Types of MAC : (i) UNICAST → LSB of 1st byte is '0'
(ii) MULTICAST → LSB of 1st byte is '1'
(iii) BROADCAST → All bits in MAC are 1's. (FF:FF:FF:FF:FF:FF)

4) length : (2B) (0 to $2^{16}-1$)

In CSMA/CD

$L \geq 2T_p * B$) Ethernet standards
$L \geq 64B$ (Min)

Min data ⇒ 64 - 18 ⇒ 46B

Max data : 1500B
Framesize : 1500 + 18B

	Min	Max
Data	46B	1500B
Frame	64B	1518B

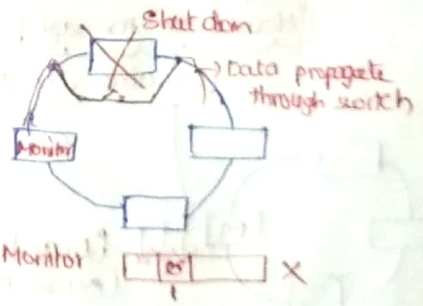
5) CRC (4B)
- 32 bit CRC generator is used.

∴ Ethernet
- Not applicable to real time applications.
- Not applicable to interactive applications (ent 'Hi' → 2B)
- No priorities in Ethernet (client server apps are not suitable)

Token ring ← sol ← Modifications
switches, star topology introduced
Full duplex :

Token Ring (IEEE 802.5)

Topology : Ring
Access Control : Token passing
Acknowledgement : Piggybacked Ack
Data rate : 4Mbps, 16Mbps
Encoding tech : Differential manchester Encoding (0 ⊓⊔, 1 ⊔⊓)
Mode : Simplex.



At Source : (Prob)
*) In Token Ring Sender should discard the sent data. It will lead to (orphan packet problem)
*) If data got corrupted in such a way that the sender might not able to recognize it (stray packet problem) → sol → Calculating CRC at monitor

At destination : (Prob)
- If destination is down
- If destination is busy
- If Packet got corrupted if destination received

→ sol → If no errors (copied) in destination

(A) Available	(C) Copied		A	C
0	0	→ Initial	0	1
1	1	→ copied		not possible
1	0	→ Busy		
0	1	→ Error		
0	0	→ Busy		

whenever we perform retransmission make monitor bit as '0' otherwise the monitor thinks that the packet is orphan | stray and discards it

Token problem:

- It leads to monopolization (one station will always send data)
- If Token is given to send data ie as long as the token is available with the station the station sends the data

then the station try to hold the token forever → (Token problem)

Captured token → Sol → Token holding time (THT)

Max Frame size:

(i) Delayed Token Reinsertion:

THT = 10ms, BW = 4Mbps, RL = 9ms

$THT = T_t + T_p$

$= T_t + RL$

$10 = T_t + 9$

$T_t = 1ms$

Here T_p is taken as RL if 'N' not given

Max frame size $= T_t * BW$

$= 1ms * 4Mbps$

$= 4Kbits$

(ii) Early token reinsertion

THT = 10ms, BW = 4Mbps.

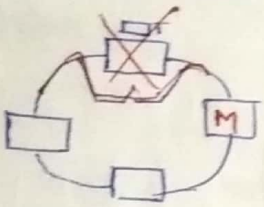
$THT = T_t$

Max Frame size $= THT * BW$

$= 10ms * 4Mbps$

$= 40Kbits.$

Token lost problem:



Sol →

Monitor calculates

Min Token return time = RL (Ring latency)

Max Token get time = RL + N * THT

→ If it Exceed → Token is lost and regenerated by monitor

Ex:- THT = 10ms, RL = 10ms, N = 10

Max TRT = 10ms + 10 * 10ms → After 110ms token is regenerated

= 110ms

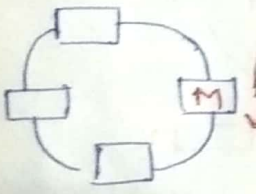
Token corrupted problem:

Token – 3B [~~~] Corrupted → Monitor doesn't recognize it as a token and discard it and generate new token after max TRT.

Monitor problem:

- If monitor is down → problems → orphan, stray, Token lost, Token corrupt } Every thing comes to halt

Sol → Heartbeat message (Im alive, Im alive, Im alive)



(AMP) (Active monitor presence)

Detected that monitor is down → Polling → Elect one station as monitor

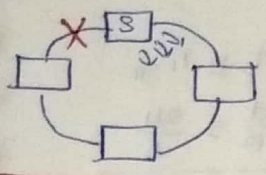
Polling → using polling frame

Monitor malfunctioning problem:

→ If monitor code got hacked by hacker then it pretends that it is alive by sending ARP packets and doesn't do any work.

→ Sol → human intervention

Problem with ring :



→ retransmit the packet for 3 times, if it doesn't reach sender again then we identify that the problem is with ring

Token :

- Used for Realtime applications bcoz of no collisions
- used for Interactive applications (can send for small amt of data)
- Stations and Tokens can be given priority (client server Arch is supported)

Token ring frame format :

Data frame :

(0 - max THT)

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

CRC is calculated upto this.

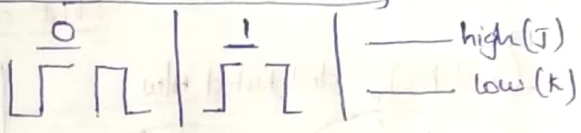
Token frame :

1B	1B	1B
SD	AC	ED

Start delimiter (SD) (1B)

J	K	0	0	I	K	0	0

In diff Manchester Encoding



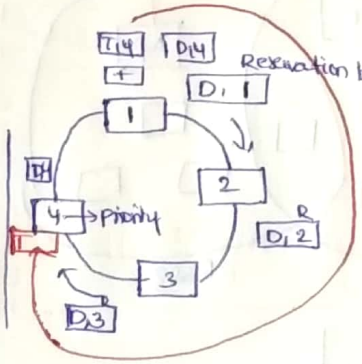
Access Control (AC) : 1B

P	P	P	T	M	R	R	R

Priority (0-7) ↓ Token ↓ Monitor ↓ Reservation

Token: 1 → Token frame; 0 → Not token

Monitor: 1 → (seen packet once); 0 → (first visit at monitor)



Frame Control : (1B).

0 0 → Data frame

1 1 → Control frame

Type of Control: AMP, polling, purge, Beckon - etc.

Destination & Source address (6B) :

MAC Addresses same as Ethernet

CRC (4B)

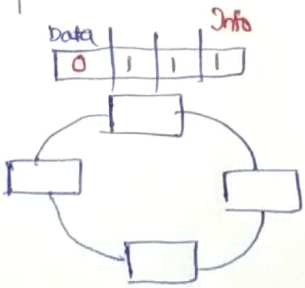
- 32 bit CRC generator is used

End delimiter : (1B)

J	K	1	1	J	K	I	E

E: 1 ⇒ Corrupted; 0 ⇒ busy

I: Info bit — 1 ⇒ More info is following; 0 ⇒ No info following



Frame status (1B)

A	C	0	0	A	C	0	0

A: Availability; C: Copied

1) Why 2 copies?

bcoz CRC is not calculated to FS bcoz FS bits changes frequently and this increases load on the stations so, we place FS outside CRC.

Minimum length of token ring :

- The token ring should capable of holding atleast 24 bit

$$\text{Capacity of wire} = T_p * BW$$

Capacity ≥ 24

$T_p * B \geq 24$

$$d \geq \frac{24 * V}{B}$$

(or)

$T_p \geq T_{t\,24}$

$\frac{d}{v} \geq \frac{24}{B}$

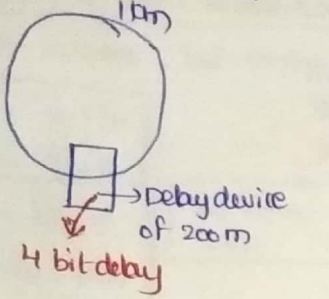
$$d \geq \frac{24 * V}{B}$$

Ex:- BW = 4 Mbps, $v = 2 * 10^8$ m/s

Sol $d \geq \frac{24 * V}{B} \geq \frac{24 * 2 * 10^8}{4 * 10^6} \geq 1200$ mts.

Min length of wire = 1200 mts

If wire of 1km is present



m → sec → B (/v, *B)

$\frac{200}{2 * 10^8} * 4 * 10^6 \rightarrow$ 4bit delay

BW = 4mbps. $v = 2 * 10^8$ m/s

How many meters is 1 bit equivalent to

$d \geq \frac{24 * V}{B}$

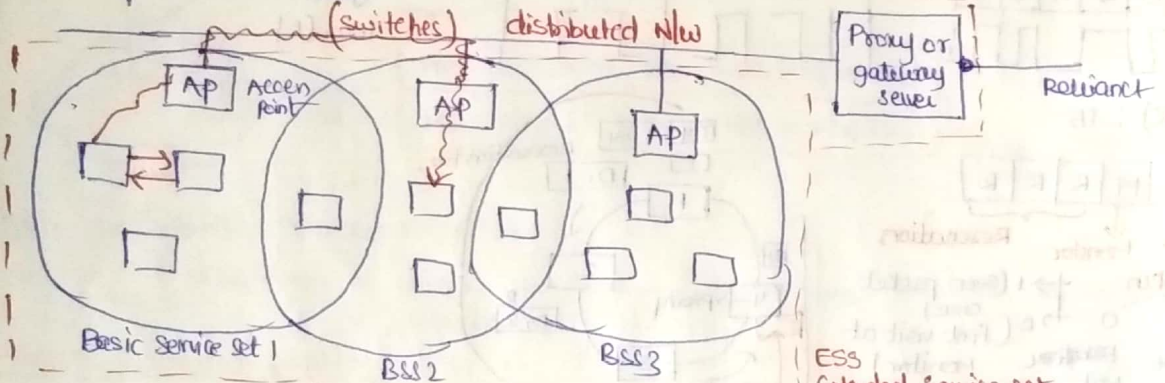
$\geq \frac{24 * 2 * 10^8}{4 * 10^6} \geq 1200$ m

24 bits → 1200m

1 bit → $\frac{1200}{24} \rightarrow 50$ m

Wifi (IEEE 802.11)

- Wifi operates at DLL



- Access Control : CSMA/CA
  Mode : Half duplex

Wifi

802.11	802.11a	802.11b	802.11g
Tx : 2 Mbps	54 mbps	11 Mbps	20+ Mbps
BW : 2.4 GHz	5 GHz	2.4 GHz	2.4 GHz

X — The End — X