Computer network

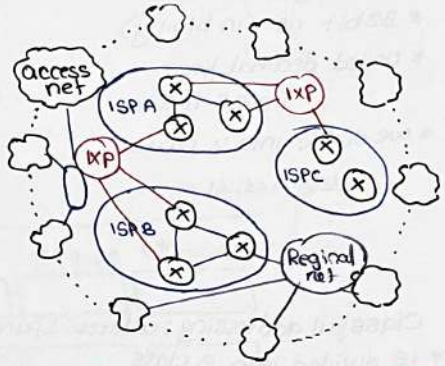
Avg. 8 marks.

- Reference Book
- Forouzan
- Kurose ross Top down approach
- Tanenbaum

- what is computer n/w?
  it's a n/w of computer
- Internet works (Internet) is what you get when you join n/w together

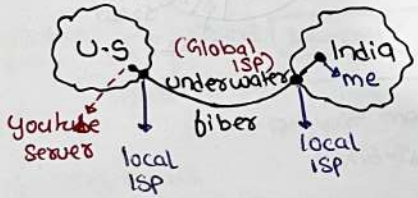
- what is Internet?
  A collection of computer n/w that use protocol to exchange data.

Internet structure:



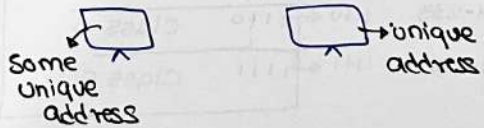
IXP: facility at which n/w connect

- using Internet service using Satellite would be time taking b/c of long dist. b/w them.
- connection b/w to country



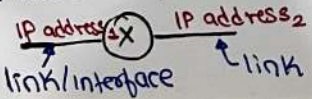
- protocols & layers
  is main structuring method used to divide up n/w functionality
  ↳ we need them for maintaince

- IP addressing
  need a globally unique way to "address" hosts.
  IP address



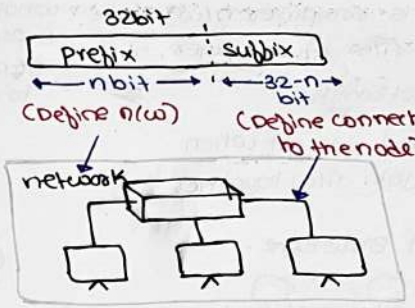
Point to remember

- IP address is for connection not for host or router. b/c if device moved to another n/w the IP address may be changed
- if your device is connected to 2 connection of Internet via 2 n/w then it will have 2 I/P address.



• Hierarchy in Addressing

[Ip address is divided into prefix, suffix]



• Summary

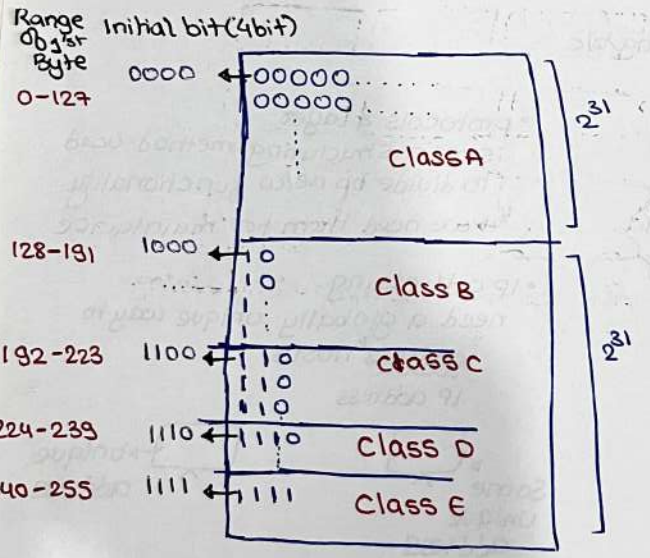
* 32bit no. (in binary)
* Dotted decimal form
  i.e 128.11.3.31
* we divide into 2 parts.

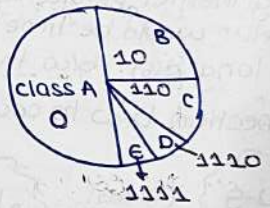
128 . 11.3.31

Suppose these division

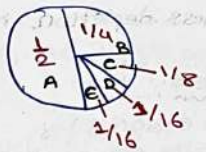
Classful addressing : address space
* is divided into 5 class
  A, B, C, D & E

total IP address : 2^{32} address space





- Suppose we uniformly randomly pick an IP address, then what is the prob. it belong to class E? $\frac{1}{16}$

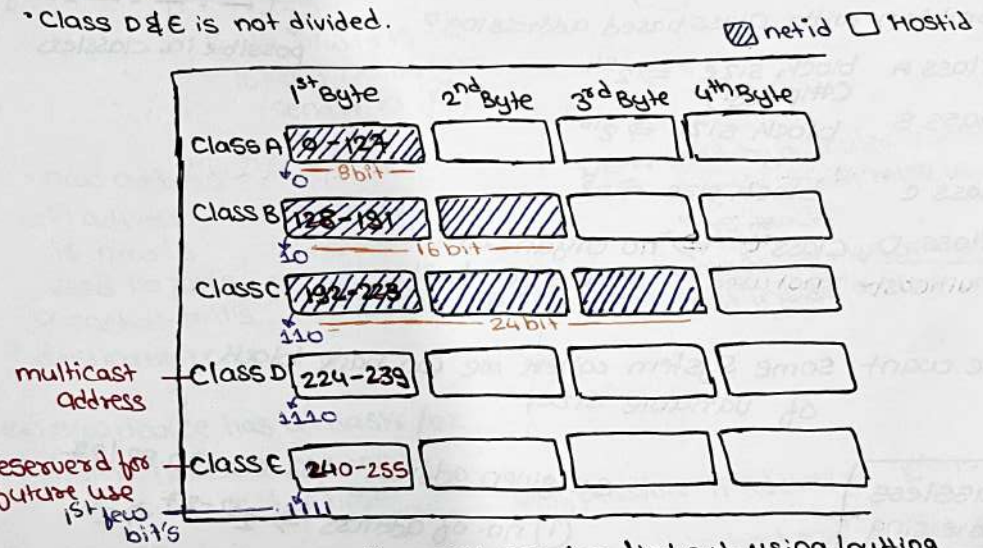


$\Rightarrow \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} = 1$

- given any IP address we can tell
  1. class of the IP address
  2. n/w id of the IP address

IP address is divided into
- Netid
- Hostid

- Class D & E is not divided.



- Default mask can be find out using./putting netid ⇒ All 1's & host id ⇒ All 0's.

- finding n/w id (beginning address)

[given IP address] & [Default mask of that class]

n/w id

• Address depletion is there so we don't use we use classless addressing

▸ Class B
were designed for mid size organization but many are remained unused

▸ Class C
complete different blaw in design b/c each n/w has 256 address which is very small for any organiza^

▸ Class E
almost never used

• In classful we can divide in 5 class only
• if we want to divide anywhere we can't do that

division
[—·—+—·—]
possible in classless

• problem with class based addressing?

Class A block size (#IP add.) $\Rightarrow 2^{24}$

Class B block size $\Rightarrow 2^{16}$

Class C block size $\Rightarrow 2^{8}$

Class D, Class E $\Rightarrow$ no organ^ can take for host add.
multicast not used.

• we want some system where we can have block of variable size.

Classless addressing

• there is no class hence n/w id (prefix) is not implicit.
• we need to specify prefix explicity

/n

ex: 12.24.76.8 → IP address

12.24.76.8/8 → one network
↳ length of n/w id

12.24.76.8
n/w id — remaining are host id

ex: given address 167.199.170.82/27

① no. of address $\Rightarrow 2^{32-27} \Rightarrow 2^5$

167.199.170.82
010.....
27 bit

• first address: 167.199.170.64

• last address: 167.199.170.95

range of IP

64

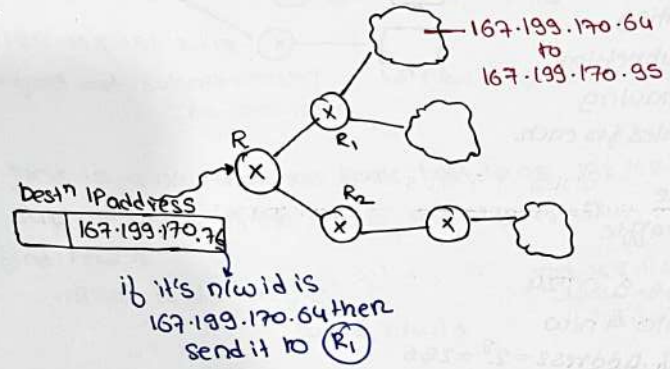
n/w id — 167.199.170.01000000

Broadcast address — 167.199.170.01011111
95.

nlw id

167.199.170.010 00000 /27 ← I want to go to this host — 167.199.170.76 } IP address

64



- nlw address.
- 1st address is nlw id used to route a packet to it's destination nlw
- each nlw is identified by it's nlw address.

DBA: used by a router to send a msg to all host in the block

LBA: broadcast msg only to host inside a nlw

ex: nlw device has a mask /28

then ip address possible. for host

$2^{32-28} \Rightarrow 2^4$

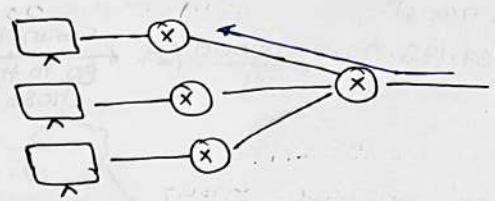
1st ipaddress: nlw id

last ip address: Broadcast address (DBA)

# of possible ip address = $2^4 = 16$

# of possible host = $2^4 - 2 = 14$

Subnetting



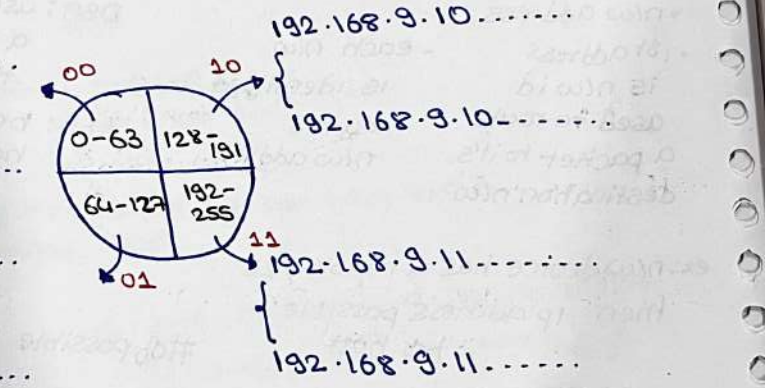
more level of hierarchy can be created using subnetting

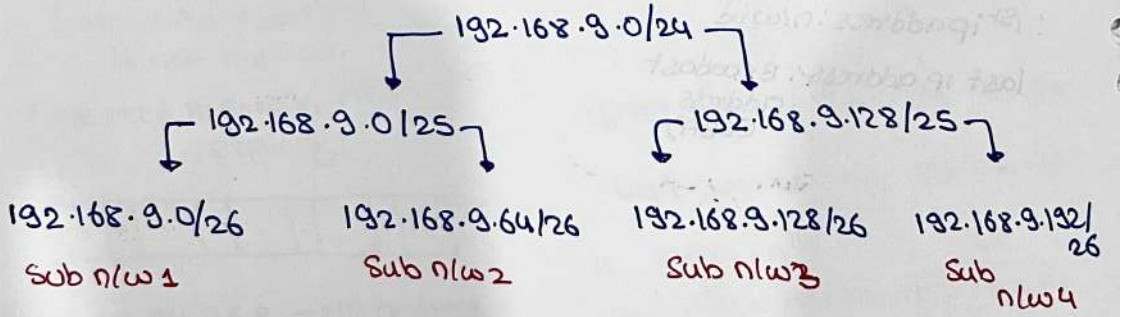
Purpose of subnetting

→ useful for having specific rules for each router.
→ maintainence
→ to reduce traffic

Divide 192.168.9.0/24 into 4 n/w

no. of address = $2^8 = 256$





Subnet mask:
put all 1's in nw id

ex: 192.168.9.0/24
↓
1111...1 1111...1 1111...1 000...0
Subnet mask of this nw : 255.255.255.0

ex: 192.168.192.1/17
↳ find out subnet-mask?
255.255.128.0
↳ nw id ⇒ 192.168.128.0

ex: there is a nw that has some host ip as 172.168.224.3 &
Subnet mask of the nw is 255.255.192.0
find nw id.
initial 18 bit are fixed part of nw id

172.168.11000000.0
nw id.

Slash notation: 172.168.192.0/18.

172.168.224.0 ← ip address of some host
& 255.255.192.0 ← Subnet mask of nw
nw id 172.168.192.0

ex: Divide 192.168.9.0/24 such that nw has 10 host

earlier in question we divide the ip into 4 network
192.168.9.0

no. of host bit required = 4
192.168.9.0
host id

▸ we have 8 bit for host

192.168.9. 4bit 4bit
Subnet id host id

Subnet 1: 192.168.9.0000____ /28

I/p address
- 192.168.9.00000000 } Subnet id → 0
- ⋮
- 192.168.9.00001111 } subnet broadcast address → 15

192.168.9.0/24

0-15	96-111	.....
16-31	112-127	
32-47	⋮	
⋮		
80-95		

16 id's
⇒ 16-2
⇒ 14 host possible

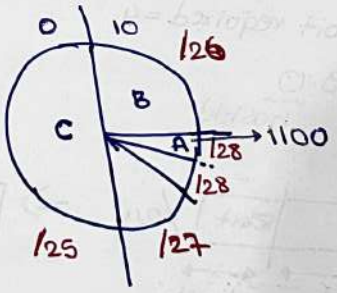
ex: An organization block address that start with 14.24.74.0/24

14.24.74.0/24
↓
256 address

They need 3 subblock of address use in 3 subnets

Subnet 1 - 60
Subnet 2 - 60
Subnet 3 - 120

one subblock is 10 address
----11------ is 60 ... 11---
----11------ is 120 address.



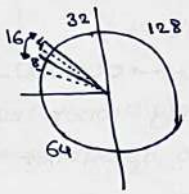
we have divided into 5 part out of which 3 are allocated to A, B, C

A: 14.24.74.192/28
B: 14.24.74.128/26
C: 14.24.74.0/25

• in previous questn it was given that no. of subnet=4 no. of host in each nlwrk=10

ex: given ip address

210.168.200.0/24 ⟶ 2^8 = 256 ips



- Subnet 1 — 2 host — 4 ips
- Subnet 2 — 28 host — 32 ips

we round up to power of 2 as ip

So, Subnet$_1$: 210.168.200.0/24+6 = 30 /30

6 subnet bit, 2 host bit

Subnet$_2$: 210.168.200.0/24+3 = 27 /27

3 subnet bit, 5 host bit

ex: Divide 172.16.0.0/16 into 7 nlw such that

net1 500 ip's ⟶ 2^9 ⟶ Subnet id: 7
net2 200 ip's ⟶ 2^8
net3 100 ip's ⟶ 2^7
net4 60 ip's ⟶ 2^6
net5 20 ip's ⟶ 2^5
net6 2 ip's ⟶ 2^1
net7 2 ip's ⟶ 2^1

nlw id	subnet	host id
16	7	9

172.16.0.0/16 ⟹ 2^{16} address available

assign a block of 2^9 address

no. of subnet bit = 7 (7bit, 9bit)

to start with we are 1st borrowing 7bit from host id.

no. of Subnet = 2^7

172.16.0.0

172.16.254.0 ← last Subnet id.

1bit

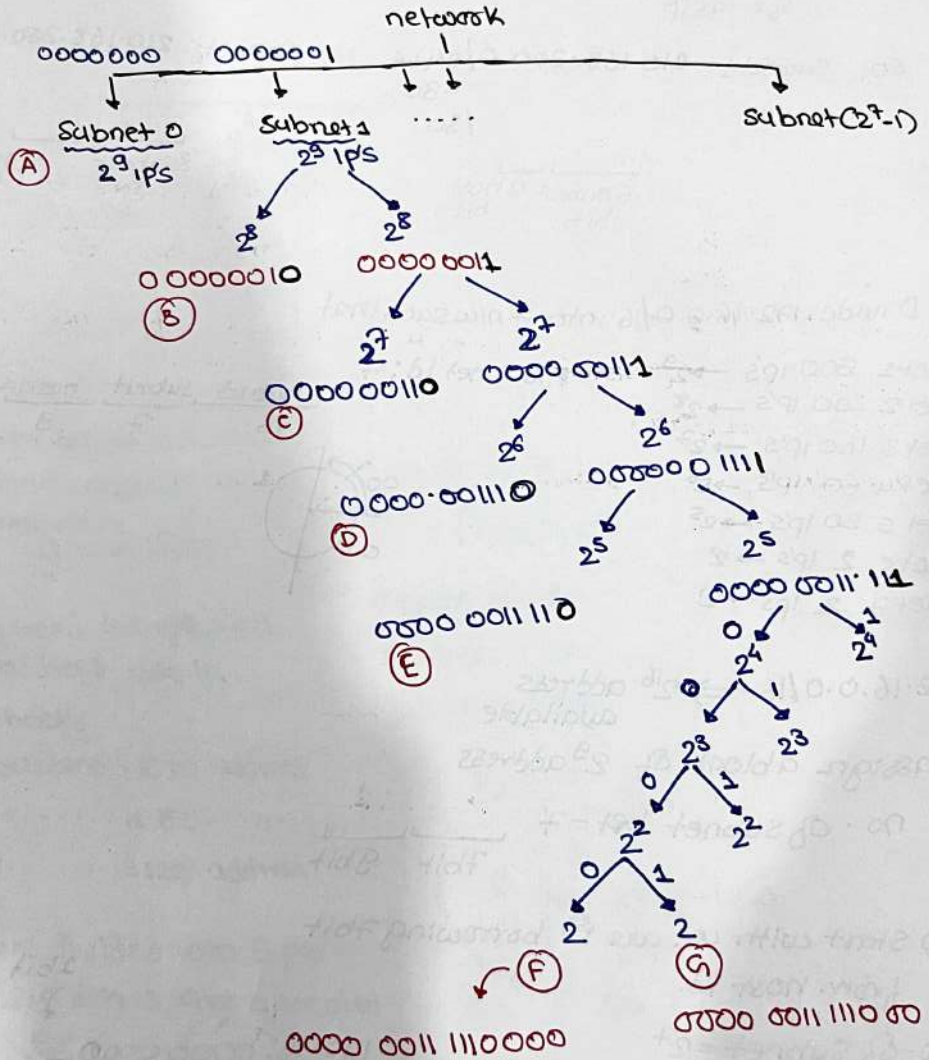
172.16.0000 000 — . ——
000 0001 — . ——
⋮
1111111 — . —— 8bit

2^7 subnet

A ⟶ 000000 — · ~~~~

every subnet has ⇒ 2^9 ip's

no. of subnet ⇒ 2^7 subnets



In previous ques.

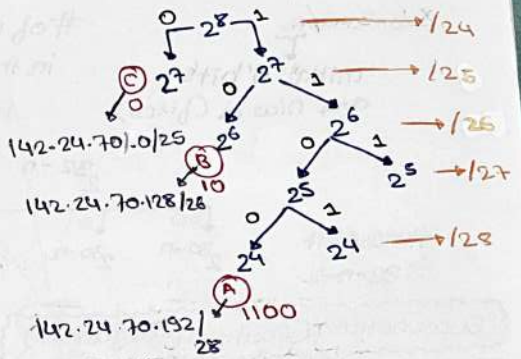
142.24.70.0/24 $\Rightarrow 2^8$ address (ip's) available

& we need subblock of

(A) Subnet$_1$: 10 $\Rightarrow 2^4$

(B) Subnet$_2$: 60 $\Rightarrow 2^6$

(C) Subnet$_3$: 120 $\Rightarrow 2^7$



Gate CSE

2012

An ISP has chuck of CIDR based IP address.

245.248.128.0/20 $\Rightarrow 2^{12}$ I'p address we have

↳ half is given to organizat^n (A)

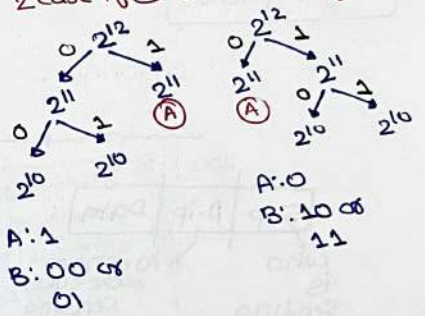
↳ quarter to organ^n (B)

↳ & retaining the remaining with itself.

So, 245.248.128.0/20 $\Rightarrow 2^{12}$

128 → 1000

2 case if (A) goes two one of the group



So A/c to option

245.248.136.0/21 & 245.248.128.0/22.

136 → 10001000

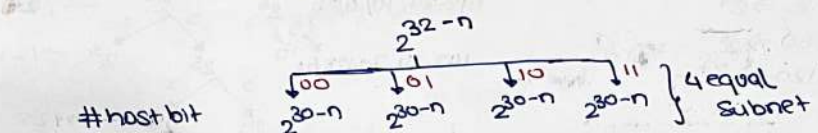
128 → 10000000

x.y.z.w/n

initial 'n' bit's are n/w id (fixed)

# of ip address in this n/w : 2^{32-n}

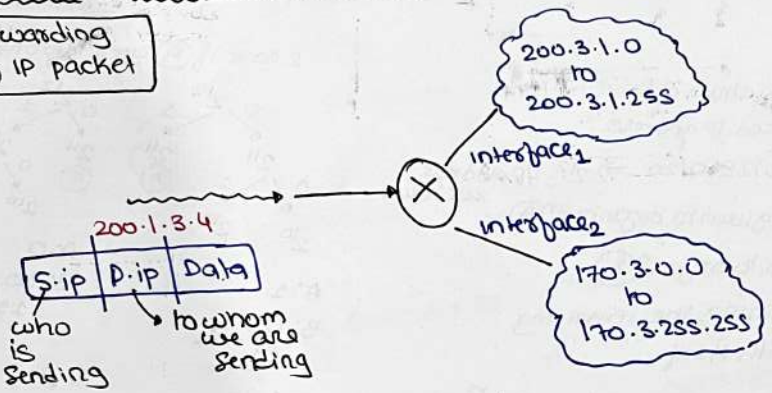
Host bit : 32-n



host bit
⇒ 32-n-2

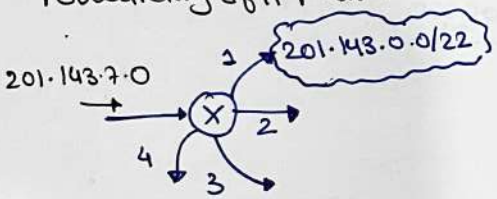
Supernetting (Route aggregation)

Forwarding of ip packet



Based on given dest. ip address, router will send it to interface 1

Forwarding of IP packet



201.143.7.0 need to be transfer

forwarding table

Prefix	Port
201.143.0.0/22	1
201.143.4.0/24	2
201.143.5.0/24	3
201.143.6.0/23	4

Range:

201.143.0.0 to 201.143.3.255

201.143.4.0 to 201.143.4.255

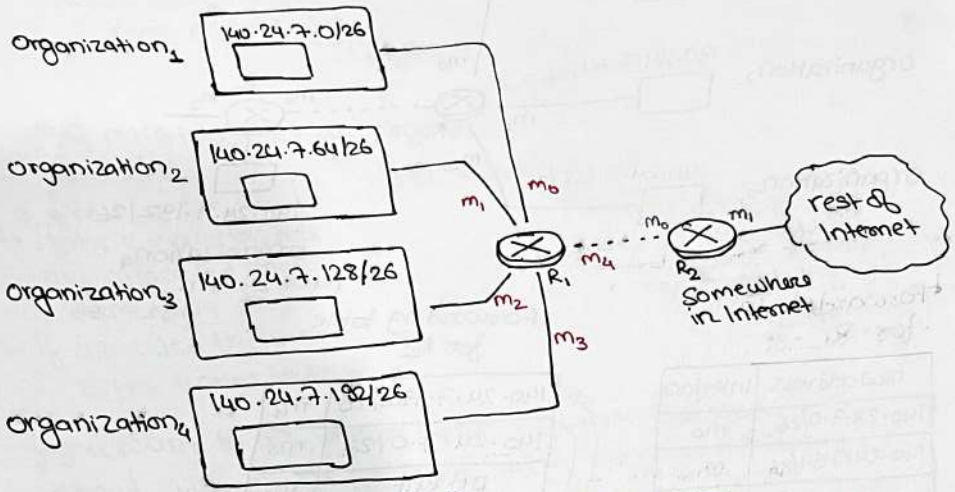
201.143.5.0 to 201.143.5.255

201.143.0000 0110.0 to

201.143.0000 0111.255

we will send using port 4 201.143.6.0 to 201.143.7.255

ex: forwarding table for R_1?



For R_1

140.24.7.0/26	m_0
140.29.7.64/26	m_1
140.24.7.128/26	m_2
140.24.7.192/26	m_3
Default	m_4

⇓

no common interface
So we can't aggregate

For R_2

140.24.7.0/26	m_0
140.24.7.64/26	m_0
140.24.7.128/26	m_0
140.24.7.192/26	m_0
Default	m_1

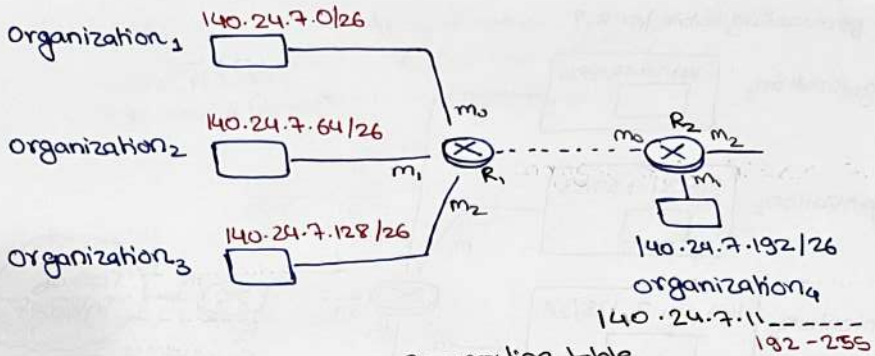
$4 \times 2^6 = 2^8$

⇓

140.24.7.0/24	m_0
Default	m_1

- whatever we haven just seen to aggrate R_2 entries is called supernetting (or) Route Aggregation

- Subnetting is a physical concept in the sense that we actually need to put router to have different Subnet But Route Aggregation is just logical Concept at Router forwarding table.



Forwarding table for R_1

n/w address	Interface
140.24.7.0/26	m_0
140.24.7.64/26	m_1
140.24.7.128/26	m_2
Default	m_3

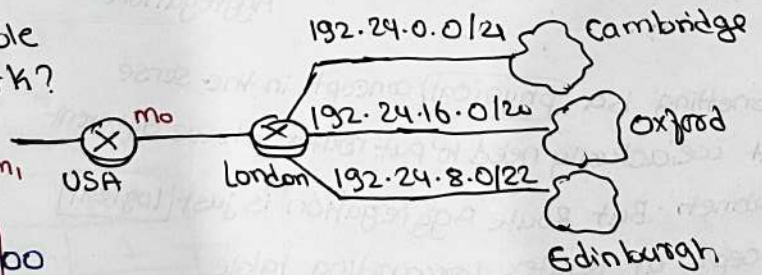
Forwarding table for R_2

140.24.7.192/26	m_1
140.24.7.0/24	m_0
Default	m_2

140.24.7.200 will be matching to both but will be sent to the longer one (longest matching)

- So here we are combining organization 1,2,3 together & making one entire for that in table
- one separate (entire) row for organizat^n 4 as it missing as it is somewhere else

ex Routing table for new york?



192.24.00000
192.24.000
192.24.000010

/19

190.24.0.0/
Default

Separately 3 n/w has: $2^{11} + 2^{12} + 2^{10} \Rightarrow 2^{10}(2+4+1)$
6×2^{10}

now aggregated

↳ 2^{13} ip's $= 2^{10} \times 2^3$
$= 8 \times 2^{10}$

# of extra ip's after aggregation

↳ $8 \times 2^{10} - 6 \times 2^{10} = 2 \times 2^{10}$

$= 2^{11}$ → missing or extra ip's is covered by 192.24.0.0/19

1. 2 case
   if these extra ip's are not allocated then do nothing
2. if allocated then have extra entire for that

- if you want to aggregate some n/w then just go ahead just check till what point thing are matching

140.	24.7.00_____	140.24.	7.00	140.24.7.	00_____
140.	24.7.01_____	140.24.	7.01	140.24.7.	01_____
140.	24.7.10_____	140.24.	7.10	140.24.7.	10_____
140.	24.7.11_____	140.24.	7.11	140.24.7.	11_____

most specific option

if after aggregation, you cover extra ip's then mention that entry separately & our longest prefix match will takecare.

Misconceptions

1. addresses should be contigious
2. no. of subnet to be combined should be in 2^k

ex:

Destination	nxt hop
10.1.0.0/24	R_3
10.1.2.0/24	eth1
10.2.1.0/24	eth0
10.3.1.0/24	R_3
20.2.0.0/16	R_2
20.1.1.0/28	R_2

20.2.00 to 20.2.255.255

Destination	nxt hop
10.1.0.0/24	R_3
10.1.2.0/24	eth1
10.2.1.0/24	eth0
10.3.1.0/24	R_3
20.0.0.0/8	R_2

↳ not contigious but still combining

- routing table help.
  - ↳ combining routing table entires whenever all node with same prefix share same hop
  - ↳ longest prefix match for forwarding.

	Address	mask	next hop	
1	128.42.222.3	255.255.255.0	R_1	/24
2	128.42.128.4	255.255.128.0	R_2	/17
3	18.0.0.0	255.0.0.0	R_4	/8
4	0.0.0.0	0.0.0.0	R_4	/0
5	128.42.127.3	255.255.248.0	R_1	/21
6	128.42.216.0	255.255.248.0	R_1	/21
7	128.42.128.4	255.255.0.0	R_3	/16

we can combine 1,5,6

1. 128.42.1101 1110.3/24
5. 128.42.0111 1111.3/21
6. 128.42.1110 0010.0/24

after combing it would be 128.42.0.0/16 is wrong as /16 is already there for 128.42.128.4/16.

we try to combine with 1 & 6.

1. 128.42.222.3/24 → 128.42.192.0/18
6. 128.42.216.0/21 →

	Address	mask	nxt hop	
1	128.42.128.4	255.255.128.0	R_2	2
2	0.0.0.0	0.0.0.0	R_4	3+4
3	128.42.127.3	255.255.248.0	R_1	5
4	128.42.216.0	255.255.248.0	R_1	1+6
5	128.42.128.4	255.255.0.0	R_3	7

Default entry every incoming packet match here.

aggregated entries.

Default: 0.0.0.0/0

Practice quest.

ex IP: 18.26.0.127

Class A

① if it break into 32 diff. Subnet than subnet mask?

#Subnet: 32 $\Rightarrow 2^5$

5bit

#bit n/wid: 18.0.0.0 → have n/w mask as 255.0.0.0

after dividing the subnet mask will be

Subnet-mask: 255.11110000.0.0

$\Rightarrow$ 255.248.0.0.

the Subnet to which it belong to is

18.00011000.0.127/13

Subnet 3

② IP: 14.24.74.0/24

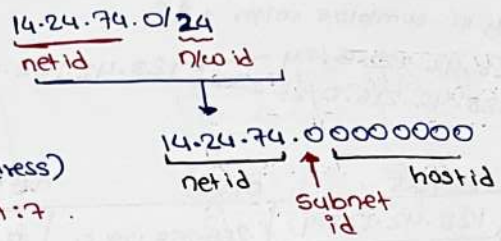
3 subblock division

A: 10 address → 2^4

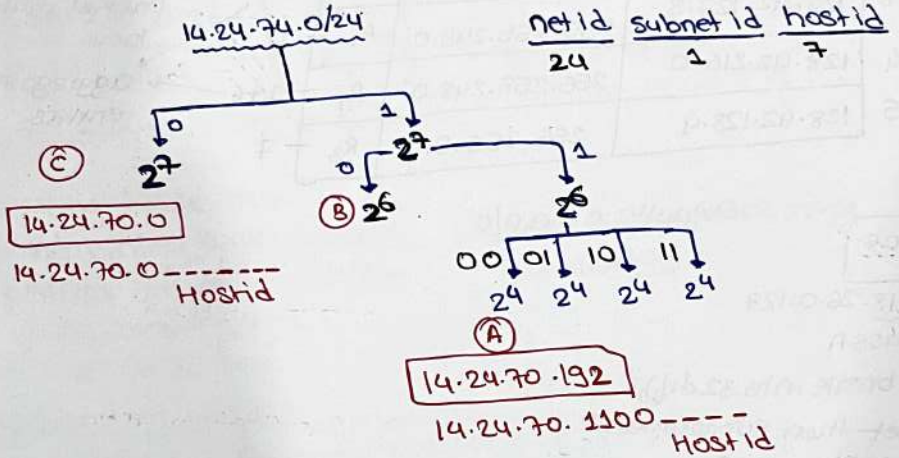
B: 60 ... → 2^6

C: 120 ... → 2^7 (ip address)

↳ host bit : 7



# of subnet = 2



gate CSE 2003

ex: Subnet mask : 255.255.31.0 . which pair of IP belong here

(a) 172.57.88.62 & 172.56.87.23. b/c 1st 2byte are different.

(b) 10.35.28.2 & 10.35.29.4 → so after "4" with subnet mask they will be different

even odd

(c) 191.203.31.87 & 191.234.31.88

(d) 128.8.129.43 & 128.8.161.55.

↓

129 : 10000001

161 : 10100001

28 : 11100

29 : 11101

gate 2006

ex: Two computers

C_1 has IP: 203.197.2.53

netmask: 255.255.128.0

C_2 has IP: 203.197.75.201

255.255.192.0

C_1 netid: 203.197.0_ _ _ _ _ _ _ ← subnetting

C_2 netid: 203.197.01_ _ _ _ _ _

[C_2 think C_1 is in diff. n/w

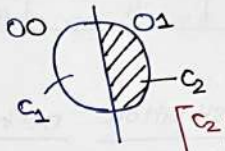
C_1 think C_2 is in my n/w

203.197.0.0/17

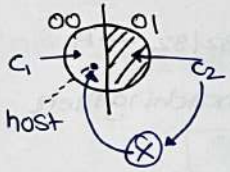
C_1 is in this n/w

203.197.64.0/18

C_2 is [illegible] n/w

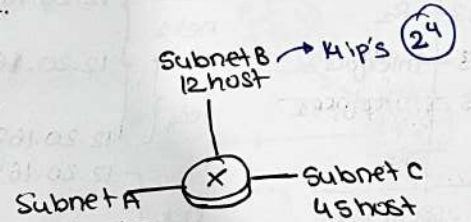


[C_2 will think C_1 is outside the n/w]



- if some host is in different n/w & I want to send a packet then 1st I send to default gateway (Router).
- if C_1 want to send to host 'H' then it can send directly.
- if C_2 want to send to host 'H' then it has to send to default gateway

③ 3 subnet A, B, C.



A: 255.255.255.226

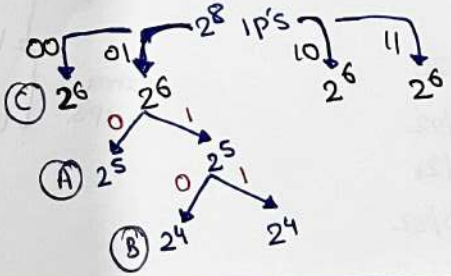
C: 255.255.255.128

x.y.z. 6bit

Subnet id

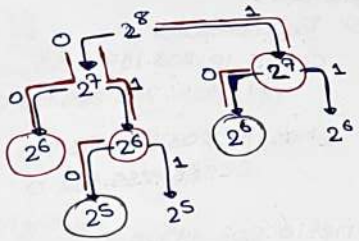
x.y.z.0	0
x.y.z.0	10
x.y.z.0	110

x.y.z.0/25



④ 8 bit address

Prefix	Interface	#host
00	0	64
010	1	32
01	2	64-32=32
1	3	128-64=64
10	4	64



⑤

new Destination	nxt hop
142.150.64.0/20	A
142.150.71.128/28	B
142.150.71.128/30	C
142.150.0.0/16	D

(i) IP address with 140.150.71.132 should go to Hop 'A'

add in table

140.150.71.132/32 — A

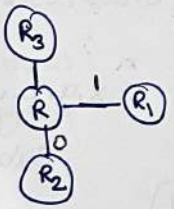
(ii) if nothing is matching then redirect to C.

0.0.0.0/0 — C

gate 2022

ex:

Subnet no.	nxt hop
12.20.164.0/22	R_1
12.20.170.0/23	R_2
12.20.168.0/23	Interface 0
12.20.166.0/23	interface 1
default	R_3



assuming;

$R_1 \equiv$ Interface 1

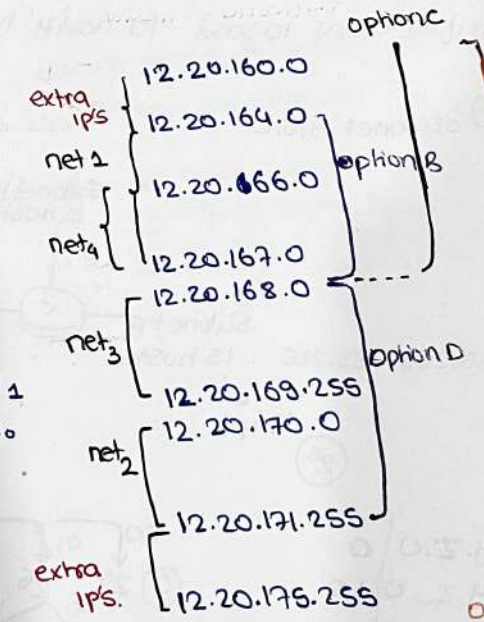
$R_0 \equiv$ Interface 0

~~a~~ 12.20.164.0/20

~~b~~ 12.20.164.0/22

~~c~~ 12.20.164.0/21

~~d~~ 12.20.168.0/22



Gate 2008

ex: given:

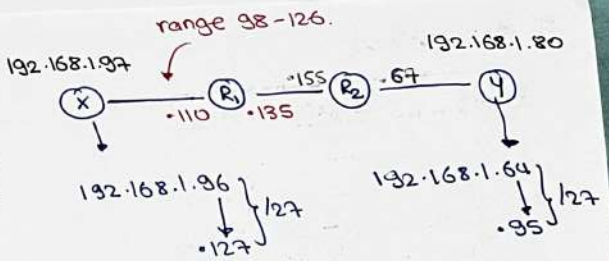
Host x ip's: 192.168.1.97

Host y ip's: 192.168.1.80

R_1 has IP: 192.168.1.135 & 192.168.1.110

R_2 has ip's: 192.168.1.67 & 192.168.1.155

netmask: 255.255.255.224



• unique range we have ≡ no. of subnet

So, host x: 192.168.1.97

01100001

④

netmask: 255.255.255.224

⇒

net id 192.168.1.011_____/27

range

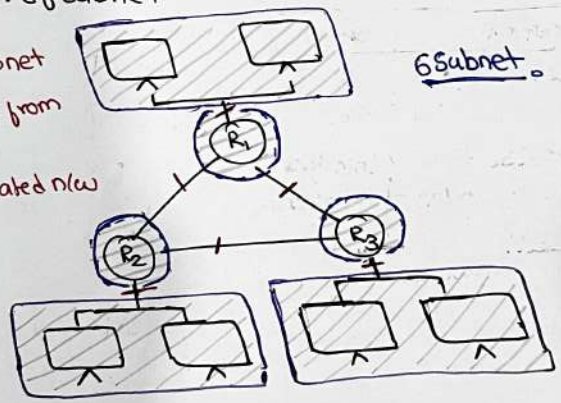
98 – 126

⑥ how to find no. of subnet

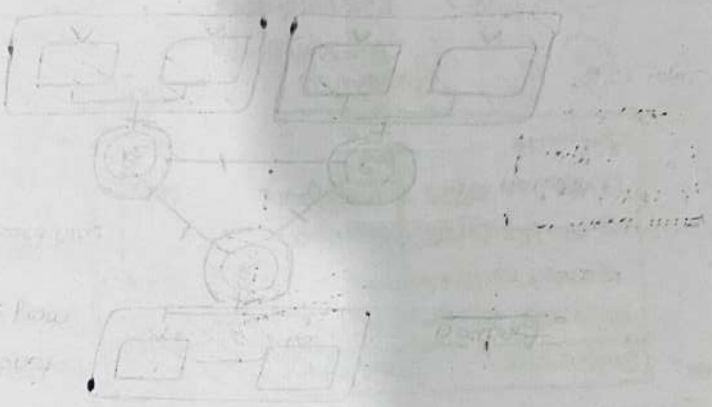
to determine the subnet

detach each interface from its host or router

• creating island of isolated n/w



6 subnet.

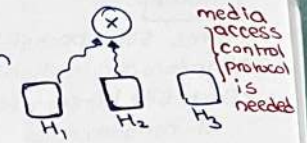


Media Access control (Aloha)

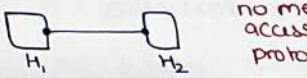
Point to point
dedicate pairwise
Communication

vs

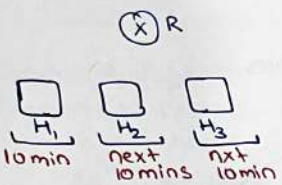
Broadcast medium
Shared wire or medium



media access control protocol is needed

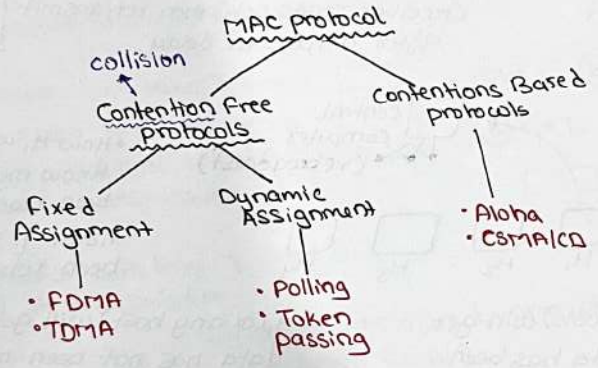


no media access control protocol is needed



medium access control
A distributed algo. running at node to determine who should transmit over a shared link

- Channel partitioning
  - FDMA
  - TDMA
- turn taking
  - polling
  - Token passing
- Random access
  - Aloha
  - CSMA/CD



- Random access protocols

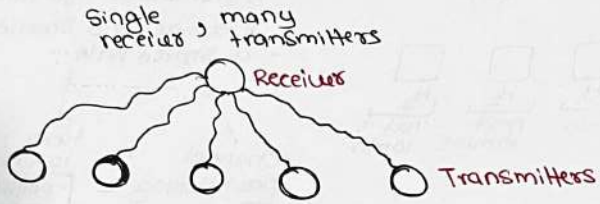
(1) what if node decide to transmit randomly
Collision can happen

(2) when node has packet to send
- Transmit at full Channel data rate "R"
- no a priori coordination among node

(3) collision possible when two or more transmitting node choose to send Simultaneously

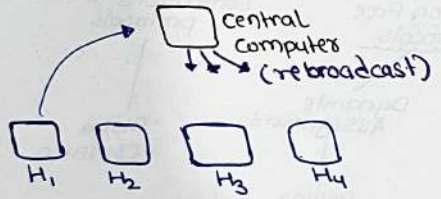
Aloha

- Sites send packet to hub
  - Random access channel
  - Each site transmit packet at random times
  - if packet not received (due to collision), site resend.
- Hub sends packet to all sites
  - → Broadcast channel
  - → Sites can receive even if they are also sending



- Aloha protocol in a nutshell:
  - when you have data send it
  - if data doesn't get through (receiver sends ack) then retransmit after a random delay.

} in collision then retransmit after random time



- How H_1 will get to know my data has been reached to H_3?
  listen if data has been rebroadcast.

- H_1 (or any host) will get to know data has been reached using Rebroadcast
- H_1 (or any host) will get to know data has not been reached using Collision

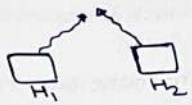
- Pure Aloha

Station sends a frame whenever it has a frame to send

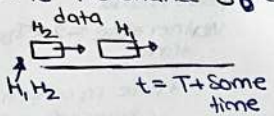
if there is a collision then resend it (after some random time)

b/c if both colliding stations send at fixed time then there will be a collision again

Any overlap of packet ⇒ Collision



- if frame of different stations are going in same direction (to central node) then after transmition there is no chance of collision.



- Collision can happen ⇒ whenever 1000 (or more) Station transmitting simultaneously

B transmission will take T_t time

at t=0 B start it transmission

Tell us, till what time another station can not start transmission

(a) T_t (b) $2T_t$

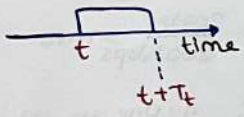
(b) $T_t + T_p$ (d) $5T_t$

- B transmission will take T_t time at t

B start it transmission

till what time after t, any other station should not start?

$t + T_t$



- Suppose B want to transmit at some time "t" then give the range of time in which there should be no other transmissions? (Any other time which is not in the range doesn't create any problem)

(a) $[t, t+T_t]$

(b) $[t_t; t+5T_t]$

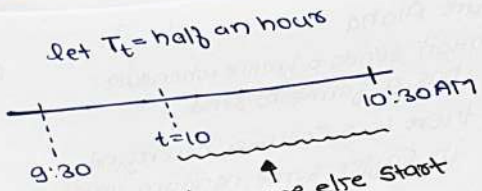
(c) $[t-T_t, t+T_t]$

(d) $[t, t+2T_t]$

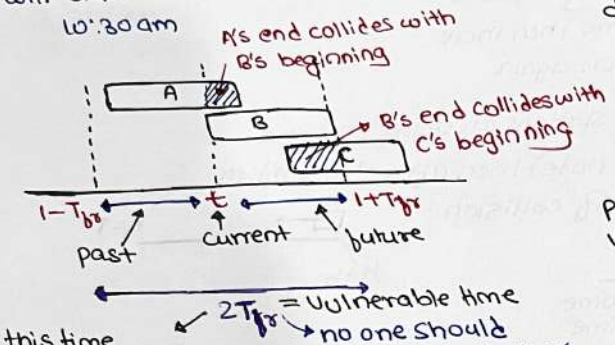
1. we need to make sure that no one has already started transmission after 9:30am

2. we need to make sure no one will start the transition before 10:30 am

let T_t = half an hour



if anyone else start transmission then definetely collision



$2T_{fr}$ = Vulnerable time

In this time Just one station Should transmit

no one should transmit other than given station.

Pure Aloha Vulnerable time = $2 \times T_{fr}$

time in which chance of collision

- pure Aloha -

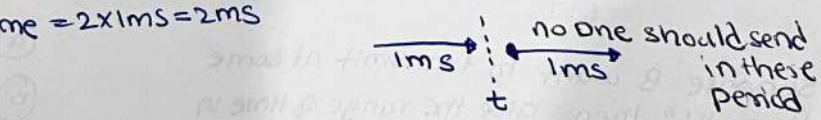
when will the transmission at time "t" is successfull in pure Aloha? If only one station transmits in $[t-T_t, t+T_t]$ time

ex: pure Aloha,

n/w transmits 200bit frame on a shared channel of 200Kbps requirement to make collision free

Avg. frame transmission time (T_{fr}) is $\frac{200bit}{200Kbps} = 1ms$

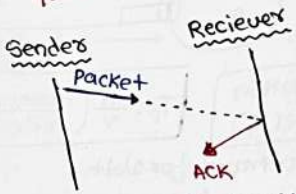
Vulnerable time = $2 \times 1ms = 2ms$



Flow Control

assume there is no errors, can host1 keep on sending data to host2?

what if host2 doesn't enough buffer? there could definetely be the case that channel is noisy

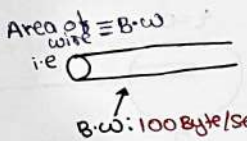


Basic idea of any flow control (ARQ)

techenical defination

1. Bandwidth

max. amount of data that can be transfer per time unit

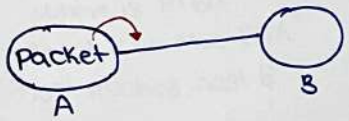


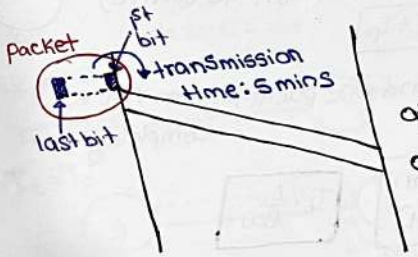
in one sec we can put 100 Byte on wire

ex: then in 5 sec., how much amount of data we can put on channel?

⇒ 5 sec × 100 Byte/sec = 500 Byte

2. Transmission & propagation delay

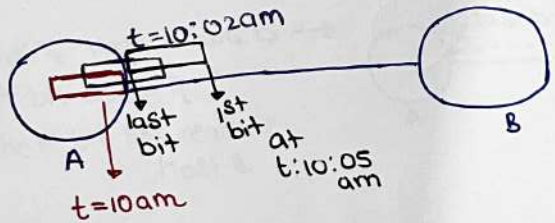




at 10 am

at what time, last bit of data will be put on channel?

10:05 am



transmission time depend on length of data & also B·w of channel

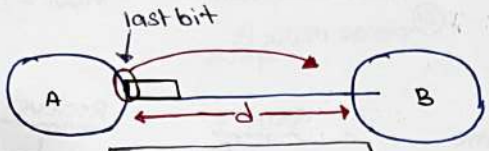
$T_t = \frac{L}{B \cdot w}$ ← Bytes

↳ byte/sec

(T_t is in sec)

ex: How much time would you take to transmit 1000B of data on a channel having B·w: 100B/sec

$T_t: \frac{1000}{100} = 10sec$



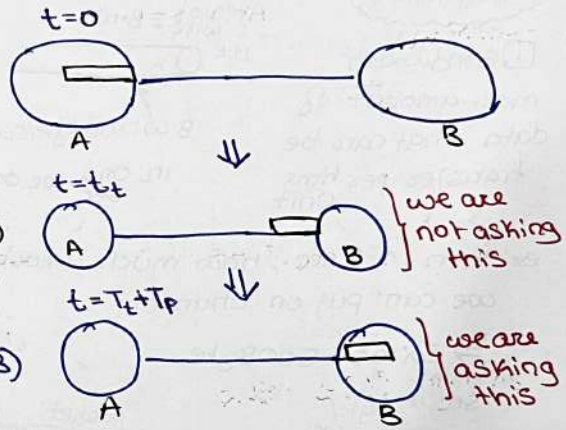
Propagation time (T_p)

$T_p: \frac{d}{v}\left(\frac{distance}{velocity}\right)$

time to travel for 1bit

↳ Depend on distance & speed at which bit is travelling

ex: How much time it takes from when A thinks (just start the transmission) about the transmission to B Completely get the packet (even the last bit is with B)



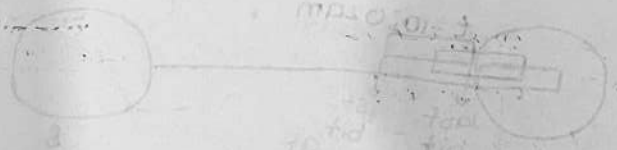
$T_t + T_p$

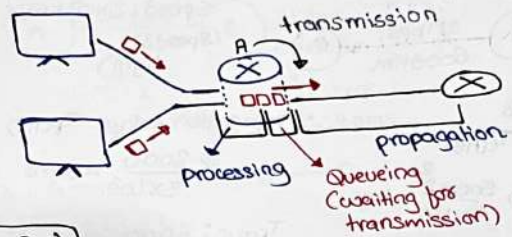
time to send one packet from A to B Completely

transmission delay (T_t)

$T_t: \frac{L}{Bw}$

↳ Time taken to push all the bits of a packet into a link





End to End Delay

Summation of all delays

$T_t + T_p + T_{queueing} + T_{processing}$

generally: $T_t + T_p + T_q + T_{process}$ (0, 0)

$T_t + T_p$

ex: consider 2 host A & B connected by single link

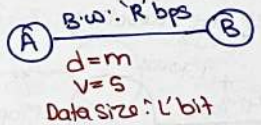
B.w: 'R' bps

Dist.: 'm' meters

propagation speed: 's' m/sec

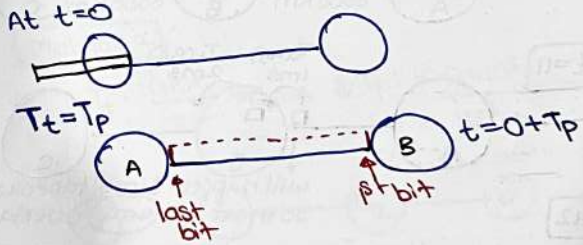
packet size: 'L' bit

(a) host A start transmit at t=0 then at $t=t_t$ where is last bit? just leaving host A



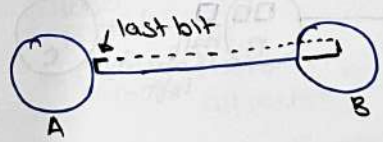
(b) $T_p > T_t$ then at $t=T_t$ → where is 1st bit.

1st bit is on the link not reached host B

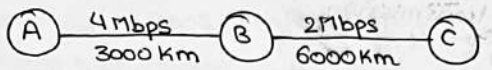


(c) $T_p < T_t$ then where is 1st bit at $t=T_t$.? →

the 1st bit has reached host B.



ex:



Speed: 3×10^5 km/s (Speed of light)

- if A sends 500 Byte to B

  transmission delay $T_{t(AB)}$

  L: 500B = 500 x 8 bit

  B.w: 4×10^6 bps

  $\Rightarrow \frac{500 \times 8}{4 \times 10^6} = 1ms$

- propagation delay $T_{P(AB)}$

  $\Rightarrow \frac{3000}{3 \times 10^5} = 10ms$

  $T_{P(BC)}$: $\frac{6000}{3 \times 10^5} = 20ms$

- end to end delay if A send 500B to B

  1 + 10 = 11ms

- end to end delay if A send 500 Byte to C.

  A — B + B — C

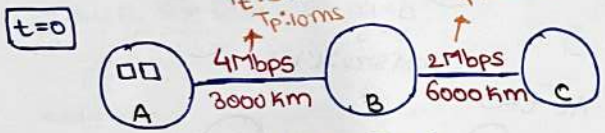
  $T_t + T_p$ $T_t + T_p$

  1+10 + 2+20 $\Rightarrow$ 33ms.

  1ms to transfer one packet

each node "store & forward" every packet

- Diagram to showcase how 2packet are traveling

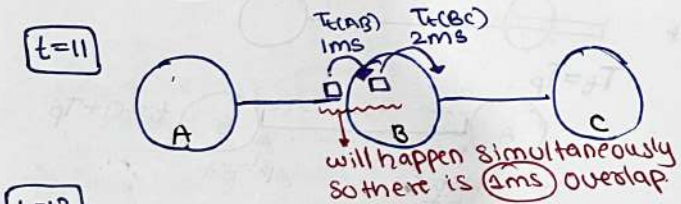


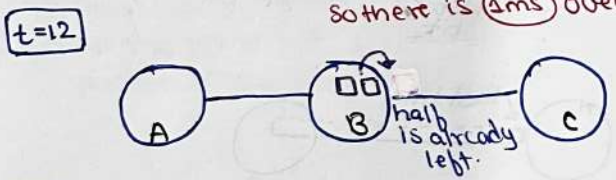
2ms + 10ms $\Rightarrow$ 12ms

to transmit both packe

P_d

means that last bit of packet reach B



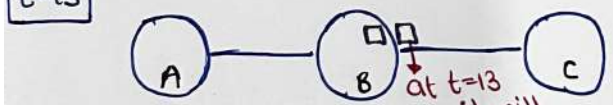


time taken to push on 2nd link

So, 13 + 2 + 20

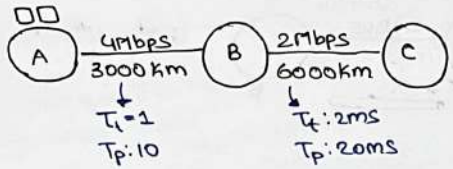
push 2nd packet on 2nd link

Tp time so last bit to reach



t = 13 + 2 + 20

$\Rightarrow$ 35



So, A — B

time for last bit to transfer to get started for last packet: 1+1

1+1+10 total time to transfer.

B — C

1+2+20 total time taken to transfer

we only took 1ms b/c 1ms was already used for 1st bit.

Gate 2012

ex: Source comp.(S)

file size: 10^6 bit

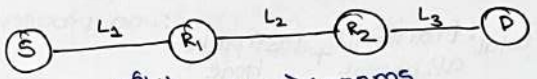
Dist. of each length: 100km

Speed: 10^8 m/s

B.w: 1Mbps

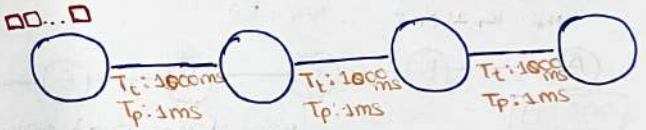
file broke into 1000 packet of size 1000bit

total sum of $T_t + T_p$?



T_t: $\frac{10^6 \text{bit}}{10^6 \text{b/s}}$ = 1sec ⇒ 1000ms

T_p: $\frac{100 \times 10^3}{10^8 \text{m/s}}$ ⇒ 1ms



last packet time:

$3 \times T_p + 2 \times T_t$ ⇒ $(3\times1)+(2\times1)$

⇒ 5ms.

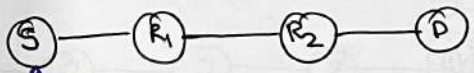
Transmission time of source is already included in 1000ms.

So, Total time ⇒ 1000+5

⇒ 1005ms.

Method 2

T_t: $\frac{1000}{10^6}$ = 1ms & T_p: $\frac{100 \times 10^3}{10^8}$ = 1ms

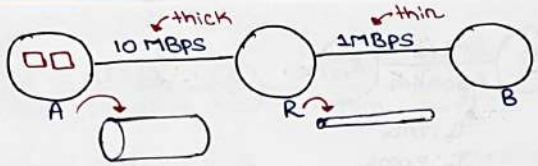


1000 packet

$T_{end\text{-}end}$: transmit all packet + last packet journey

$1000T_t + T_p + 2(T_p + T_t)$

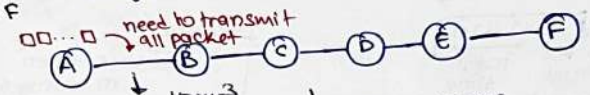
⇒ 1005ms.



ex: we want to send 20kB from A to F

Packet size : 1kB
B.w : 10 Mbps
Dist. link: 10km
• time taken to send entire file
Propagation speed : 2×10^8 m/s

# of packet : 20 , B.w = 10Mbps , d: 10km



$T_p : \frac{10 \times 10^3}{2 \times 10^8} = \frac{1}{2 \times 10^4}$ sec $= 50$MS

$T_t : \frac{10^3 B}{10 Mbps} = \frac{10^3 \times 8 \text{ bits}}{10 \times 10^6 \text{ b/s}} = 800$MS

∴ no processing no queoeing delay.

T_{total} : transmit all packet + last packet time

$\Rightarrow 20T_t + T_p + 4(T_p + T_t)$ (last packet at P')

$\Rightarrow 24T_t + 5T_p$

one big packet

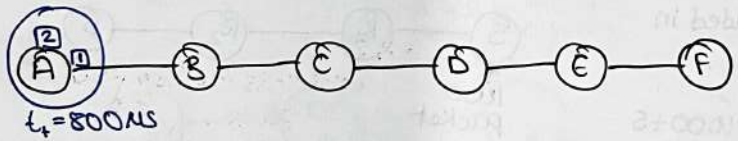
T_p : 50MS

new T_t : $20T_t$

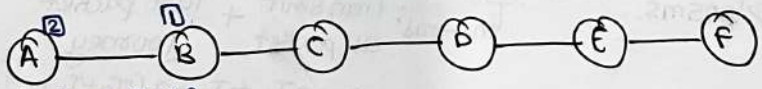
time if you send only as one big packet

$5(T_p + \text{new}T_t)$

$\Rightarrow \boxed{100T_t + 5T_p}$

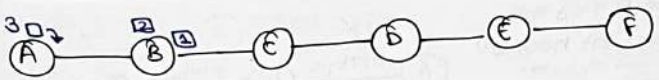


$t_t = 800$MS



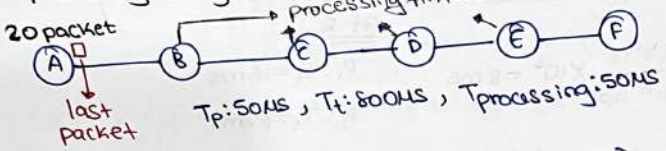
$t = 800 + 50$MS

$t = 800$MS $+ 50$MS $+$ 750MS



Ex: Assume now we also have.

Processing delay: 50ms.

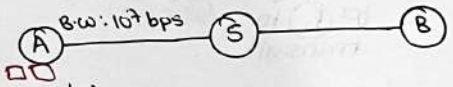


T_p: 50ms, T_t: 800ms, $T_{processing}$: 50ms

total time: $20T_t + T_p + 4(T_{process} + T_t + T_p)$

gate2015

ex:



T_p: 20ms

$T_{processing}$: 35ms

packet size: 5000bit

⇒ 10,000 bit data is to be send.

- time elapsed b/w the transmission of the 1st bit & reception of last bit of the data

T_t: $\frac{5\times10^3}{10^7} \times 10^6$ ms = 500 ms

T_p: 20ms

$T_{processing}$: 35ms

⇒ $2T_t + T_p + T_{process} + T_t + T_p$

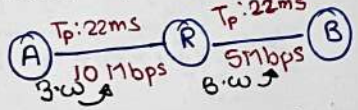
(last bit take to reach 'S'; last bit take to reach 'B' from 'S')

⇒ $3T_t + 2T_p + T_{processing}$

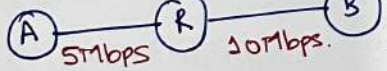
⇒ 1500 + 40 + 35

⇒ 1575ms

ex: Host A & B via 'R'

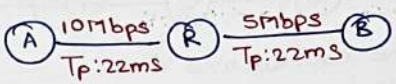


- A send. 30KB to B. which is divided into packet P_1 & P_2 → 10KB, 20KB



no chance of queuing delay.

(9) diff. b/w the A.T of the 1st & 2nd packet at host B?



20KB | 10KB

At A

P_1: $T_t = \frac{10\times10^3\times8 \text{ bits}}{10\times10^6 \text{ bps}} \times 10^3 = 8ms$

P_2: $T_t = \frac{20\times10^3\times8}{10\times10^6} \times 10^3 = 16ms$.

t=0

P_1: $(8+22) + (16+22)$ (T_t (A), T_p, T_t (B), T_p)

$\Rightarrow$ 68 ms

At R

P_1: $T_t = 16ms$

P_2: $T_t = 32ms$.

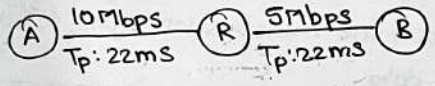
t=0

P_2: $8 + (16+22) + (32+22)$ (T_t (A), T_p, T_t (B), T_p)

waiting for (P_1) to transmit. $\Rightarrow 68 + 32ms$

Variation:

P_1, P_2 Same length = 10KB each



At A

P_1: T_t: 8ms

P_2: T_t: 8ms

At R

P_1: $T_t = 16ms$

P_2: $T_t = 16ms$

t=0

P_1: $8 + 22 + 16ms$ ← it will leave R_1 completely

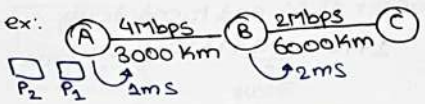
t=0

P_2: $8 + 8 + 22$ ← P_2 will reach to R_2

wait

we have to queue for 8ms.

ex:



T_t: $\frac{500\,B}{4\times10^6\,bps}$ = 1ms

$T_{P(AB)}$ = $\frac{3000\,km}{3\times10^8\,km/s}$ = 10ms

A send 500Byte to C.

P_1 t=0 : 1+10+2 } P_1 will leave B

P_2 t=0 : 1 (wait) +1+10 } P_2 will reach B.

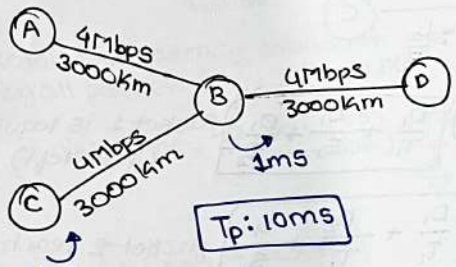
∴ hence there is a queuing delay of 1ms.

total time: transmit all packet + last packet journey

2+10+1+2+20 = 35 ms.

2 ← to send 2 packet; 1 ← queuing delay.

ex:



Tp: 10ms

- A send '2' 500Byte packet to 'D' at t=0 & 'C' send a single 500 Byte to 'D' 1.5ms later

For 1st packet of A

t=0 : 1+10+1 } ← to leave B

for 2nd packet of A

t=0 : 1 (wait for 1st bit to transmit) +1 +10 } to Reach B

for 1st packet of C

t=0 : 1.5 (wait) +1+10 } to reach B.

For 2nd packet of A

t=0 : 1 (waiting 1st bit to be transmit) +1+10 +1 } leave B

▸ queuing delay for C : 0.5ms.

For 1st packet of C:

t=0 : 1.5 +1+10 } to reach B.

For packet of A:
end to end delay

$2+10+1+10 = 23ms$

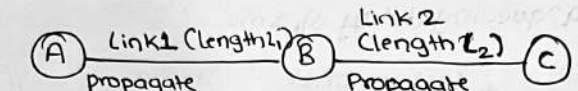
(2 = all packet transmission)

For packet of C: end to end delay

$1+10+0.5+1+10 \Rightarrow 22.5ms$

(0.5 = queue delay)

ex

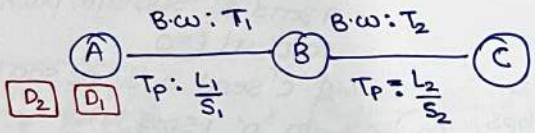


Transmission rate = T_1 bps

Trans. rate = T_2 bps.

At $t=0$

A send P_1 with D_1 Byte & then another packet P_2 with D_2 Byte after immedⁿ pushing P_1



first packet of A (t=0): $\frac{D_1}{T_1} + \frac{L_1}{S_1} + \frac{D_1}{T_2}$ Packet 1 is leaving B (just left)

2nd packet of A: (t=0) $\frac{D_1}{T_1} + \frac{D_2}{T_1} + \frac{L_1}{S_1}$ packet 2 reach B.

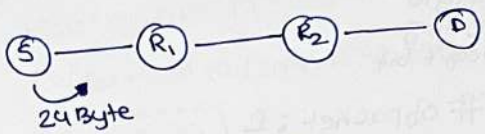
no queuing delay $\left\{ \frac{D_1}{T_1} + \frac{D_2}{T_1} + \frac{L_1}{S_1} \geq \frac{D_1}{T_1} + \frac{L_1}{S_1} + \frac{D_1}{T_2} \right.$

$$\frac{D_2}{T_1} \geq \frac{D_1}{T_2}$$

Optimal packet size

gate 2005

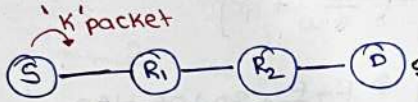
ex: packet switching n/w source to dest. along with two Routers msg. size: 24 Byte & each packet contain 3Byte then optimum packet size.



let packet size: x

# of packet(k) = $\frac{24}{x}$

Size of packet + header = x+3



Size of each packet: x+3

assume B·ω = 1

$T_t = \frac{x+3}{B\cdot\omega}$ & $T_p : \underbrace{\frac{D}{v}}_{\text{assume}}$

total time taken by all packet : transmit all packet + focus on last packet

total time $\Rightarrow k\cdot\left(\frac{x+3}{100}\right) + \cancel{T_p} + 2\left(T_p + \frac{x+3}{B\cdot\omega}\right)$ } ignoring T_p b/c it's indep. of x

($\frac{d}{v}$)

$\Rightarrow \frac{1}{B\cdot\omega}\left[k(x+3)+2(x+3)\right]$

↳ Differentn

$k(x+3)+2(x+3)=0$

$\Rightarrow \frac{24}{x}(x+3)+2x+6$

$\Rightarrow 24+\frac{72}{x}+2x+6$

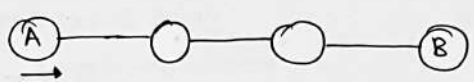
different^n ⟵ $\Rightarrow 2x+\frac{72}{x}$

$2-\frac{72}{x^2}=0$

$\Rightarrow \frac{72}{x^2}=2 \Rightarrow \boxed{x=6}$

ex

B·ω = R bps at each link



no queuing delay.

trying to send large file of 'F' bit

given that file is segement into S and also have 80 bit's as header

L = 80 + S

# of packet : $\frac{F}{S}$

size of each packet + header (L)

$\Rightarrow S + 80$

• transmission rate. (B·ω)

total delay : $\frac{(S+80)}{R} \cdot \frac{F}{S} + T_P + 2\left(\frac{S+80}{B\cdot\omega} + T_P\right)$

$\frac{1}{R}\left[\frac{(S+80)F}{S} + 2(S+80)\right] \Rightarrow F + \frac{F}{S} * 80 + 2S + 160.$

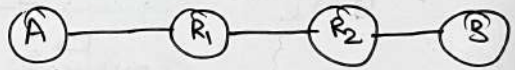
$\Rightarrow \frac{-F}{S^2} * 80 + 2 = 0 \Rightarrow \frac{F}{S^2} * \overset{40}{\cancel{80}} = \cancel{2}$

$S^2 = 40F$

$S = \sqrt{40F}$

ex: Store & forward packet switched n/w.

B·ω : 10^6 Bps each link



Send file size 10^3 Byte's

header

case 1: $10^3 B + 100$

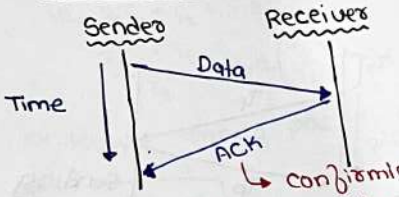
case 2: $10^2 B + 100$

case 3: $50B + 100$.

	Size of each packet	no. of packet	total time
case 1:	10^3B+100	1	$3(10^3+100) \Rightarrow 3300$
case 2:	10^2B+100	10	$10(10^2+100)+2(10^2+100) \Rightarrow 2400$
case 3:	$50B+100$	20	$20(50+100)+2(50+100) \Rightarrow 3300$

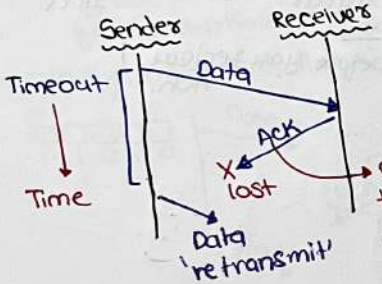
$T_2 < T_3 = T_1$

Simple idea : ARQ



↳ confirming that 'I' have got the packet

- How sender can know if packet has been reached?
  ⇒ ask for ACK from recievers.



confirming that I have got the packet

- Simplest idea for any flow control.
  Send data & wait for ACK
  if ACK arrive (within time)
  Send nxt packet
  otherwise retransmit.

ARQ → Automatic Repeat request.

protocol in which sender wait for ACK before advancing to nxt data item.

↳ timeout at sender

↳ ACK at recievers.

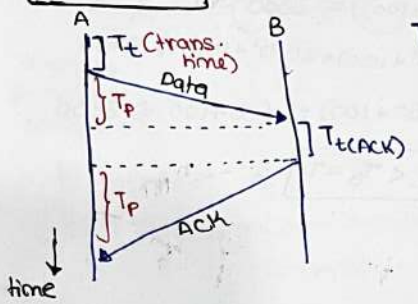
- after timeout we are automatically sending

- How many packet you can transmit

(i) in T_t time Just 1 packet

(ii) in $5T_t$ time $\frac{5T_t}{T_t} = 5$ packet

Round-trip time (RTT)



$T_t + T_p$: total time to transfer packet

- Total time from the moment we start transmission till we get Ack

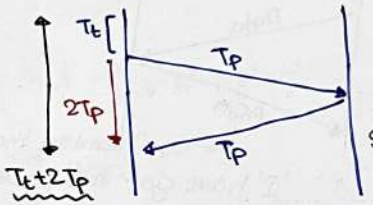
$RTT = T_t(data) + T_p + T_t(ACK) + T_p + T_{queue} + T_{process}$

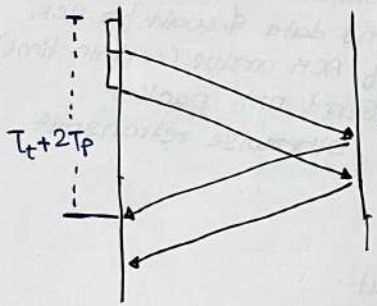
$RTT : T_t + 2T_p$

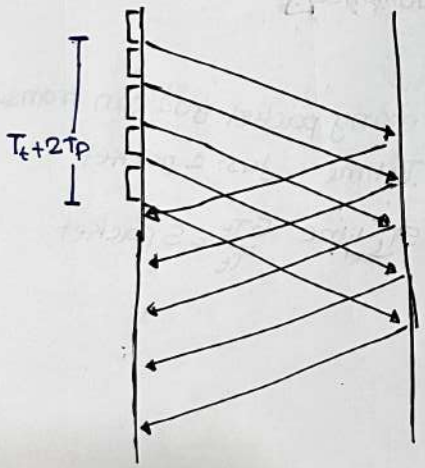
- how many packet can you transmit in $2T_p + T_t$ time

$\Rightarrow \frac{2T_p + T_t}{T_t} \Rightarrow 2\left(\frac{T_p}{T_t}\right) + 1$

$\Rightarrow 1 + 2a$







$KT_t = T_t + 2T_p$ → time in our hand

Case can send only one or multiple packet in $T_t + 2T_p$

$K = 1 + 2a$

max. no. of packet we can send in RTT.

One packet → Stop & wait ARQ

multiple packet → Sliding window
- → GBNARQ
- → SR ARQ

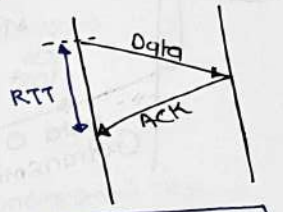
- If you only have ack, and no time out
  Suppose ACK is lost, how long sender has to wait?
  ↳ we should have time out.

- time from the moment we start the transmission. till the moment sender get ACK.

$2T_p + T_t + T_t(ACK)$

≈ 0 (generally)

$\Rightarrow 2T_p + T_t$



timeout > RTT

ex: How many packets you can transmit in $2T_p + T_t$ time

$$\frac{2T_p + T_t}{T_t} = 1 + 2\left(\frac{T_p}{T_t}\right) \Rightarrow 1 + 2a$$

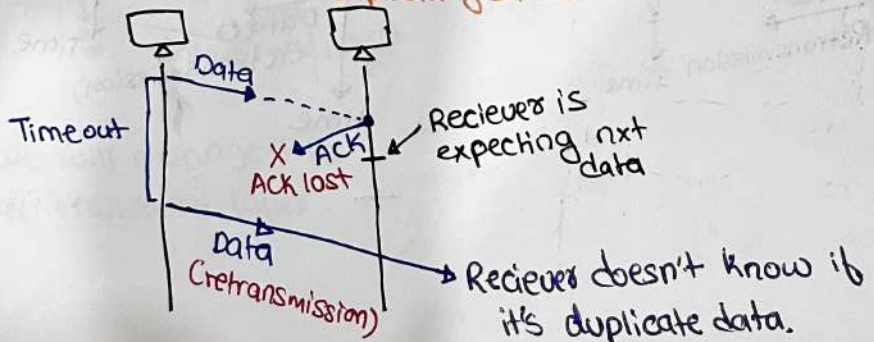
ex: How many packet you can transmit in T_p time? (a)

Stop & wait

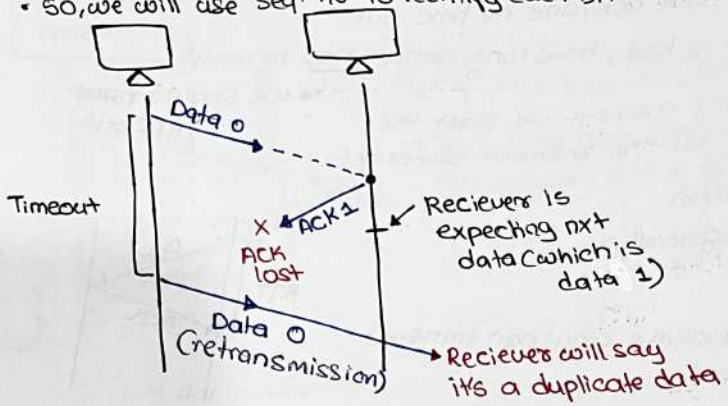
only one outstanding frame at a time

Diagram for stop & wait

(we will use this convention)
the reciver is expecting data 1.



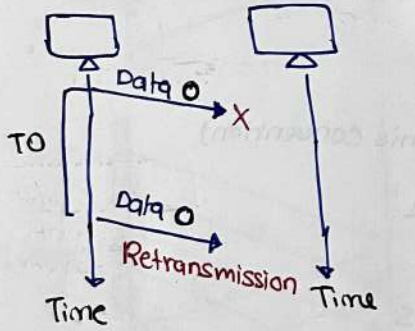
- So, we will use seq. no. to identify each bit



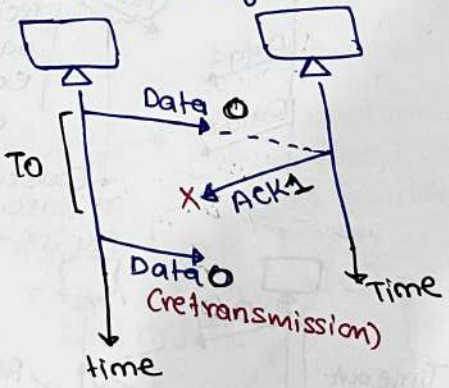
Few Case's about Stop & wait ARQ

1. Data is lost
2. ACK is lost
3. Data is corrupted
4. ACK is corrupted

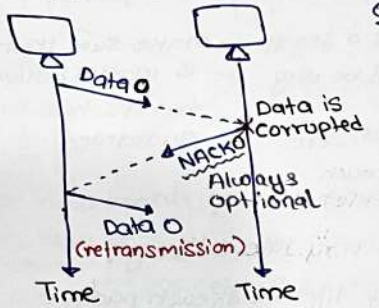
1. Data is lost
retransmit after TO (Timeout)



2. ACK is lost
Retransmit after TO

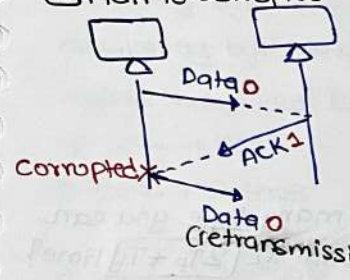


3 Data is corrupted



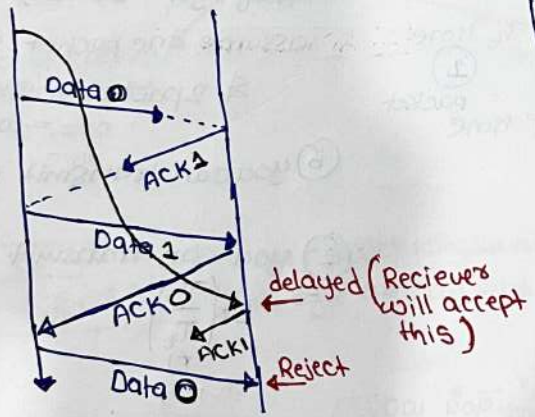
Options for recievers
- Silently discard it ← Same as Data lost
- Send NACK (optional) used for fast retransmission
  - the data you sent get corrupted.

4 ACK is corrupted

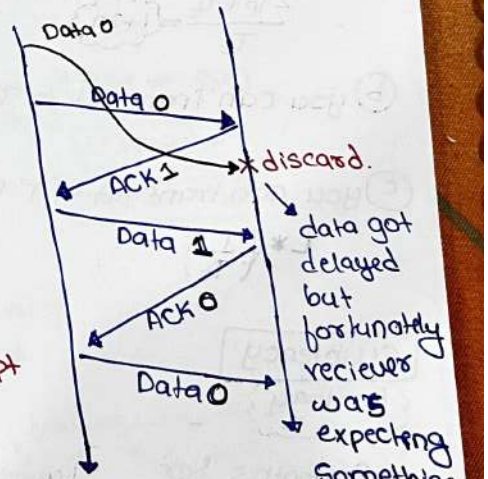


→ Sender doesn't understand ACK
→ Send the data again.

- what if packet are delayed?

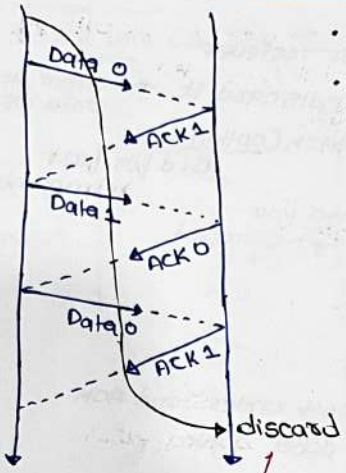


- this situation is managable



data got delayed but fortunately recieves was expecting something else.

- we will manage this in transport layer.



Reciever is expecting: 1
& got : 0 ← discard.

- problem if packet are delayed?
- never reuse a seq # b/c it could be very big
  i.e 2^{32}, 2^{29}... etc.
  (not be much scalable)
- make sure that there is inorder delivery. but it's hard to guarantee
- prevent very late delivery.?
  ↳ Limit lifetime of each packet

ex: How many packets you can
(a) transmit in $2T_p + T_t$ time?

$\frac{2T_p + T_t}{T_t} = (1+2a)$

(b) you can transmit in T_t time (1) packet

(c) you can transmit in 'T' time

$\left(\frac{T}{T_t}\right)$

ex: How many Byte you can
(a) transmit in $2T_p + T_t$ time?

# of packet: $1+2a$
# of byte: $L(1+2a)$
assume one packet: 50B
& 2 packet ⇒ 2x50
= 100B

(b) you can transmit in T_t time (1) byte time

(c) you can transmit in 'T' time

$L * \left(\frac{T}{T_t}\right)$

Efficiency

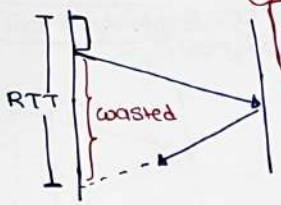
Analogy

- 8 months for gate prep.
- Studied only 2 month (wasted 6 months)

were you 100% effiecent. = no

$\frac{2}{8} = \frac{\text{useful work}}{\text{total}}$

$\Rightarrow \frac{1}{4} = 25\%$



① $\eta : \frac{T_t}{T_t + 2T_p}$ → How much time we are utilizing / Total time in hand

$RTT = T_t + 2T_p$ ← in your hand

ex: How many packet you are sending : 1

ex: How many ---11---11--- could have sent : $1+2a$

② $\eta : \frac{1}{1+2a}$ → packet we are actually sending / Total packet we can send.

How many byte we are sending : L

How many byte we could have sent : ~~$L(1+2a)$~~ $B \cdot \omega (T_t + 2T_p)$

Suppose we know B.w : 500 byte/sec (500 Bps)

$\Rightarrow$ 1S ⟶ 500B

$\Rightarrow$ 2S ⟶ 2×500B

⋮

$\Rightarrow$ T sec ⟶ T×500B

So, in $T_t + 2T_p$ # of Byte

$\Rightarrow (T_t + 2T_p)\,500$

$B \cdot \omega (T_t + 2T_p)$

$T_t + 2T_p$

no. of packet we can Send

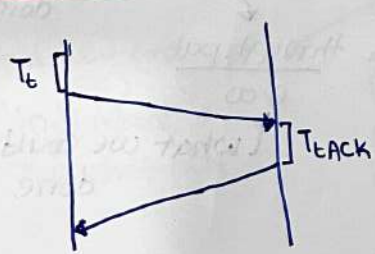
$\Rightarrow 1+2a$

no. of Byte we can Send

$L(1+2a)$ ✓

$B \cdot \omega (T_t + 2T_p)$

③ $\eta = \frac{L}{B \cdot \omega (T_t + 2T_p)}$ → Byte we are actually sending / Total Byte we can send.



$\eta = \frac{T_t}{T_t + 2T_p + T_{tACK}}$

data ↘ T_t } bit (we only utilise this much)

$T_t + 2T_p + T_{tACK}$ } may change depending on question.

↑ data

Conclusion

$$\eta = \frac{T_t}{T_t + 2T_p} = \frac{1}{1+2a} = \frac{L}{B \cdot \omega (T_t + 2T_p)}$$

(time: T_t; Packet: 1; Byte: L)

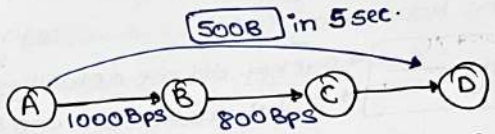
↳ Unit of efficiency: nothing

Throughput

how much byte or Packet we are actually sending per unit time

what could be unit of throughput.

- ~~Packet/sec~~
- ~~Packets/min~~
- ~~Bits/sec~~
- ~~Bits/min~~
- ~~Byte/min~~



assume packet size (L) = 200B.

throughput = in one sec. how much we are sending $= \frac{500}{5} = 100$ Bps

↳ 5packet/sec

In $T_t + 2T_p$ # of byte we are sending: L

In $T_t + 2T_p$ # packet we are sending:

$$\frac{\text{total data we are sending}}{\text{total time taken}} \Rightarrow \frac{L}{T_t + 2T_p} \text{ B/s}$$

$$\Rightarrow \frac{1}{T_p + 2T_p} \text{ Packet/sec}$$

throughput: $2T_p + T_t$ time —— L Byte

1 —— $\frac{L}{T_t + 2T_p}$ Bps

throughput: $\eta \times B \cdot \omega$ ↓ $\frac{L}{B \cdot \omega (T_t + 2T_p)}$

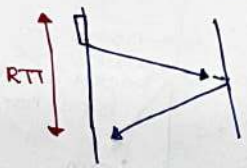
$$\eta = \frac{\text{throughput}}{B \cdot \omega}$$

(throughput: what we are actually doing; $B \cdot \omega$: what we could have done.)

$w_s : 3$

(+) Sender Side

Go-back-N ⇒ RTT
⇒ $T_t + 2T_p$



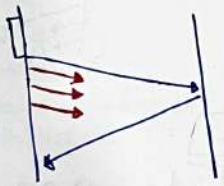
Sliding window protocols.
→ GBN protocol
→ SR protocol

this is an idea that say, we can send multiple frame in an RTT time.

Sender window

0	1	2

Sender window size
$(S.w.S) \geq 1$



Reciever window

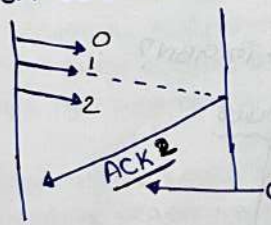
Reciever window size(R.w.S)=1

- we can sent new packet before ACK
- Sender can sent `N' packet w/o ACK

N=4

0	1	2	3

ACKnowledgement
GBN use cumulative ACK.



Cummulative ACK
(I have got till 1)

Type of ACK

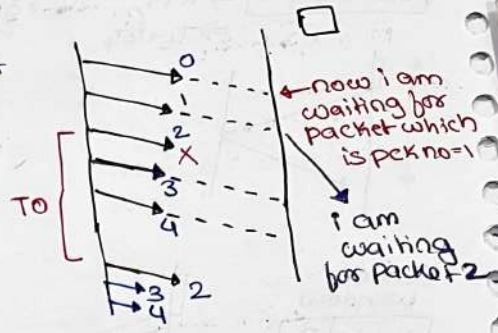
(Stop & wait and SR)
1 Individual
each packet ACK
(we use module-2)

(GBN)
2 cumulative
ACK says that it got everything upto "x-1"
ex: ACK 2
you got (0,1)

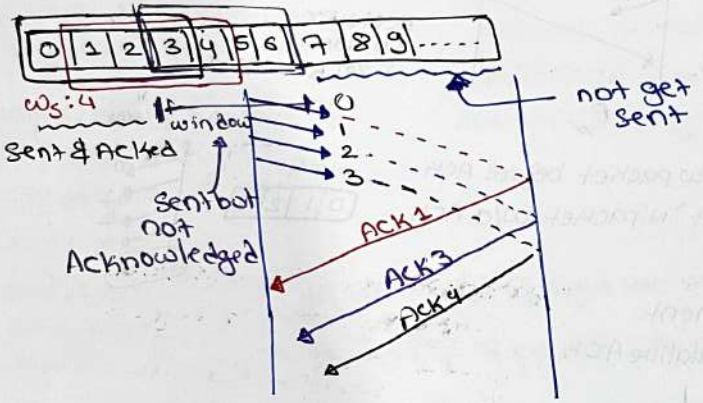
is always optional
3 negative (NACK)
ACK that you didn't get x

Timer in GBN

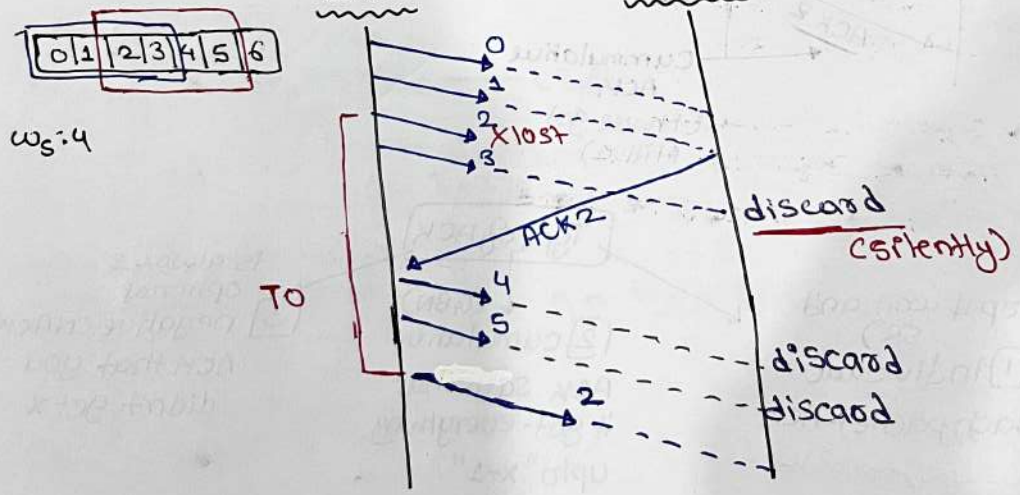
- we can have timer for each packet but 1st outstanding packet timer expires 1st. So, we need to resend all outstanding packet when timer expires.



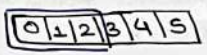
- Normal working of GBN (How window get slide)



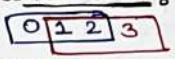
ex. what happens if data/ACK get lost in GBN?



w_s: 3

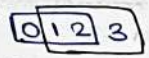


(i) Sender Side



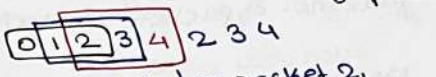
Recieved ACK1

(ii)



Packet #2 has lost,
Recieved ACK2

(iii)

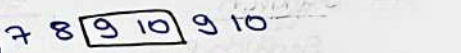


till this moment Sender has sent '8' packet

timeout for packet 2,
resend which packet? 2,3,4

ex: S.w.s = 3
10 frame in total
- every 5th transmission is lost (but no ACK)

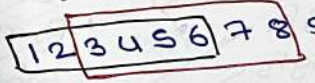
how many data transmission in total if GBN used.



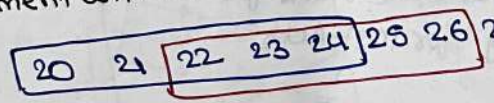
⇒ 10+8 = 18 total transmission.

ex: GBN ARQ protocol
window size: 6
Seq. no. 1,2,3,4,5,6 have been sent.
Segement 7,8,9,10 are waiting to be sent
the sender received an Ack for segement 2

- packet Sender will send immediately after receiving ACK for 2. 7 & 8



ex: segements 20,21,22,23,24 & 25
however segement 22 lost
what segement will be resend.



Retransmission: 22,23,24

ex. Consider sending 10,000 packet using GBN with N=10

Scenario where Sender lost, happen 1% of the time.

max. no. of packet to be sent.

$1000 \times \frac{1}{100} = 10$ packet lost again $\times N = 100$

$\Rightarrow 100 \times 1\% = 1$ packet $\times N$

$\Rightarrow 10$

total packet being lost

$\Rightarrow 10{,}000 \times 1\%$

$\Rightarrow 100$ ~~Retransmission~~ packet lost

Retransmission $= 100 \times N$

$\Leftarrow \Rightarrow 100 \times N = 100 \times 10$

$= 1000$

Total transmit

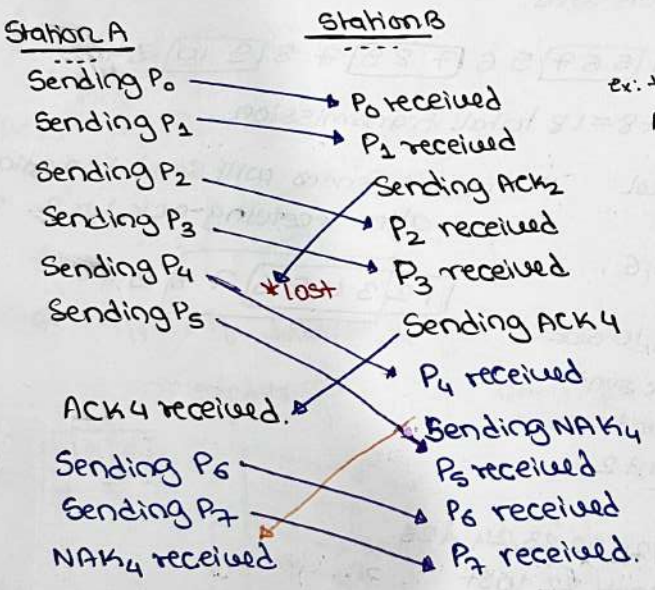
$\Rightarrow 1000$

100

10

1110

ex



ex: the receiver send ACK$_2$ & ACK$_4$ & ACK$_2$ get lost while ACK$_4$ is successfully received by the Sender.

the action that sender undertake for lost ACK$_2$ will not do anything for lost ACK$_2$

Sliding window Protocol's → in one RTT, we can transmit multiple packet (unlike stop & wait)

SR protocol

*can accept out of order

Go back N

w_S — w_R

(Recieves is always waiting for only one frame)

Selective Repeat

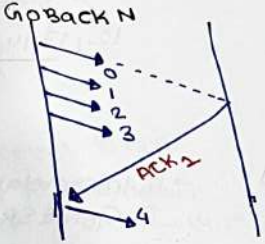
w_S — w_R

(Recievs can have window size ≥ 1)

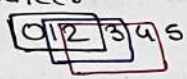
Selective Repeat protocol

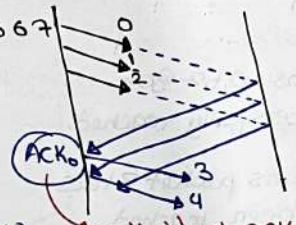
0 1 2 3 4

Go back N

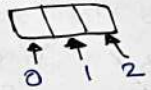


Senders





Reciever

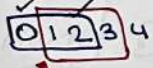


we will keep on sliding Senders window, for every Ack we recieve in sequence

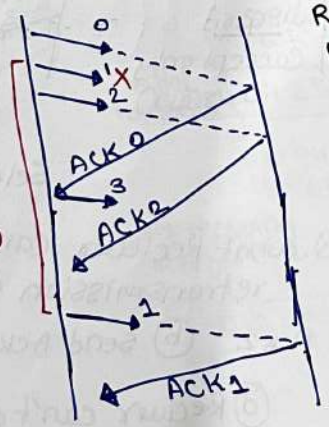
Individual ack telling 0th frame has been recieved.

• what if packet or Ack lost in SR protocol.

Sender

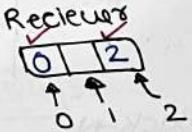


current window

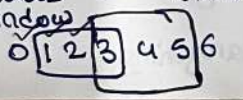


To

Reciever



Previous window — Current window



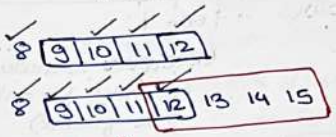
ex: Suppose sender send 8,9,10,11

window size: 4

Assume ACK for 9 is lost & no other loss

Consider that sender 9 retransmission & also recieved ACK for it

- the packet no. others than 8,9,10,11 already sent or will be immediately sending after ACK 9.



already sent

12, 13,14,15

sending immediately after recieving ACK 9

Forouzan.

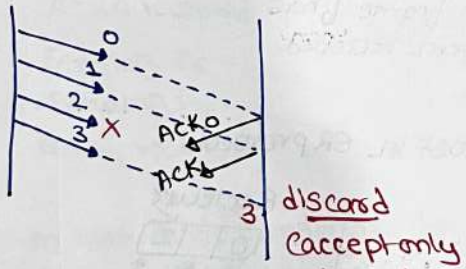
ex assume a sender send 6 packet: 0,1,2,3,4,5.

Sender recieve an ACK with ACKno. =3

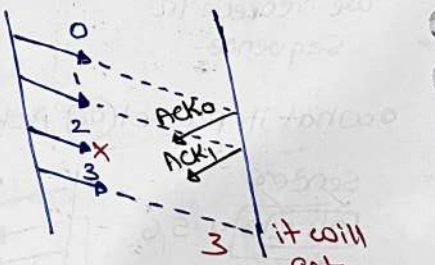
- if interpretation we use is GBN & SR.

GBN: ack3 → it means 0,1,2 got successfully reached.

SR: ack3 → it means packet 3 has been reached.



Go-Back-N



Selective Repeat

Q. what Reciever can do for fast retransmission of packet 2.

(a) Send ACK2 (b) Send ACK3 (c) Send NACK2

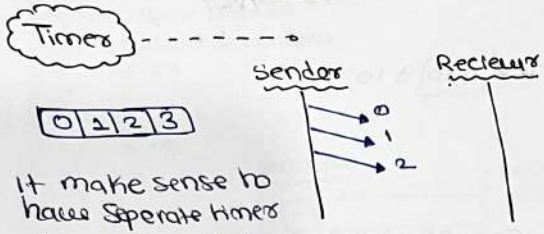
(d) Reciver can't do anything

ex: ACK to handle GBN & SR. (difference)

ack for cumulative Ack.

Ack for individual packet

if we use cumulative ACK then we degrade to GBN.



It make sense to have seperate timers for each packet in window

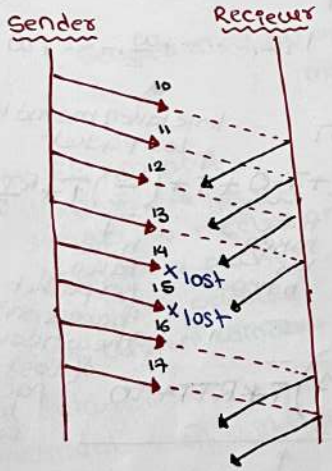
ex. sender send segement 10 to 17

receiver get segement 10,11,12,13,16,17 ↳ 14,15 (missing).

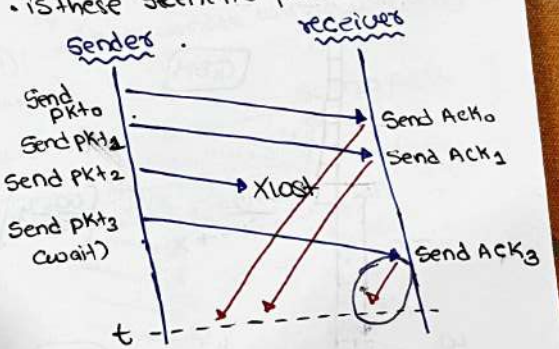
& generate there ACK

Assume that missing segement are lost.

using SR ARQ.



• is these seceneio possible in GBN



Sender window

initial window 0 1 2 3 4 5 6 ...

Sent but not acked.

after receiving ack for 0,1

0 1 2 3 4 5 6 7 ....

Receiver window

0 1 2 3 4 5 6 7 ...

marked

waiting for these frames

frame #3 is marked

ex: S.W.S = 3
- we sent 10 frame in total
- every 5th transmission is lost

In SR protocol.

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

total packet: 10+2 ⇒ 12//

in GBN.

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

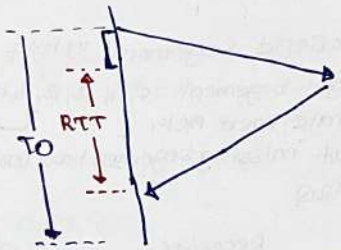
Total packet = 10+3+3+2 = 18

~~Imp~~ ex: Suppose 'w': window size is an even no.

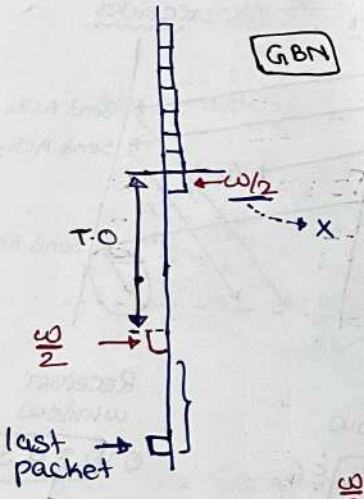
Seq. no': 1, 2, 3 .....

- if packet $\frac{w}{2}$ is lost

time taken for 1st 'w' packet to arriving at receiver in GBN



$1, 2, \ldots, \frac{w}{2}, \ldots, w$

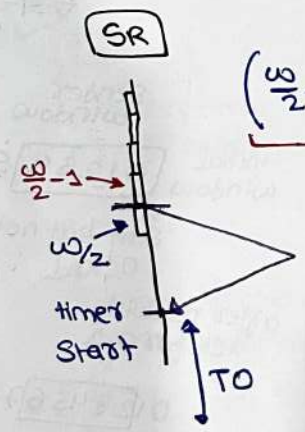


$\left[\frac{w}{2}-1\right]T + \text{RTT} + T.O + T + \left(\frac{w}{2}\right)T + \frac{RTT}{2}$

time taken to send the lost packet

time taken per timeout

time taken per packet that are on the window of lost packet to send.



$\left(\frac{w}{2}-1\right)T + RTT + TO$

before transmission

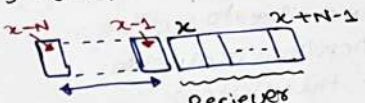
$\left(\frac{w}{2}\right)T + TO + T + \frac{RTT}{2}$

ex: Sender & receiver in a pipelined reliable data transfer. ... sliding window

window size : N

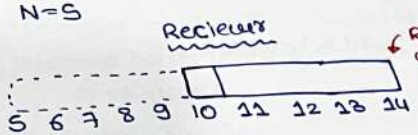
Seq. no. of the segment at the base of window at receiver is X.

• range of seq. no. in the sender window?

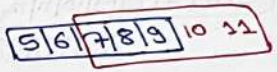


Reciever window
[x, x+N-1]

ex: X=10
N=5

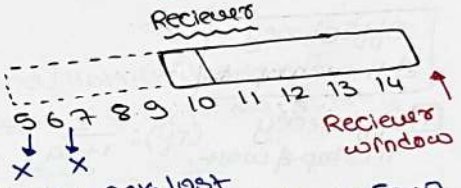


All ACK get lost then
Sender window = [5,9]



5|6|7|8|9|10,11,12,14

if no ack get lost
Sender window = [10,14]

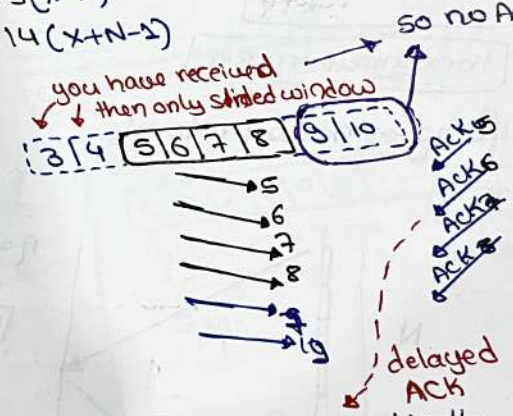


if 5,6 ACK lost
then Sender window = [7,11]

Sender window min: 5 (X-N)
max: 14 (X+N-1)

ex: SR protocol,
it's possible for the sender to receive an ACK for a packet that falls outside of its current window.

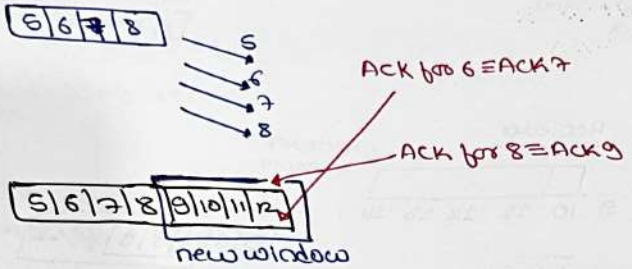
if delayed ACK then protocol itself will not work.
(so we will handle these in TCP)



Automatically expires.

[worry about
→data lost
→ACK lost

ex In GBN is it possible for sender to receive Ack that is outside the window.



Efficiency & throughput

1. efficiency in stop & wait (η): $\frac{1}{1+2a} = \frac{L}{RTT \times Bw}$

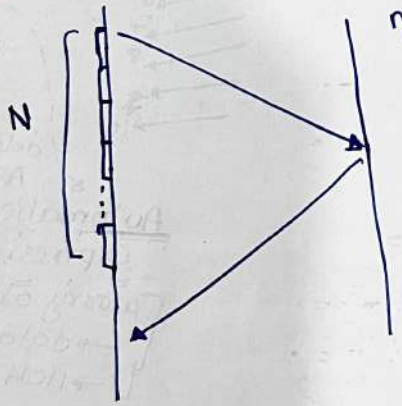
Bandwidth-delay product

$throughput = \frac{L}{RTT}$

$throughput = \eta \times B \cdot \omega$

- **Sliding window protocols**

$\omega_S = N$



$\eta = \frac{N T_t}{RTT} = \frac{N}{1+2a}$ } no packet loss

$RTT \cdot 1 + 2a$

$N \equiv S \cdot \omega \cdot S$

- efficiency for GBN & SR protocol.

ex: n/w connecting 2 host with an RTT of 100ms
Bottleneck link rate of 30Mbps
Assume packet are 1500B

(a) max. throughput b/w 2 hosts? 30Mbps

(b) In a standard 1-packet Stop & wait protocol. %age of max. throughput

RTT: 100ms & L = 1500 Byte

$\Rightarrow \frac{0.12}{B.W} = \frac{0.12}{30} = 0.4\%$

means efficiency.

So, 100ms → 1500 B
1000ms → 15×10^3 B
$\Rightarrow 15 \times 10^3 \times 8$
$\Rightarrow \frac{15 \times 10^3 \times 8}{10^6} \times 10^6$ B/s
Mbps
$\Rightarrow$ 0.12Mbps

ex: 1ms to send packet & 10ms one way P_d Bw Sender & receiver
S.W.S (N): 4

- channel/link utilization?

T_t: 1ms & T_p: 10ms & N = 4

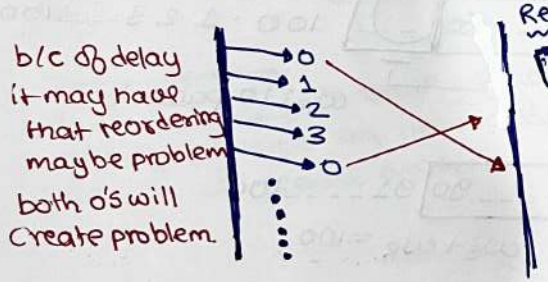
$\eta = \frac{4}{1+2\left(\frac{10}{1}\right)} = \frac{4}{21}$

Relationship b/w windowsize & ASN

Sender

0	1	2	3	0	1	2	3	0	1	2	3

b/c of delay it may have that reordering may be problem
both 0's will create problem.

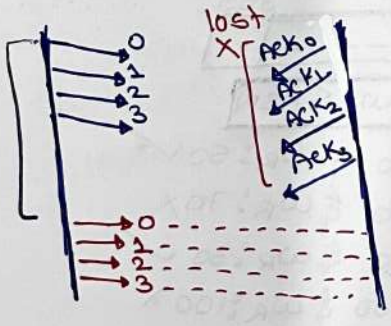


Receiver

0	1	2	3	0	1	.....

we can't have two packet in flight with same Seq. no.

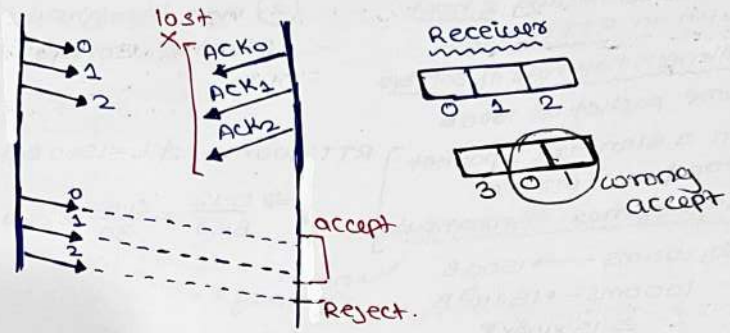
Timeout



Receiver

0	1	2	3

- Receiver is waiting for nxt 0,1,2,3 but sender is sending different 0,1,2,3.



- we can't send multiple packet with same Seq. no. at once (we can't have same Seq. no. packet in flight at once)
- But it also seems that we can't even send all distinct seq. no. at once

Sender

0 1 2 3 0 1 2 3 0 1 2 3

0 1 2 3 4 5 | 6 7 8 9 0 | 1 2 3 4 5 6 7 8 9

✗ not possible

1 2 3 4 5 ..... 50 51 52 . 60 | 61 70

100 . 1 2 3 ---- 100

w_R : 10 packet

Available Seq. no. = 100 (ASN)

1 ------ 90 | 91 ---- 100

$w_S + w_R = 100$

300 399

ASN = 100

$w_S + w_R \leq ASN$

(a) w_S : 50 & w_R : 50 ✓

(b) w_S : 70 & w_R : 70 ✗

(c) w_S : 30 & w_R : 30 ✓

(d) w_S : 100 & w_R : 100 ✗

ex: if ASN are 100
then max. s.w.s in
GBN

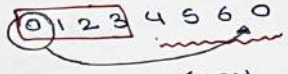
$w_S + w_R \leq ASN$

$w_S \leq 100 - 1$

$w_S \leq 99$

N≥99

ex: SR protocol
Seq. no. available 0,1,2,3,4,5,6.
window size (N): 4
will the protocol work properly.
no. will not work



$w_S + w_R \leq ASN$

ASN=7

ex: Sliding window
S.w.s = 6 & R.w.s = 4
Seq. no. are based 0.

0 . . . . . N

$w_S + w_R \leq ASN$

ASN ≥ 10

(a) Smallest value of
Max. seq. no. used for
successfull transmission

(9)

ex: Sliding window protocol
Sufficiently large window
to avoid Stop & wait

- window size: 200
no. of bit in seq. no.

(1) GBN $w_S = 200$ & $w_R = 1$

$\Rightarrow 201 \Rightarrow \log_2\lceil 201 \rceil$

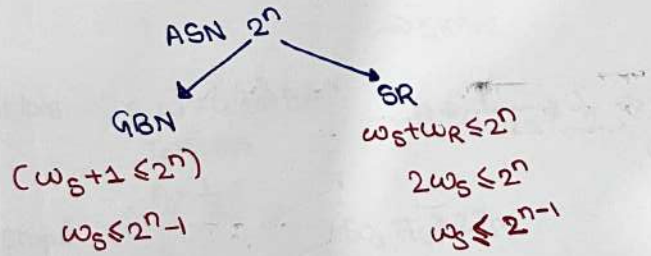
= 8bit

(2) SR

$w_S = 200$ & $w_R = 200$

$\Rightarrow 400 \Rightarrow \lceil \log_2 400 \rceil = 9$ bit

$w_S + w_R \leq ASN$ we strictly follow there
rule if not then seq. no. would repeat.
which will cause problem.



Question on Shop & wait.

Time: gate CSE

28.00 – 2015
37.48 – 2005
38.41 – 2015 set 1
47.43 – 2016 set 1

lecture 14

lecture 15.

15:00 — 2016 set 1.
30:00 — 2017 set 1
1:11:48 — 2023
1:16:35 — 2016

Question on stop & wait

Recap: $T_t + 2T_p$] time we have

- $1+2a$ packet, $a = T_p / T_t$
- $B\omega (T_t + 2T_p) \Rightarrow L(1+2a)$

$\eta = \frac{\text{throughput}}{B \cdot \omega}$

$T_t + 2T_p$ sec $\Rightarrow$ L byte

1 sec $\Rightarrow \frac{L}{T_t + 2T_p}$ Byte/sec

throughput

& 1 packet/sec $\Rightarrow \frac{1}{T_t + 2T_p}$

(Efficiency)

• link utilization by stop & wait

① 1000 mile link
& 1Mbps capacity
link used by a single
sender 1250B frame
(Single propag^n is
roughly 5MS per mile)

D = 1000 mile
L = 1250 Byte
$B \cdot \omega$ = 1Mbps
T_p = 5MS × 1000 b/c of per mile.
= 5000MS

$T_t = \frac{1250 \times 8}{1 \times 10^6} \times 10^6$ ms
$\Rightarrow$ 10^4 MS

$a = \frac{5000 MS}{10,000 MS} \frac{1}{2}$

$\eta = \frac{1}{1+2a} \Rightarrow \frac{1}{1+2 \times \frac{1}{2}} \Rightarrow \frac{1}{2}$

$\eta = 50\%$

② 10KB/s link
50ms one way
Latency
frame size: 1KB
max. link Utilization

$B \cdot \omega$: 10KB/s
L = 1KB

$T_t = \frac{1}{10} = 100ms$

$a = \frac{T_p}{T_t}$

$\Rightarrow \frac{50ms}{100ms} \frac{1}{2}$

$\eta = \frac{1}{1+2a} \Rightarrow \frac{1}{1+2 \times \frac{1}{2}} \Rightarrow \frac{1}{1+1}$

$\Rightarrow 50\%$

A/c to this throughput = 5KB/s

RTT = $T_t + 2T_p \Rightarrow 100 + 2 \times 50 = 200ms. = 0.2sec$

throughput = 5KB/s

0.2sec — 1KB
1sec — $\frac{1}{0.2}$ KB/s
$\Rightarrow$ 5KB/s

③ bit rate: 4Kb/s
P_d: 10ms
η: 50%
frame size of Stop & wait.

$B \cdot \omega = 4 \times 10^3$ b/s
T_p = 10ms
$\eta = \frac{1}{2}$

$\eta \Rightarrow \frac{1}{1+2a} \geq \frac{1}{2} \Rightarrow 1+2a \geq 2$

$a \geq \frac{1}{2}$

So, $T_t \geq 2T_p$

$L \geq 2T_p \times B \cdot \omega$

$2 \times 10ms \times 4Kb/s$

$L \geq 80$ bits

$T_t =$

④ packet size: 1000bit
transmission rate 1Mbps
P_d: 15ms
Ack & processing time is negligible
throughput if stop & wait used.

T_t=1ms & T_p=15ms

RTT = 1+2×15 = 31ms

method 2: using (η)

$\eta = \frac{1}{31}$ & B·ω = 10^6 bps

so, throughput = $\frac{10^6}{31} \approx 32258$ bit/sec

L: 1000bit, B·ω: 1Mbps } $T_t = \frac{1000}{1\times10^6} \times 10^3 = 1$ms

T_p: 15ms

method 1: using (η)

throughput = η × B·ω

method 2: using RTT

method 2: using RTT

31ms —— 1000bit

1ms —— $\frac{1000}{31}$ bits/sec = 32.258 bits/ms

convert to sec.

$\Rightarrow \frac{32.258 \text{ bits}}{\text{ms}} \times 10^{-3}$ sec

$\Rightarrow 32.258 \times 10^3$ sec

$\Rightarrow$ 32258 bit/sec

⑤ chain topology

A---B----C---D---E

A send packet to E using reliable transport protocol.

each link transmit one packet per second. (no other delay except transmission delay)

(a) RTT b/w A & E = 8 sec.

B·ω: 1 packet/sec

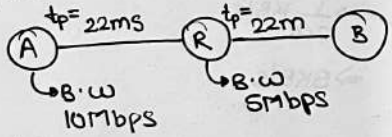
T_t = 1sec & T_p = 0

(b) throughput of a stop & wait

8sec —— 1 packet

1sec —— $\frac{1}{8}$ packet/sec.

⑥ given



Packet size

P_1: 10KB

P_2: 20KB

(1) (P_1)

transmission at A:

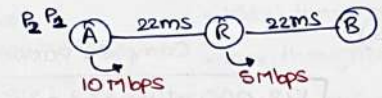
A: $T_t = \frac{10KB \times 8 \times 10^3}{10Mbps} = 8$ms

R: b/c B·ω is half so T_t will be double = 16ms.

(P_2)

A: $T_t = \frac{20^{KB} \times 8 \times 10^3}{10 \times Mbps} \Rightarrow 16$ms

R: T_t = double T_t b/c B·ω is half $\Rightarrow$ 32ms



P_1 $t=0$ A to R (t_p) + R to B (t_p)
$8 + 22 + 16 + 22 = 68$
A to R T_t + R to B T_t

P_2 A to R (t_p) + R to B (t_p)
$8 + 16 + 22 + 32 + 22 = 100$
A to R T_t + R to B t_t
waiting for P_1 to transfer.

So, Arrival time difference $= 100 - 68$
$= 32$ ms.

(2) How long would it take to send the entire file.

P_1 to $(t=0)$ leaving R $\Rightarrow 8 + 22 + 16$
P_2 $t=0$ (entering to R) $\Rightarrow 8 + 16 + 22$
no diff. so, no queuing delay

So, total transmission of packet + last packet + total time taken to transfer from R to B.

$\Rightarrow \underline{8+16} + 22 + \underline{32+22}$

$\Rightarrow 100$ ms.

(3) throughput b/w A & B

$\Rightarrow \frac{30\times KB}{100ms} = \frac{30\times10^3 MB}{10\times10^{-3} sec} \Rightarrow 0.3$ MBps.

(7) transfers file from

B → L

Size of file 122,000 Byte

transfers in 1500 B data packet including 40B header.

Size of ACK packet is 40B including header

B·ω = 12 Mbps (bidirectional)

$T_p = 12$ ms

L_{Data}

122,000 Byte → 1460B + 40B
1460B + 40B
⋮

$L_{ACK} = 40B$ / $B\cdot\omega = 12$ Mbps

$L_{Data} = 1500B$ / $T_p : 12$ ms

820 B + 1500 Byte

one packet — 83 packet

$L_{ACK} = 40B$

B.w = 12 Mbps.

Partial packet

$T_t = \frac{820 \times 8}{12 \times 10^6}$

Complete packet

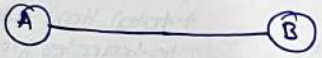
$122,000 = 1460 \times 83 + 820 * 1$

$T_t = \frac{1500 \times 8}{12 \times 10^6}$ sec

$L_{Data} = 1500B$

$T_p = 12ms$

$T_t = 1ms$



Total time for 1 partial + 8 full packet.

full packet.

$T_t = 1ms$

Partial packet: $T_t = 0.573$ ms

ACK: $T_t = 0.027$ ms

Full packet:

$T_t + T_p + T_{tACK} + T_p$

83 full packet: $83 (T_t + T_p + T_{tACK} + T_p)$

+

Partial packet: $T_{tpacket} + T_p + T_{tACK} + T_p$

$\Rightarrow 2101.8ms.$

throughput $\Rightarrow \frac{\text{data sent}}{\text{time taken}} = \frac{122000B}{2101.8ms}$

$\Rightarrow 58.045$ KBps.

8. RTT : 100 ms
Bottleneck B.w : 30 Mbps
(Communicⁿ is one direcⁿ)
packet size 1500 Byte
(no queueing delay in the routers)

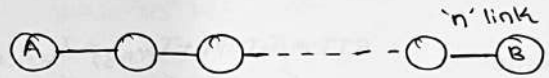
- Max. throughput b/w the 2 host. 30 Mbps
- Max. throughput in stop & wait.

$$\eta = \frac{1500 \times 10^3 \times 8}{30 \times 10^6}$$

$$\eta = 0.4\%$$

100 ms — 1500 B
1 sec — 15×10^3 Byte

9.



B.w : R bps
L_{data} : S bits
L_{ACK} : K bits
d = D meter
v = C mps
$T_{processing}$ = P sec
prob. of packet loss = p

(a) T_t b/w A & B = $\frac{S}{R}$ (for one hop)

(b) P_d b/w A & B = $\frac{nD}{C}$

(c) RTT b/w A & B

$$\Rightarrow T_t + T_{P(data)} + \text{Processing delay for ACK} + T_{t(ACK)} + T_{P(ACK)} + \text{Processing delay for ACK}$$

(packet)

$$\Rightarrow \underbrace{\frac{nS}{R} + \frac{nD}{C} + (n-1)P}_{\text{for data}} + \underbrace{n\left(\frac{K}{R} + \frac{D}{C}\right) + (n-1)P}_{\text{for ACK.}}$$

(d) prob. that packet sent by A reach B.

$(1-p^n)$

ACK can be lost with prob. = q

prob. that A send data & recive Ack successfully

$(1-p)^n (1-q)^n$

10 diff. B·w in each direction

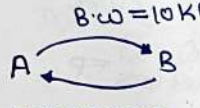
② node's A & B.

B·w from A to B : 5KBps
& P_d : 120ms

& B·w from B to A : 10KBps
& P_d : 80ms

Packet size (including header) : 500Byte

ACK packet size of 100Byte



$L_{Data} = 500B$

$L_{ACK} = 100B$

- numeric expression for the throughput A can achieve in transmitting B

(you can treat a 500B data packet as transferring 500B of useful data)

$RTT = T_t Data + T_{P(AB)} + T_{t(ACK)} + T_{P(BA)}$

$\Rightarrow \frac{500}{5\times10^3} \times 10^3 + 120 + \frac{100}{10\times10^3} \times 10^3 + 80$

for millisec.

$RTT \Rightarrow 100 + 120 + 10 + 80 = 310ms$

method 1:

$throughput = \frac{500\,Byte}{310ms} \Rightarrow \frac{500\times10^3}{310sec} = 1613.9\ B/s \approx 1613\ B/s$

$\eta = \frac{T_t}{RTT} \Rightarrow \frac{100}{310}$

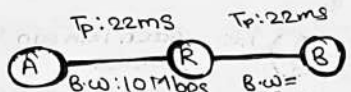
T_t : useful time, RTT : Total time

method 2:

throughput At A $\Rightarrow \eta \times B{\cdot}w$

$\Rightarrow \frac{100}{310} \times 5KBps \Rightarrow \frac{100}{310} \times 5\times10^3 \approx 1613\ B/s$

11



30KB data is to be sent

- the file is divided into 6 packet of same size.

time taken to send entire packet & new packet can't be sent before ACK of pervious (ignore T_t of ACK)

- $L_{data} = 5KB$

$$T_t = \frac{5\times10^3\times8 \text{ bits}}{10\times10^6 \text{ bps}} = \frac{40}{10^4}\text{ sec} \times 10^3 = 4ms$$

- for 6 frame it could be $6\times100 \Rightarrow 600$ ms.

So, $T_{t(AR)} + T_{P(AR)} + \underbrace{T_{t(RB)}}_{\text{double b/c of B.w}} + T_{P(RB)}$

$\Rightarrow \underbrace{4+22+8+22}_{\text{one packet to reach B}} + \underbrace{22+22}_{\text{ACK from B to A}}$

$\Rightarrow 100$ ms.
(for one frame)

MIT

12 B.w: 40Kbps
Connecting the earth to moon

- moon is 1.5 light-sec from earth.

$B.w = 40Kbps$ $L = 1KB$
$D = 1.5$ light/sec $T_p = 1.5$ sec.
$= 1.5\times3\times10^8$ m

$RTT = T_t + 2T_p$.

$T_t = \frac{1\times8}{40} \Rightarrow 0.2$ sec

- Data size: 1KB

using stop & wait data transfer rate can be achieved.

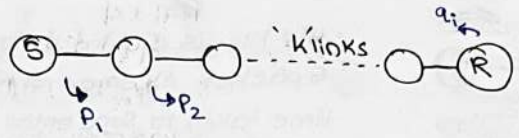
$\eta = \frac{0.2}{0.2+2\times1.5} \Rightarrow \frac{0.2}{3.2} = \frac{1}{16}\times100 = 6.25\%$.

In RTT some time (authors dependent)

$T_t + 2T_p$ — we will use these

$2T_p$ — we don't include T_t

13



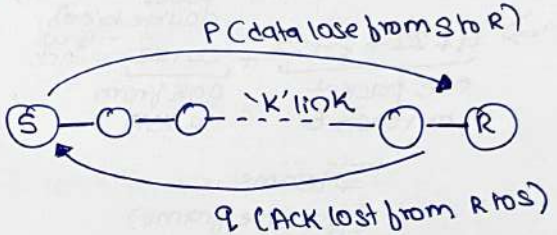
each link can loss a packet
Link 'i' has p_i as loss rate in one direction & q_i in another direct^n

(a) prob. that a data packet doesn't reach R when sent by S is 'P'
prob. that an ACK packet sent by R doesn't reach S is 'q'

$P = 1 - \underbrace{(1-p_1)(1-p_2)\cdots(1-p_k)}_{\text{reaches } R}$

$q = 1 - \underbrace{(1-q_1)(1-q_2)\cdots(1-q_k)}_{\text{ack reaches } S}$

(b) Stop & wait
expected no. of transmission of a packet before S can send the nxt packet in seq.



with what prob. S will be sure packet has reached? $(1-P)(1-q)$

14 Stop & wait
the expected no. of retransmission needed for a single packet with dropout prob. of "p"?

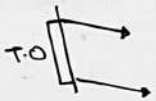
$\frac{1}{P}$] total transmission

$\Rightarrow \frac{1}{P} - 1$ total

50 + 1
trail real one

15) A Station — A ——— B

- file size $L_{data} = 1KB$
- #of frame = 100
- RTT : 50ms.
- Timeout timer



(a) time taken for transmission of packet (if no error then)

50ms × 100 = 5sec

(b) n/w drop each frame with prob. of 10%. indep. of each other

(expected duration of the transmission).

Prop. that packet may loss $\Rightarrow \frac{1}{10}$ & $P = \underbrace{\frac{1}{10}}_{\text{packet may loss}} \times \underbrace{\frac{1}{10}}_{\text{Ack may loss}}$

& $P = \underbrace{\frac{9}{10}}_{\text{packet success reach}} \times \underbrace{\frac{9}{10}}_{\text{Ack Safely reach}} = 0.81$

total transmissions for just one packet $= \frac{1}{P}$ → total transmission

$\frac{1}{P} - 1$ (time out) , 1 (RTT)

total time taken to send 1 packet : $200 \times \left(\frac{1}{P} - 1\right) + 50$

$\Rightarrow 200\left(\frac{1}{0.81} - 1\right) + 50 \Rightarrow 96.91ms$

So, for 100 packet $\Rightarrow$ 96.91ms × 100 $\Rightarrow$ 9.69 sec.

16) Stop & wait ARQ

P_f : Prob. of frame error. from A to B

P_f & processing delay are negligible

RTT : t

Time out timer : twice the RTT

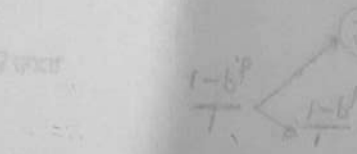
- expected time to send a single frame.

RTT : t & TO : 2t

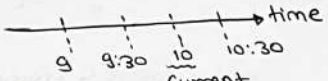
$\frac{1}{1-P_f}$ — $\frac{1}{1-P_f} - 1 = \frac{P_f}{1-P_f}$ } T.O = 2t

1 — RTT = t

$\Rightarrow t + 2t\left(\frac{P_f}{1-P_f}\right)$



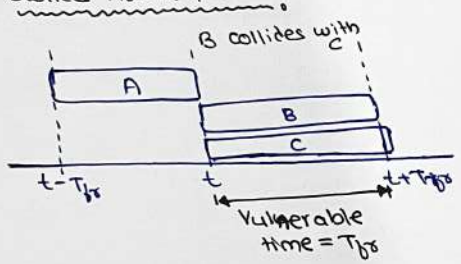
Slotted Aloha In slotted Aloha we divide the time into slots of T_{fr} sec.



- Pure aloha can send at any time
- if someone wants to send at 9:15am then it can't send & has to wait till beginning of the nxt slot
- if station A starts transmission at time 10am then it will collide only if some other station start the transmission at time
  - (a) 9:45am
  - (b) 10:15am
  - (c) 10:30am
  - (d) 10:00am

Station can't even start Transmission here

- Vulnerable time for slotted Aloha protocol.

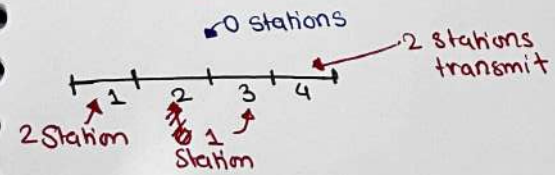


assumptions: Poission distribution
1. transmission time of all station & all frame is same
2. Slot time is equal to transmission time
   → Binomial distribution

- efficiency = Throughput (in packet/slot) = Prob. of slot "not wasted"
- Aloha efficiency derivation case.

Case 1: no. of station in n/w is very large
Practically ≈ 40-50 Station

Case 2: no. of stations in n/w is finite.

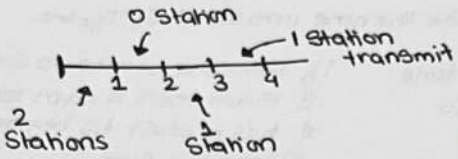


In slot no. 3, there is a successfull transmission

$$\frac{2+0+1+2}{4} = \frac{5}{4} = 1.25 \text{ frame/slot}$$

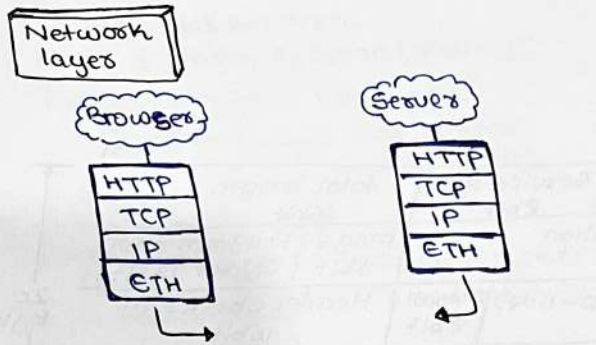
on an avg.

G: Avg. no. of frames, generate by system in any slot out of 5 frames, only one got transmitted.

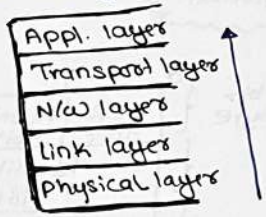


$G = \frac{2+0+1+1}{4} = 1$ frame on an avg. in each slot.

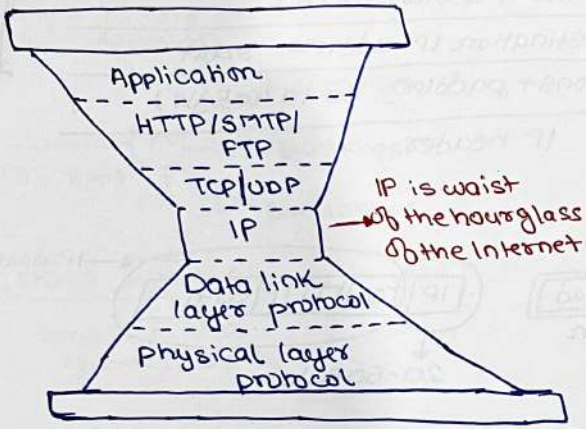
out of these 4 frame, how much actually got transmitted.



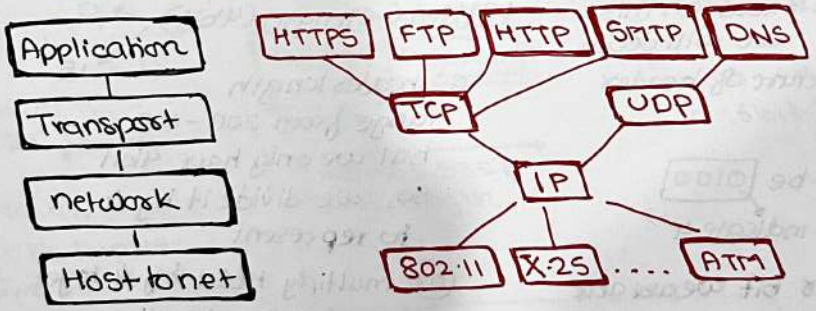
• where we are in course.
↳ moving upto n/w layers



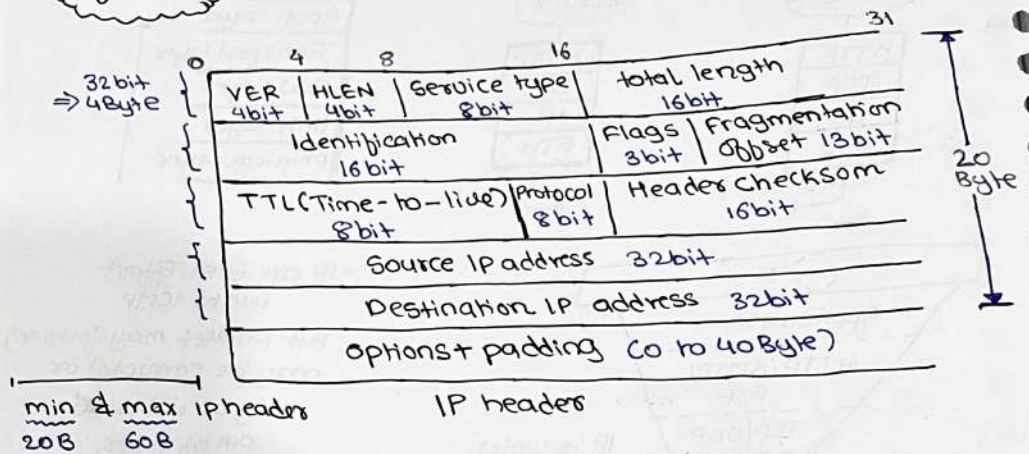
• IP are Best Effort Due to ICMP
But packet may lost here, may be corrupted or may be delivered out of order.

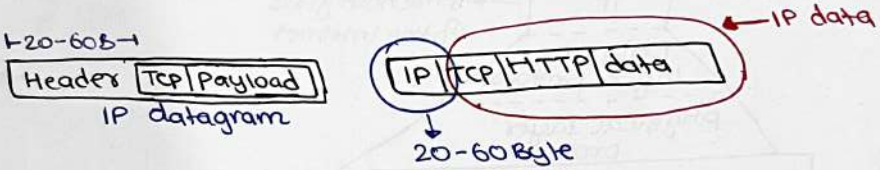


IP is waist of the hourglass of the Internet



IP header format





1. Version (4bit) : Based on this we will decide the structure of header
   i.e IPv4
   ver bit will be 0100
   indicate 4

* Based on ver bit we are able to now what are the nxt field.

2. Header length (4bit) → 0 … 15
   as header length range from 20B–60B but we only have 4bit so, we divide it by 4 to represent
   (or) multiply HLEN by 4 to get header length

- if HLEN field contain : 15
  header length : 15x4
  ⇒60B (max)

- header length is 20Byte
  value of HLEN field
  ⇒ 20/4 = 5 (0101) Binary (min)

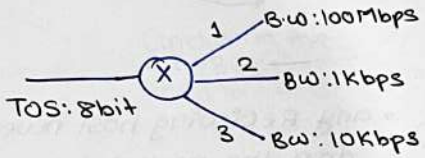
- if HLEN has 1101 then
  what is length of option+padding?
  header length is = 13×4 = 52
  52 − 20 = 32 Byte are options + Padding

never been used (useless)

(3) Service type (8bit)

- Allow different packet to be treated differently.

to tell routers about which link to send (how to take routing decision)

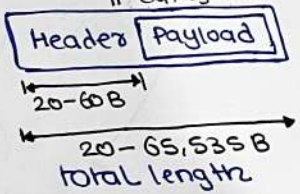


0000 0100 = 4

send it to max. throughput link.

(4) Total length (16bit)

- no. of Byte in packet
- though most links impose smaller limits.



max size

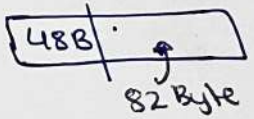
$2^{16} - 1 = 65,535$

max no. we can represent in 16bit

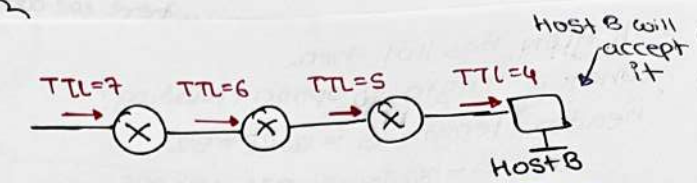
Length of data = total length − (HLEN)×4

ex: Assume total length field is
0000 0000 1000 0010
& HLEN field is 1100 → 12×4 = 48 Byte header length

total len. = 128+2
⇒ 130 byte

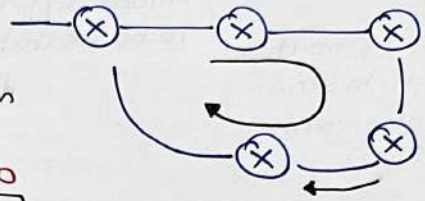


5 TTL (Time to Live)

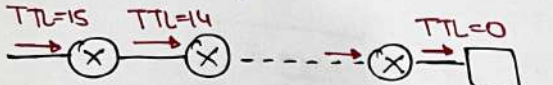


- used to identify packet stuck in forwarding loop & eventually discard them from the n/w

if intermediate router get TTL=0 packet, it will drop it.



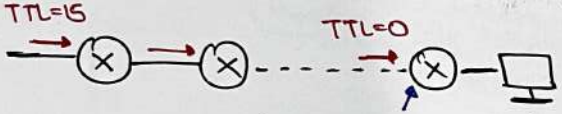
ex: Can a host recieve a packet with TTL=0? yes



∴ Host B will Accept

- any Recieving host never drop the packet b/c of TTL

ex:



this. router will drop the packet

6 protocol's (8 bits)

some protocol value

ICMP	01
IGMP	02
TCP	06
UDP	17
OSPF	89

don't remember this.

not required for gate

Example of option

Just like type of Service we don't often use options

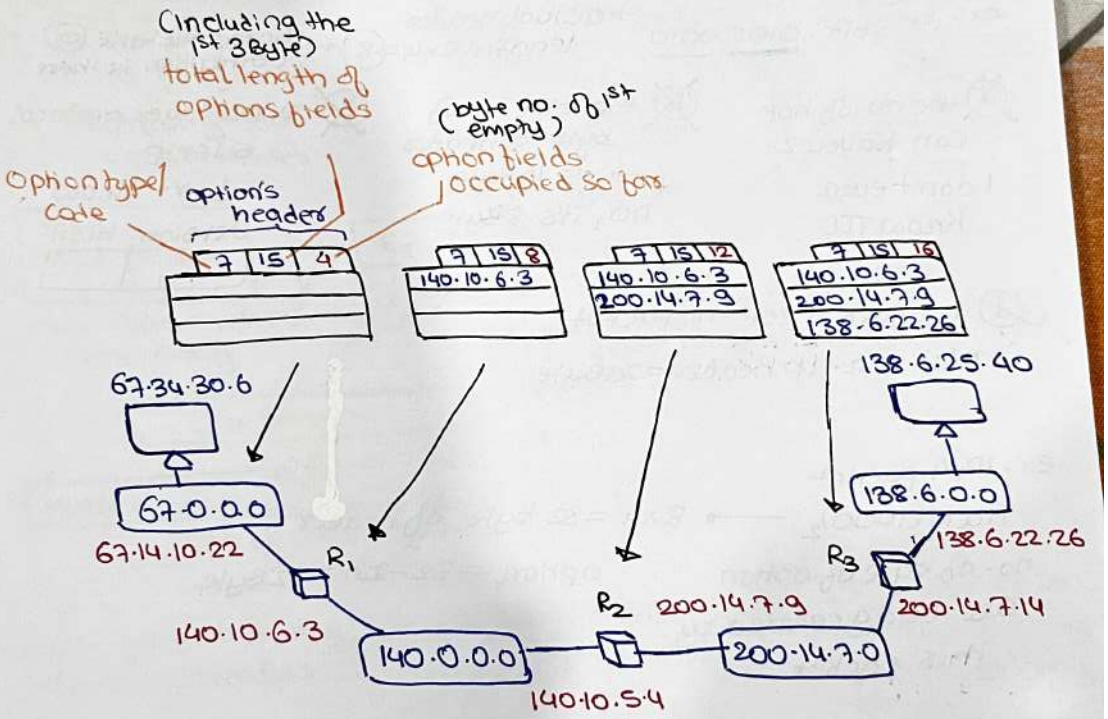
- Record Route
- Strict Source Route
- Loose Source Route
- Timestamp
- Traceroute
- Router Alert
- .....

Source & Dest. IP address remain unchange until it's final destination

Option → is used for special purpose's

(a) Record Route option.
used to trace route

1. Source create empty field for IP address.
2. Each router that processes datagram insert its outgoing IP address.



(b) Timestamp options

Similarly to Record Route + records datagram end processing time by each router (in ms)

(c) Strict Source Route option

(d) Loose Source Route option.

gate 2015.

field that IP header is not modified

(a) checksum — b/c TTL is changed

(b) source address — we can't touch it

(c) TTL

(d) length may be padding (c) Record route option enable.

ISRO 2014

ex: 1st 8bit: 0100 0010 → actual header length: 2×4 = (8) ← some mistake (c) correction is there

(a) the no. of hop can travel 2.

I don't even Know TTL

(b) the total no. of byte in header is 16 Byte

↓

no, it's 8 Byte

(c) upper layer protocol is ICMP

I don't Know

Version | HLEN

(d) receiver reject the packet

b/c min. IP header = 20 Byte

ex: IPv4 packet

HLEN $(1000)_2$ → 8×4 = 32 byte. of header

no. of byte of option are being carried by this packet.

option = 32 − 20 = 12 Byte.

ex: IPv4 packet

HLEN is 5 → 5×4 = 20 Byte

total length is $(0028)_{16}$ → $2\times16^1 + 8\times16^0$

⇒ $32+8 = (40)_{10}$

how many byte of data are being Carried by this packet

data length

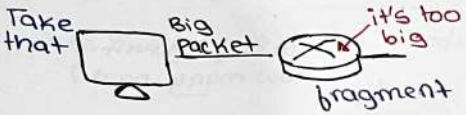
⇒ 40 − 20

⇒ 20 Byte.

ex IPv4 packet
arrived 1st few Hexadecimal (4500002800010000 | 01 | 06 ...)16
Protocol
TTL

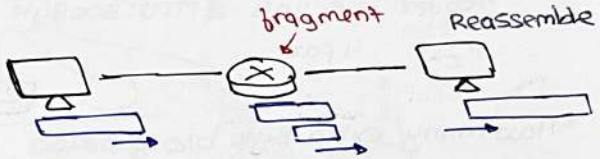
④ no. of hopes it can travel.
skip 8 byte → then we get
01 ⇒ 1 hop

Fragmentation



Problem: how to connect n/w with different max. packet size?
↓
need to split up packet

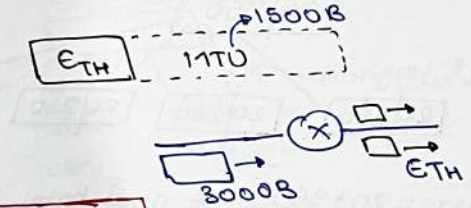
IPv4 fragmentation

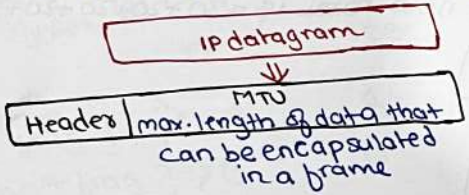


- divide large packet into fragment
- Receiving host reassembles

MTU (max. transfer unit)

max. data that DLL can carry.





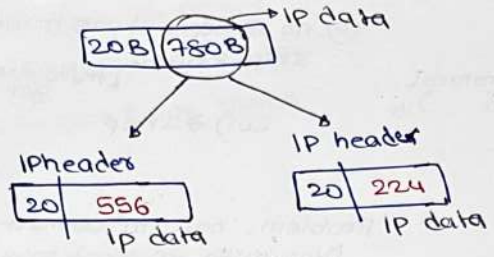
- each fragment has it's own header
- fragment itself can be fragment
- fragmentation can be done by source or any router.

- ip datagram total length is = 800Byte.
  ip header: 20Byte
  ip data = 800 − 20 = 780B

20	780

MTU: 576Byte. → 576 − 20 = 556B

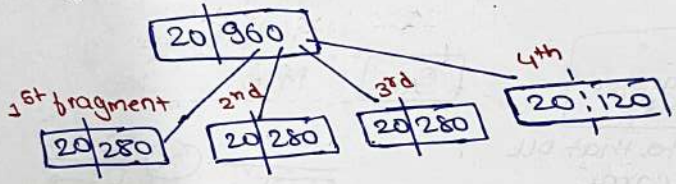
ETH	576B



ex: Suppose a datagram carry ip data (excluding header) = 960B
Header is of 20B & MTU: 300Byte
4 part.

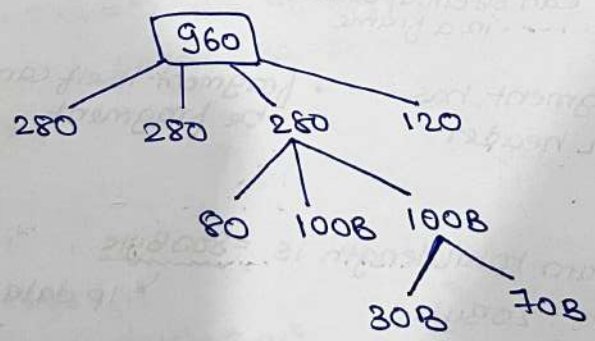
• it will fragment into how many part?

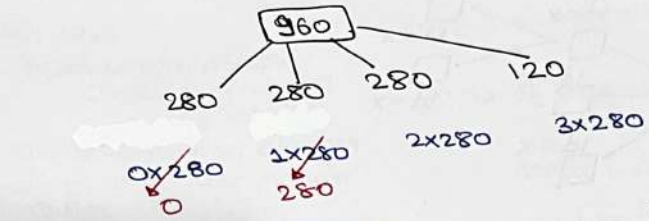
• How many extra Byte b/c of header we need to carry including all fragments.

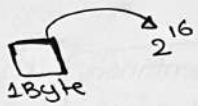


Earlier total was = 20+960 & now total is = 20+20+20+20+960

the difference is 60Byte

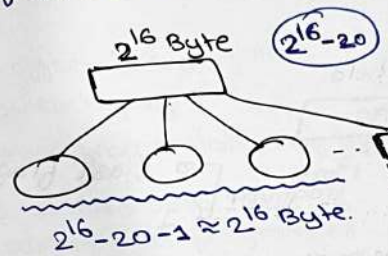






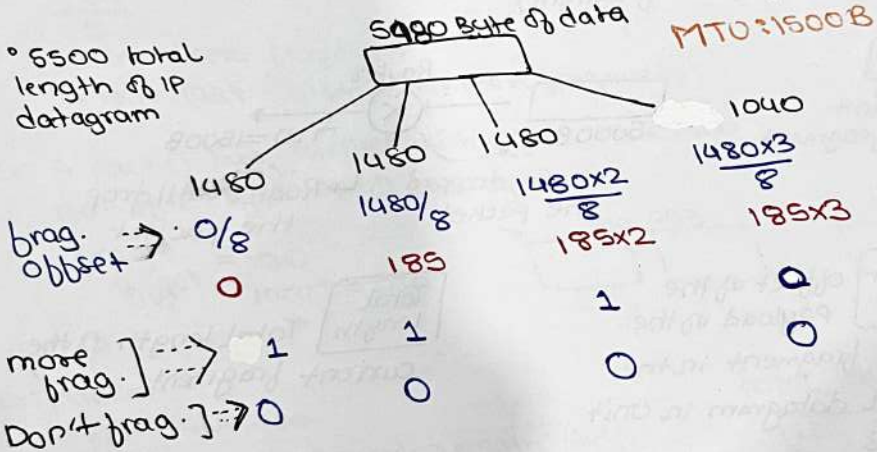
Fragment offset

this is 13bit field $\Rightarrow 2^{13}-1$



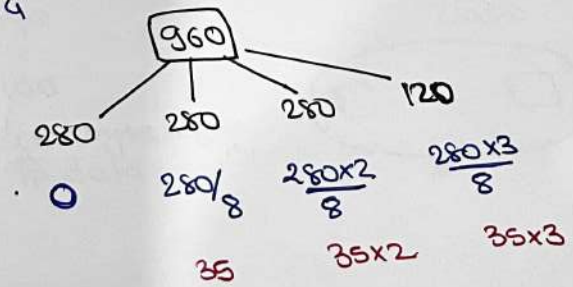
which can't be represented in 13bit so, we divide it by 8.

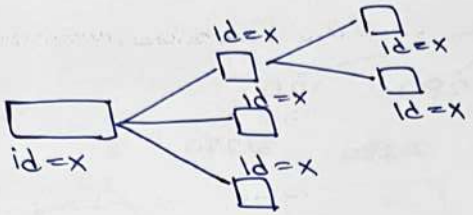
frag. offset ≈ 2^{16} Byte.



④ no. of fragment: 4

ex:



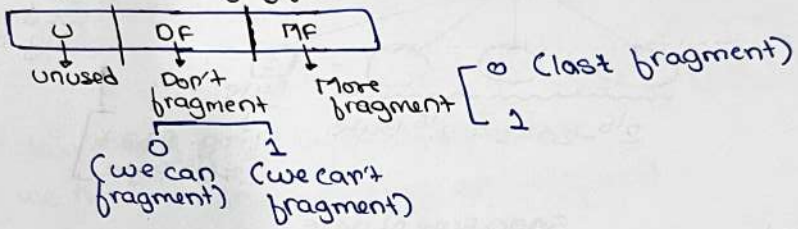


Fragmentation Related field's

1. Identification to match up with other fragments.

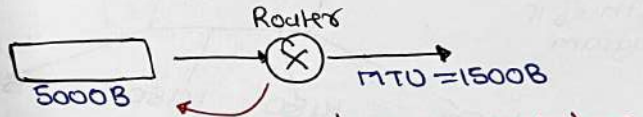
2. Flag's

3 subfield in the flag field.



DF=1

I can't fragment



I have dropped the packet

↳ Router will drop the packet

Fragment offset offset of the payload of the current fragment in the original datagram in unit of 8B

Total length Total length of the current fragment

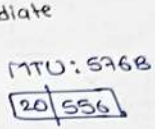
(4)

ex: IPv4
total length: 1500B
(including header) of 20 Byte

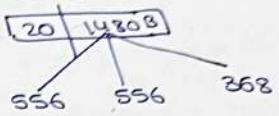
arrive through an ethernet link at an intermediate router.

- next link only allows datagram upto 576B MTU
- no of fragment are created

MTU: 576B



total length: 1500



ex: Why use a Byte offset for fragment rather than no. each frag.?
Allow further fragmentation of fragments.

ex: A packet has arrived with M bit : 1
Can we say packet was fragmented? (yes)

M bit
1

means ithese fragment is not last (atleast one more fragment)

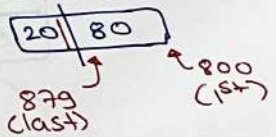
ex: A packet has arrived with offset value : 100
no. of 1st Byte? no. of last byte?

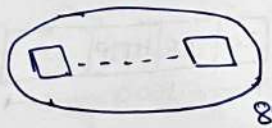
frag. offset = 100
100 × 8 = 800

ex: packet
offset value : 100
HLEN: 5 → header length: 5×4 = 20
total length : 100
no. of 1st & last Byte

frag. offset : 100
⇒ 800 byte of ip data

total length: 100





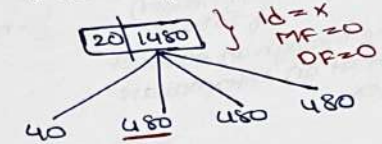
800 Byte.

ex: IPv4 packet
router receive 1500 Byte

20 Byte IP header

Id=x
MF=0
DF=0
MTU=500 Byte

- what's ID, length, offset, flag field contain of headers of 3rd fragment



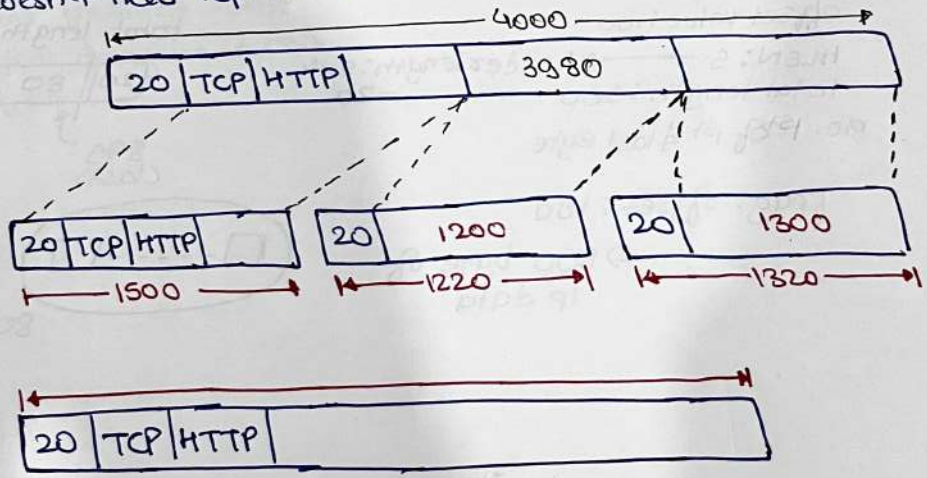
Id = x
MF = 1
DF = 0

total len. = 500

frag. offset : $\frac{480 \times 2}{8} = 120$

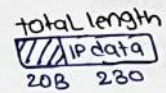
ex: IPv4 packet include a TCP segment
IPv4 packet need to fragmented
which fragment include the original IP header.
which fragment include the TCP header.
↳ only the 1st one

- Intermediate routed anyway doesn't need TCP header.

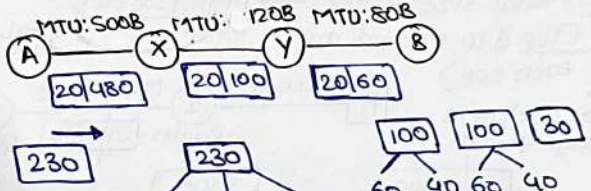


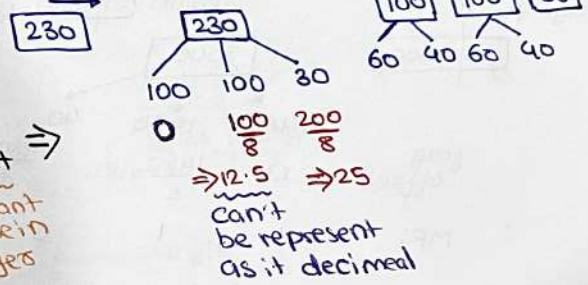
question on fragmentation & Reassembly

① packet with total length: 250B
(no options) → means header length: 20B
id field: 17
Sent from A to B via Router x & y
& subnet connected to A has MTU: 500B
& subnet connected to B has MTU: 80B
& subnet ----11---- b/w x & y has MTU: 120B.
& DF: not set (can't frag.)



- no. of fragment routers x divide into & each have length?

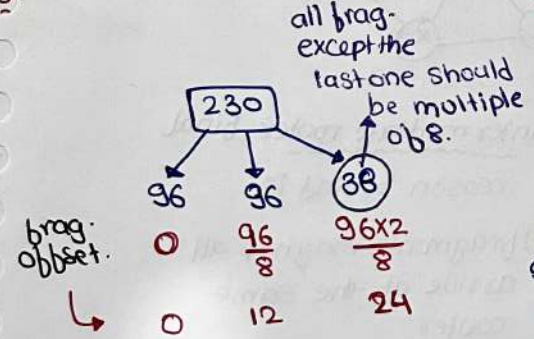


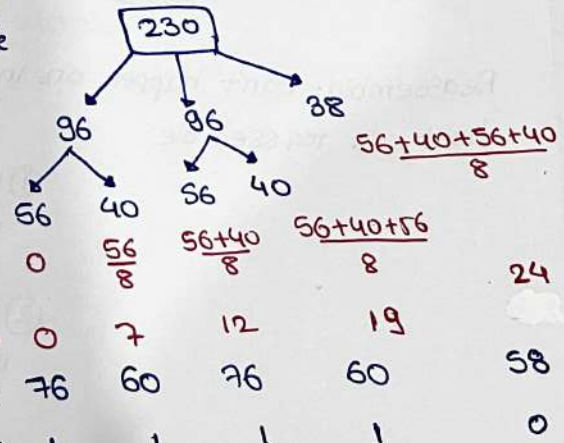


frag. offset ⇒ we want it to be in integers

can't be represent as it decimal

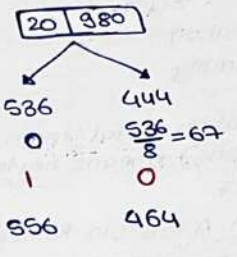
So we need to divide in such a way that we can represent in binary (closet divisible of 8)



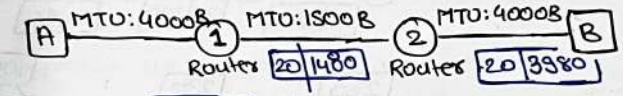


② IP packet
w/o options] header length: 20B
link with MTU: 560B
Before fragmentation

- Header length = 5
- Total length: 1000
- Id: 998
- Flags: DF: 0, MF: 0
- TTL: 64
- frag. offset: 0

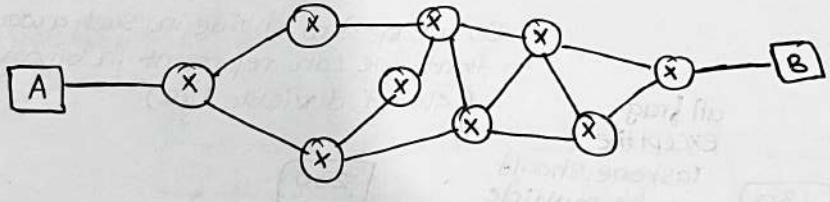


③ total size: 3020B → 3020 (IP | TCP | 2980) (20 | 20) total data
only 1st contain TCP
(TCP & IP header are each 20B)
not useful in frag. question



frag. offset: 0, 1480/8 = 185, 1480×2/8 = 370
MF: 1, 1, 0

- Routers does not need TCP header



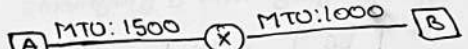
Reassembly can't happen on intermediate router final host will reassemble.

reason behind is

① fragments may not all arrive at the same router

② Reassembled packet may be fragmented again

④ A send 4000B to B
A with link MTU: 1500B to x
& x to B with MTU: 1000B



(Assume the IP header must fit within the MTU but not the datalink layer)

already know

assume header length: 20B

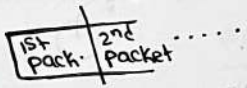
frag. offset ⇒	0	122	185	307	370	492
MF ⇒	1	1	1	1	1	0

Reassembly Algo.

```
Int frag.offset=0;
while(MF==1){
    Search for frag.offset
    frag.offset=(total len_header.len)/8
}
```

Within Same identifier

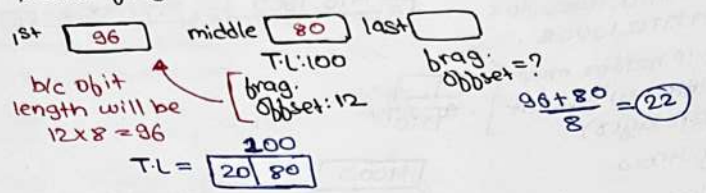
- to find next packet we find it's frag. offset.



- Sorting them & then arranging will be a bad idea as few packet might be lost or have not been reached.

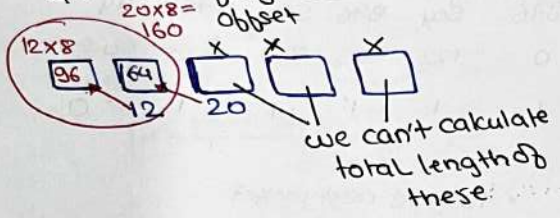
⑤ Packet frag. into 3 fragments

IP header size : 20

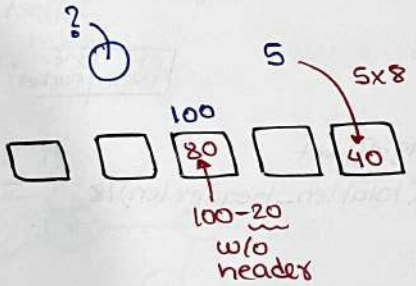


⑥ Packet divide into 5 fragment

2nd & 3rd frag. are 12 & 20



⑦ frag. offset
total length



- which of the following additional information is sufficient to calculate TL of 4th fragment

(a) frag. offset of 3rd fragment

(b) ——//—— 2nd fragment

(c) frag. off of 5th fragment & total length of 4th fragment

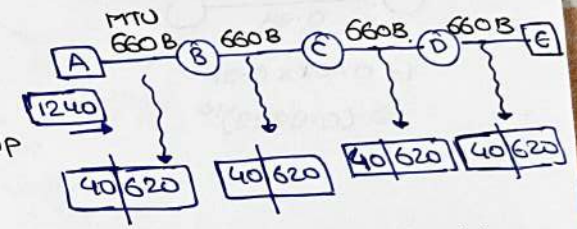
- If a fragment is lost by an intermediate router then the whole TCP segment is send again

(8) identification no. of the frag. be same if frag. is a source node? no.

each time a source node fragments a packet it use a different id. no. for each frag. packet destined to a specific node.

the source is expected to use an id. no. different from any already sent packet for the expected life time of a packet.

(9) A send packet to E
1280B
with 3 Router & 4 link
MTU 660B & 50% packet drop due to error.
IP header: 40B



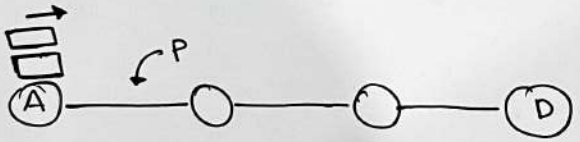
Prob. it will arrive Successfully at E

we want that 1st & 2nd packet to reach Successfully

1st frag. 2nd frag.

So, $\left(\frac{1}{2}\right)^4 \times \left(\frac{1}{2}\right)^4$

$\Rightarrow \frac{1}{2^8} = \frac{1}{256}$



P(success) = P of 1 datagram

no frag. then A to D Successfully reached $= p^3$

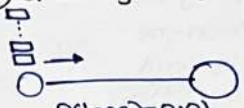
[fragmentation with 2 frag.
P(successful of datagram) $= p^3 \times p^3 = p^6$]

↳ decrease

Tricky
(10) question.

Packet frag. into 10 fragment each with an independent loss prob. of 1%.

(a) a datagram (each frag.) is transmitted once.

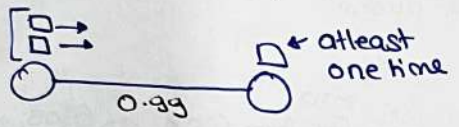


P(loss) = 0.01
P(success) = 0.99

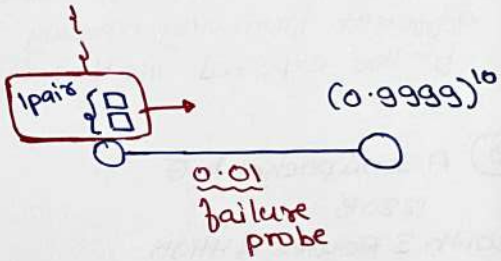
$0.99 \times 0.99 \times \dots \times 0.99$
$\Rightarrow (0.99)^{10}$
≈ 0.904

P(datagram is successfully transmitted)

(b) the datagram (each fragment) is transmitted twice?



$1 - 0.01 \times 0.01$
$\Rightarrow (0.9999)^{10}$



$(0.9999)^{10}$

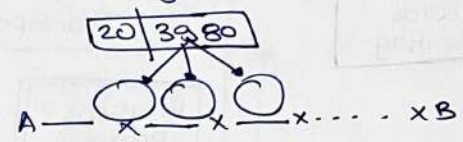
P(atleast one of the copy reaches) = 1 − no copy reaches

$(0.9999)^{10}$.

$P(A \cup B) = 1 - P(\overline{A \cup B})$
$\Rightarrow 1 - P(\bar{A} \cap \bar{B})$
$\Rightarrow 1 - P(\bar{A}) \cdot P(\bar{B})$
$\Rightarrow 1 - 0.01 \times 0.01$
≈ 0.9999

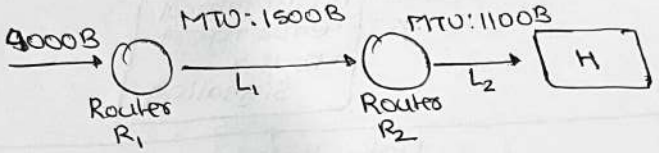
ex: IPv4 datagram
4000byte
which traverse 10 link
A to B
assme link 1,2,5,8
have MTU of 5000Byte
& other link MTO 1500 Byte

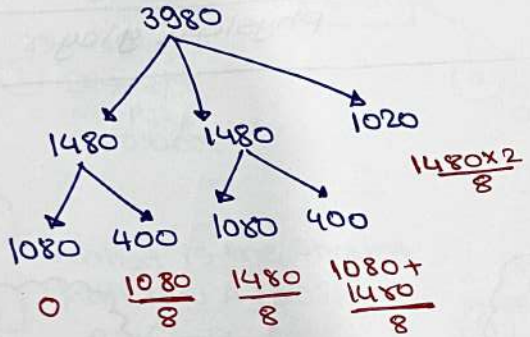
total length: 4000B



the packet will be
divided only at 3rd link
into 3 fragment
after that no fragment.

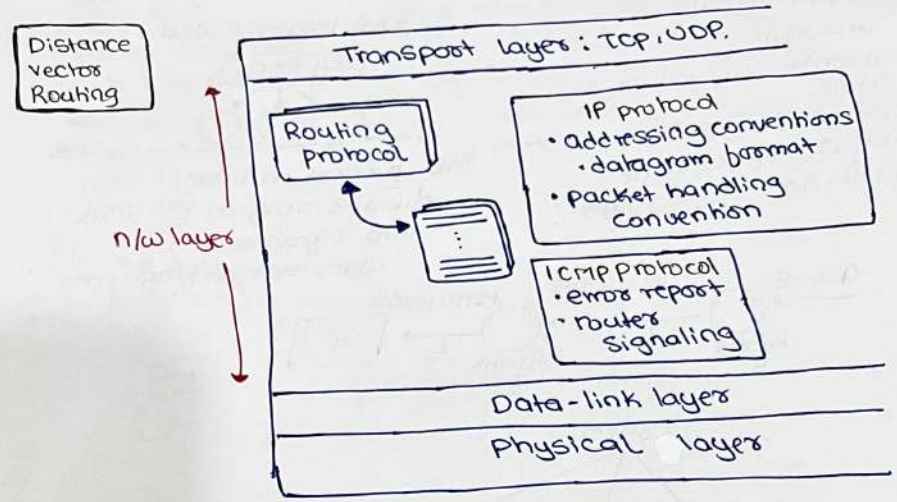
ex:

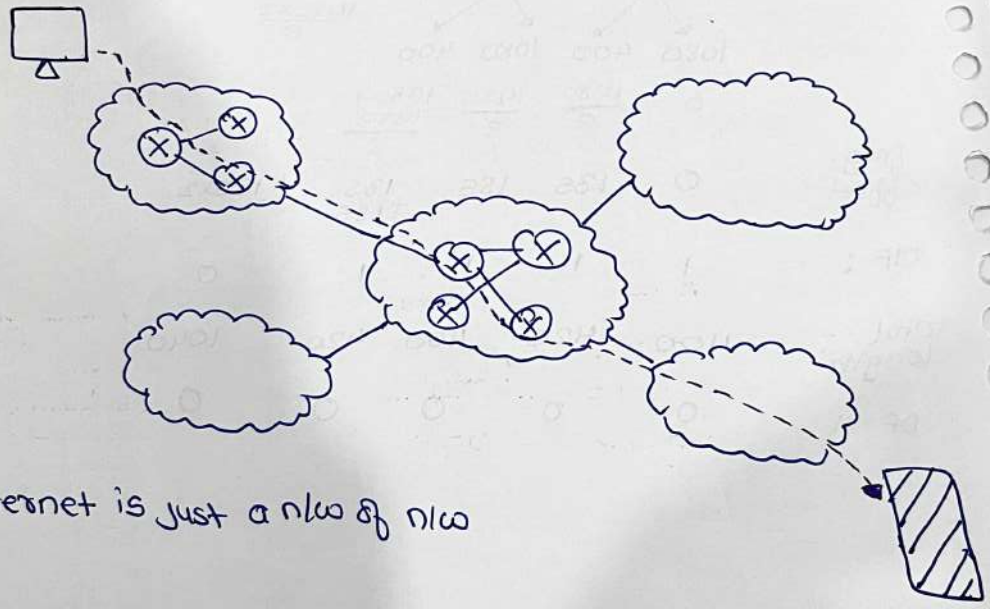




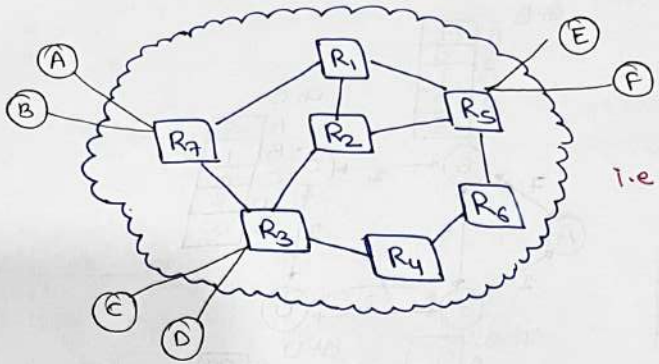
frag. offset :	0	135	185	135 +185	185x2
MF :	1	1	1	1	0
total length :	1100	420	1100	420	1040
DF :	0	0	0	0	0

Distance Vector Routing

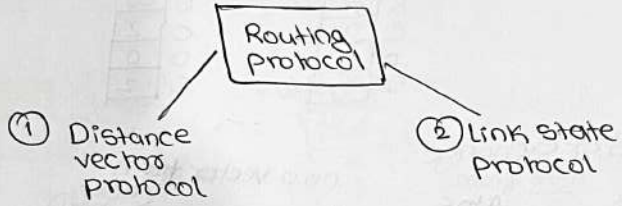




Internet is just a n/w of n/w



i.e Autonomous System(AS)
(or)
Domain

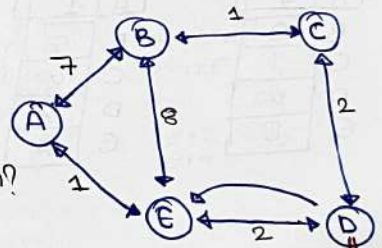


Distance Vector Routing

- what is the shortest path b/w A to C?

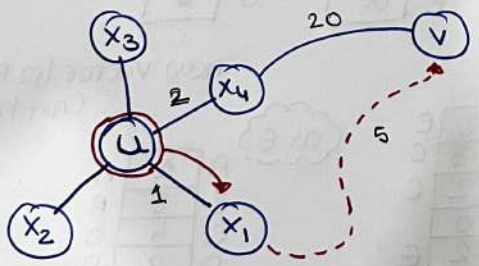
A–E–D–C

How 'A' get to know shortest path?



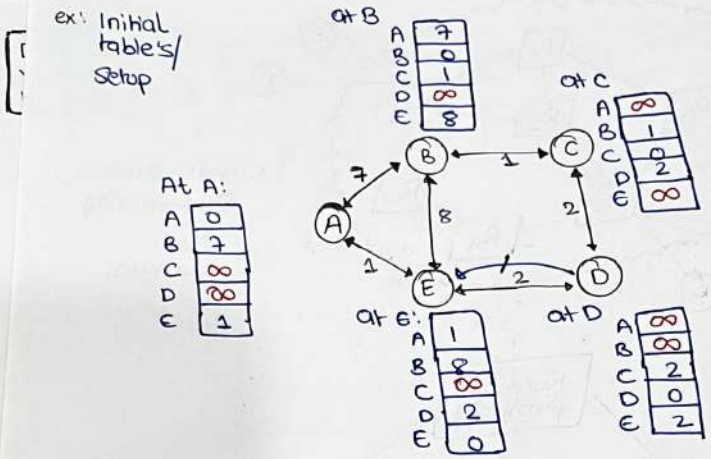
I'm 2 from C
& '0' from D
& '2' from E

Goal → each router "u" must compute for every other router 'v', the nxt hope neighbor 'x' that is on the least cost path from u to v.

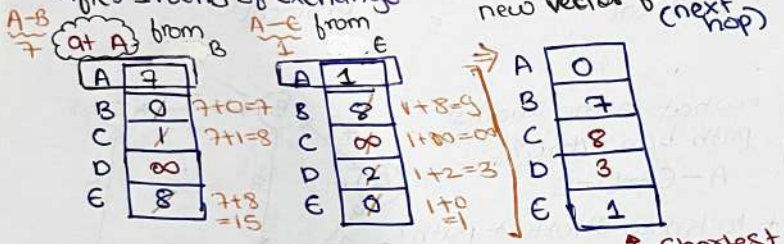


- each node announces dist. to each dest.

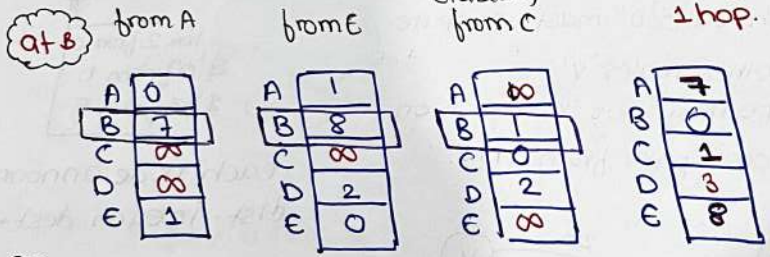
ex: Initial table's/ setup

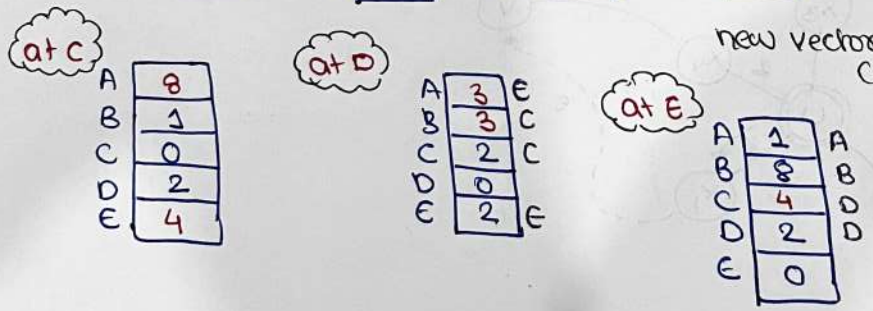


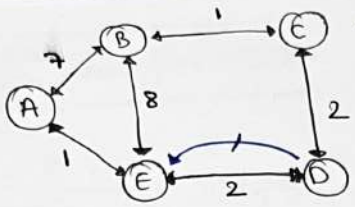
after 1 round of exchange



↑ shortest path from A to every node using just 1 hop.



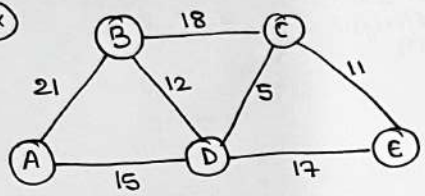




no. of iterations DVR algorithm need to converge?

=> After 2 iteration it coill converge.

Ex



routing table for E

- Initial table

Dest.	Cost	nxthop
A	-	-
B	-	-
C	11	C
D	17	D

- after 1 iteration (using 1 hop)

Dest.	Cost	nxt hop
A	32	D
B	29	B or D
C	11	C
D	16	C

- after 2 iteration

Dest.	Cost	nxt hop
A	31	C
B	28	C
C	11	C
D	16	C

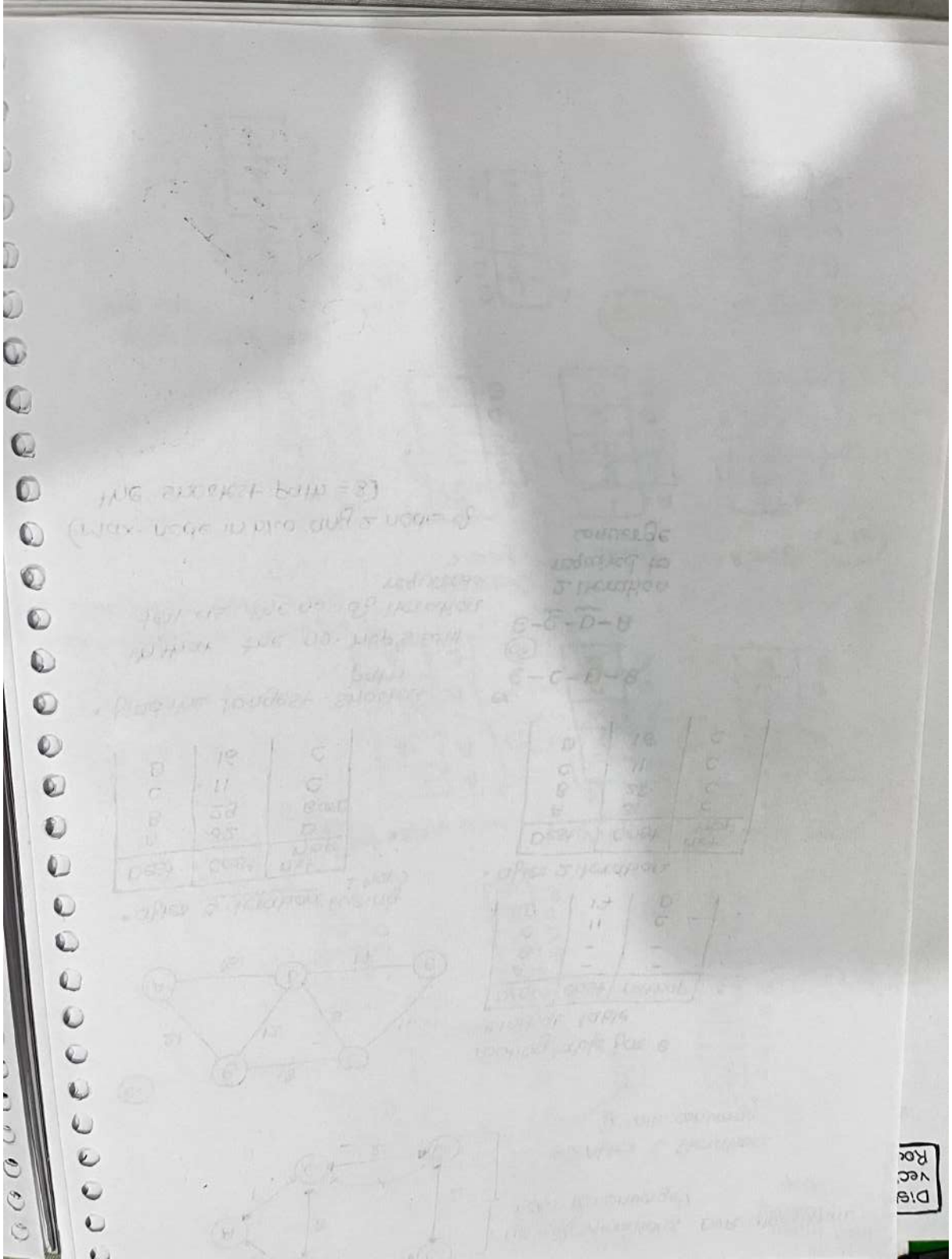
- find the longest shortest path

in that the no. hop's will tell us the no. of iteration required.

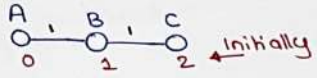
ex:
E-C-D-B
OR
E-C-D-A

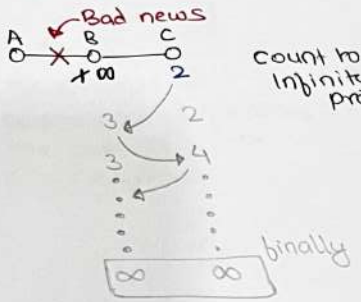
2 iteration required to converge

(Max. node in b/w any 2 nodes of the shortest path = 3)

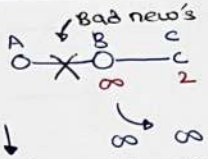


Count to infinity: slow convergence

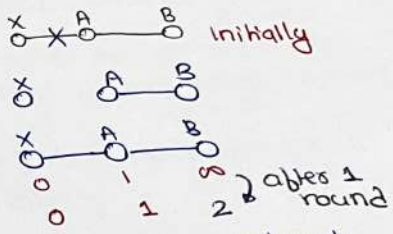




count to infinity problem



- if B advertise 1st then there is no problem of count to infinity



good news travel fast.

- good news travel quickly, bad news slowly
  But we want both news to travel fast.

ex: under some condition
a dist. vector protocol finding min. cost path suffers from the "count to infinity"

(a) count to infinity prblm may arise in a dist. vector protocol when the n/w get disconnected
True

(b) count to infinity prblm may arise in a dist. vector protocol, even when the n/w never get disconnected False.

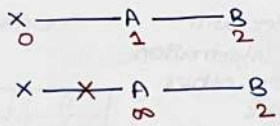
- In DV routing, can count to infinity happen when a link is added to the n/w

(no) b/c adding length of the shortest path can only decrease, never increase.

Di
ve
Re

Split horizon

Don't send route back to where you learned it from.



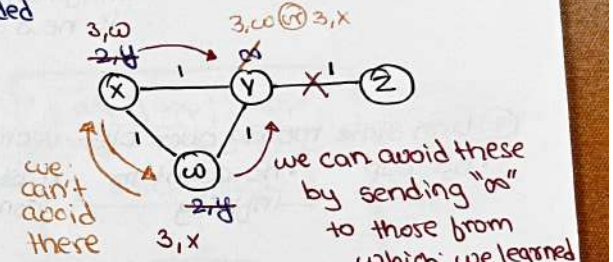
Fixes to count to infinity prblm

1. Split horizon
   - A router never advertises the cost of a dest. to a neighbor

2. Split horizon with poisonous Reverse.
   - if X routes traffic to Z via Y (instead of not telling anything at all)
   - X tell Y that it's distance to Z is infinity

- 3 node instability
  2 node instability can be avoided using split horizon + poison reverse

But
3 node stability can't be guaranteed.



Link State Routing → every router will flood the information to every other router

more useful

- every router knows whole topology

⇒ Shortest path algo ⇒ Dijkstra algo

- Link state packet

A	
B	4
E	5

B	
A	4
C	2
F	6

C	
B	2
D	3
E	1

D	
C	3
F	7

E	
A	5
C	1
F	8

F	
B	6
D	7
E	8

An LSB packet contain

A
seq.
age

why age? work as TTL

we find out if LSB is new or not using seq. no. (Greater the seq. no. means newer the packet)

distinguish if packet is new or old

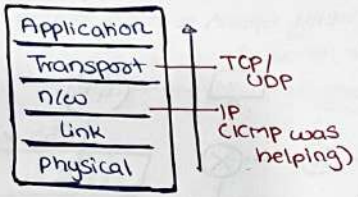
① Link state routing over Dist. vectors

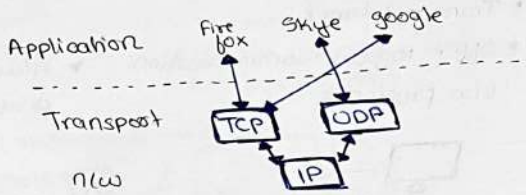
- no loop
- no count to infinity
- faster convergence

② Distance vector over Link state

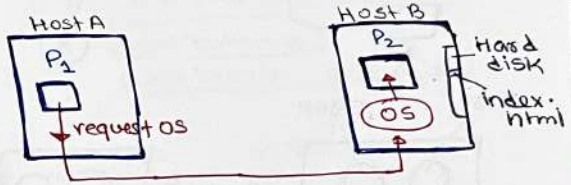
- Smaller update traffic
- simpler implement

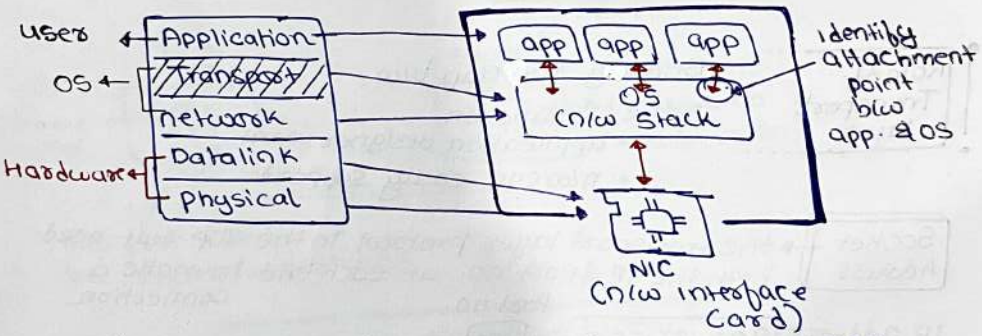
Transport layer

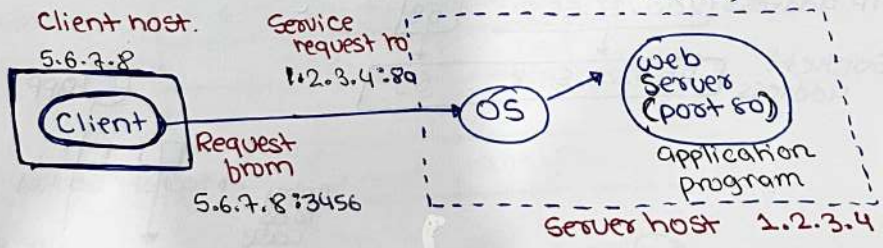




Inter process Communication (IPC)



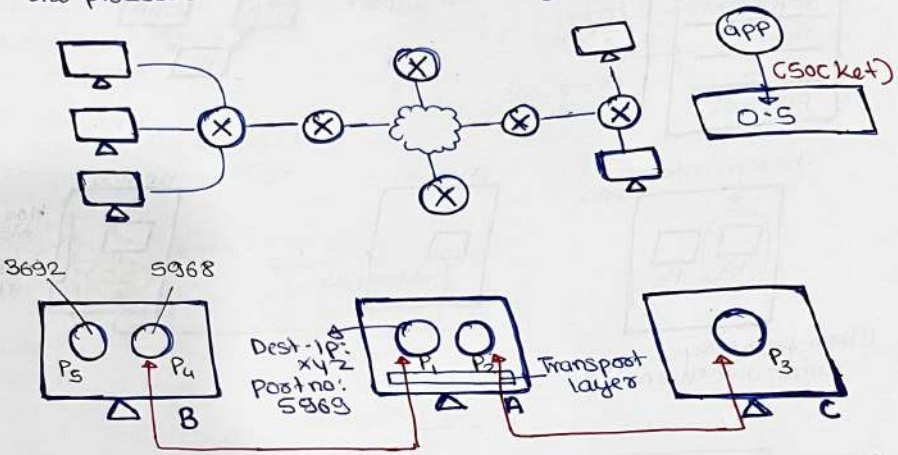




- node to node (Data link layer)
- Host to host (n/w layer)
- process to process (Transport layer)

Socket: IP + port. no.

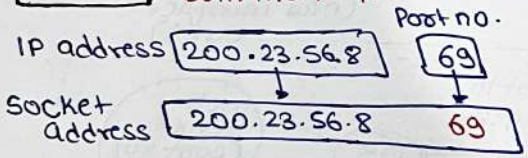
- Transport layer
- offer logical communication b/w processes
- n/w processes think that they directly speak to each other.



Role of Transport layer

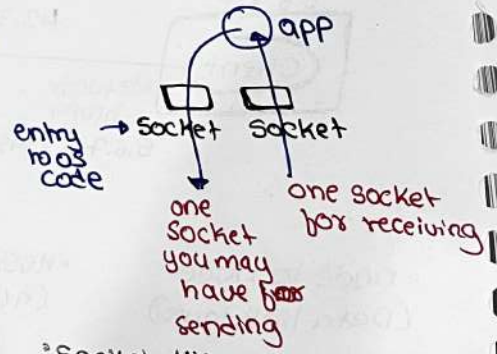
- Bridging the gap b/w
  - → the abstraction
    - application designer want
    - n/w can easily support

Socket Address → the transport layer protocol in the TCP suit need both the IP & port no. at each end to make a connection



① to whom send data (Dest. IP, Dest. port)

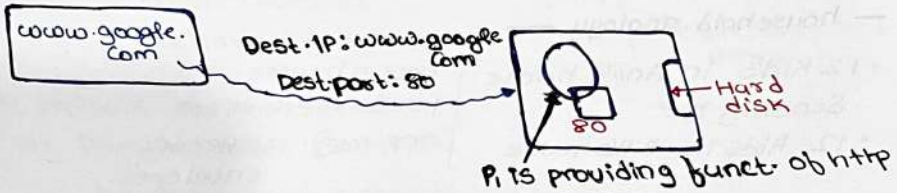
② what is the data (size of data (pointer to data) as well)



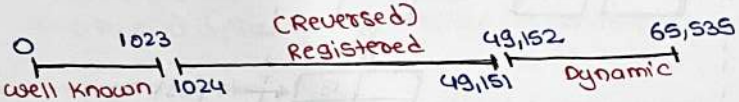
- Socket like a system call once open then no need of O.S now we can communicate using socket

Port no. → unsigned integer (will be allocated to process using O.S)

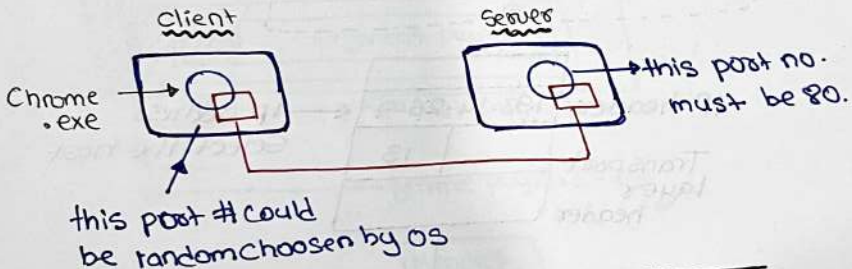
- Servers typically bind to well-known port no.
- Client use OS assigned temporary port.

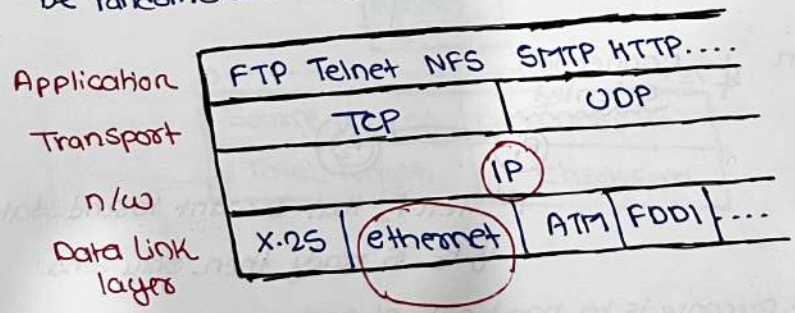


Send to this IP
Point to data, size of data



Port no.: 16 bit unsigned integer





Transport (Vs) n/w Layer service & protocols

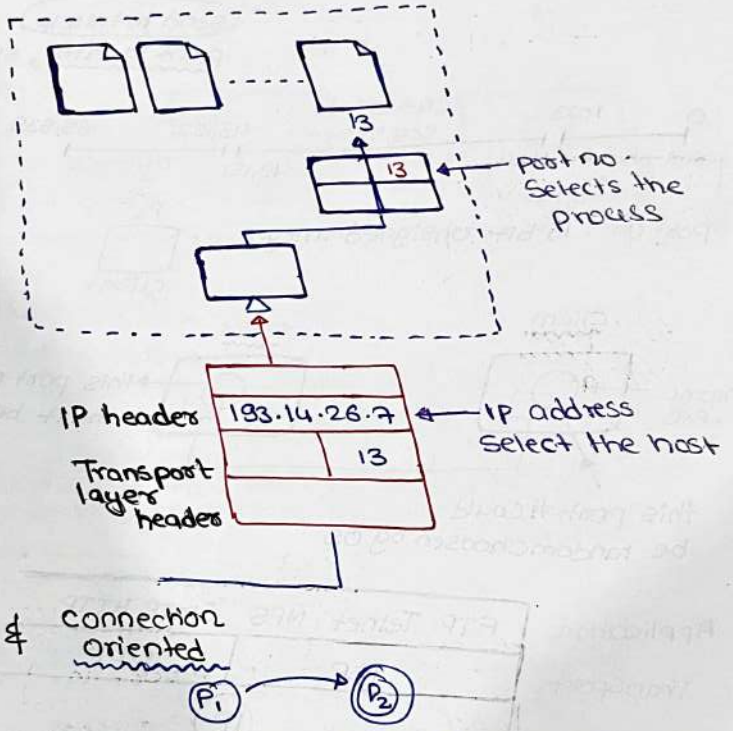
logical communication b/w processes

logical communication b/w host

— household analogy —

- 12 kid's in Ann's house sending to
- 12 kids in Bill's house

host = houses
processes = kids
app msg. = letters in envelopes.



Connection less & Connection oriented

1st tell P_2 that I want to send data, if P_2 is ready then only send.

Analogy purpose is to hand out chocolates to people.

↳ connection less

Who ever stops by to your Station, you handout Chocolate no matter what

↳ connection oriented

Before you handout chocolate you ask, after the agreement you then hand out the Chocolate A/c.

Transport layer → mission: Transfer a segment from one process to another.

Service:
- Multiplexing–Demultiplexing
- Error Detection
- Reliable data transfer

UDP → connectionless service

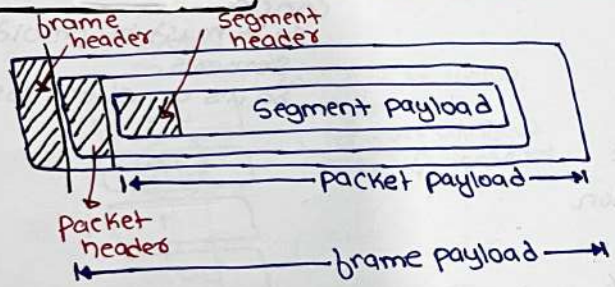
TCP → connection oriented.

- TL protocol are implemented only at terminal node's (Computer)
- Routers & switches don't exam this layer

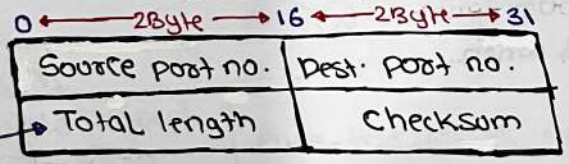
Transport layer services

- responsible for end to end delivery
- TCP & UDP: reliable, unreliable

User datagram protocol (UDP) → connectionless (directly send)
→ unreliable (doesn't care about packet loss)



Header format



Including header

Header length = 8 Byte

UDP Checksum

Goal: detect errors in transmitted segment

	1st no.	2nd no.	Sum
Transmitted:	5	6	11
Received:	9	6	11

Received: 9 6 — Receiver computed checksum

11 — Sender computed checksum

$\neq$ (1)

ex: UDP header in hexidecimal format

CB84 000D 001C 001C

Source port | dest. port | total length

001C ⇒ $16^1 \times 1 + 16^0 \times 12$

⇒ 20

0 1 2 ... A-10 B-11 C-12

- What is the length of data? 20
- Is packet directed from client to server or vice versa?
  client to server
  - as Dest. port is well known port from 0 to 1023 (000D) so it must be providing server's. so it's a client to server

Why use UDP?

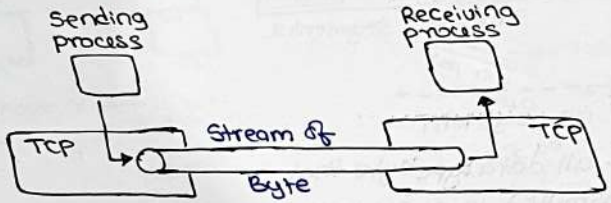
- no connection state
- no delay for connection establishment
- Small packet header overhead (header only 8 B long)
- Fine control over what data is sent & when.

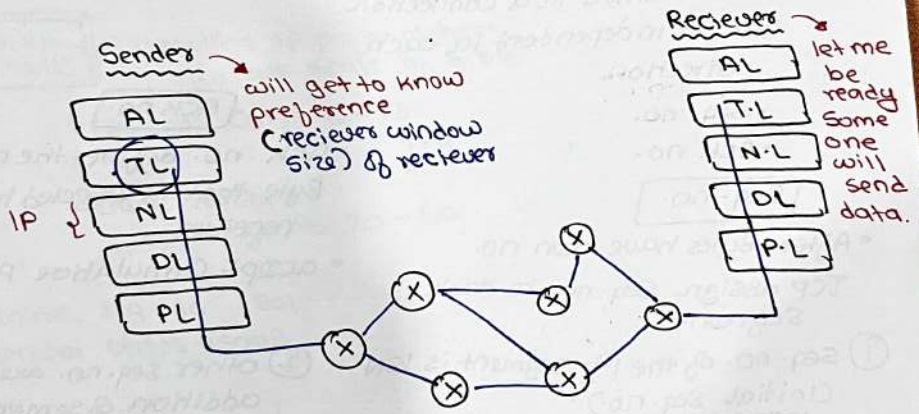
Transmission control protocol (TCP)

UDP
- ↳ connection less protocol
- ↳ unreliable
- ↳ don't use seq. no.
- ↳ no flow control

TCP
- ↳ connection oriented (before sending the data, we will establish connection)
- ↳ reliable protocol (this is a responsibility of TCP to transfer data error free & inorder (to upper layer))
- ↳ use seq. no.
- ↳ flow control is enforced.

TCP
- ↳ is full duplex (using one connection we can send/ receive data)
- ↳ Byte oriented protocol (use no. every byte)

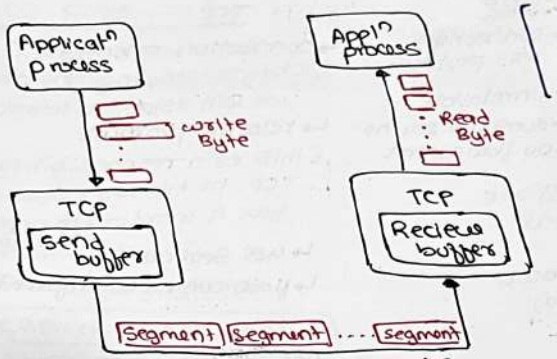




ex: at nlw layer, we have unrelaible protocol which is IP, is it still possible to have relaible protocol at Transport layer. yes,

even though NL may provide corrupted or out of order data, still at TL we can take care of this.

- TCP at Datalink layer are frame oriented.



TL: segment
NL: Datagram
DLL: frame

TCP feature ---------- no. system

TCP no. all data Byte that are transmitted in a connection no. is independent in each direction

- Seq. no.
- Ack no.

Seq. no.

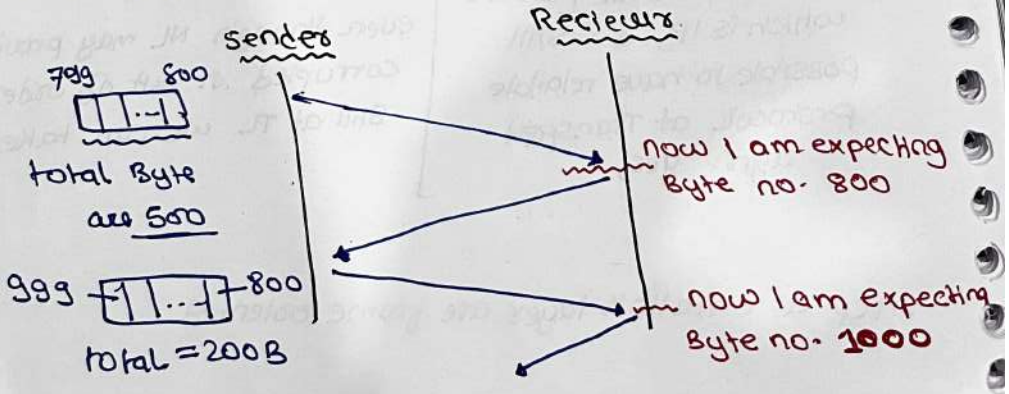
- After Bytes have been no. TCP assign Seq. no. to each segment

(1) Seq. no. of the 1st segment is ISN (Initial Seq. no.). random no.

Ack no.

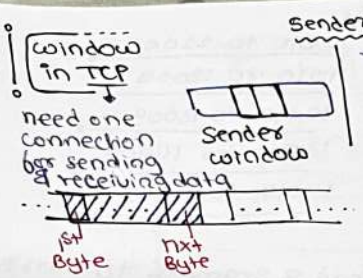
- Ack no. define the nxt Byte that is expected to receiver.
- accept Cumulative Ack no.

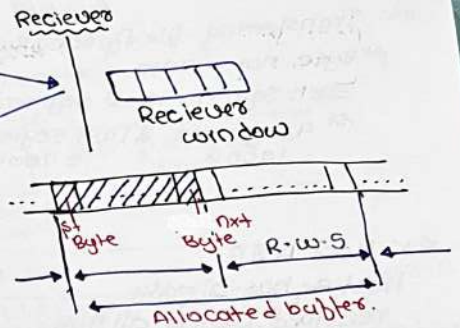
(2) other seq. no. are addition of some no.



window in TCP

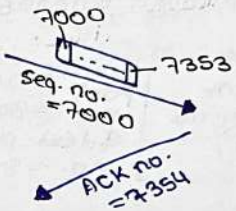
need one connection for sending & receiving data





Source port: x ← 16 bit port no.

dest. port: y



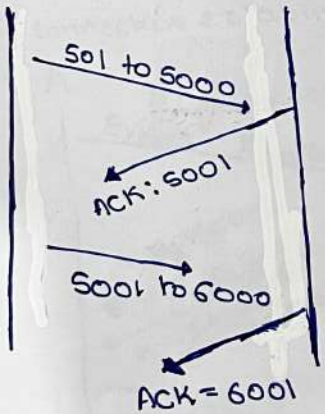
HLEN (4bit) → just like IP protocol here also we scale to 4 b/c

this can carry 0–15 ×4

Actual header length is 20–60

ex: Initial seq. no: 501

Sender sends 4500B



ex: Transferring file of 6000 Byte.

1st Byte no. = 10010

each segment is 5 segment

1st 4 segments 1000 B & last segment is 2000 B

10,010 to 11009

11010 to 12009

12010 to 13009

13010 to 14009

14010 to 16009

ex: Host A & B

Host B has already received from A all Byte 0p through byte 126.

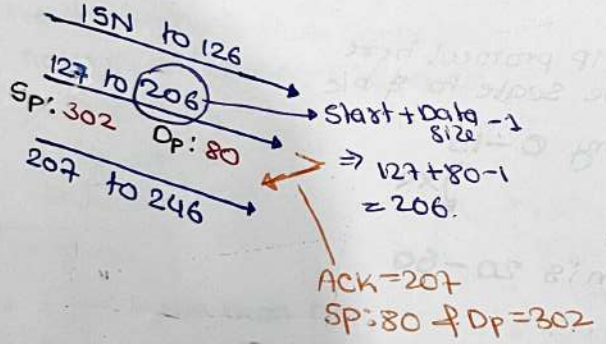
- Host A send 2 segment to Host B back to back

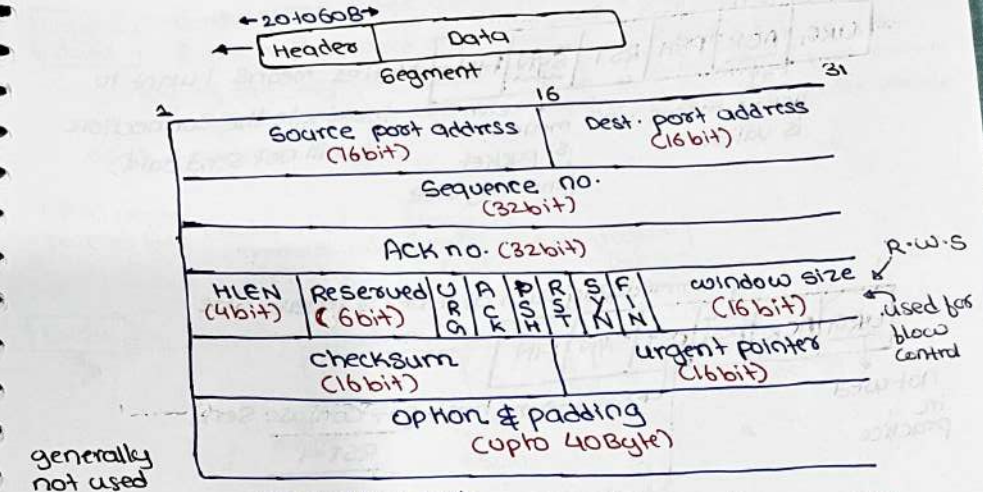
  1st contain 80 Byte, 2nd segment 40 Byte

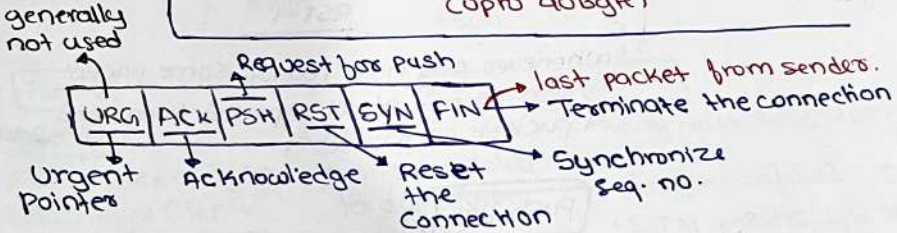
In the 1st Segment no. Seq. no. (127)

Source port no. = 302 & dest. port no. = 80

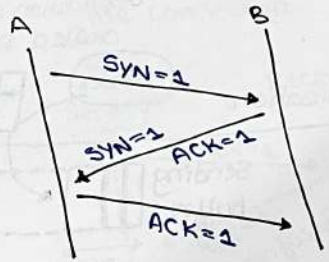
- Host B send an Ack coheneuer it receive a segment from Host A.



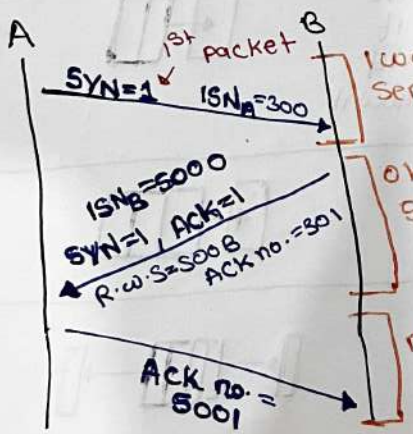


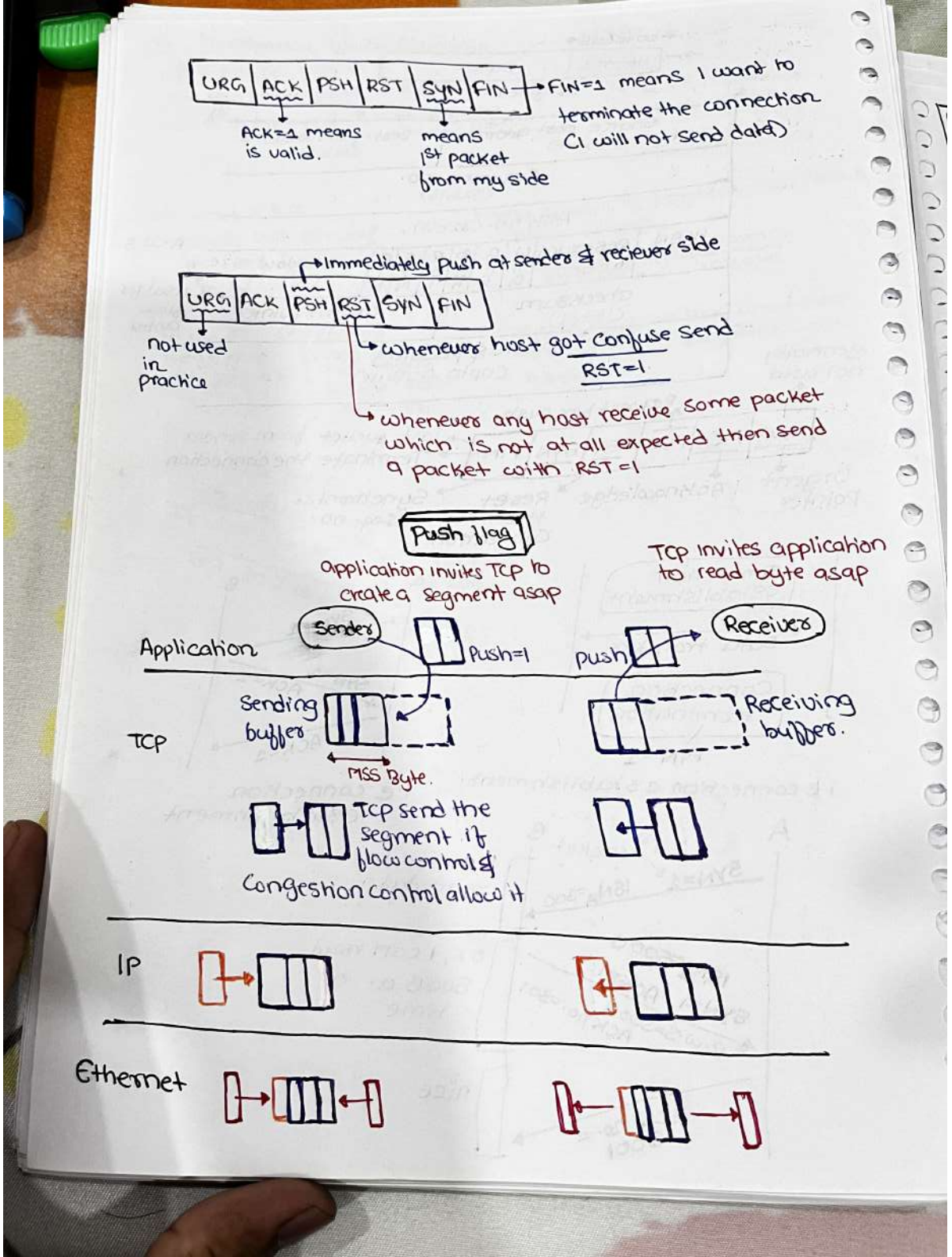


i.e connection establishment



i.e connection establishment




URG
ACK
PSH
RST
SYN
FIN
FIN=1 means I want to terminate the connection (I will not send data)
ACK=1 means is valid.
means 1st packet from my side
Immediately push at sender & reciever side
not used in practice
whenever host got confuse Send RST=1
whenever any host receive some packet which is not at all expected then send a packet with RST=1
Push flag
Application invites TCP to create a segment asap
TCP invites application to read byte asap
Application
Sender
Push=1
push
Receiver
TCP
Sending buffer
Receiving buffer.
MSS Byte.
TCP send the segment if flow control & Congestion control allow it
IP
Ethernet

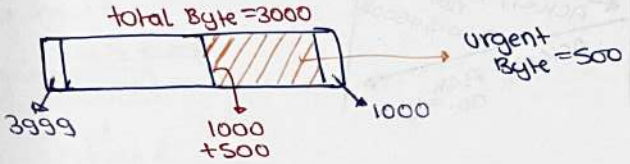
Urgent data

- Sending TCP create a segement & insert the urgent data at the beginning of the segment.
  the rest of the segment contain normal data from the buffer.

- Imp. to mention that TCP urgent data is neither a priority service nor an out of band data service

URG: urgent pointer flag (data need immediately delivery) [urgent msg end = SN + urgent pointer]

URG=1



PSH=1

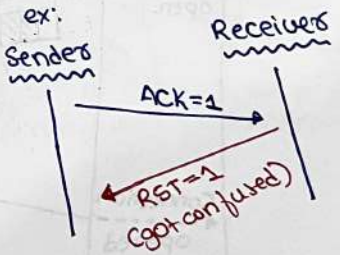
at Sender
PUSH immediately to below layer (don't wait for some MSS or anything else)

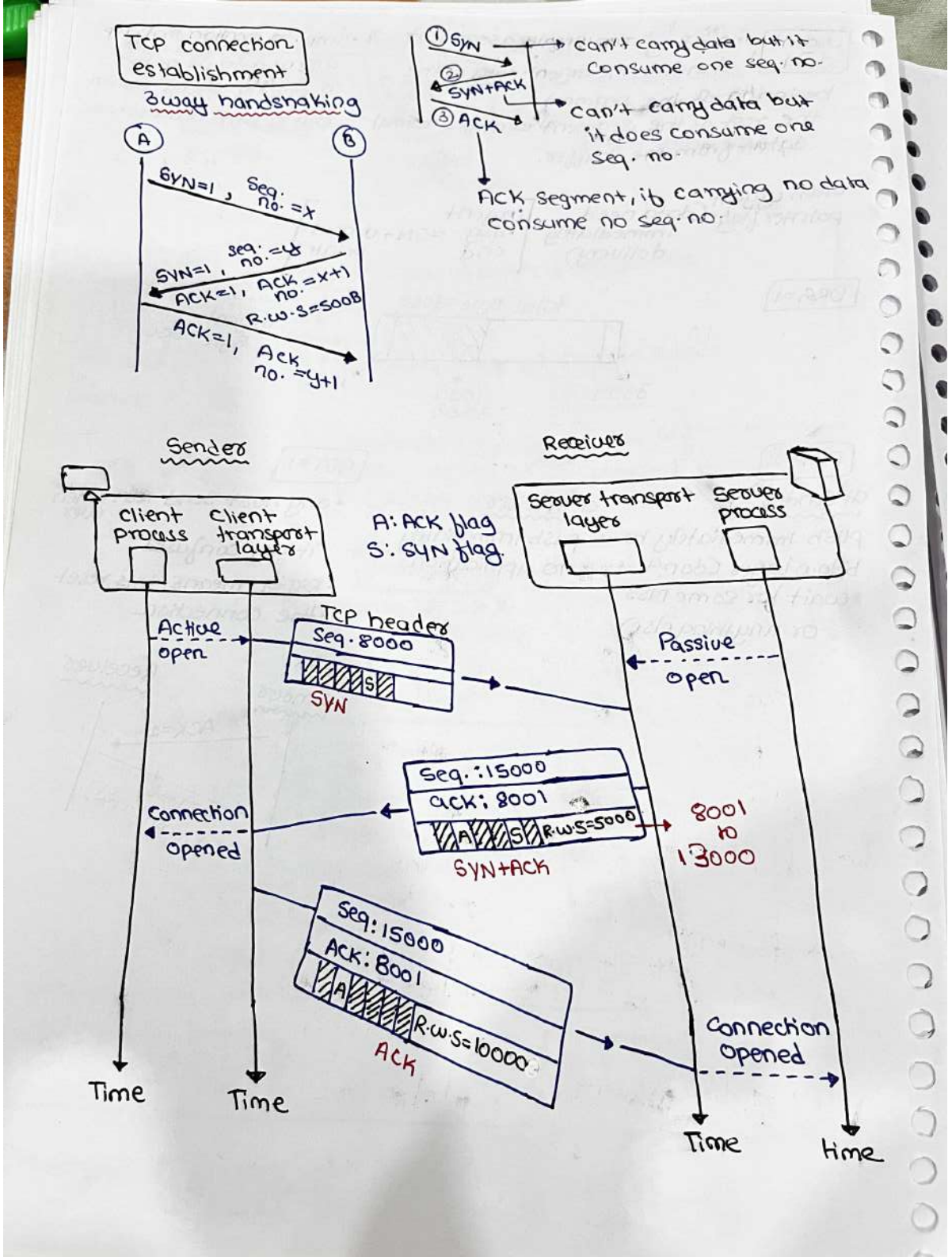
at receiver
push immediately to application

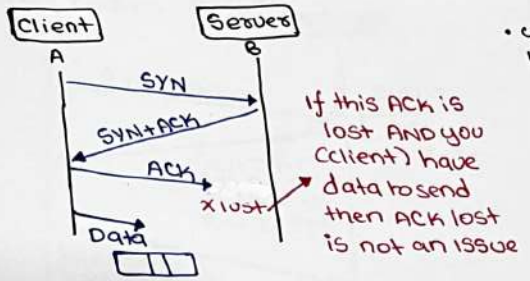
RST=1

- any host send RST whenever it get confused
- RST=1 means, lets reset the connection.

ex:



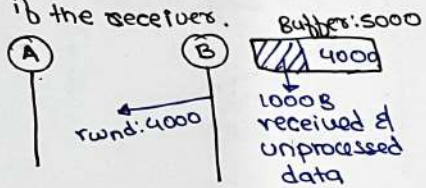

TCP connection establishment
3way handshaking
A
B
SYN=1, Seq. no. =x
SYN=1, ACK=1, seq: no. =y, ACK no. =x+1
ACK=1, Ack no. =y+1
R.w.S=5008
1 SYN
can't carry data but it consume one seq. no.
2 SYN+ACK
3 ACK
can't carry data but it does consume one seq. no.
ACK segment, if carrying no data consume no seq. no.
Sender
Receiver
Client process
Client transport layer
Server transport layer
Server process
A: ACK flag
S: SYN flag
Active open
TCP header
Seq. 8000
S
SYN
Passive open
Seq.: 15000
ack: 8001
A
S
R.w.S=5000
SYN+ACK
8001 to 13000
Connection opened
Seq: 15000
ACK: 8001
A
R.w.S=10000
ACK
Connection Opened
Time
Time
Time
Time



If this ACK is lost AND you (client) have data to send then ACK lost is not an issue

- what ultimately triggers A to retransmit the lost ACK timeout at B, trigger's a retransmission of SYN-ACK packet.

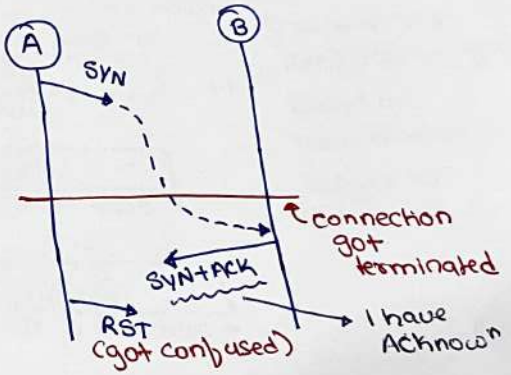
ex: rwnd value for host A if the receivers.



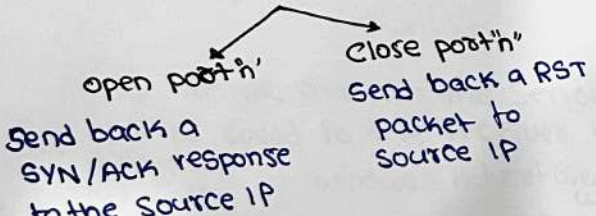
ex: 3 way handshake

- an old SYN segment from station A arrives at B

how 3 way handshake procedure ensure connection is rejected

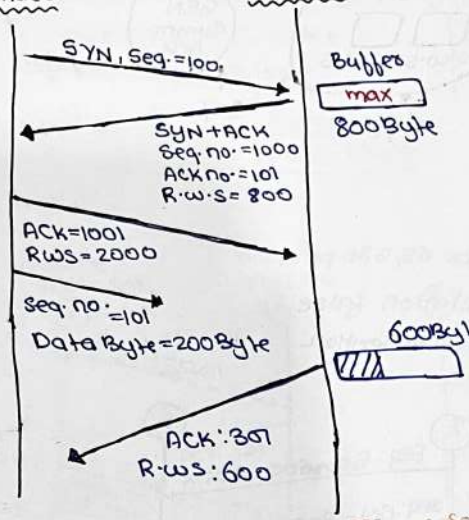


ex: "A" host that receive SYN packet to



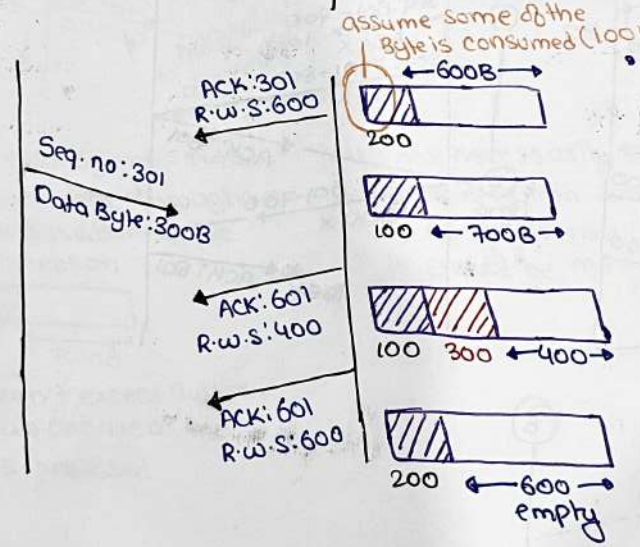
- Flow control in TCP

Sender | Reciever



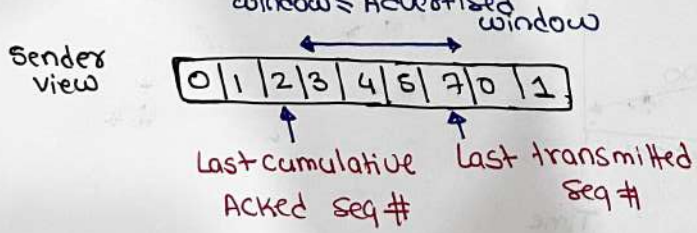
Receiver Advertised window

- Big diff. is the size of the sliding window size at the receiver is not fixed
- the receiver advertises an adjustable window size
- Sender is limited to having no more than Advertised window Byte of unAcked data at any time.

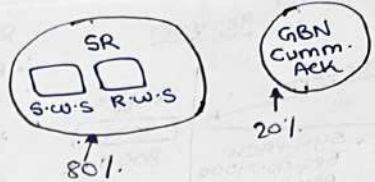


- R.w.s in TCP Keep on changing Based on how much buffer is available

- Flow control matches the Sender's write speed to the receiver read speed

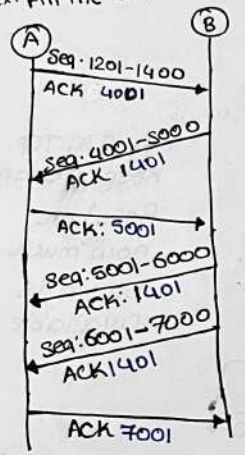


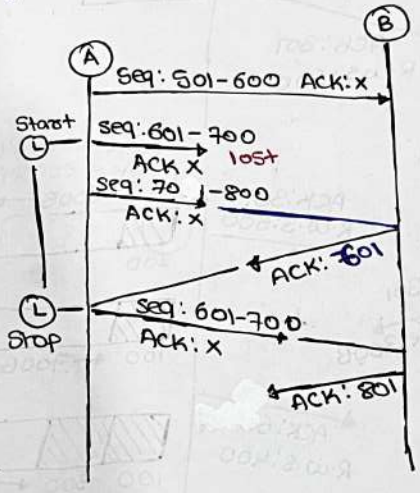
TCP use SR protocol
& cumulative ACK
(generally)
but we can use
Selective ACK if
we want

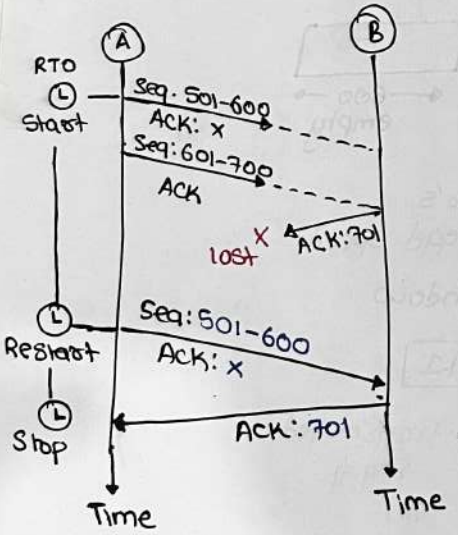


ex: the limitation in TCP for 65,536 port
is due to hardware limitation false
this is purely Software convention

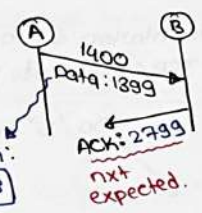
ex: Fill the value.



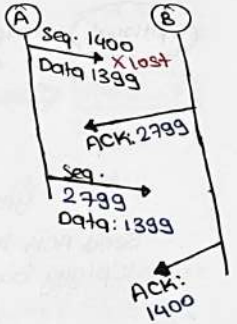




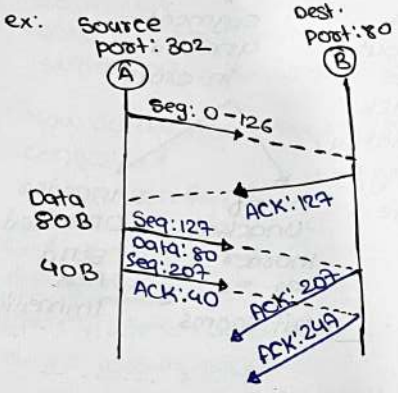
ex: A client sent seq. no. 1400 & payload included in the segment is 1399 B long.

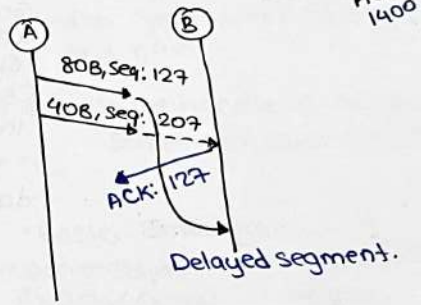


range from:
1400 - 2798



ex:



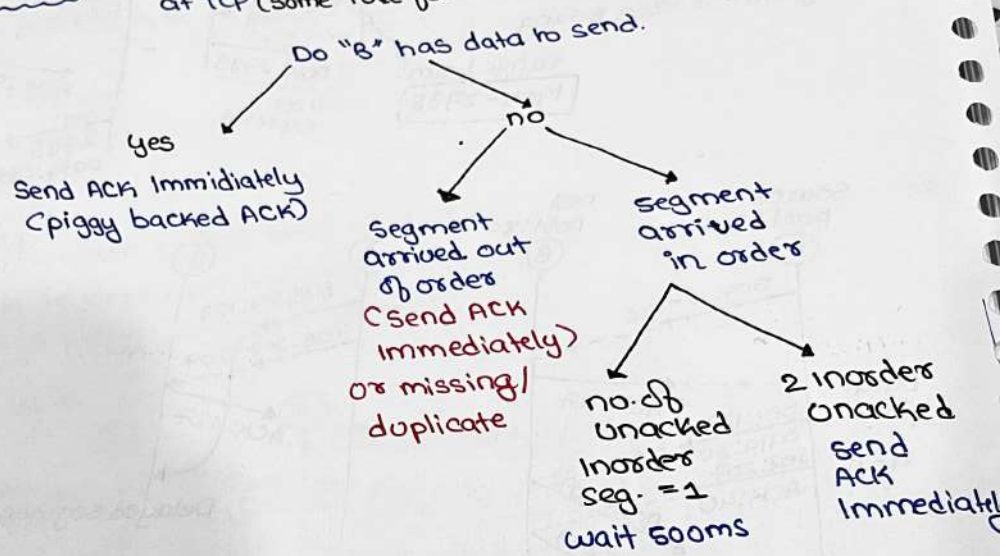


- the size of TCP rwnd can change throughout the duration of the connection

rwnd can't exceed Buffer but can change as the data is processed

- it's not necessarily be true that if seq. no. is m then subsequent seq. no. will be m+1
  it should be m + data no. of Byte

Optional — Implementation of cummulative ACK at TCP (some rule for TCP)



TCP → we generally use cummulative ACK only but we can use Selective ACK if we want to.

- Tcp connection are Byte Strea not msg. Stream.

TCP Congestion Control

- TCP delivers a Reliable, in order, byte stream.
- Flow control
  adjusting the sending rate to keep from over whelming a slow receiver
- How do we know the n/w is congested?
  Based on ACK

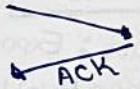
Congestion : the bottleneck is inside the n/w

- congestion control.
  adjusting the sending rate to keep from overloading the n/w.
- How do we handle of congestion?
  Sender will slow down.

Congestion detection

we can detect congestion by the loss of packet
(if there is a loss of packet then congestion)

- Basic structure
- upon receipt of ACK (of new data):
  increase rate
- upon detection of loss:
  decrease rate



Analogy

we are playing a game,

- I have one no. in my mind.
- I ask you to guess the no.

1 → no, I have greater no. in mind
2 → ------"------
3 →
:
64 → ..... lesser .....
40 → yes, these is the no.

- TCP flow control: receiver window (Rwnd) → is available in header itself
- TCP congestion control : Congestion window (Cwnd) → sender will maintain one more window
- Sender TCP window = min{Cwnd, Rwnd}
  - min: b/c we don't want to loss data
  - Cwnd: what nlw can handle
  - Rwnd: what reciever can handle

Congestion window (Cwnd)

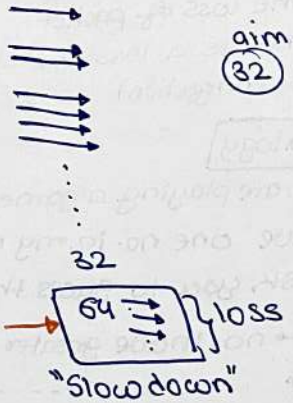
- how many byte can be sent w/o overflow routers.
- Computed by the sender using congestion control algorithm.

Flow Control window (Rwnd) (Advertised window.)

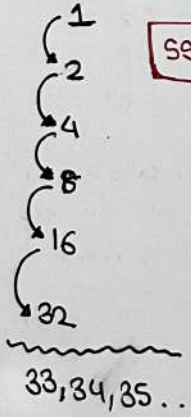
- how many byte can be sent w/o overflowing receiver buffer.
- Determined by the receiver & report to the sender.

Congestion Policies

SlowStart : Exponential Increase
Congestion Avoidance: Additive Increase

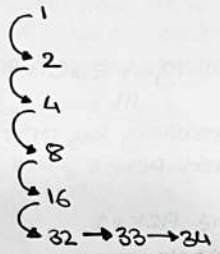


Keep on doubling sound very good but it's bound to have a loss at some point in time. we want to avoid this loss.

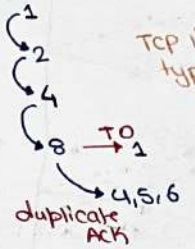


ssthreshold=32

initial ssthreshold : 32

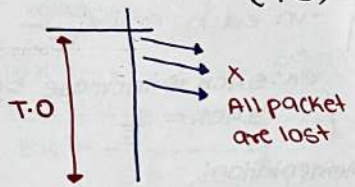


at 8, you incurr a packet loss

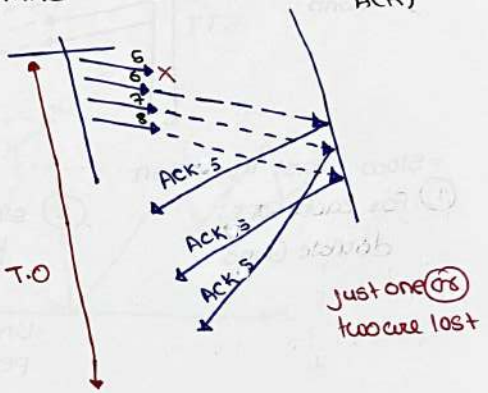


Tcp 1st identify type of loss

• Heavy Congestion (T.O)



• mild congestion. (3 duplicate Ack)



ssthreshold : 32

1 2 4 8 16 32 33 34 35.....

↑ normal case

1 2 4 8 16 1 2 4 8 4 5 6 1 2....

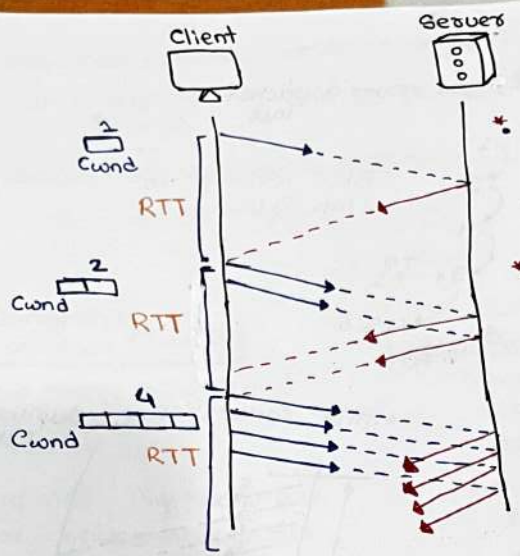
16 → Congestion (TO)

8 → Congestion (3 dup. Ack)

1 → T.O

Slow start phase

- Sender start at a slow rate
- Start with a small congestion window (Initially Cwnd = 1)
- Double the Cwnd for each RTT with no loss. till threshold.



* doubling at each RTT
  ||
  increasing by one at each ACK

* Cwnd ACK=1
  (at each RTT, we increase by one

- in each RTT ⇒ increase by 1
- in each ACK = $\frac{1}{Cwnd}$

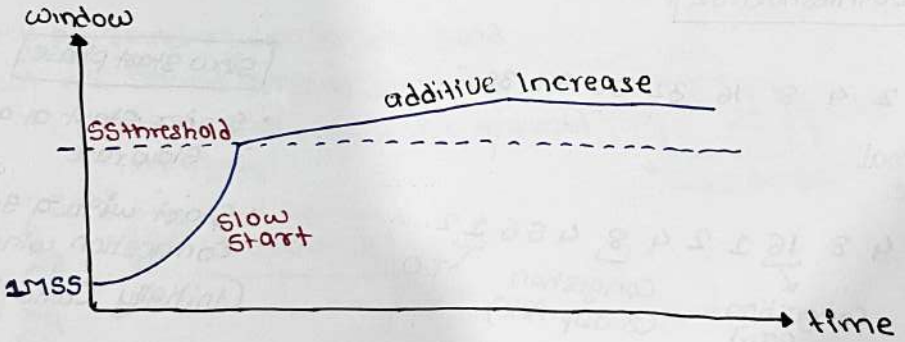
ex: 5ACK ⇒ increase by 1
1ACK = $\frac{1}{5}$

- Slow Start in Action

① for each RTT:
double Cwnd

② simple implementation:
for each ACK, Cwnd+=1

- Linear increase per ACK (Cwnd+1)
- exp- increase per RTT (2x Cwnd)

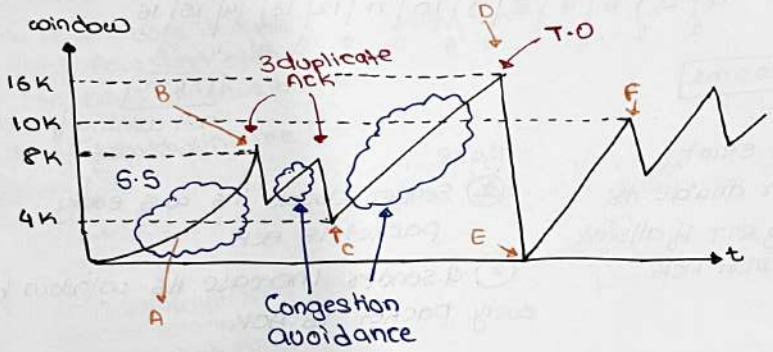


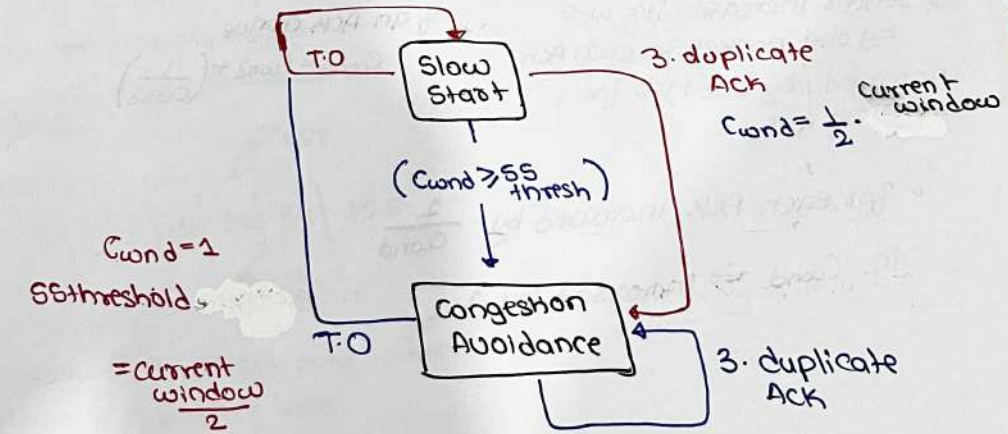
- Timeout (T.O) ACK
  (Heavy Congestion)
  ↳ if entire window was lost, buffer may be full
  ↳

- Triple duplicate ACK
  (Light congestion)
  ↳

- Not All looses the same
  - Duplicate ACK (Isolated loss)
    - Still getting ACK
    - Cwnd → Cwnd/2
      - half & then grow linears
  - Timeout (much more serious)
    - not enough dup. acks.
    - must have suffered several loss
    - Cwnd ← 1MSS & enter Slow Start mode
    - SSthreshold ← Cwnd/2

ex: 1 2 4 8 16 8 9 10 1 2 4 5 6 ....
(current Cwnd window)
16 → 3 duplicate ACK
10 → T.O





- Previous diagram of window & time.

Assume a
MSS of 1000Byte
RTT b/w sender & receiver : 100ms

- Assume at t=0 Sender attempt to open the Connection.

① RTT: 100ms MSS = 1000B

- time has progressed by point B?

Able to → 3 way question hand shake

1 | 2 | 4 | 8 400ms

1RTT + 3 RTT

- time has progressed b/w C & D?

Start with 4K means 4MSS.

4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16

1 2 3 4 5 6 7 8 9 10 11 12

1200ms

- at which round we are sending 11th segment. 12 RTT / trans.

no. of segment's we are sending

12 RTT

not asking from very beginning

- In slow start,

① a sender double it's w.s every RTT if all sent packet were ACK

false

~~②~~ Senders double it's w.s every packet is ACK

② a sender increase its window by 1 every packet is ACK

- Congestion Avoidance,

false

~~①~~ Sender increase it's w.s by one packet for each ACK

Increased by one MSS for every RTT

If an ACK arrive

$Cwnd = Cwnd + \left(\frac{1}{Cwnd}\right)$

- for each ACK Increased by $\frac{1}{Cwnd}$

in Cwnd ⇒ Increased by 1

ex: congestion window: 64KB when T.O occur
Connection is 50ms
MSS: 2KB
time taken to get back to 64KB.

Cwnd: 64KB ⇒ 32MSS

Cwnd: 1, SSthresh: 16MSS

1 | 2 | 4 | 8 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32

20 RTT

we will get Cwnd as 32.

ex: oneway latency: 50ms
to transfer 10MB file
Rwnd: 16MB] doesn't matter
TCP send 2KB packet. ≡ MSS = 2^{10}

(a) no. of RTT does it take until slow start open the send window to 1MB?

1KB | 2KB | 2^2KB | 2^3KB | 2^{10}KB
10 RTT

(b) no. of RTT to send the file?
10MB file.

1KB → 2KB → 2^2KB → 2^3KB → ---- → 2^{10} 2^{11} 2^{12} 2^{13} | 2^{14}

14 RTT

2^{10} - 1 MSS is send

Done

Major Misconception.

If SSthresh. is not given then many GATE aspirants do $SSthreshold = \frac{Rwnd}{2}$

wrong X

10MB = 10×2^{10} KB & MSS = 1KB
⇒ 10×2^{10} MSS
⇒ 2^{14} MSS

RTT
1 | 2 | 4 | 8 | 16 | 32 | 64 | 128

6th RTT

63 MSS

- we set SSthreshold only when there is a packet loss.

"Initial SSthreshold" it must be give otherwise we will not assume anything

initial ssthreshold

↑

it should be either given or we don't assume anything

ssthresh. after loss

⇓

ssthresh. = cwnd/2

ex: RTT: 10msec
& no congestion
Rwnd: 24KB
MSS: 2KB
How long does it take before it full window can be sent.

RTT: 10ms rwnd: 12MSS

1 | 2 | 4 | 8 | 12

4RTT

& 4x10 msec

⇒ 40 msec.

ex: file need to be send 250,000 Byte ⇒ 250MSS

MSS: 1MSS

TCP need to 1st establish 3 way hand shake.

1 | 2 | 2^2 | 2^3 | 2^4 | 2^5 | 2^6 | 2^7 | 2^8-1

⇒ 255MSS

1RTT for connect establish + 8RTT to send the Data

⇒ 9RTT

if ssthreshold not given that don't limit yourself.

gate IT 2004

① Max. transmit window size: 12000B ⇒ MSS = 6MSS

each packet consist of 2000B ⇒ 2000B MSS=

At some point in Slow start phase current window: 4000Byte. (no packet loss or T.O) max. value current window?

current MSS = 2MSS

in each ACK in Slow Start = +1 MSS

2MSS → 4MSS (+2) ← Based of ACK

⇒ 8000B

• More question on TCP Congestion control

① T.O occurs when Cwnd is 32KB } 16MSS
MSS: 2KB
Rwnd: 64KB } 32MSS
no. of RTT to return to 32KB } 16MSS

w.s = min{16MSS, 32MSS}
= 16MSS
SSthress = $\frac{Cwnd}{2} = \frac{16}{2} = 8$MSS

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

after 11RTT Cwnd will be (16).

Optional

② MSS: 1000B]
Cwnd: 1000B] 1MSS
SSthresh: 4000] 4MSS
alldata upto 16000 have been Acked
last Ack was 16001

1 2 4 5 6 7 8 ..... 15 16

16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 .....

ACK for MSS	16th	17th	18th	19th	20th	21st	22	23	24	24	24	25
Cwnd:	2000	3000	4000	4250	4500	4750	5000	5200	5400	5400	-11-	1000
Packet Sent:	17,18	19,20	21,22	23	24	25	26,27	28	29	X	X	25

③ given that
④ at t=0 we are at congestion avoidance mode
& Cwnd = 6MSS
no. of RTT to reach 12MSS.

6MSS | 7 | 8 | 9 | 10 | 11 | 12 (6RTT)

throughput.

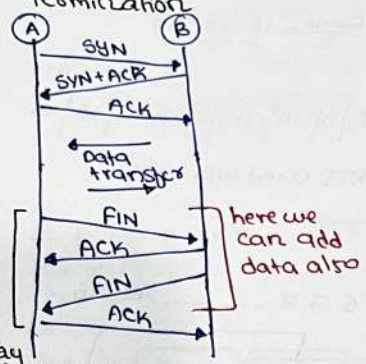
$\frac{6+7}{2RTT} = \frac{13}{2} = 6.5$MSS/RTT

⑤ avg. throughput from t=0 to 6.

So,
avg. throughput $\Rightarrow \frac{6+7+8+9+10+11}{6RTT} = \frac{51}{6}$

8.5MSS/RTT

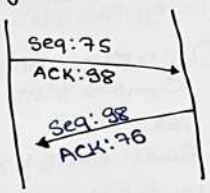
- TCP Connection termination



4 way connection termination we can convert it into 3 way

Ex

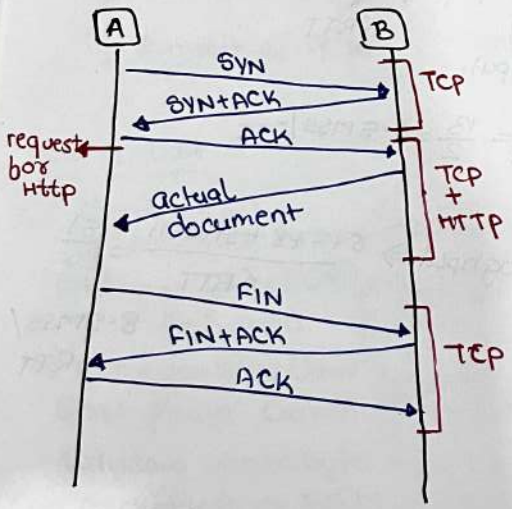
if no data then



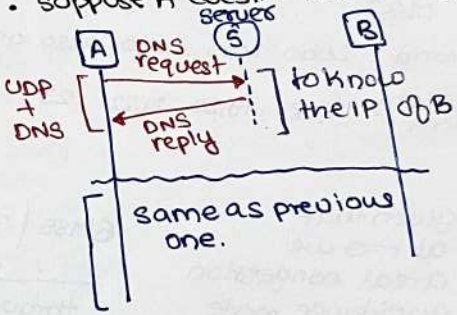
(1) "A" want to retrieve an HTML document from B

& smaller than MSS size so can be sent in one packet

assume A knows IP of B.



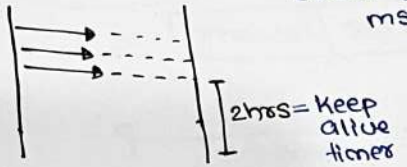
- suppose A doesn't know IP of B



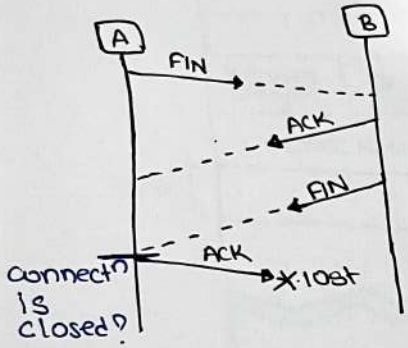
A B

DNS + TCP Connection establish + HTTP data transfer over HTTP + TCP Connectn terminatn

- Keep alive timer

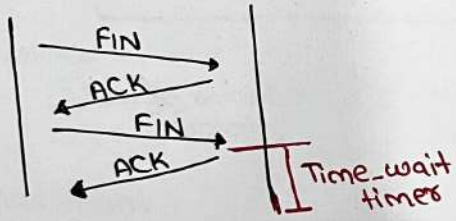


- If server doesn't get to hear anything from client for keep alive time then it send a probe msg. & close the connection

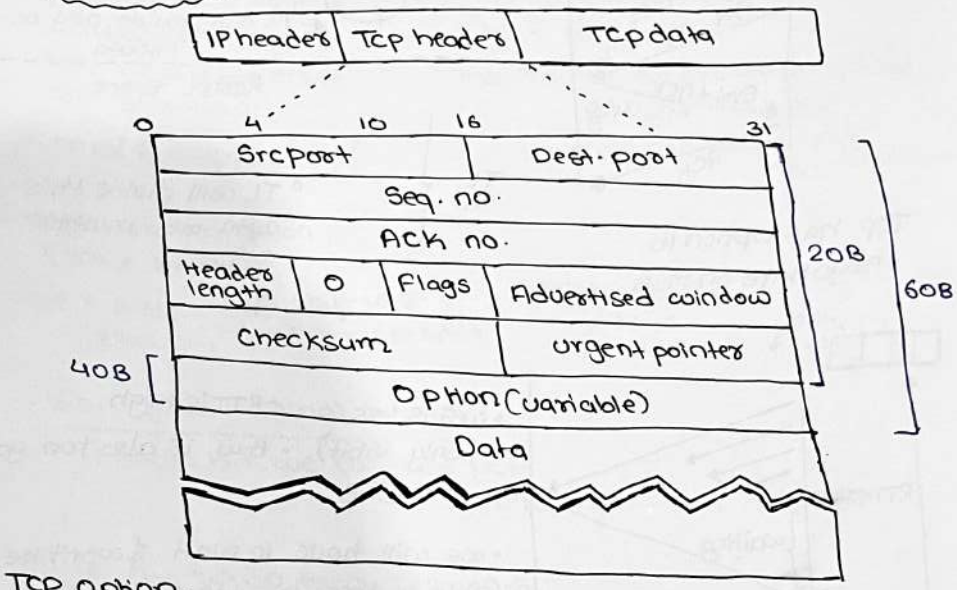


Suppose after sending last ACK "A" declares connection is closed but last ACK is lost then B will send FIN Again b/c of T.O

Solution:
A should declare connection is closed but after some time
(if ACK is lost then let B send FIN so that A can send ACK again).
resend of final ACK

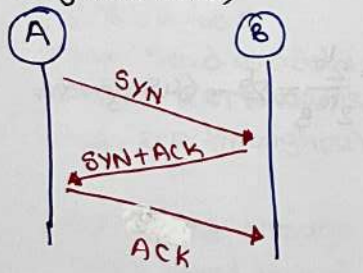


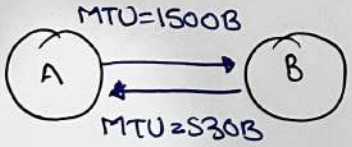
TCP Options



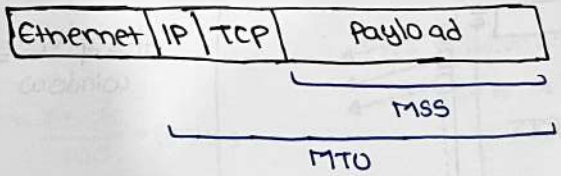
- TCP option
  1. MSS
  2. Large window option
  3. Timestamp option
  4. Selective ACK (SACK)

1. MSS (Maximum Segment Size)





- Relationship b/w MSS & MTU



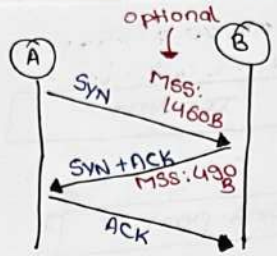
$$MSS = MTU - (\text{IP header}) - (\text{TCP header})$$

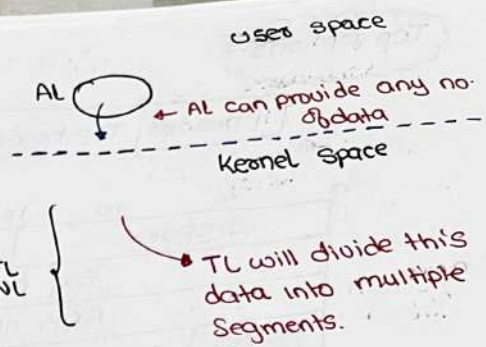
ex. MTU: 530B
20B IP
20B TCP

$$MSS = 530 - 40 = 490B$$

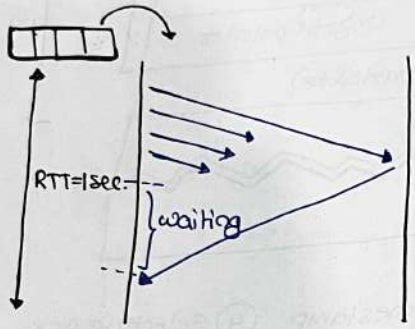
↓

is announced at setup by both ends



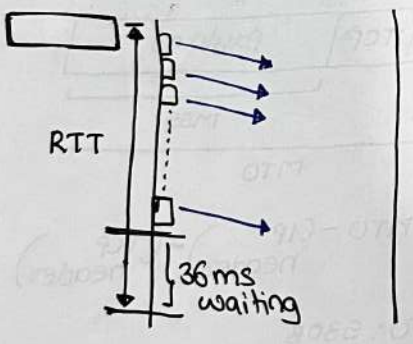


TCP has option to negotiate on MSS



- w.s is less (only 16bit) $= 2^{16}$ B
- RTT is high
- B.w is also too good
- we will have to wait & won't be able to keep on transmitting for complete RTT time

Soln : Increase window size



$$\#\text{seq. in window} = \frac{2^{16}}{2^{10}B} = 2^6 = 64 \text{ segment}$$

2^{16} Byte = w.s

assume:

MSS $= 2^{10}B = 1024B$

$T_t = 1ms$

RTT = 100ms

(1) w.s field: 5000B
& window scale option is used with 3bit then what is the actual size of window?

$5000 \times 2^3 B \Rightarrow 40{,}000B$

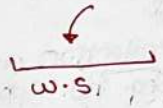
(2) In TCP, what can be size of of advertised w.s using the option field.

$2^{16}-1 \Rightarrow 65535 \times 2^{14}$

max. no. that we can write into the window field.

- we can scale are w.s to 1GB (fixed standard)

- Large window option (scale window)
- use a scale factor
- left shift w.s field by upto 14bit
- window of upto 2^{30} Byte



16 bit from original header
+14 more bit we can use from options.

- if we use option then TCP · w.s

$\Rightarrow 2^{16+14} = 2^{30}$

16 → Original header
14 → options.

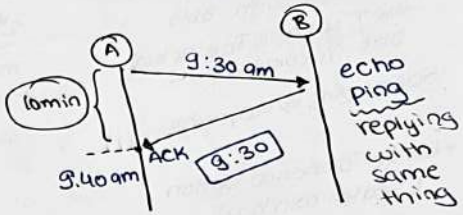
(ex) 16 bit w.s
Max. Send window = 65535B
Suppose a RTT: 100ms
Max. TCP throughput = $\frac{65KB}{100}$ = 5Mbps

- not good enough for modern high speed links!

(ex) TCP connection b/w A & B parameter limits the no. of Byte that a segment from Host A can carry as payload.

MSS advertised by Host A & Host B.

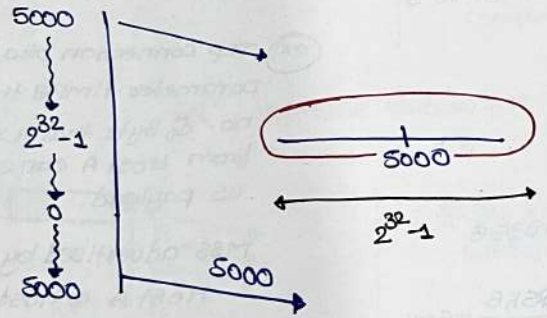
- Timestamp option — Dual purpose: (1) calculate RTT (2) protect against wraparound
- timestamp added to segment by the sender
- Echoed by the received
- Sender can then compute RTT
- Also can be combined with seq no.



- Protection against wrapped Seq. no. (PAWS) — Provided by timestamp options

Seq. no. Space

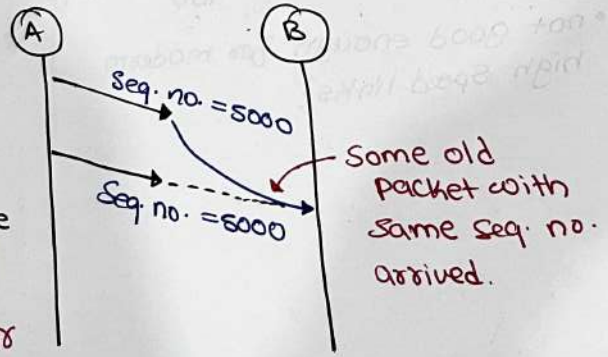
↳ 32 bit seq. no.



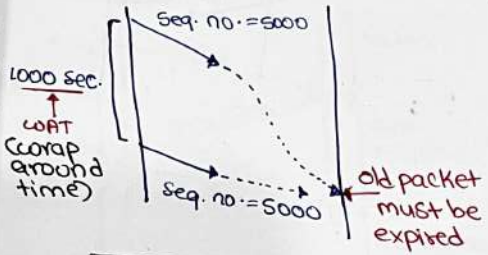
- How to protect from this particular scenerion

Solution: assume Max. seg. life (MSL) = 120 sec

↳ If packet is the n/w for more than MSL time then we directly discard that.

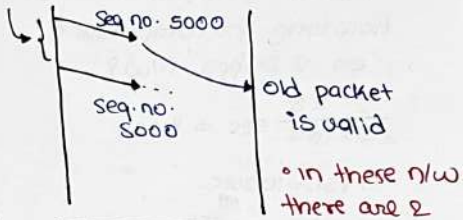


MSL = 120sec



WAT > MSL

WAT = 10 sec.



- in these n/w there are 2 valid packet with same seq. no.

WAT < MSL
10sec. 120sec.
creating problem

- Seq. no. = 32bit

# of different seq. no. = 2^{32}

2^{32} Byte we can send w/o reusing any seq. no.

B.w ⇒ 2^{30} bps

2^{32} Byte

B.w = 2^{30} bps (≈ 1Gbps)

after how long you have reuse seq. no.?

Sending all 2^{32} Byte = $\frac{2^{32} \times 8}{2^{30}}$ sec = 32sec.

After 32sec. we have to repeat seq. no.

- At some point in time, we have to repeat seq. no.

we want to repeat only after MSL

- protecting against wraparound

(Relevance of the 32-bit seq. no. space)

① Seq. no. may wraparound

② A Byte with sequence x could be sent then later time a 2nd byte with seq. x could be sent

③ packet can't survive in the Internet for longer than the MSL = 120s.

④ Seq. no. must not wrap around within 1 MSL.

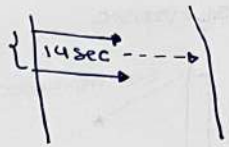
⑤ we need to make sure that seq. no. doesn't wraparound within a 120sec. period of time

⑥ Depends on how fast data can be transmitted over the Internet

ex: 32bit seq. no. $\Rightarrow 2^{32}$ Byte

How long for wraparound on 2.5Gbps n/w?

$$\frac{2^{32} \times 8}{2.5 \times 10^9} \text{ sec} = 14\text{sec}$$

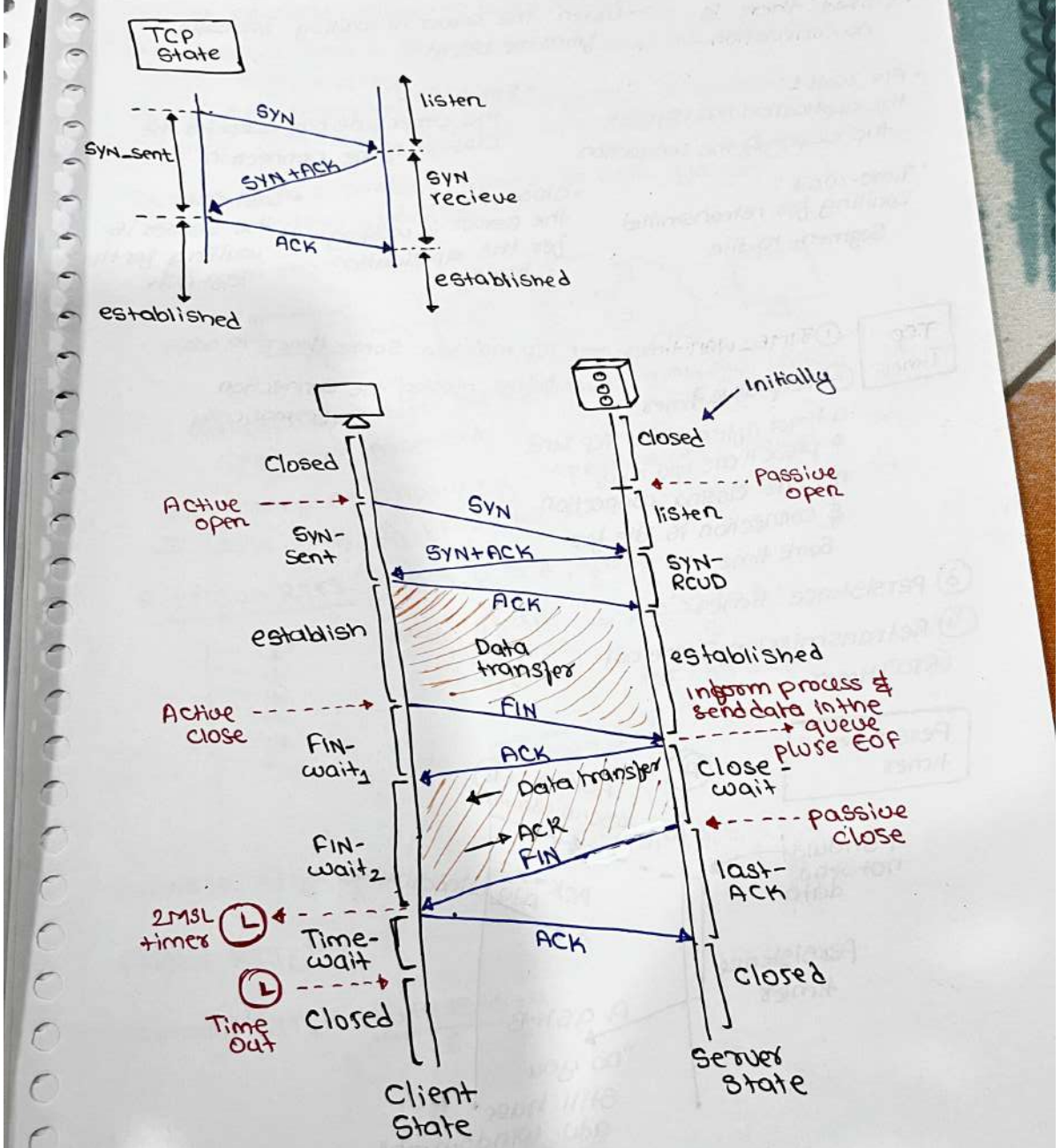


MSL = 100sec.

(can also use 2^{30} instead of 10^9)

Tcp extensions

- Selective ACK (SACK)
  - Optional header field used to ACK additional header
  - Sender can then resubmit only missing block.


TCP State
SYN
SYN_sent
SYN+ACK
ACK
listen
SYN recieve
established
established
Initially
Closed
Passive open
Active open
SYN
listen
SYN-Sent
SYN+ACK
SYN-RCVD
ACK
establish
Data transfer
established
Inform process & send data in the queue
Active close
FIN
ACK
FIN-wait1
Data transfer
Close-wait
Pluse EOF
passive close
ACK
FIN
FIN-wait2
last-ACK
2MSL timer
Time-wait
ACK
Closed
Time Out
Closed
Client State
Server State

- Closed: there is no connection
- Listen: the server is waiting for calls from the client
- FIN-wait 1: the application has request the closing of the connection
- FIN-wait 2 the other side has accepted the closing of the connection
- Time-wait: waiting for retransmitted Segment to die
- Close-wait: the server is waiting for the application to close
- Last-ACK the server is waiting for the last ACK.

TCP Timers

① TIME_WAIT timer → TCP maintain some timer to wait before closing the connection permanently

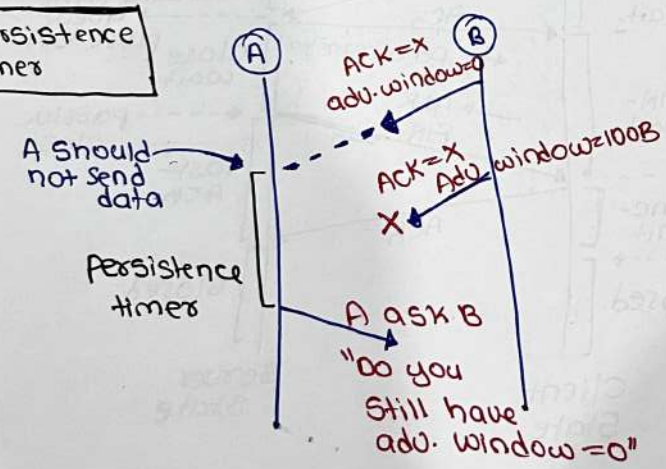
② Keep-alive timer

a timer after which TCP send a probe "are you alive"? no one closing connection & connection is idle for some time

③ Persistence timer

④ Retransmission Time out (RTO) timer

Persistence timer



- Receiver advertise window = 0 to stop transferring
- later, Receiver send a segment with window advertisment, but a segment is lost
- it no any mechanism, transferring is stopped & Sender will think receiver still has window size of 0. (Deadlock)
- Sender set a persist timer to ask for a new window updated.

we can measure RTT

Here, whatever we measure, may go invalid in very nxt moment.

RTT keep on changing in transport layer.
⇒ take avg. of previous RTT to guess next RTT

Previous RTT's : 5, 3, 4, 5, 6, 4, 5, 8

Avg. of all previous RTT's $= \frac{5+3+4+5+6}{5}$

$\Rightarrow \frac{23}{5} = 4.6$

T.O = 2RTT

: 5, 3, 4, 5, 6, 4, 5, 8

4.6 4.5 4.57

Estimated $RTT_6 = \frac{23}{5} = 4.6$

Actual $RTT_6 = 4$

Estimated $RTT_7 = \frac{5+3+4+5+6+4}{6}$

$\Rightarrow 4.5$

Estimated $RTT_8 = \frac{5+3+4+5+6+4+5}{7}$

$\Rightarrow 4.57$

Estimate RTT_6: $\frac{5+3+4+5+6}{5}$

Estimate RTT_7: $\frac{(5+3+4+5+6) + \text{Actual } RTT_6}{6}$

$$\frac{5 \cdot \text{Estimate} + \text{Actual } RTT_6}{6}$$

Estimate $RTT_7 = \frac{5}{6}$ Estimate $RTT_6 + \frac{1}{6}$ Actual RTT_6

Estimate RTT_n: $\frac{(n-2)}{(n-1)}(\text{Estimate } RTT_{n-1}) + \frac{1}{(n-1)}$ Actual RTT_{n-1}

↓ these is nothing but avg. of all prev. Actual RTT's

Estimate $RTT_n = (1-\alpha)(\text{estimate } RTT_{n-1}) + \alpha\,(\text{Actual } RTT_{n-1})$

$$0 \leq \alpha \leq 1$$

Ques. How long should Sender wait?

Sender set a timeout to wait for an ACK.

- Too short
  wasted retransmission
- Too long
  excessive delay when packet lost

Ques. how to set TCP time out value

- too short
  premature timeout,
  unnecessary
  retransmission.
- Too long
  slow reaction to
  segment loss.

① A timeout much smaller than the estimated RTT will lead to unnecessary retransmissions.

② A timeout that is roughly equal to the RTT give the senders enough time to receive ACK's from the receivers

③ A timeout much larger than the estimated RTT may result in poor n/w utilization

④

⑤

Basic Algorithm

$$\text{Estimated RTT}_{new} = (1-\alpha)\,\text{Estimated RTT}_{old} + \alpha\,\text{Sample RTT}$$

$$\text{RTO} = 2 * \text{Estimated RTT}$$

- How All Layer works together.