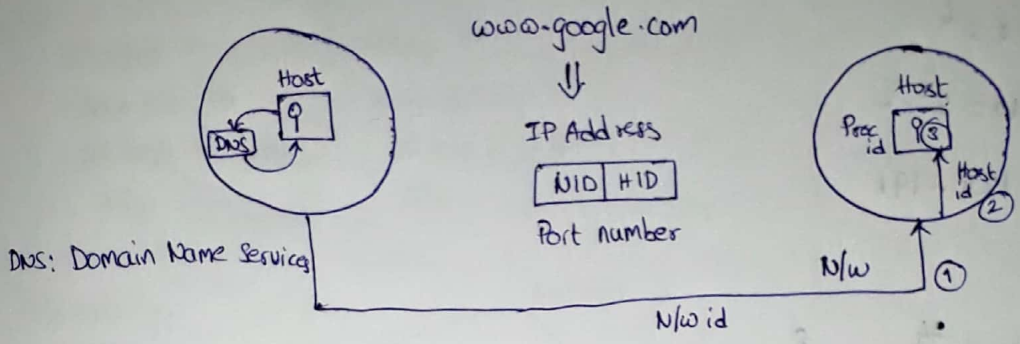
02/12/20

Computer Networks

Introduction:



→ Port number is used to identify a process in host. Well known services have predefined fixed port numbers.

→ The overhead involved in translating domain name to IP address using DNS is called DNS overhead.

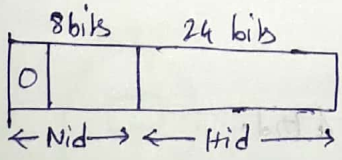
Classfull IP address classification:

Class A: 0___ ; Class B: 10___ ; class C: 110___ ; Class D: 1110___ ; class E: 1111___

→ Size of IPV4 address is 32 bits.

→ It is represented using dotted decimal notation where every 8 bit part of the address is represented in decimal form.

Class A:



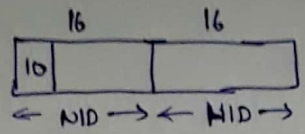
→ Nof networks = 128

→ no of IP add per n/w = 2^{24}

→ no of hosts per N/w = $2^{24} - 2$

→ Out of 128 N/ws we don't use starting Nid (all 0's) and last N/w (all 1's) ⇒ Range: 1.___.___.___ to 126.___.___.___

Class B :



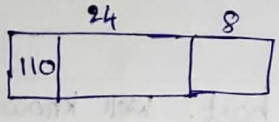
→ N/Ws : 2^{14}

→ IP/Nw = 2^{16}

→ Hosts/Nw = $2^{16}-2$

→ Range: 128 - 191

Class C :



→ NWs : 2^{21}

→ IP/Nw : 2^{8}

→ Hosts/Nw : $2^{8}-2$

→ Range : 192 - 223

Class D :

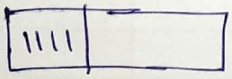
→ Here addresses are not classified into Nid and Hid.

→ This class of addresses are used for multicasting.

→ No of address : 2^{28}

→ This class is a drawback because 2^{28} is a large number for multicasting. Range : 224 - 239

Class E :



→ These are also not divided into Nid & Hid.

→ These are reserved for special purpose.

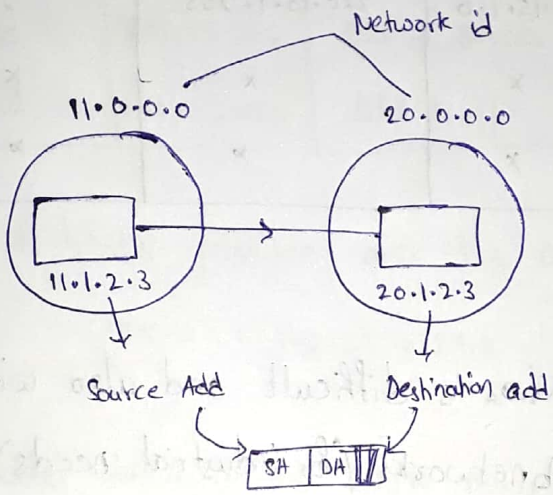
→ Range : 240 - 255.

Note : Under every Nlw, 1st & last IP addresses are used for special purposes (i.e., for network id & ~~multicasting~~ Direct broadcast)

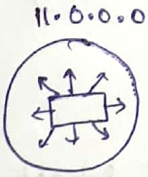
Casting:

- Unicast: Sending packet from one host to other host
- Broadcast: sending a packet from one host to many hosts.
  - Limited
    - send pckt to all hosts of same N/w
  - Directed
    - send pckt to all host of diff N/w

Unicasting:



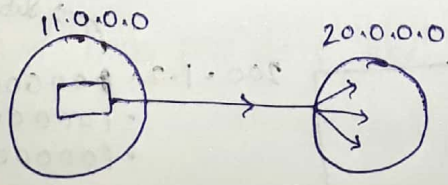
Limited Broadcasting:



Source Add: 11.1.2.3

Destination Add: 255.255.255.255

Directed Broadcast:



Source Add: 11.1.2.3

Destination Add: 20.255.255.255

Note:

limited broadcast add: 255.255.255.255

Directed broadcast add: Nid.1111....1

Network id: Nid.000...0

(81)

IP Add	Class	Nid	DBA	LBA
1.2.3.4	A	1.0.0.0	1.255.255.255	255.255.255.255
10.15.20.60	A	10.0.0.0	10.255.255.255	"
130.1.2.3	B	130.1.0.0	130.1.255.255	"
150.0.150.150	B	150.0.0.0	150.0.255.255	"
200.1.10.100	C	200.1.10.0	200.1.10.255	"
220.15.1.10	C	220.15.1.0	220.15.1.255	"
250.0.1.2	E	x	x	x
330.1.2.3	invalid	x	x	x

Subnets:

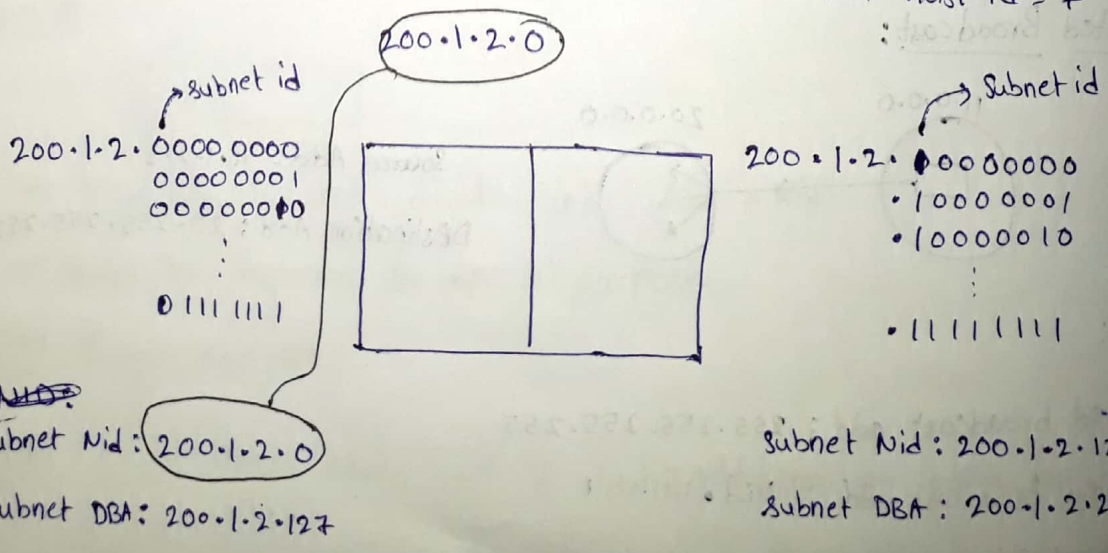
→ Maintainence of large Nlws is difficult and also within a Nlw we need small subnetworks (for industrial needs).

→ Subnetting is ~~one~~ process of dividing large Nlw into smaller networks.

→ But subnetting ~~to~~ incurs more time in identifying host.

Eg: Consider class c Nlw 200.1.2.0

Now lets divide it into 2 subnets ⇒ no of bits in subnet id = 1
no of bits in host id = 7



→ Now there is a confusion that whether

200.1.2.0 is Nid or subnet Nid

200.1.2.255 is DBA of subnet or DBA of whole N/w.

→

Sender	Address	Interpretion
within N/w	200.1.2.0	subnet N/w id
ouside	200.1.2.0	N/w id
inside	200.1.2.255	DBA of subnet
outsid	200.1.2.255	DBA of N/w

→ Now no of hosts possible over this organization is

$(128-2)+(128-2) = 252$

→ Now consider dividing the same network into 4 subnets.

⇒ no of bits in subnet id : 2

no of bits in host id : 6

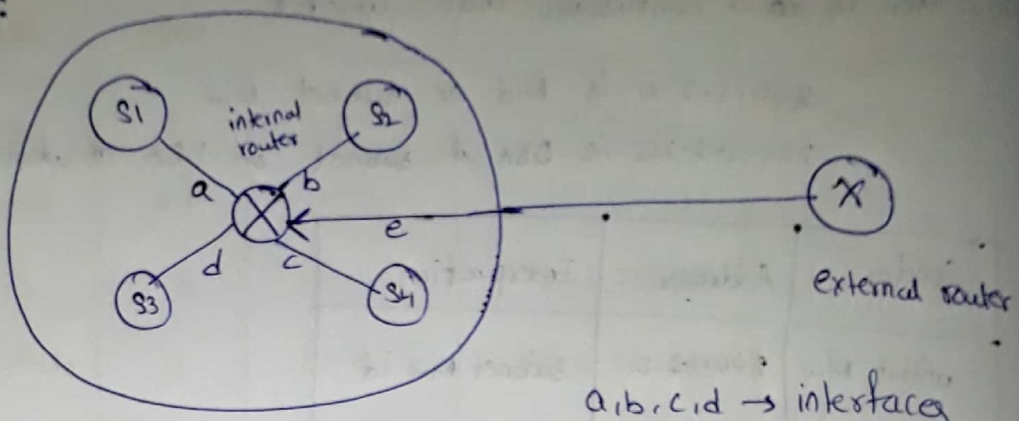
S1	S2
200.1.2.0 to 200.1.2.63	200.1.2.64 to 200.1.2.127
200.1.2.128 to 200.1.2.191	200.1.2.192 to 200.1.2.255
S3	S4

no of hosts possible : 256 - (2*no of subnets)

= 256 - (8) = 248

→ Generally router gets the packets and forwards to the correct host and makes correct interpretation (in the case when there is confusion)

Routing:



a, b, c, d → interfaces

Subnet mask: It is a 32-bit number used by router to identify the subnet to which a given address belongs. to. within subnet mask,

all bits corresponding to Nid & subnet id are 1's
" " " " host id are 0's

Eg: For current example subnet mask is

Subnet mask is also known as network mask

11111111	11111111	11111111	11000000

≅ 255.255.255.192

we obtain subnet network id by performing bitwise and with given i/p address.

For example consider ip add 200.1.2.130

Subnet mask: 11111111 11111111 11111111 11000000
ip add : 11001000 00000001 00000010 10000010

bitwise and : 11001000 00000001 00000010 10000000

i.e., 200.1.2.128 (subnet id)

→ whenever router gets an address, it performs bit wise and b/w ip add. & subnet mask.
After this it finds subnet network id and forwards the packet to the correct interface accordingly. This process is done using routing table.

Routing table:

Subnet id	Subnet mask	Interface
200.1.2.0	255.255.255.192	a
200.1.2.64	"	b
200.1.2.128	"	c
200.1.2.192	"	d
0.0.0.0	0.0.0.0	e

→ when ever i/p add is obtained, it is anded with subnet mask and check if it matches with corresponding subnet id or not.

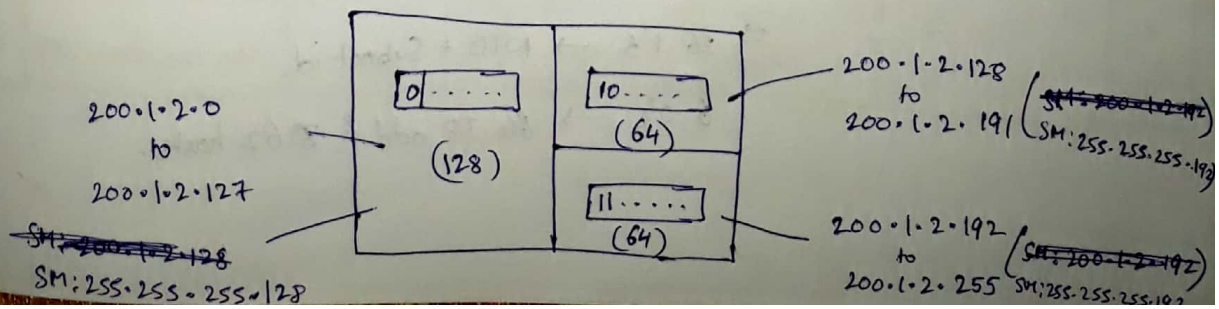
→ If all the entries doesn't match, then default entry (0.0.0.0) is used.

→ If an address match with more than one pair then choose the pair with longest subnet mask (i.e., the one with more no of 1's)

→ when sizes of all the subnets are same, it is called Fixed Length Subnet Masking (FLSM):

→ when sizes of subnets differ, it is called Variable Length Subnet Masking (VLSM)

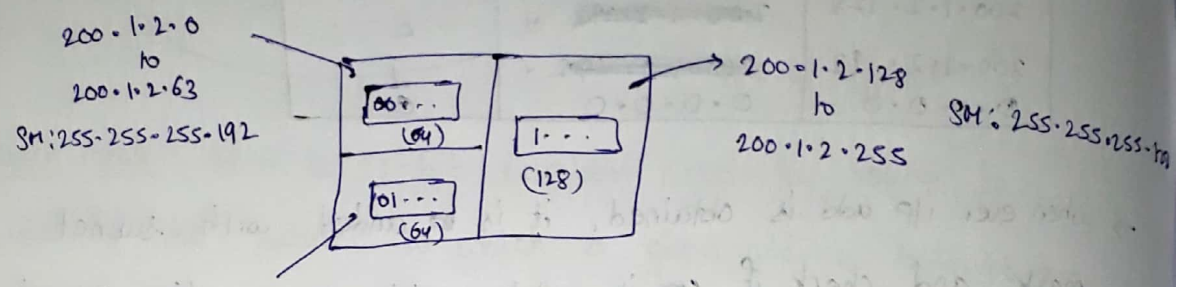
→ Now consider subnetting of previous example into 3 subnets with 128, 64, 64 IP addresses.



→ Subnets of equal sizes will have same value of ~~sm~~ subnet mask.

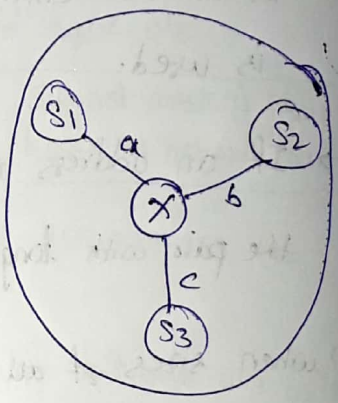
→ ~~Smaller~~ Larger the size of subnet, smaller the size of subnet mask is.

→ Another variant can also be used ~~where larger subnets get larger value of subnet i.e. as shown below~~



Routing table:

Subnet N/w id	SM	interface
200.1.2.0	255.255.255.128	a
200.1.2.128	255.255.255.192	b
200.1.2.192	255.255.255.192	c
0.0.0.0	0.0.0.0	d



03/12/20

→ Given a subnet mask we can find no of IP add. in the the subnet.

Eg: Consider 255.255.255.192

11111111.11111111.11111111.11000000

⇒ 26 1's → NID + Subnet id

6 0's → 64 IP add & 62 hosts.

If class of IP add is given then we can find ↑ subnet id also

Class A:

NID: 8 bits Sid: 18

Class B:

NID: 16 Sid: 10

Class : NID: 24 Sid: 2

→ Practically MSB bits of HID are only used for subnet id. But theoritally any bits of HID can be used.

→ If LSB bits are used for subnet id then we can't get proper ranges

Eg: Consider dividing 200.1.2.0 into 2 subnets. If LSB ↑ bit is used as subnet id then one subnet will have all even IP addresses and other will have all odd IP addresses.

(Q2)

	Subnet Mask	No of hosts Per subnet	Subnets in class A	Subnets in class B	Subnets in class C	Subnet if no of bits in HID = 10
1	255.0.0.0	$2^{24}-2$	1	–	–	–
2	255.128.0.0	$2^{23}-2$	2	–	–	–
3	255.192.0.0	$2^{22}-2$	2^2	–	–	1
4	255.240.0.0	$2^{20}-2$	2^4	–	–	2^2
5	255.255.0.0 (Default subnet mask of class B)	$2^{16}-2$	2^8	1	–	2^6
6	255.255.254.0	2^9-2	2^{15}	2^7	–	2^{13}
7	255.255.255.0 (Default subnet M of class C)	2^8-2	2^{16}	2^8	1	2^{14}
8	255.255.255.224	2^5-2	2^{19}	2^{11}	2^3	2^{17}
9	225.255.255.240	2^4-2	2^{20}	2^{12}	2^4	2^{18}

1) 255.0.0.0

Nid + Sid : 8 bits

Hid : 24 bits ⇒ no of hosts = $2^{24}-2$

Class A : Nid + Sid = 8

8 + Sid = 8

Sid = 0

i.e., no subnets.

Entire network is one subnet

Class B :

Nid + Sid = 8

Nid of Class B : 16 bits

∴ not possible

Class C :

here also not possible

2) 255.128.0.0

11111111 . 10000000 . 0 . 0 . 0 . 0

⇒ Nid + Sid : 9 bits

Class A : $2^{23}-2$ ⇒ 15 bits for sid

Class B : not possible

Class C : not possible

CIDR : (Classless Inter Domain Routing)

→ The problem with classful classification is that it not flexible for assigning IP addresses.

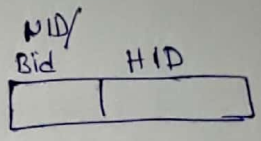
→ So we use CIDR technique which doesn't have any classes.

→ whenever a set of IP addresses are needed, the available IP address set is assigned. This set of IP addresses is called block.

→ In classful, given an IP add we can determine ~~no of b~~ network id to which the address belongs to (Rnd in class of address)

But in ~~CD~~ CIDR we need to mention no of bits in ~~i~~ NID field explicitly.

CIDR representation: a.b.c.d/n



n → no of bits in N/w id part

32 − n → no of bits in host id

no of IP addresses = 2^{32-n}

Rules for CIDR blocks:

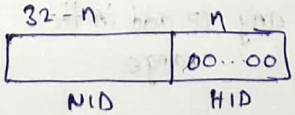
(i) All the IP addresses should be contiguous.

(ii) Block size is always in power of 2.

(iii) First IP address in the block should be divisible by size of the block.

i.e., ~~If~~ starting address should have all 0's in HID.

For example if block size is 2^n then least significant n bits should be all 0's.



This lets IP addresses of block range from NID.00...0 to NID.111...1 } 2^n

Eg: Check whether below set of IP addresses form a valid CIDR block or not.

100.1.2.32
100.1.2.33
⋮
100.1.2.47

All the addresses are contiguous

No of size of block = 16 = 2^4

starting address: 100.1.2.00100000

least significant 4 bits are 0's.

∴ All 3 rules are satisfied

∴ It is a valid CIDR block.

Eg: 20.10.30.32
20.10.30.33
⋮
20.10.30.63

Contiguous
size = 2^5 = 32
starting host id: 00100000

∴ valid CIDR block.

Eg: 150.10.20.64
150.10.20.65
⋮
150.10.20.127

Contiguous
size: 64: 2^6

01000000
6 bits

∴ valid CIDR block

This represented as 150.10.20.70/26

This can be any IP add in the range

no of bits in NID. Using this Network id can be found

→ Here also, the first address of block is used to represent block id or network id.

The last address of the block is used as directed broadcast address.

Eg: Find the CIDR block of addresses using below CIDR representation.

20.10.30.35/27

Sol:

no of bits in NID = 27

no of bits in host id : 5 ⇒ LSB 5 bits must be 0 for starting add

no of IP add : $2^5 = 32$

20.10.30.00100000

= 20.10.30.32 is that starting address

∴ The range of addresses is 20.10.30.32
20.10.30.33
⋮
20.10.30.63

Eg: 100.1.2.35/28

100.1.2.00100011

Range of add : 100.1.2.00100000 ≅ 100.1.2.32
⋮ ⋮
100.1.2.00101111 100.1.2.47

Eg: 100.1.2.35/20

Ans: 100.1.0.0 to 100.1.15.255

Subnetting in CIDR:

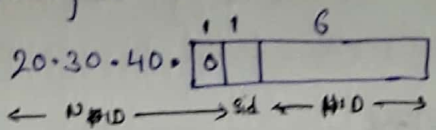
→ Consider forming 2 subnets from block 20.30.40.10/25

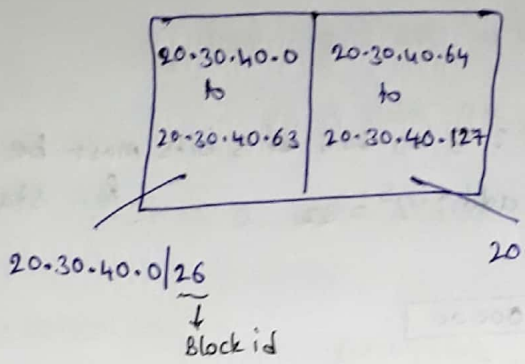
⇒ no of bits in host id = 7

20.30.40.00001010

⇒ N/w id: 20.30.40.0000 0000
Host id

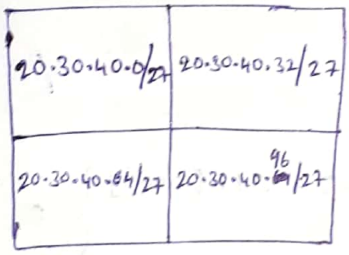
Now by subnetting we have

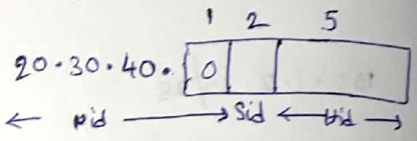




In CIDR representation ~~the~~ ~~redacted~~ no bits in block id of subnet are represented. Also these bits represent ~~subnet~~ subnet mask also.

Similarly we can divide it into 4 subnets



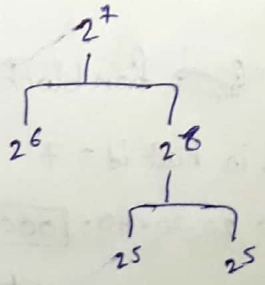


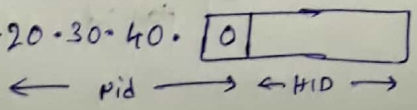
VLSM in CIDR Blocks

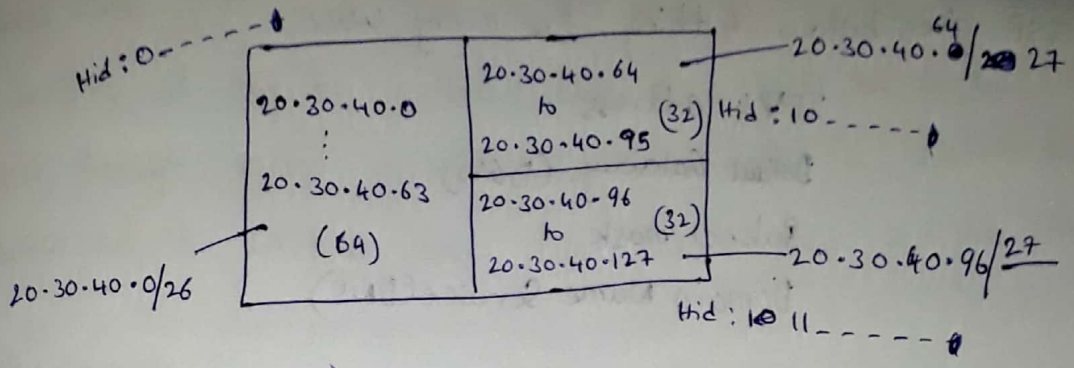
→ Consider 20.30.40.10/25

⇒ no of hosts: 2^7

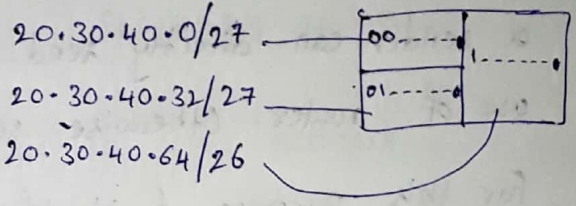
Consider following division



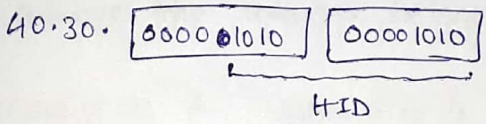




Another way of subnetting is



Eg: Consider 40.30.10.10/20

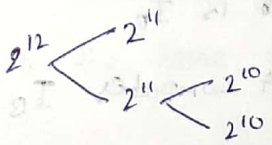


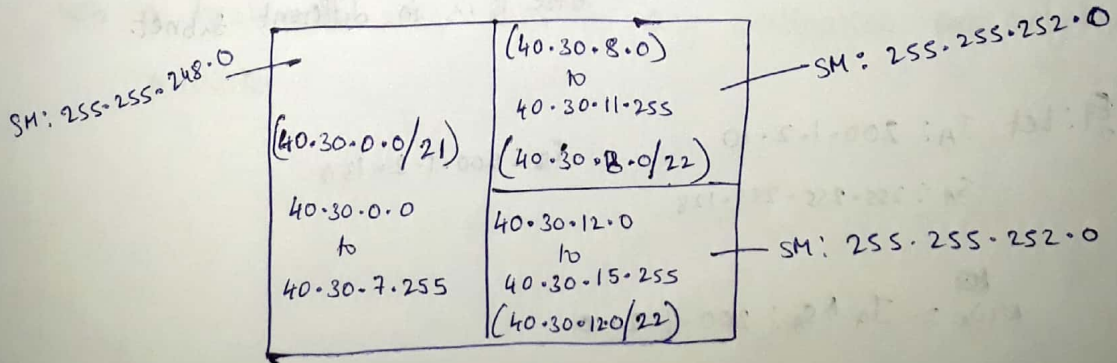
∴ NID: 40.30.0.0

no of IP add: 2^{12}

Range of add: 40.30.0.0 to 40.30.15.255

Consider subnetting this into 3 subnets as shown below





→ ISP gives below 4 thing for every user

IPV4 address
Default Gateway (DGW)
Subnet mask
Domain Name Service (DNS)

→ ~~Using subnet mask a~~ If destination is in same N/w then a sender can directly send packet to destination without use of router. Otherwise sender sends packet to router. For this purpose of identifying whether destination is within same ~~netwer~~ subnet or not, the host is provided subnet mask.

→ Consider

let IP add of A is I_A and subnet mask be S_A ~~and n/w id be~~ and N/w id be Nid_A

$$Nid_A = I_A \wedge S_A$$

let IP add of B is I_B

now A computes $I_B \wedge S_A$

if $Nid_A = I_B \wedge S_A$ then B is in same subnet
else B is in different subnet.

Eg: Let I_A: 200.1.2.10 I_B: 200.1.2.130

S_A: 255.255.255.128

$NID_A = I_A \wedge S_A$: 200.1.2.0

$I_B \wedge S_B$: 200.1.2.128

$NID_A \neq I_B \wedge S_B$

∴ B is in diff subnet. So A forwards pckt to DGW.

→ Subnet mask given to a host by ISP may or may not be correct. Sometimes it may be needed that a pckt has to be sent through router even if the destination is within the same subnet. In that case false subnet mask will be given.

Eg: Consider

I_A : 200.1.2.10 I_B : 200.1.2.69

S_A : 255.255.255.128 S_B : 255.255.255.192

Here A knows that A & B are in same network.

But for B, it gets an illusion that A is in different subnet

If we assume the address belong to class C

According to A, subnetting is 

According to B, subnetting is 

→ Thus using subnet mask, security is provided

i.e., Some host must send data through router even if the destination is in the same network.

→ When subnet mask is 255.255.255.255,

a host must send data to any destination only through a router

Supernetting or aggregation:

→ In a routing table,

no of entries = no of networks connected to the router

→ For large no of networks, size of routing table grows exponentially.

→ So we perform supernetting using which create an illusion to router that all the small networks belong to one network.

→ Supernetting doesn't really combine Nws. It is an illusion.

Rules for aggregation:

(i) Addresses in subnets must be contiguous.

i.e., contiguous blocks can only be aggregated

(ii) Size of blocks must be same (sum of all block sizes is power 2 provided given blocks are valid)

(iii) The first network id should be divisible by size of total aggregated block.

Eg: Consider supernetting of below 4 networks

200.1.0.0/24
200.1.1.0/24
200.1.2.0/24
200.1.3.0/24

Sol:

Addresses are contiguous

size of all blocks are same i.e., 2^8

size of total aggregated block = $2^8 \times 2^2 = 2^{10}$

∴ starting IP add 200.1.0.0 is divisible by 2^{10}

i.e., least significant 10 bits are 0's.

∴ supernetting is possible.

Now after supernetting the block is represented as

Supernet id : 200.1.0.0/22 ~~SM: 255.255.252.0~~

Now the router will have only one entry instead of 4 entries.

For this block we have supernet mask

Supernet Mask : ~~200.1~~ 255.255. 11111100 . 00000000

= 255.255.252.0

→ The internal router will have 4 entries
external router will have 1 entry

Eg: 200.1.32.0/24
200.1.33.0/24
⋮
200.1.47.0/24

(i) Contiguous

(ii) sizes are same. i.e., $2^8 \times 16 = 2^{12}$

(iii) ID of 1st network

200.1. 00100000 . 00000000

size of supernet: 2^{12}

Nid 1st network is divisible by 2^{12}

∴ Supernetting is possible.

⇒ Supernet id: 200.1.32.0/20 → $24 - \log_2(16) = 24 - 4 = 20$

Supernet mask : ~~200.1~~

255.255. 11110000 . 00000000

255.255.240.0

* Eg: 100.1.2.0/25
100.1.2.128/26
100.1.2.192/26

(i) Contiguous

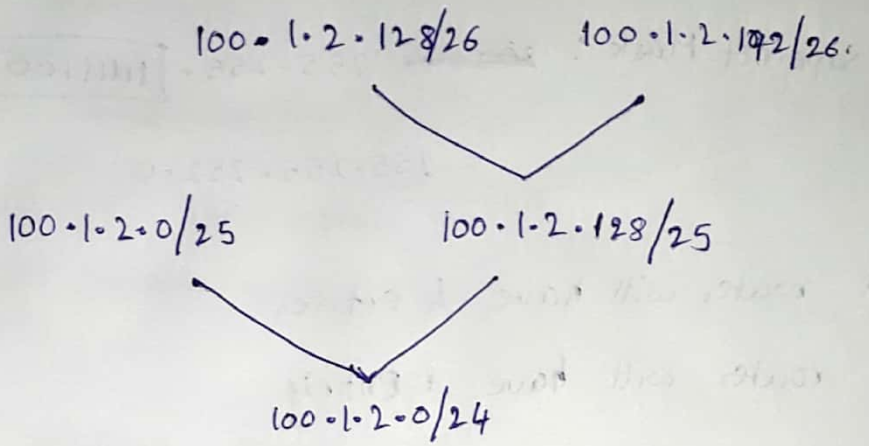
* (ii) Though size is not ~~same~~ same, we can first supernet the 2 small networks and then supernet with the larger one.

(iii) size of supernet

$\because 2^7 + 2^6 + 2^6 = 2^8$

1st N/w id is divisible by 2^8

∴ Supernetting is possible



Supernet id: 100.1.2.0/24

Supernet mask: 255.255.255.0

Q3/G-12 An Internet service provider has the following chunk.....

Sol:

The two variants possible are:

(i) 245.248.128.0/21 → A
(ii) 245.248.136.0/22
(iii) 245.248.140.0/22

(i) 245.248.128.0/22
(ii) 245.248.132.0/22
(iii) 245.248.136.0/21 → A

Flow Control Method

Delays in Computer Networks:

Transmission Delay (T_t)

It is time taken by the host to put that data packet on outgoing link.

Eg: If bandwidth = 1 bps

then for 10 bit data, transmission delay = 10 sec

If Data is L bits and bandwidth is B bits/sec then

$$\text{transmission delay} = \frac{L}{B}$$

Note:

Whenever k is given with data, it is 1024.

k is given with bandwidth, it is 1000

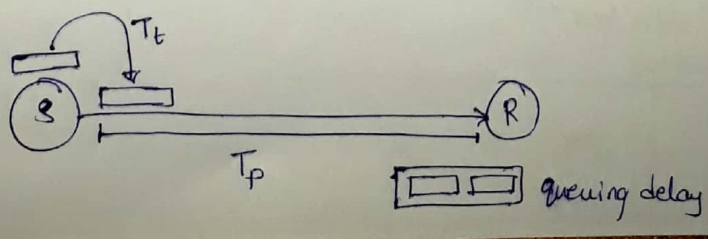
Silly for M & G

Eg:

L = 1000 bits BW = 1kbps	L = kb BW = 1kbps
$\Rightarrow T_t = \frac{1000}{1000} = 1$ sec	$\Rightarrow T_t = \frac{1024}{1000} = 1.024$

Propagation Delay:

It is amount of time required for data to travel from source to destination after data being placed on the link.



If 'd' is the distance b/w the ~~sour~~ hosts and v is the velocity of signal then

$$T_p = d/v$$

Generally velocity of signal in optical fibre is 70% of velocity of light

$$\Rightarrow V = 2.1 \times 10^8 \text{ m/s}$$

Total time to send a packet = transmission delay + Propagation delay

$$= T_t + T_p$$

Queuing delay:

The packets that are sent are not processed immediately by the reciever. The sent packet will wait in a queue. The amount of time the packets wait in the queue, before they are processed, is called queuing delay.

Processing delay:

The amount of time required by the receiver to process the packet is called ~~pdo~~ processing delay.

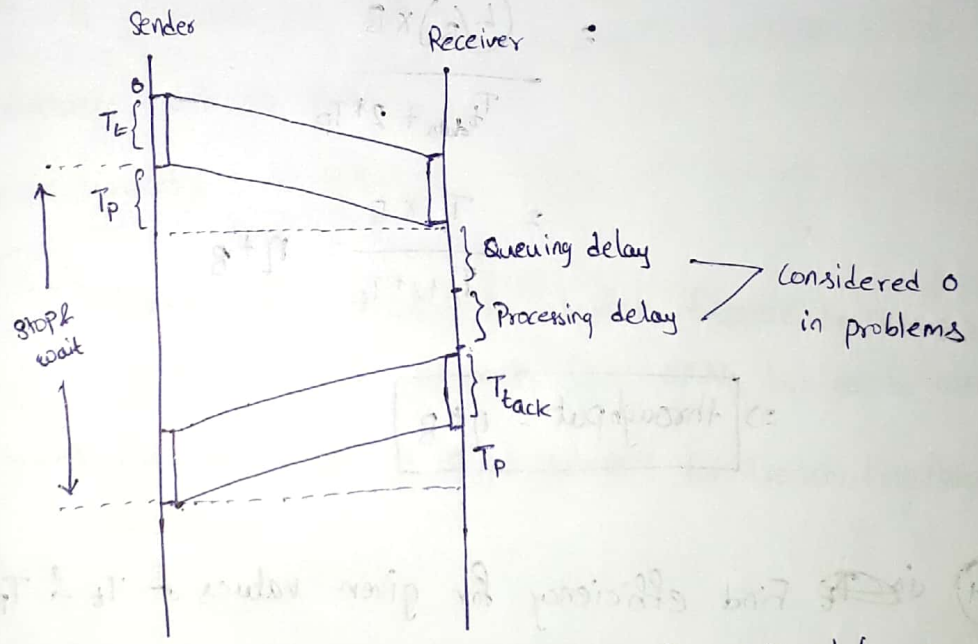
→ Queuing delay & processing delay depend on the speed of the receiver. In numerical, if not given, queuing delay and processing delay are considered 0.

Flow Control Methods:

→ If receiver is slow and sender is fast, then packets sent by sender may get lost.
So the speed at which sender has to send the data depends on recieiver. This is known as closed loop protocol.

Stop & Wait Protocol:

→ Here sender sends a packet and waits for the acknowledgment from receiver



Propagation delay is same for both data & ack

∴ Total time to send one packet

$= T_{t_{data}} + T_p + T_{t_{ack}} + T_p$

∴ Total time in stop & wait $= T_{t_{data}} + T_{t_{ack}} + 2 * T_p \cong T_{t_{data}} + 2 * T_p$

$T_{t_{ack}}$ → negligible compared to $T_{t_{data}}$ (without piggy backing)

$= T_{t_{data}} + 2 \times T_p$

$$\text{efficiency } (\eta) = \frac{\text{useful time}}{\text{total cycle time}} = \frac{T_{t_{data}}}{T_{t_{data}} + 2 * T_p} = \frac{1}{1 + 2\left(\frac{T_p}{T_{t_{data}}}\right)}$$

$$= \frac{1}{1+2a} \quad \left(a = \frac{T_p}{T_{t_{data}}} \text{ This just for simplicity}\right)$$

throughput: no of bits (effective) sent per second.

(also known as effective bandwidth / bandwith utilization.

$$\text{throughput} = \frac{L}{T_{t_{data}} + 2 * T_p}$$

L is size of data

$$= \frac{(L/B) \times B}{T_{t_{data}} + 2 * T_p}$$

$$= \frac{T_t \times B}{T_t + 2 * T_p} = \eta * B$$

$$\Rightarrow \boxed{\text{throughput} = \eta * B}$$

(Q4) Find efficiency for given values of T_t & T_p

(i) $T_t = 1ms \quad T_p = 1ms$

Ans: $1/3 \Rightarrow 33.33\%$

(ii) $T_t = 2ms \quad T_p = 1ms$

Ans: $1/2 \Rightarrow 50\%$

(Q5) Find relation blw T_p & T_t if min efficiency is 0.5

Sol:

$$\eta \geq 0.5$$

$$\frac{T_t}{T_t + 2T_p} \geq \frac{1}{2}$$

$$2T_t \geq T_t + 2T_p$$

$$T_t \geq 2T_p$$

Factors affecting efficiency in stop & wait:

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

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

for a given medium v & B are fixed

So η depends on d & L

distance, $d \uparrow \Rightarrow \eta \downarrow$

size of packet, $L \uparrow \Rightarrow \eta \uparrow$

$\therefore$ Stop & wait is good for shorter distance transmissions

$\therefore$ It is good for networks like LANs but not for WANs

Also stop & wait is good ~~while~~ for transmiting larger packets.

Note:

→ If sender doesn't receive ~~a packet~~ acknowledgment it may be due to 2 reasons

(i) Data is lost

(ii) Ack is lost

Data packet lost Problem:

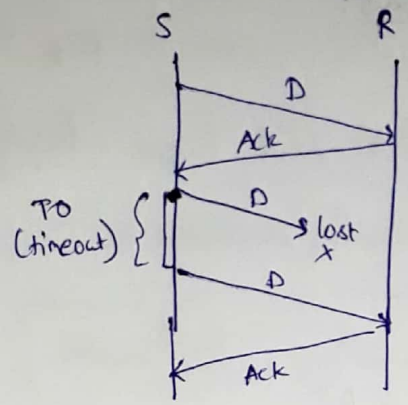
Assume a case where sent packet is lost. Then receiver won't send ack and sender assumes data is lost.

So sender waits for a fixed amount of time; timeout and resends the same data.

This variant of stop & wait is called Stop & Wait ARQ

↓

Automatic Repeat Request



→ without using this variant, the sender & the receiver may go to deadlock (sender waits for ack & receiver waits for data)

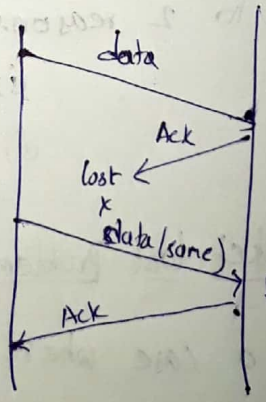
Duplicate Packet Problem:

→ This problem occurs when ack sent by receiver is lost.

→ In this case, sender assumes data is lost and resends it.

→ Now receiver receives the same packet again. This problem is called duplicate packet problem.

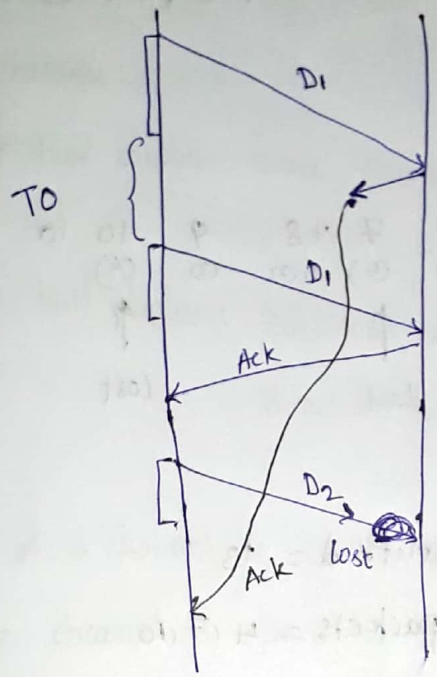
→ To solve this problem we use sequence number for every packet.



→ So here we have timeout along with sequence number.

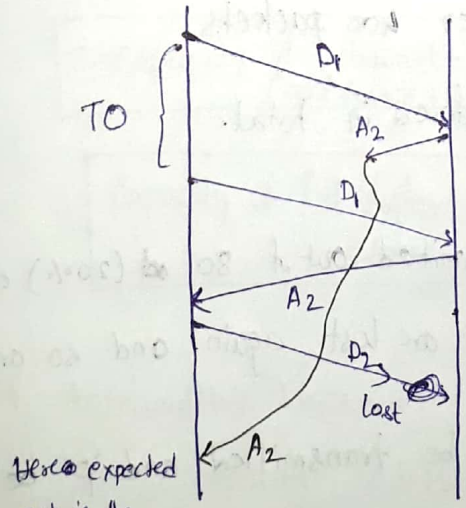
Delayed Acknowledgment (Missing packet problem):

→ Here ack is not lost but it is delayed such that timeout occurs and sender retransmits the data.



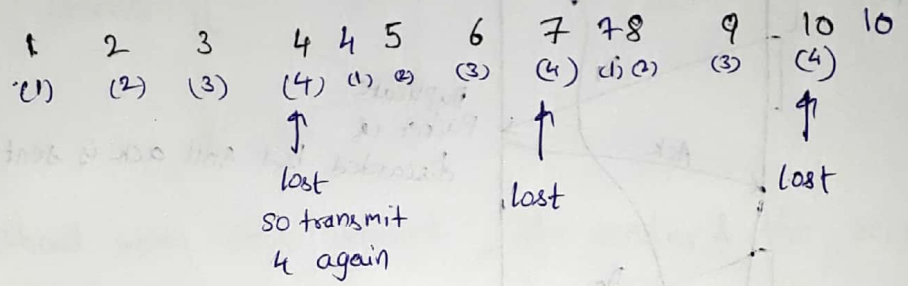
→ Here even though D_2 is lost, sender gets the ack of D_1 and thus sender thinks that D_2 is received by the receiver.

→ To avoid this problem we give sequence number for ack also.



(86) Consider using stop & wait protocol and transmitting 10 packets. Assume every 4th packet that is transmitted is lost. Find total no of packets that are transmitted and also find the packets that are retransmitted.

Sol:



∴ no of packets transmitted = 13

retransmitted packets = 4, 7, 10

Note:

→ Consider a channel b/w source & destination in which error probability of channel = 0.2

Assume we need to transfer 400 packets.

Find no of packets transfered in total.

Sol:

Initially 400 are transmited out of 80 (20%) are lost.

Among these 80, 20% are lost again and so on...

→ If n is no of pckts to be transmitted and 'p' be error probability of channel,

$$\text{no of pckts transmitted} = n + np + np^2 + \ldots$$
$$= n\left(\frac{1}{1-p}\right)$$

Here $n=400$; $p=0.2$

no of pckts transmitted = $400\left(\frac{1}{1-0.2}\right) = 500$ pckts.

Capacity of channel/wire/link:

types of channels:

- Full duplex: Data can be transimitted in both the directions at the same time.
- Half duplex: Data can be transimitted in both the directions but only in one direction at a time.

→ Capacity of a channel is no of bits present on the channel when the channel is operating at its maximum capacity.

→ After a bit is placed on the channel it takes T_p time (propagation delay) to reach the other end. That means all the bits placed on the channel during this time T_p are present on the channel.

$$\therefore \text{Capacity of channel (half duplex)} = \text{Bandwidth} * T_p \text{ bits}$$

$$\text{Capacity of full duplex channel} = 2 * \text{Bandwidth} * T_p \text{ bits}$$

Eg: If bandwidth = 1 bps and $T_p = 10$ sec

Capacity = 10 bits (half duplex)

→ Channels with high capacity are called thick channel/thick pipe

→ One more disadvantage of stop & wait, it is not good with thick channels. But it is good with thin-pipes

∴ Stop & wait thick channels → less efficiency

stop & wait, thin pipe → high efficiency.

$$\eta = \frac{1}{1+2*\frac{T_p}{T_t}} = \frac{1}{1+\frac{2\times T_p \times B}{L}} = \frac{1}{1+\frac{\text{Capacity}}{L}}$$

Capacity ↑ ⇒ efficiency ↓

Pipelining / Sliding Window Protocol:

→ This is a technique used to increase efficiency.

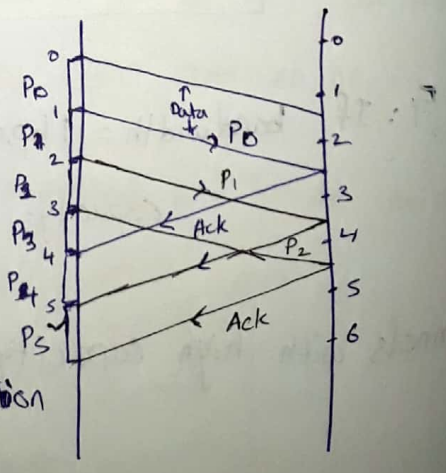
$$\eta = \frac{1}{1+2a}$$

This mean only 1 unit of time is used in (1+2a) units. which means we can increase efficiency by transmitting packets in that 2a units of time also.

This is known as pipelining.

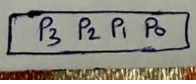
Eg: Consider $T_t = 1ms$ $T_p = 1.5ms$

$$\eta = \frac{1}{1+2a} = \frac{1}{1+2\left(\frac{1.5}{1}\right)} = \frac{1}{4}$$



Here ack of P_0 is received after 4 units of time.

So sender has to save P_0 until that time so that retransmission can be done if P_1 is lost



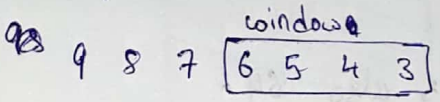
So minimum window size required = 4

↳ group of packets saved by sender so that retransmission can be done if needed.

→ Packets in window are called outstanding packets.

→ After 4 units of time, packet P_0 will be discarded from buffer (window) and P_4 will be stored.

→ Packets in the window are those that are waiting for ack.



As the time progresses, the window slides towards left.

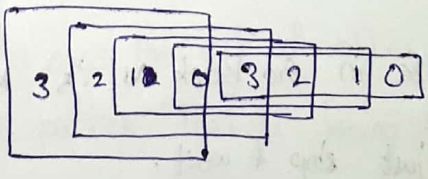
* → Minimum size of window required (Ws) = $1+2a = \frac{1}{\eta}$ packets
* ↳ (ceil if not integer)

→ Sequence number has to be stored in the header of packet.

So we cannot use arbitrarily large no of bits for seq. num.

Initially pckts in window are P_3, P_2, P_1, P_0

later when P_0 is disearded, we can again use sequence number 0.



* * Thus no of bits needed for representing seq. num = $\lceil \log_2(1+2a) \rceil$

Q7) $T_t = 1ms$; $T_p = 49.5ms$; Find min window size req & min no of bits req in seq num.

Ans: 100 & 7

→ But sometimes due to protocol overhead, no of bits in seq num field may be less.

i.e., for example in (Q7) no of bits may be just 6. but not 7

in this case we won't get max efficiency.

So here efficienty = $\frac{\text{no of pckts sent}}{\text{max possible no of pckts that can be sent}} = \frac{64}{100} \cong 64\%$

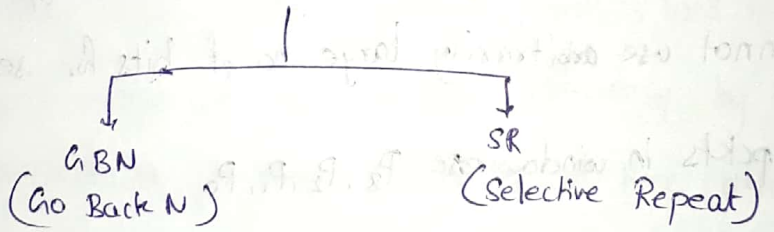
So here window size is also 64.

→ If size of seq number is n bits, then

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

04/12/20

→ Sliding window protocol is a theoritical concept. It can be implemented in 2 ~~two~~ two ways:



Go Back N: (N>1)

i) Sender window size in Go Back N is = N and N > 1

If N=1 it is just stop & wait.

Eg: Consider using GB 10 and $T_t = 1ms$ & $T_p = 49.5ms$. find η BW = 40 Mbps

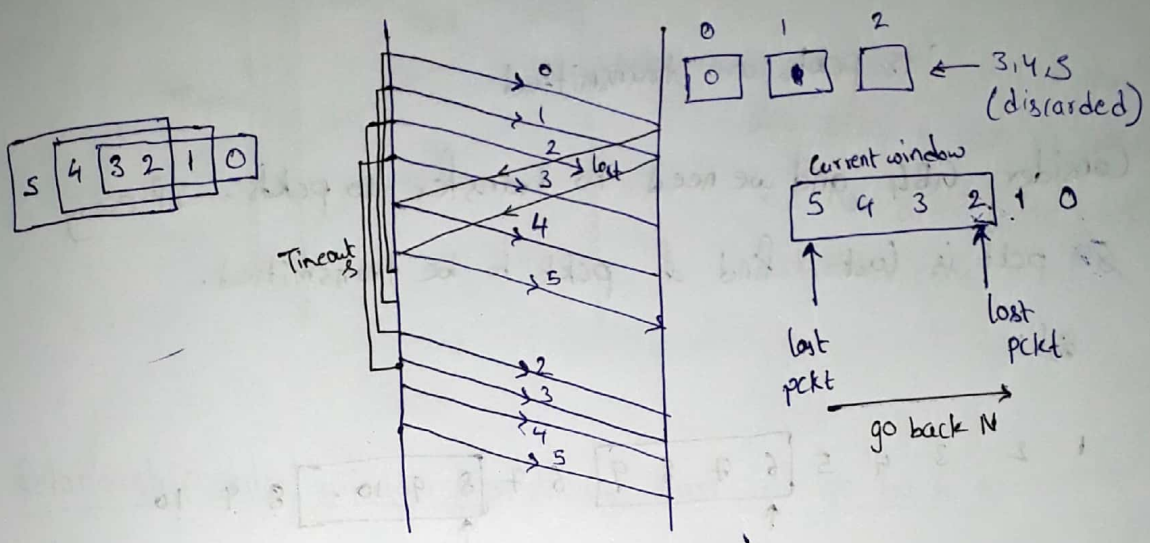
Sol:

$$\eta = \frac{\text{no of pcks sent}}{\text{max possible}} = \frac{10}{1+2a} = \frac{10}{100} \Rightarrow \text{i.e., } 10\%$$

$$\text{throughput} = \eta * B = \frac{10}{100} \times 40 = 4 \text{ Mbps}$$

ii) Receiver window size = 1

Eg: Consider below situation & let sender window size = 4



→ Consider a situation where receiver receives pckts 0 & 1 but pckt 2 is lost. By the time timeout of Packet 2 occurs sender transmit pckts 3, 4, 5.

~~After the timeout~~

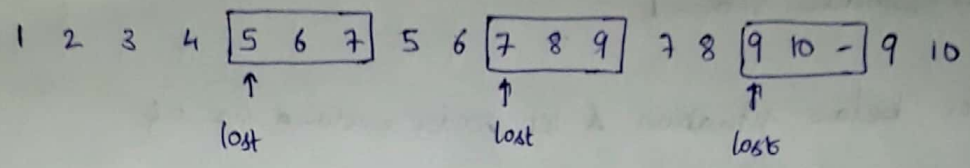
→ Here receiver first receives pckts 0, 1 and waits for pckt 2. pckt 2 is lost and receiver will reject pckts 3, 4, 5.

→ So considering this, after timeout of pckt 2, sender again sends pcks 2, 3, 4, 5 by going back 4.

→ Here the window slides once the ack of oldest pckt of window is received

Eg: In GB3, if every 5th packet that is being transmitted is lost and if we have to send 10 packets, then how many transmissions are required.

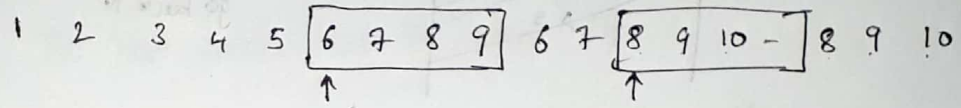
Sol:



∴ 18 pckts are transmitted

(Q6) Consider GBN and we need to transfer 10 pckts. If every 6th pckt is lost, find # pckts to be transmitted.

Sol:



∴ 17 pckts are to transmitted.

(iii) Go Back N uses cumulative acknowledgment:

Ack is of two types:

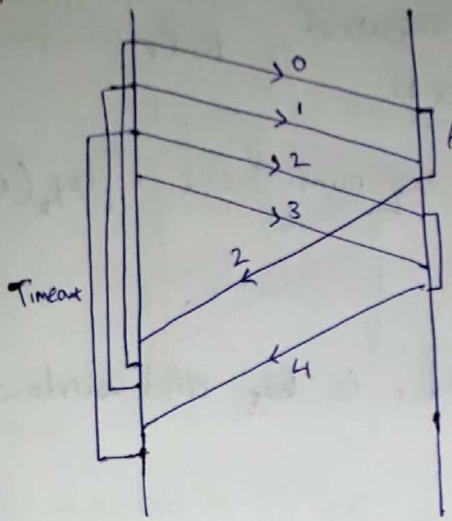
- Cumulative ack: One ack is sent for group of packets
  - adv: less traffic
  - disadv: less reliable (∵ even if one pckt is lost, ack is not sent and all has to be retransmitted)
- independent ack: one ack is sent is for each packet.
  - adv: more traffic
  - disadv: more reliable

→ Receiver has an acknowledgment timer. Once a pckt is received the receiver starts timer and once the timer expires receiver send ack for all the pckts received in that time.

→ Ack timer should be so large that time out occurs on sender side.

∴ time out timer > ack timer.

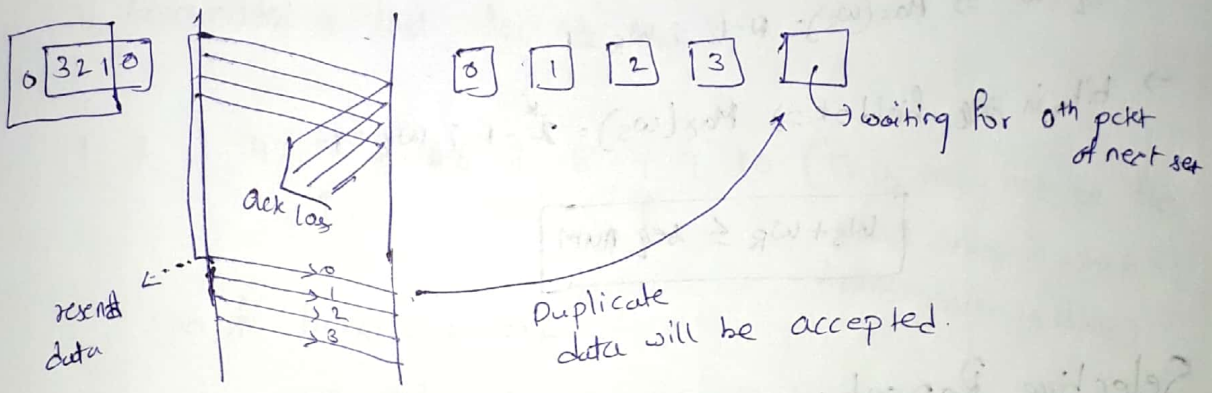
→ Also ack timer should not be too small.



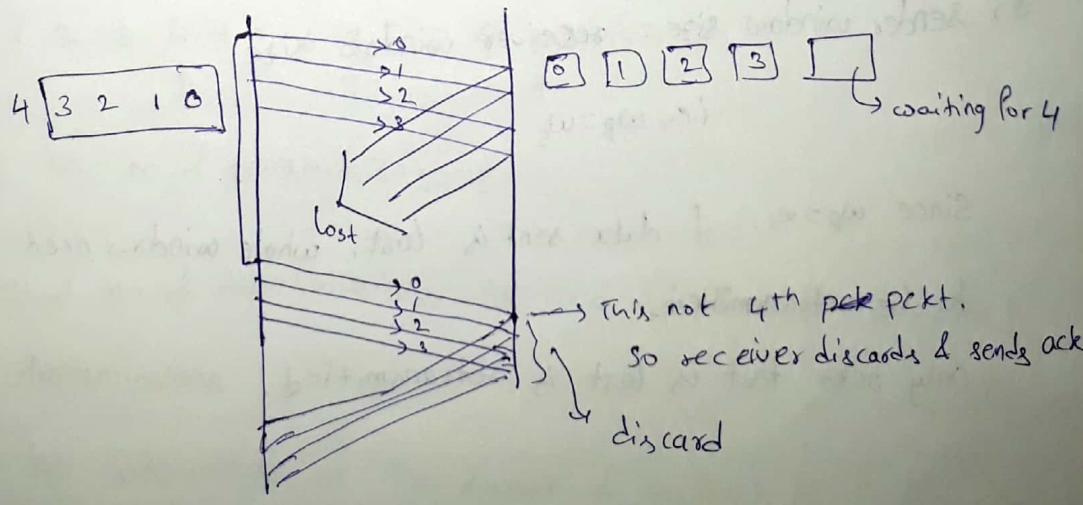
Ack time → new ack timer starts only after a pckt is received but not after the previous ack timer ends.

Relationship blw window size & seq num in go back N:

Consider GB4, and let no of seq numbers available = 4



But if take ~~2~~ no of seq numbers = 5 then we will not have this problem



So here no of seq numbers required $\leq N+1$

in go back N

no of bits in seq num field = $\lceil \log_2(N+1) \rceil$

Note:

Sly if window size of sender is W_S and window size of receiver is W_R,

no of seq numbers req = $W_S + W_R$

no of bits in seq num field = $\lceil \log_2(W_S + W_R) \rceil$

Note:

In GBN,

→ $W_S = N$ $W_R = 1$ ⇒ seq = N+1

→ Seq = N ⇒ Max(W_S) = N-1 ; $W_R = 1$

→ bits in seq field = k ⇒ Max(W_S) = $2^k - 1$; $W_R = 1$

$W_S + W_R \leq$ seq num

Selective Repeat:

(i) Sender window size > 1

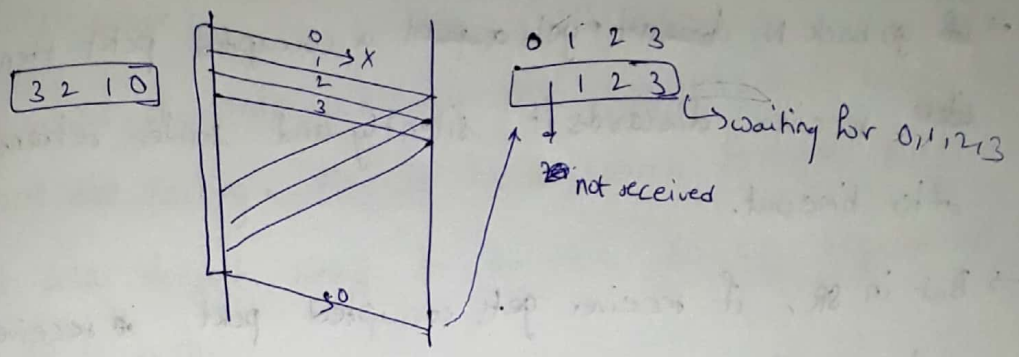
(ii) Sender window size = receiver window size

i.e., $W_R = W_S$

Since $W_R = W_S$, if data sent is lost, whole window need not to be retransmitted.

Only pckt that is lost is retransmitted.

Eg: let $w_R = w_S = 4$



Assume data pckt 0 is lost:

Still receive can accept pckts 1,2,3 (at it has sufficient window.

So here no of retransmission req will be less.

Eg: Consider we need send 10 pckts where every 5th pckt that is transmitted is lost. let $w_S = w_R = 3$

1 2 3 4 5 5 6 7 8 9 9 10 (This may not be the exact order in which transmission is done)

∴ no of transmission = 12

but go back N required 18 transmission.

Eg: Consider 10 packts to be transmitted and every 4th pckt is lost.

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

Here no of transmissions = 13

→ In terms of no of transmissions SR and stop & wait have same no of transmissions

→ So SR has advantages of stop & wait & go Back N.
But there are some disadvantages too.

3) SR uses independent acknowledgment

→ In go back N if receiver gets a corrupted pckt then, receiver discards it silently and sender retransmits after time out.

→ But in SR, if receiver gets corrupted pckt, receiver sends negative ack to sender so that sender retransmits the corrupted pckt even before the time out.

Comparision blw the protocols:

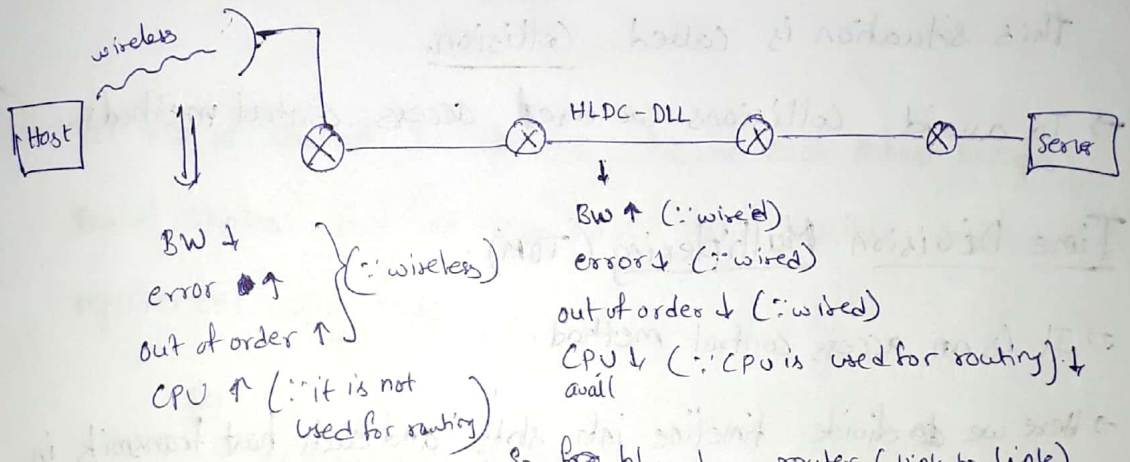
	Stop & wait	GBN	SR
Efficiency	$\frac{1}{1+2a}$	$\frac{N}{1+2a}$	$\frac{w_s}{1+2a}$
Buffer req (sender + receiver)	1+1	N+1	$w_s + w_s$
Min seq num required	1+1=2	N+1	$w_s + w_s$
Retransmission per pckt loss	1	N	1
BW req / BW consumption	low	high	moderate
CPU requirement	low	moderate	high
Implementation difficultly	low	moderate	high
ack	independent	Cumulative	independent

→ BW requirement is high for GBN, because of more no of retransmissions

→ In SR, out of order pckts are also accepted. So it needs to perform sorting. Also it has to search for last pckt. But GBN doesn't need to do this. So CPU requirement in SR is high.

→ If BW is high, CPU is less efficient, buffers are less then GBN is used
If BW is moderate, CPU is more available, buffers are more then SR is used

Example:



So for end to end connection, we used SR protocol

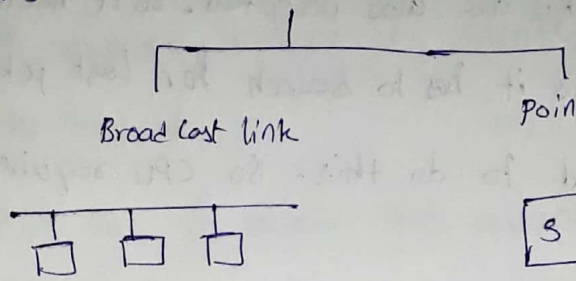
So b/w two router (link to link) GBN is used

→ A system may have several processes running. Each process has its own buffer. These buffers are operated with SR protocol.
All the packets (from diff processes) will be sent to DLL. At this layer, the buffers are operated with GBN.
So for higher level layers SR is used and for lower level layers GBN is used.

TCP — (SR (75%), GBN (25%))

Access Control Methods:

→ Links are of 2 types



All hosts connected to same link

Dedicated link is present blw two systems.

→ In broadcast link, there may be a chance that multiple station may transmit data at the same time. This situation is called Collision.

→ To avoid collisions we used access control methods.

Time Division Multiplexing (TDM):

→ It is an access control method.

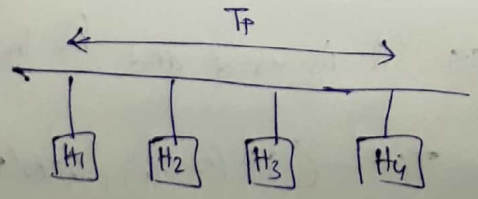
→ Here we divide timeline into slots and each host transmits in its allocated slot. in round robin fashion.

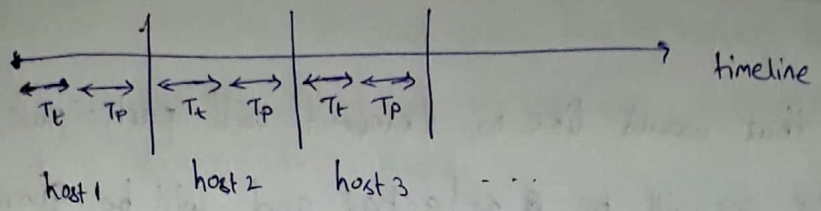
→ Allocated slot should be large enough that a host can transmit data.

i.e., ..time slot $\geq T_t + T_p$

Assuming that every station send same amount of data

Propagate delay in the worst case. i.e., when sender and destination are at two different ends of the channel.





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

Eg: $T_t = 1ms$, $T_p = 1ms$ & BW = 4Mbps. Find η & effective BW (throughput)

$$\eta = \frac{1}{1 + \frac{t}{T}} = \frac{1}{2} = 50\%$$

$$EBW = \eta \times BW = 2Mbps$$

Eg: Let BW of channel be 2Mbps. Assume each host needs BW of 2kbps. Then find max no of hosts possible satisfying this requirement using TDM

Sol:

$$N * 2kbps = 2Mbps$$

$$\Rightarrow N = 1000$$

Disadvantages:

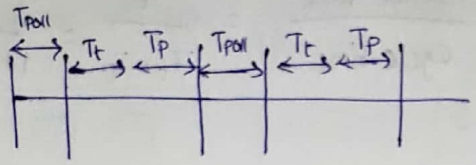
→ If a station doesn't want to transmit data, then the slot reserved for that station is wasted.

→ This disadv is overcome is using polling

Polling:

→ The system that would like to transmit will participate in polling

→ From these one will be selected and will be given time to transmit.



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

↳ taken as 0 if it is negligible.

Disadv:

→ Time for polling is a wastage.

CSMA/CD: (Carrier Sense Multiple Access Collision Detection)

→ Every host, before sending, senses the carrier and if it is free then it transmits data.

→ However a host can sense only the part that it is connected to the channel.

→ Also two stations may sense that the channel is empty and initiate transmission at the same time.

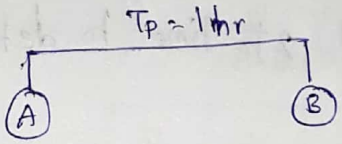
→ CSMA/CD doesn't use acknowledgement since it is mostly used in LANs. So we need to detect that collision has occured without using acknowledgment.

→ Collision produces a signal with different frequency from data. So a device can detect collision, if it sees this signal.

→ A device thinks collision has occured and its data is corrupted only if it senses collision signal while it is transmitting data.

→ So we make sure that every device transmits long enough that even in the worst case the collision signal will reach the device.

For example consider



let t = 10:00 AM

Let A & B both sense that channel is free and started sending data.

Then collision occurs at 10:30

and both A, B will detect collision signal at 11:00 AM

so this collision can be detected if $T_{t_A} > 1$ hr.

However this not the worst case.

Consider A has transmitted data to B and when its just about to reach B (after 1hr) B has started transmission Then this collision will be detected by A by A after 1hr.

A transmission initiation : 10:00 AM

collision : 11:00 AM

detection by A : 12:00 PM

∴ To detect any collision, $T_t \geq 2T_p$

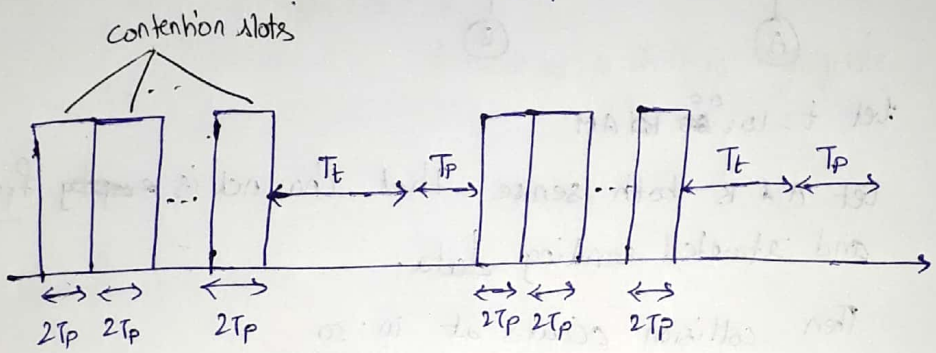
i.e., min data to be transfered, $L \geq 2 \times T_p \times B$

→ If we don't have sufficient amount of data to transfer then we need to pad extra bits.

Efficiency of CSMA/CD:
↳(ethernet)

→ Before a successful transmission, a series of collisions may occur. (we don't know the no of collisions)

In the worst case it takes $2T_p$ time to detect collision



$$\text{efficiency} = \frac{T_t}{C*(2T_p) + T_t + T_p}$$

$C \rightarrow$ no of collision slots in each cycle.

Finding C:

Assume we have n stations & every station transmits with probability p.

So there will no collision if only one station transmits the data.

$$\Rightarrow P_{success} = {}^{n}C_1 \times p \times (1-p)^{n-1} = np(1-p)^{n-1}$$

$$\frac{dP}{dp} = np(n-1)(1-p)^{n-2}(-1) + n(1-p)^{n-1}$$

$$= n(1-p)^{n-1} - n(n-1)p(1-p)^{n-2} = 0$$

$$= n(1-p)^{n-2}[(1-p) - (n-1)p] = 0$$

$\Rightarrow (1-P) - (n-1)P = 0$

$np - P - 1 + P = 0 \Rightarrow$ $\boxed{P = 1/n}$

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

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

If n is sufficiently large

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

$$\boxed{P_{max} = 1/e}$$

$\Rightarrow$ no of times we should try (collisions) before getting the first success $= \frac{1}{P_{max}} = e$

i.e., e collisions

$\therefore$ value of $c = e$

$$\therefore \text{Efficiency}, \eta = \frac{T_t}{(2e+1)T_p + T_t} = \frac{1}{1 + 6.44a}$$

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

$d \uparrow \Rightarrow \eta \downarrow$

$\therefore$ CSMA/CD is good for ~~CAN~~ LANs

$L \uparrow \Rightarrow \eta \uparrow$

$\therefore$ CSMA/CD has good efficiency for larger pckts.

Backoff Algorithm:

→ This algorithm is used to give waiting time (back off time) to stations that involved in collision.

→ This is also known as binary exponential back off algorithm.

→ This algorithm applies only for 2 stations.

→ Every packet is given a collision number.

i.e., the number collisions the pckt has involved in.

→ After a pckt gets collided, its collision number is increamented.

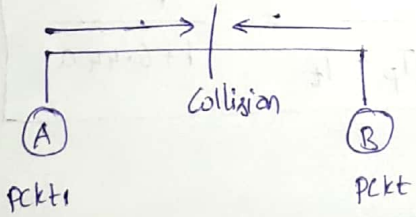
→ ~~Once a collision~~ After this both the hosts selects a number in the range $[0, 2^n - 1]$

where n is collision number.

Let the selected number be k.

Then waiting time = (k * time slot)

Eg: Consider a case where two stations A & B wants to transmit data



Collision number for pckt1 = 1

Collision number for pckt1 = 1

choose a num blw (0, 2^1 - 1)

= (0, 1)

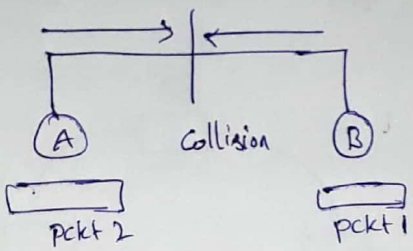
choose a num blw (0, 1)

A	B	
0	0	→ collision
0	1	→ A transmits
1	0	→ B transmits
1	1	→ collision

Let P(A), P(B), P(collision) be probabilities that A transmits, B transmits, collision occurs

$P(A) = 1/2$; $P(B) = 1/2$; $P(\text{collision}) = 2/4$

Assume A has transmitted here and it is ready with packet 2 and B is ready with pckt 1



Collision num = 1

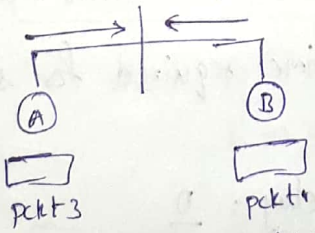
Choose k b/w (0, 1)

Collision num = 2

Choose k b/w (0, 1, 2, 3)

A	B	
0	0 → collision	⇒ P(A) = 5/8
0	1 → A	P(B) = 1/8
0	2 → A	P(collision) = 2/8
0	3 → A	
1	0 → B	
1	1 → A	
1	2 → A	
1	3 → A	

Now if A is successful again, we have below scenario



Collision num = 1

Collision num = 3

⇒ $P(A) = 13/16$ $P(B) = 1/16$ $P(\text{collision}) = 2/16$

Observations:

→ As collision number increases, the probability of collision is decreasing exponentially

→ If a system wins in transmitting successfully, then its probability of winning for further transmission increases exponentially while the probability of other system decreases exponentially. This is a disadvantage.

This is known as capturing effect.

Token Passing:

Note:

→ In some cases propagation delay is given in bits.

If propagation delay is n bits, it means that propagation delay is equal to transmission delay of n bits.

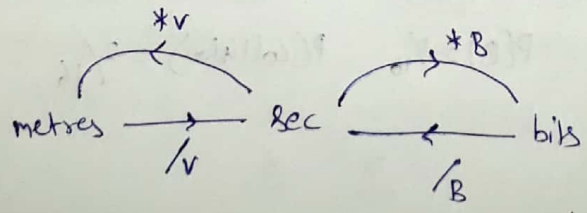
i.e., propagation delay of n bits = $\frac{n}{B}$ sec

B → Bandwidth.

→ In some cases, propagation delay is given in meters.

If propagation delay is n metres, it means that propagation delay is equal to amount of time required for signal to travel n meters.

i.e., propagation delay = $\frac{n}{v}$



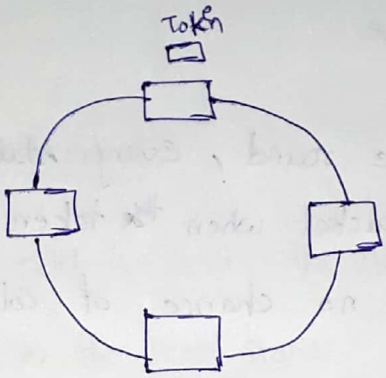
Token Passing:

→ It is an access control method.

→ All the hosts are connected in a ring topology.

A token is present and this token moves around the ring. This is known as token ring

whichever host has the token can transmit the data.



Ring latency: it amount of taken for a bit to travel around the ring and come back to the same point.

A bit while travelling around the ring, the bit has to be read by every station.

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

where N → no of stations

b → delay at each station in bits

$$\Rightarrow \text{ring latency} = \left(\frac{d}{v} + \frac{N * b}{B}\right) \text{seconds}$$

(B → Bandwidth)

(or)

$$\text{ring latency} = \left(\frac{d}{v} * B + N * b\right) \text{bits}$$

Token holding time : It is amount of time for which a token is held at any station.

$$\text{Cycle time} = \left(\frac{d}{v} + N * THT\right)$$

$\frac{d}{v}$ (T_p) → time req for a token to travel around the ring for one time

THT → token holding time

→ when token make one round, every station transmits exactly one ~~packet~~ packet when the token comes to it. So here there will be no chance of collisions.

Efficiency, $\eta = \frac{\text{useful time}}{\text{cycle time.}}$

$$\eta = \frac{N * T_t}{T_p + N * THT}$$

→ There are 2 mechanisms for token passing.

(i) Delayed token reinsertion

(ii) Early token reinsertion → Default consideration

(i) Delayed token reinsertion:

After getting the token, system transmits data and waits until data returns to the same system. Once data comes back, token is passed to next ~~host~~ station.

Here $THT = T_t + \text{ring latency}$

$= T_t + T_p + N * b$

$= T_t + T_p$ (assuming $b = 0$)

So in this strategy

$$\text{efficiency}, \eta = \frac{N + T_t}{T_p + N(T_t + T_p)}$$

$$\eta = \frac{1}{1 + \left(\frac{N+1}{N}\right)a}, \quad a = T_p/T_t$$

(ii) Early Token Reinsertion:

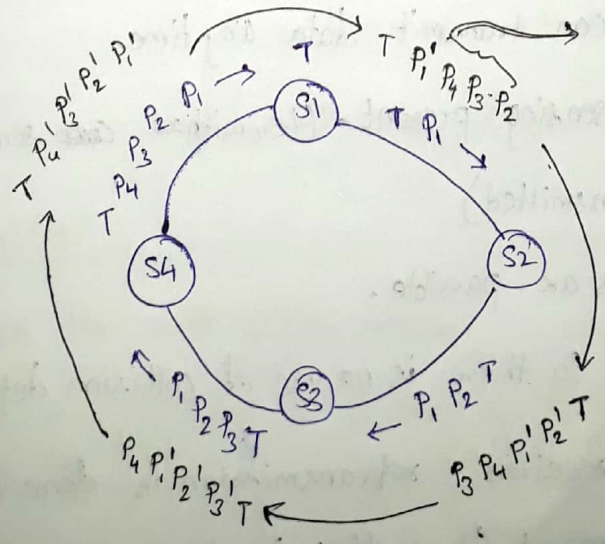
As soon as pckt is sent, the station ~~sends that to~~ passes the token to the next station.

→ Assume token is with S1. So S1 sends P_1 and then places the token.

Now S2 after receiving P_1 & T, S2 forwards P_1, then ~~it~~ sents its data P_2 and then finally forwards T.

Silly ~~S3~~ S3 forwards P_1 then P_2 then its pckt P_3 then the token T.

Also, in token ring ~~there~~ ~~data~~ sender has to remove the pckt from ring once the pckt comes back to the sender.



So here, THT = T_t (i.e., token is held until pckt is transmitted)

So efficiency, $\eta = \frac{N * T_t}{T_p + N * T_t}$

$$\eta = \frac{1}{1 + \frac{a}{N}}, \quad a = T_p / T_t$$

- For efficiency it is clear that early token reinsertion is good

→ ETR is efficient but less reliable (∵ more than 1 pckt is present at a time on the channel)

→ DTR is less efficient but more reliable

→ In ETR, at any given time n (no of hosts) number pckts are present in the ring.

In DTR, at any given time only one pckt is present. ~~at any given time~~

ALOHA:

→ It is another access control method.

→ (i) Here any station can transmit data any time.

Also no carrier sensing present. (i.e., without sensing the carrier data is transmitted)

(ii) So here collisions are possible.

Acks are used. So there is no use of collision detection.

(iii) If no ack is received, retransmission is done after some random amount (back off time) of time.

ALOHA is of 2 types:

(i) Pure aloha

(ii) Slotted aloha

Pure aloha:

→ All the 3 rules apply here.

→ If a transmission has to be done without collision, no one should have transmitted before T_t time and no one should transmit for next T_t time.

∴ This is known as vulnerable time.

Here vulnerable time $= 2T_t$ (neglecting T_p)

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

where $G \rightarrow$ no of stations who wants to transmit in T_t slot.

$\frac{d\eta}{dG} = 0 \Rightarrow G = 1/2$ → i.e., only 1 station transmits in vulnerable time.

$\Rightarrow \eta_{max} = \frac{1}{2} \times e^{-1}$

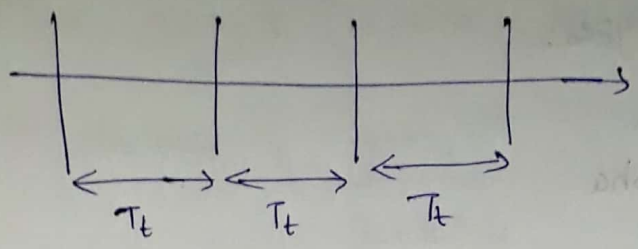
$\Rightarrow \boxed{\eta_{max} = \frac{1}{2e} = 0.184}$

∴ Max efficiency of pure aloha = 18.4%.

Slotted ALOHA:

→ Here also the same 3 rule apply.

→ But here the timeline is divided in time slots where each slot is T_t.



Here every station is allowed to transmit only at the begining of a time slot.

∴ vulnerable time = T_t

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

$$\frac{d\eta}{dG} = 0 \Rightarrow G = 1$$

$$\eta_{max} = e^{-1} = 0.368$$

∴ Max efficiency in slotted ALOHA = 36.8%

~~5~~ 5/12/20

Error Control Methods:

Error Detection

- Detects & req for retransmission

Eg: (D+D), parity check, CRC, Checksum

send data 2 times. and if they are same then no error

Error Correction

- Error is detected and corrected

Eg: Hamming Code

In Hamming code, if d data bits to be sent then $d/2$ more data bits has to be sent. All it requires lot of computation. (Not used practically)

Cyclic Redundancy Check (CRC):

→ Here both sender and receiver will have a common number called CRC generation.

→ ~~If CRC~~ Initially, send pads n-1 bits to data (on right end) if CRC is of n-bits.

→ Now perform XOR as shown below.

Let data on sender side be 1011011

Let CRC generator be 1101

CRC generator is 4 bits.

So pad 3 bits to data

1011011000

Now perform XOR as shown below

```
1101) 1011011000 (
      1101 ↓↓↓↓↓↓
      ----------
      0110011000
       1101
      ----------
      0001111000
        1101
      ----------
      0011100
       1101
      ----------
      0001
```

start XOR again from 1st 1 from left side

0001 → Here we stop the procedure cuz we have only one 0's bit

CRC

So append it to data

i.e., Data to be transmitted = 1011011 (data) 001 (CRC)

After obtaining data, receive does the same procedure with received data. If data received is Crct, CRC will be 0.

```
1101 ) 1011011001 (
       1101
       ------
       0110011001
        1101
        -------
        0001111001
           1101
           ------
           001101
            1101
            ----
            0000
```

CRC = 0

∴ Correct data is received.

Assume data has got corrupted and received as 1001011001
(error)

```
1101 ) 1001011001 (
       1101
       ------
       0100011001
        1101
        -------
        0101011001
         1101
         ------
         0110100 1
          1101
          ----
          0001001
```

remainder ≠ 0

∴ There is an error. So req for retransmission.

→ Sometime CRC generator will be represented as a polynomial.

If degree of CRC generator polynomial = n-1, then CRC generator is of n bits and CRC is of n-1 bits.

Eg: CRC generator: $x^3 + x + 1 \equiv 1011$

Checksum:

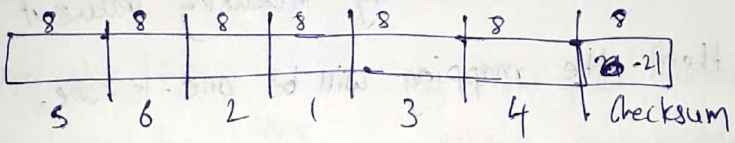
→ If we need to compute n-bit checksum, then data is divided into parts each of size n-bits.

→ Now convert each part into its decimal equivalent and add all these value.

→ Now negate the result obtain and place it in the checksum field.

Note:
while performing above steps, if data part has checksum then we take its decimal equivalent as 0.

Eg: Consider below 8-bit checksum

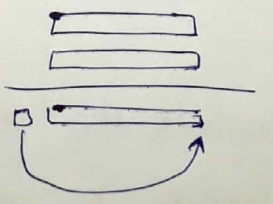


checksum 5+6+2+1+3+4=21

∴ checksum = -1

→ Now if the data received by receiver is correct then checksum sum calculated by receiver is zero.

Note:
Also while adding two numbers, if there is an overflow, then wrap around carry and add it.

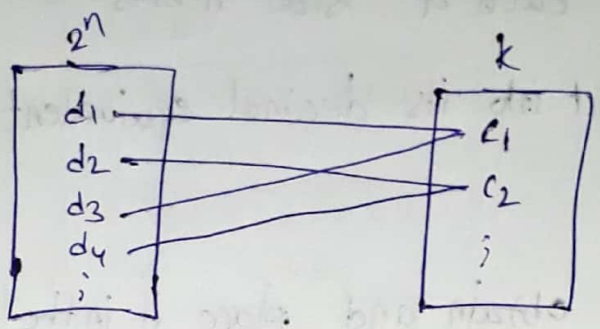


Meaningful Errors: Sometimes errors are possible in such a way that corrupted data appears as a valid data. i.e., both data & detection bits are corrupted in such a way that they form a valid combination.

Note:

Let data be of size of n bits and CRC / checksum is k bits.

$k << n$



d_1 & d_3 produces same CRC / checksum.

So if d_1 is corrupted to d_3, then this error won't be detected.

→ This problem can be minimized by increasing value of k.

If k=n then the mapping will be one-to-one.

~~All errors are~~

Even if mapping is 1-1, meaningful errors are possible.

6/12/20

ISO/OSI Stack

Functions provided by Computer networks

Mandatory	Optional
1. Error Control	1. Encryption
2. Flow Control	2. Checkpointing (while transfering large files checkpoints are used)
3. Access Control	3. Routing
4. Multiplexing & Demultiplexing	
5. Addressing	
etc.	

These function are implemented by ISO-OSI, TCP/IP, ATM,... etc.

ISO-OSI Layers

Application Layer
Presentation layer } User interachiveness
Session layer

Transport layer - More functionalities are present here (Thick)
Network layer - Complex functionalities are provided by Network layers (Routing)
Datalink layer - Contains H/w & S/w
physical layer - Contains H/w

(i) By dividing all functions into layer, we acheive Encapsulation & abstraction

(ii) Dividing layers makes testing easy because we can test only a specified layer

→ A host need not to contain network layer. Network layer is generally implemented on routers.

But transport layer is present on every host

Physical Layer:

Physical layer deals with electrical, mechanical, functional and procedural characteristics of physical links.

→ Depending on propagation medium signals has to be converted appropriately
i.e., Copper - Electrical signals; optical - light; wireless - electro magnetic waves

→ Functional properties means types of transmission modes.

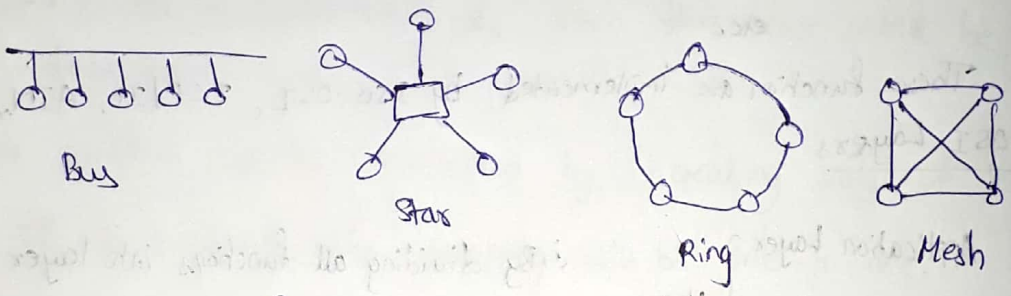
Simplex: Communication is one way

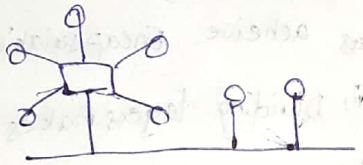
half duplex: Communication is possible in both way, but only one a time.

full duplex: Communication is possible in both ways at same time.

(In gate for all questions we consider full duplex by default)

→ Physical layer also deals with topologies





→ Physical layer also does the job of encoding:

i.e., converting bits into electrical signals

we have 2 types of encodings:

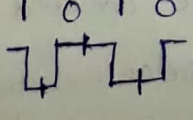
(i) Manchester encoding

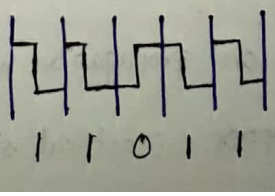
(ii) Differential Manchester encoding.

(i) Manchester Encoding:

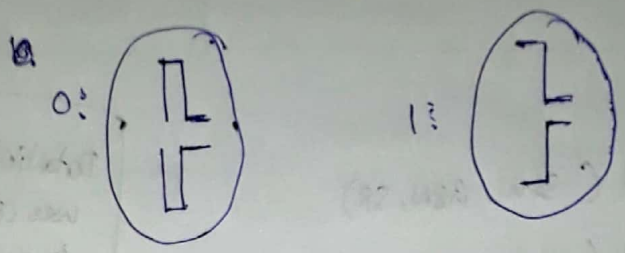
1: ⌊ 0: ⌈

Eg: 1 0 1 0



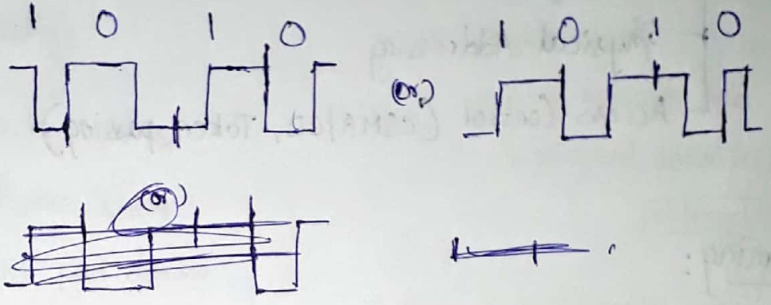


(ii) Differential Manchester Encoding

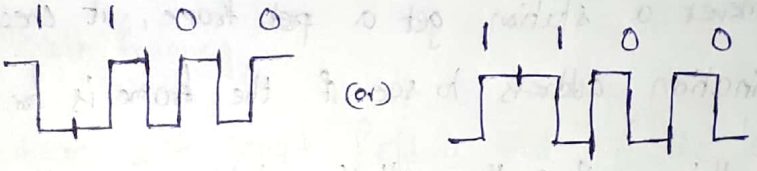


Here we have 2 representations for 0 & 1

Eg:



Eg:



Based on the representation we use for the 1st bit we get 2 different encodings.

Baud rate = 2 x Bit rate

Baud rate is no of voltages sent per second.

For each bit 2 voltages are sent.

→ PL also provides bit synchonization, bit rate control.

Data Link Layer:

Functionalities

- Flow Control (S&W, GBN, SR)
- Error Control (CRC, Checksum) — used in TCP/IP
- Framing
- Physical Addressing
- Access Control (CSMA/CD, Token passing).

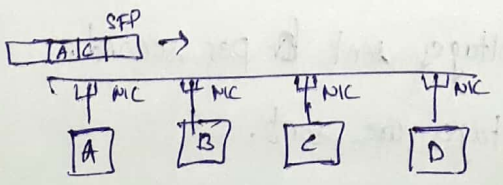
Data link layer uses CRC as it is more efficient to implement at H/w level

Framing:

→ Generally in LANs stations are connected with bus topology.

→ Whenever a station get a frame, it checks the destination address to see if the frame is for it or not.

→ So for this, rather than all the stations monitoring the channel, we used SFD (Starting Frame Delimiter) at the beginning of every frame and it will be detected by NIC and NIC will alert to station to check the destination address of the frame

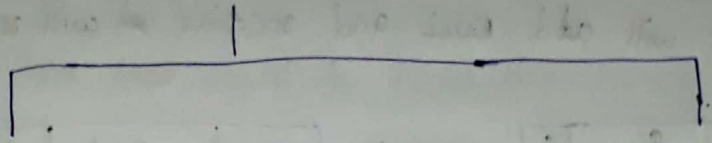


SFD is generally of for (10)*11

There will be no problem if data match with SFD because one SFD is found frame is read till end

→ Also there may be multiple frames on a channel at a time. So a station should read only the frame meant for it. For this purpose we should be able to detect the ending of frame. This detection is based on the framing technique we use.

Framing is of 2 types:



Fixed length Framing

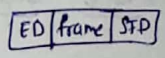
→ Here, knowing the starting of frame is enough. (Ending is known implicitly)

→ Disadv: If data to sent is less than the frame size then we need to pad dummy bits.

Variable length framing

→ Here starting is identified using SFD. Also we need to know the ending. This can be done in 2 ways:

- using length field
- using end delimiter (ED)



→ This is the mostly used framing

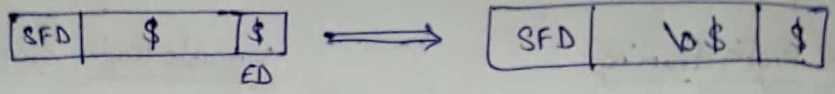
→ In variable length framing,

- the problem with length field is that, if the length field gets corrupted, then the end will be identified incorrectly. However in LANs error probability is less and hence we can use it.
- the problem with End delimeter is that, there is a chance that data pattern may match ED. This problem is dealt in two ways:
  - (i) Character stuffing / Byte stuffing
  - (ii) Bit stuffing

Character Stuffing:

→ For every pattern in data, that matches with ED with add a null character as a prefix to it.

For example if ED is $ and data has $ then sender will add NULL and receiver will remove the null



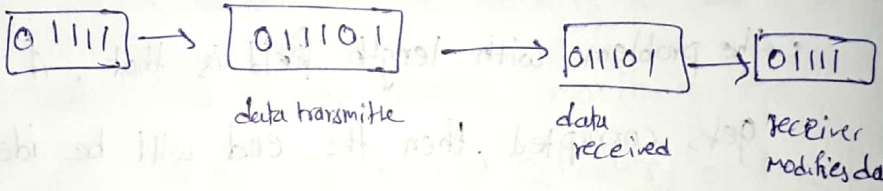
→ If data has $ \0 $ then sender will change it to \0\0\0$ i.e., for every null add a null and for every $ add a null

Disadv: Adding character requires more space.

(ii) Bit Stuffing:

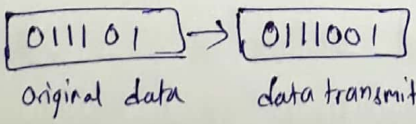
Let ED be 01111

If 01111 is found in data then sender will add 0 in this position.

∴ 

But consider the case where data transmitted itself is 011101. In this case also receiver removes the 0.

So to deal all possible cases, we come up with a strategy where we add a zero after 0111 and receiver removes zero after 0111 is found. So here sender adds 0 even in the cases where there is no ED pattern in data.



→ Similarly if ED is 011111, then we add 0 after every 01111

Eg: If ED = 01111 and data is 0111000111110

Find data that should be transmitted.

01110 0001111010

↑ ↑

Here bit is stuffed even without match

Eg: If ED is '10000, then we add 1 after every 1000

Physical Addressing:

Logical address: ~~It is address unique~~ It is an address which is unique across the world

Physical address: It is an address which is unique within a network.

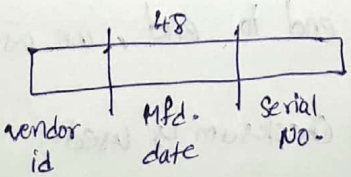
i.e., two system in two different Nw may have same physical address.

→ IP address is a logical address. It is a S/w number.

→ MAC address is a logical address. It is a H/w number.

↳ ENCODED in ROM of NIC.

MAC address is 48-bit address



Even though MAC is unique, IP add is used in communication. Because IP add has fields (NID, SID, HID) that help in routing.

→ Physical address is constant and doesn't change.

Generally MAC is used as physical address even though it is a logical address.

→ DLL is further divided into 2 sublayers.

i) Logical Link ~~Layer~~ Control (Flow control)

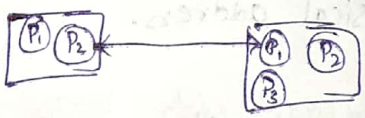
ii) Medium Access Control (Framing, access control, physical addressing)

} Error Control

Transport Layer:

Functions:

- End to End Connection: It is used to establish end to end connection. ie., process to process



It uses port numbers for this. This is known as Service Point Addressing

- Flow Control: Since it is end to end, we use SR protocol.
- Error Control: ~~It is~~ Mainly Checksum is used.
- Segmentation & Reassembly: Data received from upper layers may of different sizes. This is ~~changed~~ segmented into different sizes.
- Multiplexing & ~~Demuxing~~ Demultiplexing

Transport layer on sender side has to take

- message from many processes and send it through a single channel.
  Also TL on the order side take data from one channel and give it to different processes.
- Congestion Control

Network Layer:

Functions:

- Host to Host Connectivity: It transmits data from host to host (but not from process to process)

- Logical Addressing: Identifying hosts
- Switching: Switching is process of connecting various networks.
- Routing: Routing is building routing table.
  Later using this routing table is called switching
- Congestion Control:
- Fragmentation: Different Networks may support different maximum allowable data size.
  So in this case fragmentation is used.

Session Layer:

Functions:

- Authentication & Authourization
- Checkpoint: If connection is lost while data is being transmitted, then data from previous checkpoint can be retransmitted but not all.

- Synchronization : Eg: matching video & audio
- Dialog Control :
- Logical grouping
- Connection establishment, maintainance, termination (session)

Presentation Layer:

Functions:

- Character Translation
- Encryption & Decryption
- Compression

How all the layers work

DLL: Hop to Hop
NL: host to host
TL: end to end

On sender side:

In TL, port num of source & Destination are added
↳ well known port.

In NL, source IP & Destination IP are added
↳ found using DNS

In DLL, source MAC & Destination MAC are added
↳ found using ARP
↓
MAC of the default gate way.

Now this moved to PL and from there it will be sent to default gateway router

At Router R_1:

- Routers have only 3 layers PL, DLL, NL
- Based on the type of N/ws router is connected it may have several DLLs and several PLs but only one NL
- Before router forwards to next router, it changes source MAC & destination MAC.
- But source IP and destination IP are kept same.
- Also DLL of R_1 sends ack to DLL of host

At G:

→ NL uses UDP protocol. So no ack will be sent.
→ TL sends ack to the sender.
→ Using port number the pckt is sent to appropriate process.

07/12/20

LAN Technologies

Ethernet (IEEE 802.3)

It is one of the LAN technologies.

→ Topology : Bus

→ Access Control : CSMA/CD

→ No acks are used

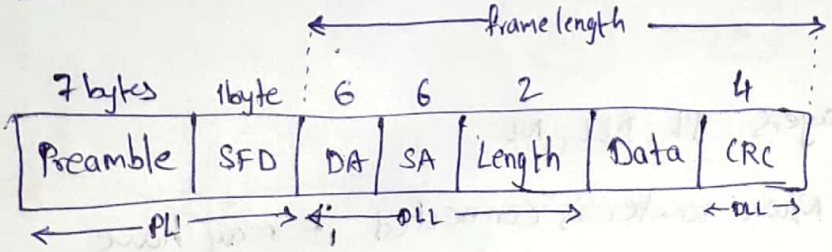
However if an application needs ack then it may send ack as if it is data.

→ Data rate/Band width : 10 Mbps, 100 Mbps, 1 Gbps

↓ fast ethernet ↓ Gigabit ethernet

→ Encoding technique : Manchester

Ethernet Frame Format :



Preamble : 10101010 ... 1010

SFD : 10101011

Data at diff layers :

AL - Message
TL - segment
NL - Datagram
DLL - Frame
PL - Single Protocol Data Unit (IPDU)

- Preamble & SFD are not added to the frame.

  PL just sends preamble & SFD just ~~be for~~ before sending it.

- Preamble & SFD ~~alerts~~ alerts other stations indicating start of the frame.

- Preamble & SFD also provides synchronization.

- DA & SA are MAC address of destination & source.

  ∴ It is 6 bytes

Types of MAC addresses:

(i) Unicast: Least significant bit of 1st byte is 0.

SA is always a unicast address. Otherwise the frame is invalid

(ii) Multicast address: Least significant bit of 1st byte is 1.

For multicasting, we assign MCA (Multicast Address) to a set of stations of the group.

Now all stations after reading DA; if the DA matches with their unicast address or multicast address, then they will read the data.

(iii) Broadcast address: All the bits are 1's

When a station sees all bits as 1's, then it will accept it.

→ Length field is used to tell the length of the frame. It is 2bytes.

Ethernet uses variable frame length. So we used length field.

Even though frame length is variable, since ethernet uses CSMA/CD, the frame must be of some minimum size.

$$T_t \geq 2T_p$$

$$L \geq 2T_p * BW$$

$L \geq 64$ bytes (from standard value of ethernet)

↳ from DA to CRC

Also maximum data (not frame) that can be sent is 1500.

This restriction is made so that one user won't be sending large frames and use the line.

i.e., to avoid monopolization.

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

→ CRC is 4 bytes i.e. 32 bits

∴ we use CRC polynomial of degree 32

Disadvantages:

→ It is not applicable for real time application.
i.e., in applications where sending data within time is important.

→ The minimum size of data to be sent is 46B.
so it is not applicable for interactive applications.

→ No priorities are present among stations
So it is not good for client-server application where server has to be given more priority.

Token Ring (IEEE 802.5)

→ It is another LAN technology that overcomes the drawbacks of Ethernet. But it has its own disadvantages.

→ Topology: Ring

→ Access Control: Token passing

→ Data sent: unidirectional (clockwise or anticlockwise)
∴ links used are simplex

→ Data rate: 4Mbps, 6Mbps

→ piggy backing acknowledgments are used.

→ Encoding: Differential Manchester encoding.

Sender side Problems:

In token ring, once data is placed on the link, it is removed by the sender only. Due to this reason there are 2 problems possible.

(i) Orphan packet Problem:

If a packet is placed on link and if the sender down by the time the packet comes back, the packet will never be removed from the link.

(ii) Stray packet Problem:

By the time sent packet comes back to the sender, if the pckt is corrupted enought that sender cannot recognize it, then the packet will remain on the link.

To solve the above two problems we introduce new station called monitor in the ring. Monitor is one of stations in the ring.

(i) A monitor bit is maintained, which is initially 0. When this pckt passes through the monitor, it set the bit to 1. when packet comes to the monitor for second time, the pckt will be removed.

(ii) CRC is present in every pckt. whenever the packet is corrupted monitor can detect it. Monitor removes the packet if it detects this change.

Problems on Destination:

(i) Destination is down:

In this case sender should not resend.

(ii) Destination is busy:

In this case destination buffers may be full or destination may be ~~bg~~ busy. So ~~used~~ sender has to resend the exam.

(iii) Packet corrupted:

In this case pckt is corrupted & sender has to resend.

(iv) Copied: Here destination receives pckt and hence there is no problem

To overcome these problems we use 3 special bits A, C, E ~~A B C~~ in header

A - Available (availability of destination)
C - Copied
E - Error

A	C	
0	0	— Initial
1	1	— Destination available & pckt is copied
1	0	— E=0, Destination busy E=1, pckt is erroneous
0	0	— Destination is down
0	1	— Invalid

→ This A & C are kind of acknowledgments.
So we say it is piggy backing acknowledgments

→ Also while retransmitting, sender has to reset monitor bit otherwise monitor will remove the pckt.

→ A & C are used ~~used~~ only for unicasting.

Token Problem

i) A station send large amount of data without releasing the token. This will lead to a problem of monopolization. This is known as Captured token.

So to overcome this, we impose restriction on maximum THT.

Default value of max THT = 10 ms

Bandwidth = 4 Mbps

→ Maximum frame size that can be sent with ETR

$= THT \times BW$

$= 10 \times 10^{-3} \times 4 \times 10^{6}$

$= 40000$ bits

→ Maximum frame size that can be sent with DTR

let THT = 10ms BW = 4mbps

RL = 9ms

within THT (10ms) transmitting and getting the pckt back should happen.

$\therefore THT = T_t + RL$

$\Rightarrow T_t = 10 - 9 = 1$ ms

$\therefore$ Max data sent $= 10^{-3} \times 4 \times 10^{6}$

$= 4000$ bits

while solving problems take ring latency as

$RL = T_p + \frac{N * b}{B}$

If bit dealy if not given then take $RL = T_p$

~~$\therefore$ In ETR, transmission can be do~~

$\therefore$ In ETR, $T_t = THT$

In DTR, $T_t = THT - RL$

(ii) Token is Lost :

Assume a case where a station takes a token and the station went down.

In this case monitor will generate a new token.

For that monitor has to be sure that token is lost but not delayed:

In best case

minimum token return time = RL

In worst case

maximum token return time = RL + N*THT
(TRT)

∴ If token doesn't return to the monitor within RL + N*THT, monitor will generate new token.

Eg : RL = 10ms ; THT = 10ms ; N = 10 ;

Max TRT = 10 + 10×10

= 110ms

(iii) Token is Corrupted :

In general, token size is 3 bytes. So since it is small we don't use any CRC for it.

So if a corrupted token comes to monitor, monitor assumes it is some pckt which has split from some data and monitor discard.

In this case also monitor generates token after maximum token return time.

Monitor Problems

(i) If monitor goes down, there will be lot of problems with stray pckts, orphan pckts etc.

Due to this communication will halt at some point.

To overcome this problem, monitor sends heart beat meassage periodically saying that it is present.

This heart beat messages are implemented using

AMP frames

↳ Active Monitor Presence

Every station expects AMP frames in regular interval of time.

~~If moni~~

If monitor goes down now, using polling one of the existing hosts will be selected as monitor.

(ii) Monitor is hacked (Monitor Malfunction):

In this case monitor is present but as it hacked it sends AMP frames without doing any work.

The only solution for this problem is human verification.

If communication is found to be halt then we will check if monitor is malfunctioning.

Advantages of Token ring:

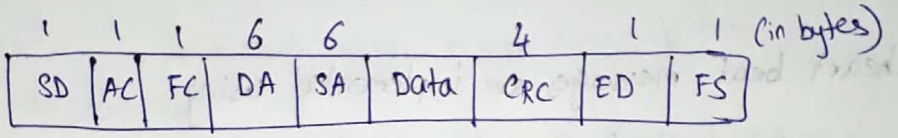
→ It is applicable for real time applications since there are no collisions.

→ It is also suitable for interactive applications as there is no limitation on minimum size of data that can be sent

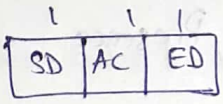
→ It is also applicable for client-server applications by giving priorities to stations.

Token Ring Frame Format:

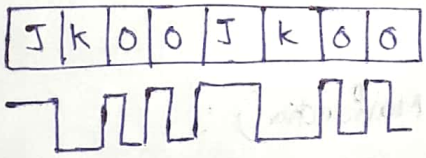
Data frame:



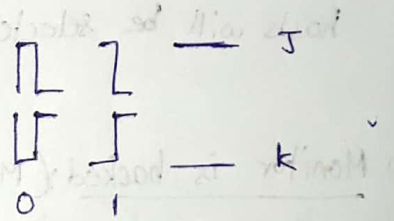
Token:



SD: Used for alerting that frame is being transfered and also performs synchronization (Start Delimiter)



Manchester Encoding.



J & k doesn't represent any data. They are just invalid patterns. So using it in SD, we can make sure that SD never occurs in data.

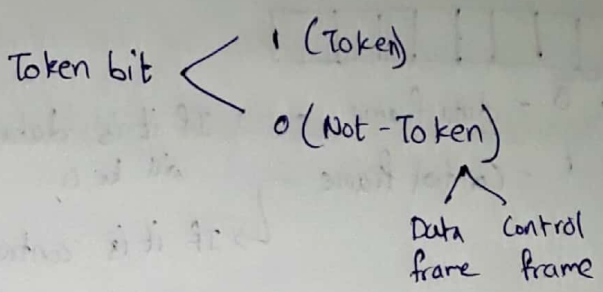
Access Control:

→ This field is of 1 byte size

→

* 0-7, eight levels of priorities are possible.

→ Token bit is used to say whether the frame is token frame or data frame



Priorities & Reservations:

Using these we implement priorities.

→ Every host is assigned a priority.

→ If a host has to transmit it will put its priority in reservation field. saying that it need that token. A host can update reservation field only if its priority is more than the content in reservation field.

→ Assume DTR is being used.

Sender sends data packet and waits until it comes back. As data pckt passes through all the hosts each host if needed will modify the reservation field.

when sender gets back the data field, it copies the reservation field of AC to reservation field of token.

Now this token can be held by a host only if its prority is greater than or equal to the content in the reservation fied.

→ In client server applications, server will be given priority 7 and all other host will be given a priority of 0.

Frame Control:

0 0 - data frame → If it is data frame then all the 8 bits will be 0

1 1 - Control frame → If it is control frame then the remaining 6 bit will say which type of control frame it is.

Examples of control frames:

- AMP
  - Polling - used at the time of polling
  - purge - used when some maintainence is going on so that no one will send data
  - beacon - It is used find if there is any cut in the frame.

DA & SA:

6 byte MAC addresses

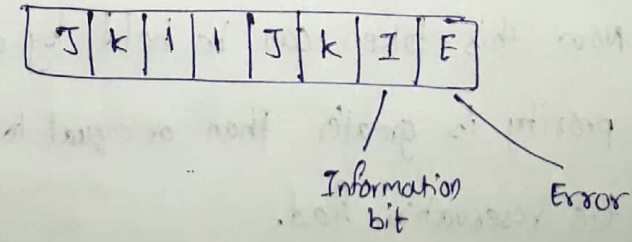
CRC:

4 bytes

∴ CRC polynomial of degree 32

End Delimeter:

Used to detect end of the frame.



→ Infomation bit = 1 ⇒ there is some of data following.

As there is restriction on max THT, a station may not be able

to send all the data in one THT

So it uses information bit to tell the destination that some more data will be sent.

→ If destination detects any error then destination will set E.

Frame Status:

A	C	0	0	A	C	0	0

→ why 2 copies of AC?

* CRC is computed only on field that are left to CRC field but not on FS & ED.
It is because A,C bit will be changed on receiver side.
So to have more reliability we maintain 2 copies of A,C.

→ Max size of data depends on max THT.

→ If ETR is used then priorities are implemented differently as token has to released as soon as sender sends the data.
So here based on the latest pckt (pckt in last cycle) seen by the sender, sender gets the reservation details and puts it on the AC field of the token.

Minimum length of token Ring:

Assume a case where all the hosts are down.
In this case monitor places token and receiving it quickly. So there is a chance that monitor get 1st bit of token back before it finishes the transmission.
In this case collision is possible.

To over come this we impose restriction on min length of ring. The min frame sent is token. Its size is 24 bits

Progration dealy ≥ transmission time of 24 bits

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

$\Rightarrow \boxed{d \geq \frac{24 \times V}{B}}$

Capacity ≥ 24

$Tp * BW \geq 24$ (∵ simplex)

$\frac{d}{V} \times BW \geq 24$

$\boxed{d \geq \frac{24 \times V}{BW}}$

Eg: $BW = 4Mbps$; $V = 2 \times 10^8$ m/sec

$\Rightarrow$ Min length of ring $\geq \frac{24 \times 2 \times 10^8}{4 \times 10^6} = 1200\,m = 1.2\,km$

But having this large link may not be possible.

~~In fact~~ Assume we have 1km link. In this case we add a delay device.

The delay ~~de~~ produce by this device should be equal to delay produce by a 200m link.

$\therefore$ delay req = 200 m

$= 200/V$ sec

~~$= 200 \times 2 \times 10^8$ sec~~

~~$= 4 \times 10$~~

$= \frac{200}{2 \times 10^8}$ sec

$= \frac{200}{2 \times 10^8} \times BW$ bits $= \frac{200 \times 4 \times 10^6}{2 \times 10^8}$ ~~bits~~

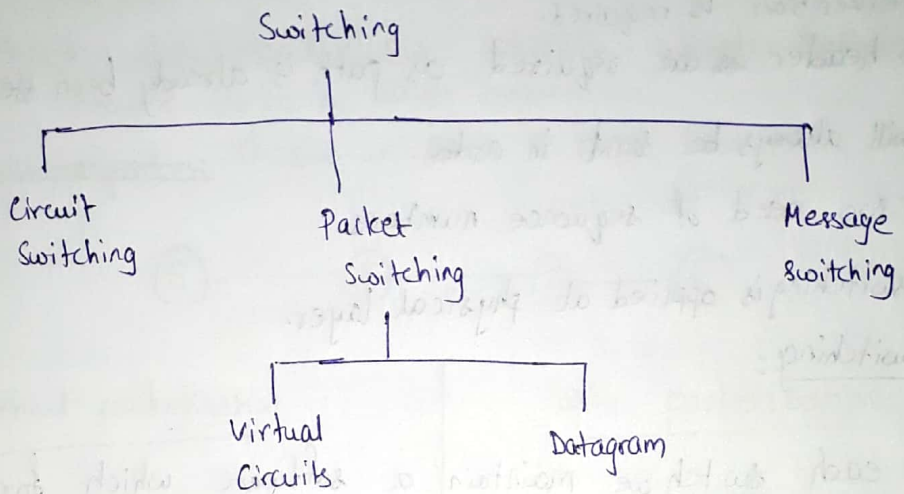
$= 4$ bit times

. Now assume that every station adds 1 bit delay, then min no of stations to be present in the ring = 4

08/12/20

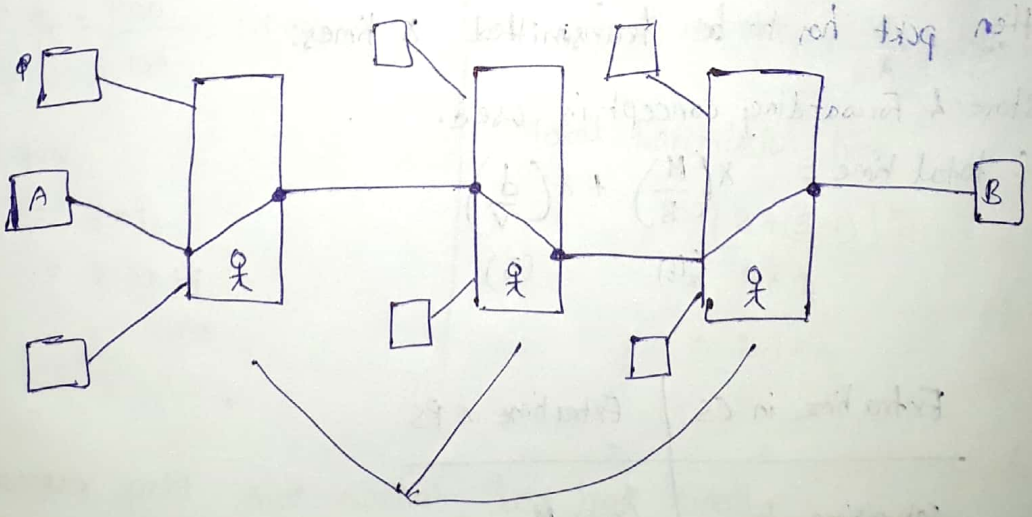
Switching:

Switching is process of connecting different networks.



Circuit Switching:

Connection and disconnection is made physically.



Telephone exchanges

∴ Total time req = setup time + $\frac{M}{B}$ + $x\left(\frac{d}{v}\right)$ + teardown time.

(T_t) (T_p)

M → size of message

B → Bandwidth

x → no of hops

d → distance blw two hops

v → velocity of signal.

setup time → time to setup connection

teardown time → time to remove connection

→ Human intervention is required.

→ Here no header ~~as~~ are required as path is already been steup.

→ Data will always be sent in order.

∴ No need of sequence numbers.

→ Circuit switching is applied at physical layer.

Packet Switching:

→ Here at each switch we maintain a software which forwards packet to appropriate switch.

→ Here pckt has to be transmitted at each switch.

→ If there are x no of hops ~~to~~ blw source & Destination then pckt has to be transmitted x times.

→ Store & Forwarding concept is used.

∴ total time = $x\left(\frac{M}{B}\right) + x\left(\frac{d}{v}\right)$

(T_t) (T_p)

Extra time in CS	Extra time in PS
setup time + teardown time	$(x-1)\frac{M}{B}$

- Thus for large data CS is ~~better~~ better

for small data PS is better

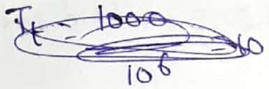
Packetization in packet switching (pipelining):

→ Here data to be transmitted is divided into packets so that pipelining can be obtained.

By this two switches will be transmitting parallelly.

Eg: Consider data = 1000 B ; Bw = 10^6 MBPS ; Header = 100 Bytes.
Assume $T_p = 0$ for easier comparision
~~without packets~~ Consider we have 3 hops blw sender & receiver

without packetization



Data to be transmitted
$= 1000 + 100 = 1100$ B

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

Total time
$= 3 \times T_t$
$= 3 \times 1.1$
$= 3.3$ ms

with packetization

Consider we ~~transmit~~ divide data into 5 pckts.
⇒ Size of each pckt $= \frac{1000}{5} = 200$
Size with header $= 200 + 100 = 300$

$\therefore T_t$ of a pckt $= \frac{300}{10^6} = 0.3$ ms

Total transmission time
$= [3 + (5-1)] T_t$
$= 7 * T_t$
$= 2.1$ ms

→ However pckt size cannot be too small.
For example if pckt size is 50 B then total time = 3.3 ms and this equal to original value.

→ So we need to use optimal pckt size (can be found by differentiation)

There are 2 types of pckt switching

(i) Virtual circuits (Eg: ATM)

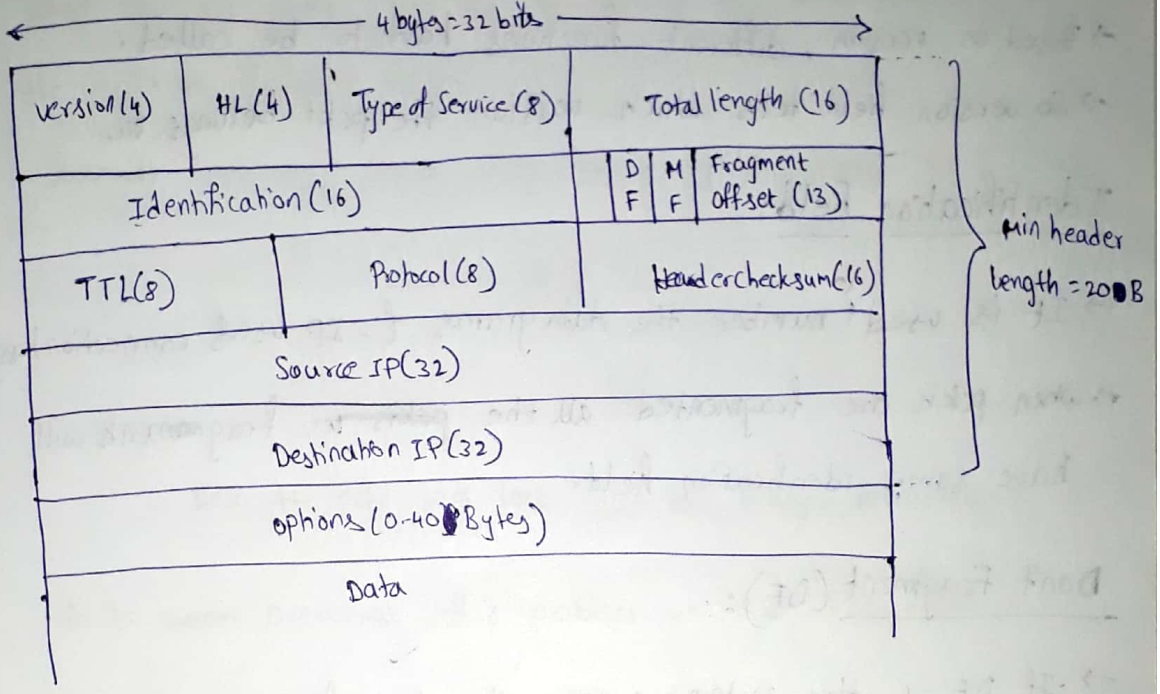
(ii) Datagram (Eg: IP)

In both the types data is divided into pckts.

Virtual Circuit	Datagram
(i) Here all the pckts follow same path	(i) Here each pckt follows different path and it is transfered independenly.
(ii) As the 1st pckt is transmitted all the remaining pckts follow the path taken by 1st pckt. So as the 1st pckt moves along the path it makes reservations at each switch for CPU, BW & buffers.	(ii) Here no reservations are made and path taken by each pckt may be different.
(iii) Connection Oriented	(iii) Connectionless
(iv) 1st pckt hay global header and remaining pckts have local header. global header help in finding path from source to destination whereas local header just helps the pckt follow the 1st pckt	(iv) All the pckts must have header
(v) Same path is followed and hence pckts arrive in order	(v) Different path are followed and hence pckts may arrive out of order.
(vi) Reliable	(vi) Not reliable
(vii) Costly	(vii) Not costly

Internet Protocol:

IP4 Header:



Min header = 20 Bytes; Max header = 20+40 = 60 Bytes

Header length (HL):

Max value possible in HL is $2^4 - 1 = 15$

But max header length is 60

so to obtain header length, we multiply header length field with 4

Header length = 4 * Header length field

Range of header length : 20 - 60

Range of header length field : 4 - 15

In exam, Based on value given, we understand which is given

Note:

If header length is 30 then Header length field = $\frac{30}{4} = 7.5$

So in this case we put 8 in HL field and pad 2 bytes to the header.

→ In worst case we may need to pad 3 bytes.

Version:

→ IP has various versions (IPV1 to IPV6)
→ Based on version, different functions have to be called.
→ So version field tells which version the pckt belongs to.

Identification Field:

→ It is used to number the datagrams (∵ IP uses connectionless)
→ when pckts are fragmented all the ~~pckts~~ fragments will have same identification field.

Don't Fragment (DF):

→ If DF = 1, then routers are not allowed to fragment the pckt.
→ In this case if router cannot forward it, the router will discard the pckt and will send an icmp error message to sender.

More Fragment (MF)

→ If MF = 1, it means that more fragments are following.

Fragment offset:

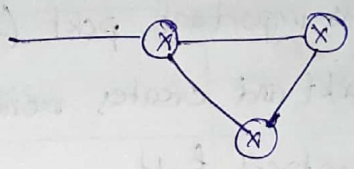
→ when packet is fragmented, ~~each~~ the fragment offset say no of data bytes ~~fragments~~ which are ahead of this fragment.

Time to live (TTL):

→ when a pckt is being routed, if a router don't know which n/w the pckt has to be forwarded to, it forwards the pckt to default router.

Sometime there may be a cycle

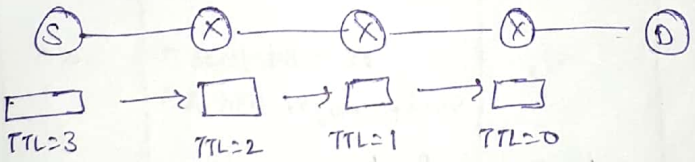
Eg:



Here the pckt will loop b/w the 3 hops infinitely.

So to overcome this problem we used TTL field.

→ The value in TTL say max no of hops the pckt can take. After every hop the value in TTL is decreamented

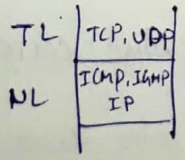


∴ Discard the pckt.

→ Destination also decrements TTL

→ The main purpose of TTL is to discard pckts that go into infinite loop.

Protocol:



The datagram may follow ICMP (or) IGMP (or) TCP (or) UDP.

To indicate which protocol is being followed we used the protocol field.

Actually identifying the protocol that is being followed is possible by looking into pckt.

But routers can't do that because we don't have transport layer in routers.

→ Also when router buffer is full it discards new incoming packets. However if incoming pckt is more important (say TCP) and router's buffer has less important pckt (say ICMP) then router will discard ICMP pckt and creates room for TCP pckt. So to do this we need protocol field.

→ The order in which pckts are discarded is

ICMP ---- least imp
↓
IGMP
↓
UDP
↓
TCP —— most imp

Header Checksum:

→ Header checksum is checksum of header part alone. (but not of data)

→ Finding checksum:

(i) Divide header part into smaller parts each of size 2 bytes.

(ii) Treat each part as a number and add them.

(iii) Now negate the result and store it in the checksum field.

while computing the header checksum, we take value corresponding to header checksum is 0.

→ At every router checksum has to be recomputed as value of TTL, ~~is changed~~ (fragment offset, MF, total length), (options, header length) may change.

Hence checksum is computed only header. If it were to be computed on data, every router has to do lot of computation.

→ However the protocol of the pckt will compute the checksum one the data part (on sender side)

Source IP & Destination IP

IP add		Description	used as	
NID	HID		Source IP	Dest IP
✓	✓	Valid IP add	✓	✓
✓	0's	Network id	X	X
✓	1's	Directed Broadcast	X	✓
1's	1's	Limited Broadcast	X	✓
1's	0's	Network Mask / Subnet mask	X	X
0's	✓	Host within same Nw	✓	✓
0's	0's	I don't have IP (A host requesting for IP	✓	X
127	✓	Loop back address	X	✓

Options:

→ This field is optional and it provide some options

(i) Record Route:

This is used to find the path followed by the pckt.

→ Every router adds its IP address if the pckt passes through it.

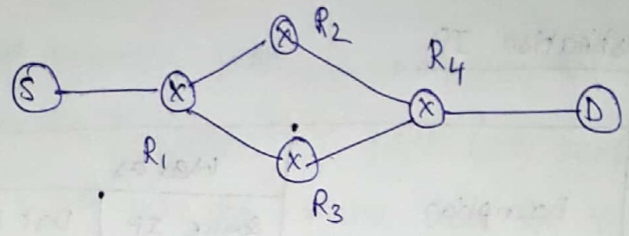
size of IP add is 4B and option field is 40B.

So max no of router address possible is 10, but we also need to mention what kind of option we are storing.

So in practice max no of router addresses possible to be stored = 9.

(ii) Source Routing:

Using this option sender can selected the path that has to be followed by the pckt. This is known as source routing.



Source routing is ~~2 types~~ said to be strict source routing if entire path to be followed is given. Otherwise it is called loose source routing.

strict source routing: Eg: R4 R2 R1

loose soure routing: Eg: R1

→ This option is used by N/w administrator to test if the path is working or not.

→ Also both the options record route, source routing are not allowed to be used by user. They are generally used by N/w administrator.

(iii) Padding:

If size of options added is not a multiple of 4 B, the extra bits are padded.

(iv) Time stamp: Using this option, we record in-time and out-time of a pckt a each router in the path. Using this we can know the delay at each router.

Total length:

This specifies length of ~~packet~~ datagram including header.

This is 16-bit field.

∴ Max possible length = $2^{16} - 1 = 65,535$ (Data + Header)

Min header size = 20B

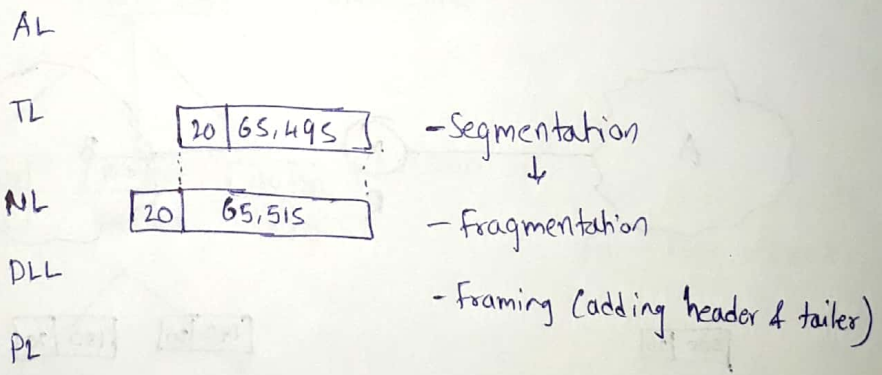
∴ Max possible size of data = 65,515

∴ Maximum segment size at transport layer = 65,515

Also TL has 20B header.

So max payload at TL = 65,495.

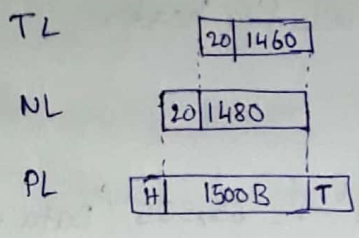
But App. layer may provide data of any size to TL. So TL does the job of segmentation



→ However if the DLL is ethernet then max frame length possible is 1500B. So for this reason NL does the job of Fragmentation.

→ In some cases DLL will be able to transmit frame of any size. If not, performing segmentation followed by fragmentation is waste of time. So it is better the TL finds least frame size / least datagram size and do segmentation accordingly

In this case segmentation done by TL is as shown below



Here only segmentation is done but not fragmentation

∴ Fragmentation is not done at sender side. Segmentation itself is done such that there will be no need of fragmentation.

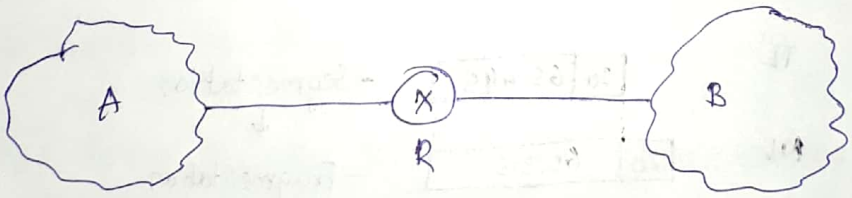
→ The maximum allowed size of frame in a Nlw is called ~~Maxi~~ Maximum Transmittable Unit (MTU).

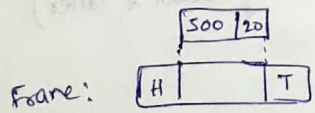
09/12/20

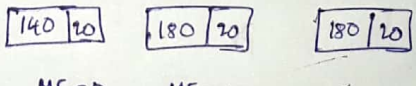
Fragmentation:

→ Fragmentation is done at routers but not sender.
MTU may be different from network to network.

Consider 2 networks A & B with $MTU_A = 520$ & $MTU_B = 200$

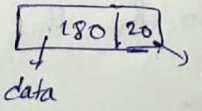






In Nlw B MTU is 200

i.e., structure is



→ So the router R has to perform fragmentation before it sends it to B

→ After fragmentation, the fragments are routed independently since it is datagram service. So the fragment may be received out of order.

→ Identification number of all the fragments of a datagram are same.

→ The fragment offset field is used to know the order of fragments.

→ For the last fragment MF is set to 0. ~~which means that~~ for remaining fragments MF is set to 1.

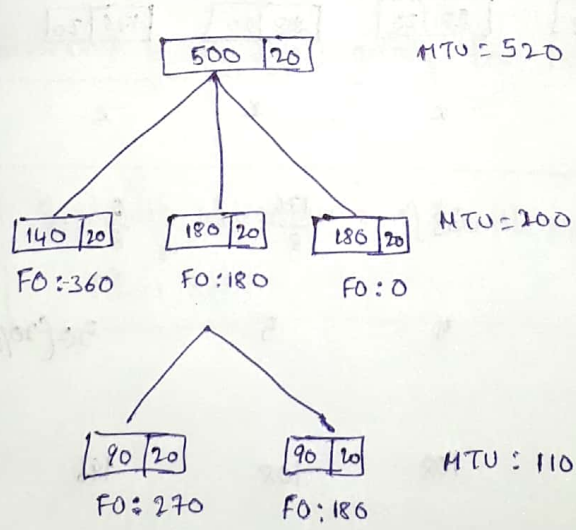
Assigning fragment offset to fragments:

Assigning fragment offset sequentially from 0 won't be scalable.

For example, consider a case where pckt P is fragmented into 3 pckts P_1, P_2, P_3 and fragment offsets are 0,1,2 respectively If we need to fragment P_2 again then we can't assign fragment offset to it. So we number fragments in a different way.

→ Fragment offset is given such that, fragment offset is equal to number of data bytes ahead of the fragment.

Eg:



∴ Final fragment offsets are (0, 180, 270, 360)

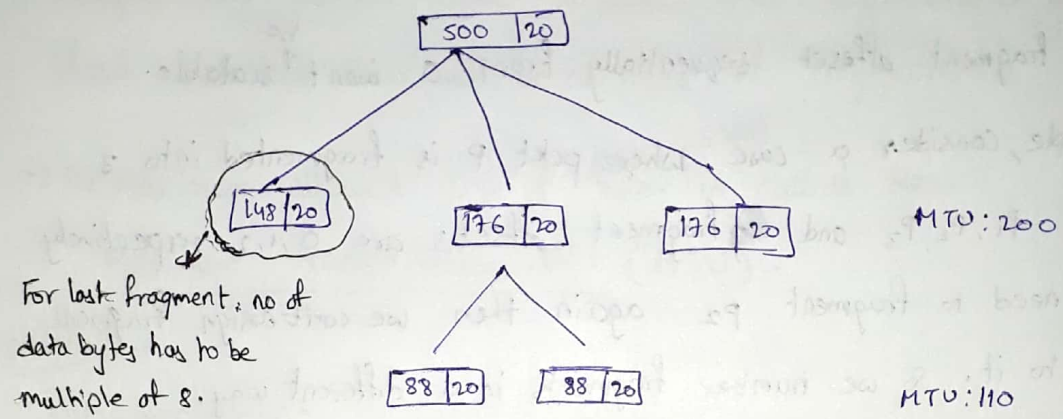
→ But here we have a problem.

Fragment offset field has 13 bits and total length field has 16 bits. In worst case if last fragment has 1 byte then fragment offset has to be $2^{16} - 20 - 1 = 65,514$

To meet this problem, we divide fragment offset with 8 ($\because \frac{2^{16}}{2^{13}} = 8$) and store it in the fragment offset field.

Now this brings a new restriction that original fragment offset has to be a multiple of 8. It means data bytes in each pckt should be a multiple of 8. (except last fragment

Now we perform fragmentation as shown below



Now the fragments are

	148 | 20	88 | 20	88 | 20	176 | 20
Identification field	x	x	x	x
fragment offset	44	33	$\frac{176}{8} = 22$	$\frac{0}{8} = 0$
Header length field	5	5	5	$5 \leftarrow (20/4)$
Total length	168	108	108	196
MF	0	1	1	1

→ Here we are totally transmitting 4 fragments.

So we transfer 4 headers.

actual data to transfer = 500 + 20 = 520

actual data that is transfer = 500 + 4 × 20 = 580

∴ 60 bytes is overhead from network layer.

Network layer overhead = (no of fragment − 1) × (Header length)

∴ Efficiency = $\frac{\text{useful data sent}}{\text{total data sent}} = \frac{500}{580}$

⇒ Bandwidth utilization = $\eta \times B$
(effec. Bandwidth)

Consider 3 networks

To find bandwidth utilization we take least bandwidth (bottleneck bandwidth) among the networks.

Bandwidth utilization = η * bottleneck Bandwidth (or) Effec. Bw

Note:

→ Fragmentation is done only at routers, but not at source.

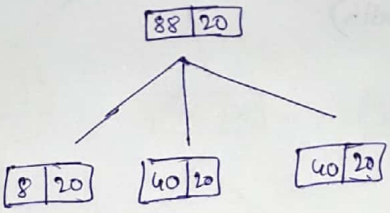
→ Reassembling is done only at destination, but not at router.
Reassembling is not done at routers because all fragments may not meet a the same router. Also even if they meet, there may be a need for fragmentation again.

→ For fragmenting or refragmenting, the given fragment and its header information is sufficient. No other extra information is needed.

Consider fragment [88|20] with

frag offset = 22; MF = 1; header length = 5; total length = 108; Id = 100;

Fragment the pckt if its passed through N/w with MTU = 60



100	100	100	ID (same as parent)
32	27	22	offset (offset of 1st fragment is same as that parent fragment)
1	1	1	MF (1 for all except last; MF of last fragment = MF of parent fragment)
5	5	5	HL
28	60	60	TL

Reassembling Algorithm:

MF	offset		
1	0	→ 1st fragment	Fragmentation done
1	not 0	→ intermediate fragment	
0	not 0	→ last fragment	
0	0	→ No fragmention is done.	No fragmentation

#Algorithm:

(i) Destination should identify that fragmentation is done (using MF, off)

(ii) Destination should identify all fragments belonging to the same datagram using identification field

(iii) Identify 1st fragment (MF = 1 & offset = 0)

(iv) Identify subsequent fragments using (TL, HL, offset)

i.e. if f_i is a fragment, then we need to identify f_{i+1}

$$\text{offset of } f_{i+1} = \left(\text{offset of } f_i + \frac{TL_i - HL * 4}{8}\right)$$

If $MF_i = 0$ then we don't find f_{i+1}

(v) Repeat (iv) until MF = 0

Eg: Consider fragment

20 | 176	20 | 88	20 | 88	20 | 148

offset	0	22	33	44
MF	1	1	1	0
HLF	5	5	5	5
TL	196	108	108	168
Id	100	100	100	100

Identify 1st fragment (MF = 1 & off = 0)

offset of 2nd fragment = $0 + \frac{196 - 5 \times 4}{8} = 22$

and 2nd fragment with offset 22 is present

offset of 3rd fragment

$$= 22 + \frac{108 - 5\times4}{8}$$

$$= 22 + 11$$

$$= 33 \quad \text{(exists)}$$

offset of 4th fragment

$$= 33 + \frac{108 - 5\times4}{8}$$

$$= 44$$

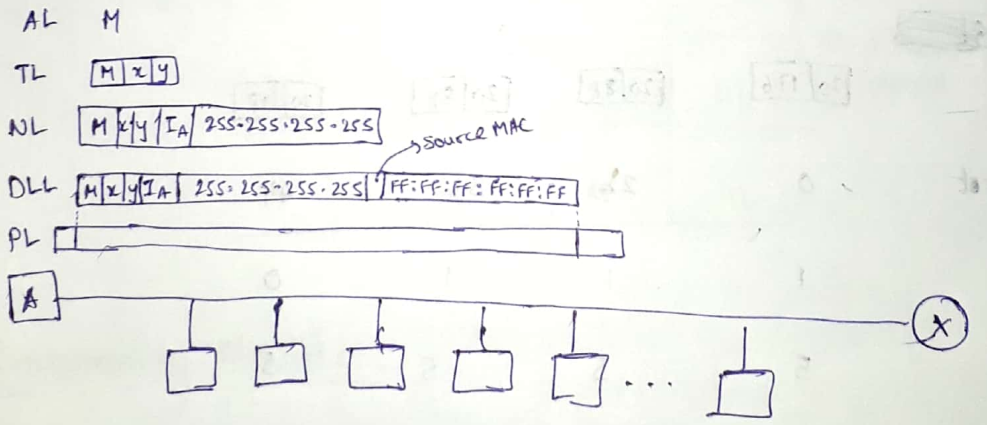
MF of 4th fragment = 0

∴ Reassembling is finished

Protocols & Concepts at Network Layer:

Implementation of broadcasting:

Limited Broadcasting:



For limited broadcasting destination IP = all 1's

destination MAC = all 1's

→ Every NIC is configured such that it accepts pckt if destination MAC is all 1's

→ when the pckt finally reaches Router R, (where the N/w ends) the router R will accepts and discards the pckt.

Directed Broadcasting:

Destination IP = [redacted] Nid + (all 1's in Hid)

Destination MAC = MAC of gateway router

→ Now this pckt is routed until it meets the router which is directly connected to the destination N/w.

→ this router forwards packet into the destination N/w.

Before forwarding it sets destination IP to all 1's

destination MAC to all 1's

→ Like in the limited broadcasting this pckt will also be discarded by the router of the network.

Address Resolution Protocol

→ ARP is used to find MAC address of a destination given its IP add.

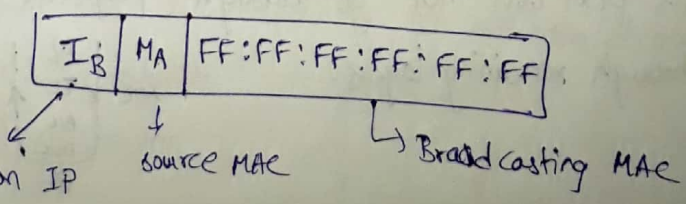
~~Receiver in same N/w~~

Receiver in same N/w:

→ On sender side, ARP can find whether receiver is on the same N/w or not using subnet mask of sender and destination IP.

→ If it is found to be on same N/w,

sender broadcasts an ARP request packet which contains destination IP and MAC of sender. whichever destination has the IP matched will send its mac address to A.



Receiver on different N/w:

→ In this case sender broadcasts ARP packet with dest IP = IP of router default to know the MAC of the router.

→ After this original data pckt is sent to router from which is sent to the destination

Note:

→ ARP request is broadcast message

→ ARP reply is unicast message.

→ ARP is used in below cases

(i) Host needs MAC of Host
(ii) Host " " " Router
(iii) Router " " " Host
(iv) Router " " " Router

→ If source and destinations are in diff networks then source can never know the MAC of destination and destination can never know the MAC of source.

Special address 127:

→ The NID 127 of class A is reserved for special purpose. It is used for testing self connectivity.

→ If a system needs to check if its NIC is working properly or not then it can set destination add as 127.-.-.-

→ This pckt will not go through physical layer but goes through remaining layers.

send ↓ AL, TL, NL, DLL, PL ↑ receive.

→ However the addresses 127.0.0.0 & 127.255.255.255 should not be used.

→ Usage: ping 127.—.—.—

If NIC is not working then we get timeout message.

→ Since pckt comes back to same system this address is called loop back address.

→ one more application is that

if we are implementing server on a host then to test if the server is working or not we also put a client processes on the same host and we can start communication blw them.

Here the processes are identified using port numbers

Message	source port	dest port	source IP	loop back add

Eg: let IP add is I_A.

port num of server is 80

port num of client is 90

M	80	90	I_A	127.0.0.1

RARP (Reverse ARP)

→ This is used know IP add of a host given MAC add of the host.

→ when a system is turned of or newly brought into Nlw, it doesn't know its IP address.

→ RARP server contains MAC add & corresponding IP addresses of the systems. The below shown table is maintained at RARP server

MAC	IP
M_1	IP_1
:	:
:	:

→ Now the host sends RARP request pckt with source IP = 0.0.0.0 and dest IP = 255.255.255.255

→ Source MAC = MAC of the host

destination MAC = All 1's (i.e. Broadcast)

→ Looking at IP = 0.0.0.0 remaining hosts in N/w discards this, but RARP server replies the host with the host's IP.

Disadvantages:

→ But for this protocol to work, there must be RARP server at every network (∵ we are broadcasting the RARP req pckt) Due to this the data is distributed access different RARP servers and if there is same data then updations has to be done at every server.

→ The IP addresses assigned are static. Even if some hosts are not active, the IP addresses are kept assigned.

∴ no of IP add ≥ no of hosts

However RARP is not currently under use.

10/12/20

BootP: (Bootstrap Protocol)

→ This also used to know IP address of a host given its MAC.

→ Here also we have a BootP server. This server will have a static table of MAC & IP addresses

→ Here also pckt is sent in the same way requesting for IP address.

→ The main difference b/w RARP & BOOT is that RARP server operates at network layer (and may not contain other higher layer) where as BOOTP server operates at application layer

→ Relay agent is used to obtain IP addresses if Bootp server is not present in the network.

If a station needs to find its IP add, it broadcasts the request pckt. Since Bootp server is not in the N/w relay agent accepts this message and unicasts this to bootp server on the other N/w. Then Bootp server replies to relay agent and relay agent replies to the station.

Adv:

→ We can have only one bootp server and hence data is centralized

Disadv: Data is static

i.e., Mapping b/w MAC & IP is static

DHCP (Dynamic Host Configuration Protocol)

→ Its working is same as Bootp. Here we have a DHCP server. Here also we have relay agents.

→ The main difference is that mapping b/w MAC & IP has 2 parts

- Static → Given to stations which runs forever
- Dynamic → Given to station that runs only for some time.

MAC	IP
:	:

Static

MAC	IP	Lease time
M_1	I_1	10 min
:	:	:

Dynamic

→ For dynamically assigning,

a pool of IP addresses is maintained. whenever a host requests, an IP address is assigned and an entry is created in the table. After the lease time finishes, the station has to renew request if it needs IP for some more time. Otherwise, the IP address will be kept back in the pool and the entry is deleted.

Adv: (i) only one DHCP Server is required.

(ii) Dynamic table

∴ no of IP add = no of hosts online

(iii) DHCP is also compatible with BootP request.

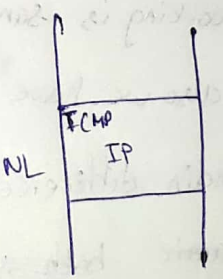
So for this purpose port numbers of DHCP & BootP are same.

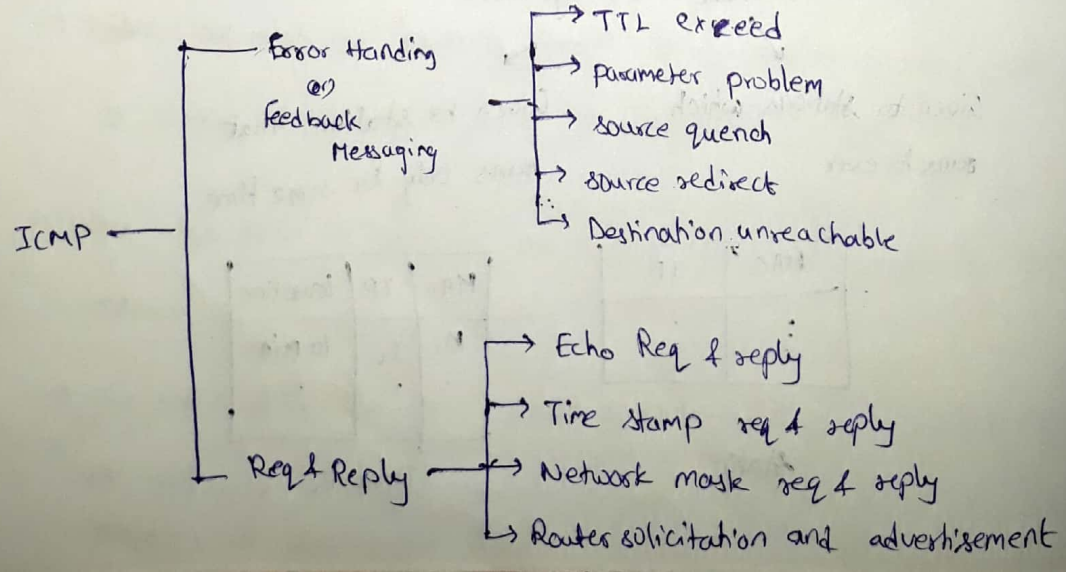
ICMP (Internet Control Message Protocol)

→ Icmp pckt is sent to sender if the sent packet is discared at router or destination.

→ Icmp pckt is also use for req & reply.

Here ICMP pckt is sent as request and an ICMP pckt is obtained as reply





→ when IP pckt is discarded, an icmp pckt is sent.

But if this generated icmp packet is discarded again, then no ICMP packet will be generated again. and in this case sender will not know that its pckt is discarded.

∴ (IP + ICMP) is still not reliable. However TL (TCP) is reliable.

ICMP Error Messages:

TTL exceed:

when TTL exceeds router discards the pcks and sends icmp packet.

Here icmp type = TTL exceed

Source Quench:

when a source (system or router) sends lot of pckts to a router and if the receiving router can't handle all the data, then source quench icmp pckt will be sent to source.

Seeing this packet source will reduce the transmission

Parameter Problem

Assume we are using strict source routing, if a router given in the path is found to be down at the time of routing then this ICMP message will sent.

Also this message is sent if the packet sent is corrupted

Destination Unreachable

Destination is down

This router tries to obtain MAC of D. But D doesn't respond.

So R_2 sends destination host unreachable message

Destination Unreachable

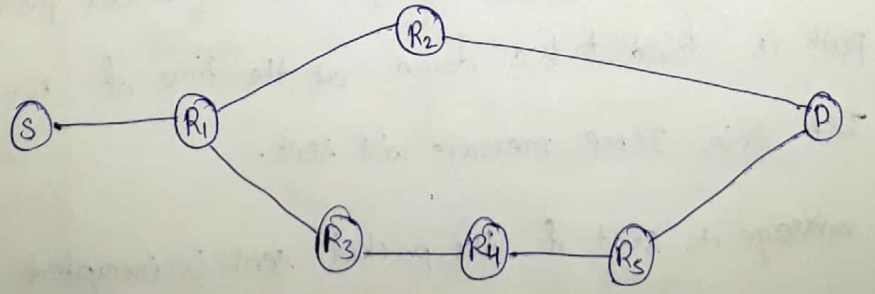
- Destination Host Unreachable → Destination Down
- Destination Port Unreachable → Destination is present but it does not have process with given port.

→ Also destination unreachable may occurs when DF = 1.

In this ICMP message, it is specified that fragmentation is required and DF = 1 and MTU of the network.

Source Redirect:

Consider



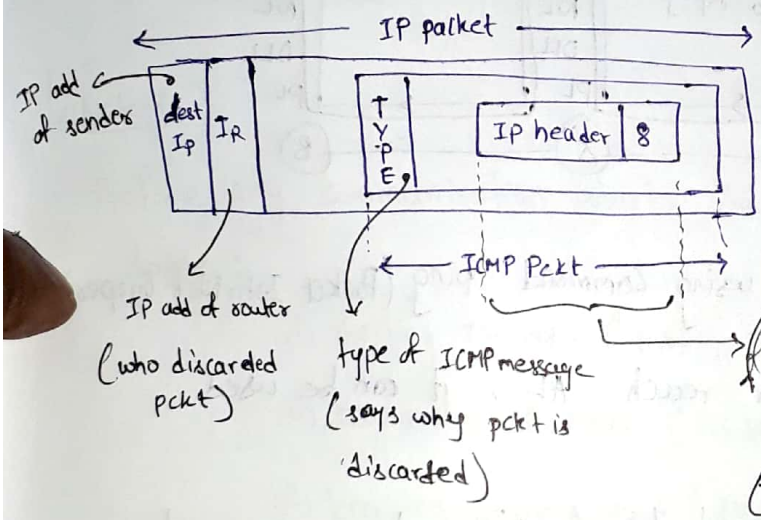
Here the best path is S - R_1 - R_2 - D

But consider R_1 has forwarded pckt to R_3.

Incase if R_3 knows that there is better path, then R_3 sends icmp ~~no~~ pckt to R_1 (warning message) saying that there is a better path.

However R_3 doesn't discard the pckt. and from next time onwards R_1 forwards to R_2

ICMP message format:



(This lets the sender know which pckt is discarded.
Also the 1st 8 bytes are send so that if it is TCP protocol then port numbers will be known and the ~~ip~~ source process will be known that the pckt is discarded)

→ ICMP is generated only if the discarded pckt is either TCP or UDP.

→ If the datagram has several fragments, ~~then etc~~ and if it is discarded then ICMP is generated only for 1st ~~pckt~~ fragment.

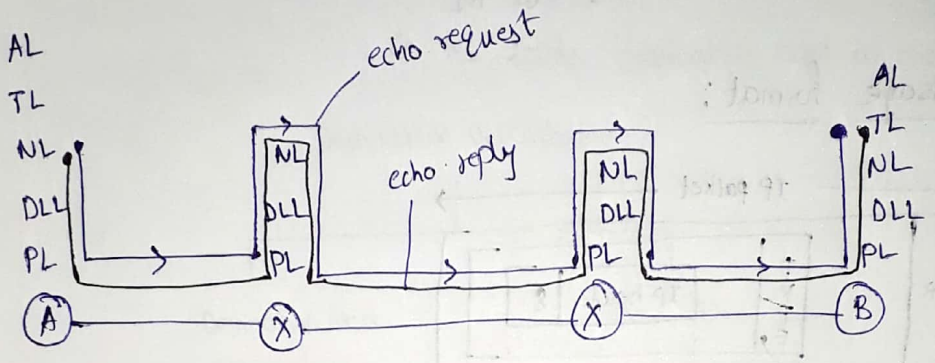
Even if 1st fragment is transmitted correctly and if 2nd fragment is discarded, then also ICMP will not be generated.

ICMP request and reply:

Echo request & reply:

It is used to see if the destination is switched ~~present (online)~~ or not.

The ICMP pckt never meets Application layer & transport layer



→ This service is obtained using command ping (Packet InterNet Groper).

→ Since these pckts never reach AL, it can be used for attacks.
For example we can send lot of pings to a server and keep it busy with echo replies and thus deny other services..

Router Solicitation:

→ This ICMP pckt is sent by a station to know ~~the~~ who is the default router.

→ So this ~~pc~~ ICMP pckt (router solicitation ICMP pckt) is broadcasted and all the routers in the N/w will reply.

→ One among these is used as default.

Router advertisement:

Whenever a new router is brought into network, this router broadcasts router advertisement ICMP pckts saying that you can use me as default router.

Network Mask Req & Reply:

This ICMP msg is sent by a host to know its N/w mask or subnet mask.

Note:

Before any communication starts the below 3 steps has to be done:

(i) Getting IP add (RARP, BootP, DHCP)

(ii) known default router (Router solicitation / Router advertisement)

(iii) knowing Network mask (Network mask Req & reply)

Time stamp req & reply

→ Two stations which are communicating may be in different time zones. So we need to synchronize the communication.

→ Time stamp req & reply is used to know the time at destination and destination uses this to know the time at sender.

→ However this is highly unreliable. and there are better ways to synchronize time.

Trace Route:

→ This is an application of ICMP.

→ This is used to know the path (routers involved in the path) of communication

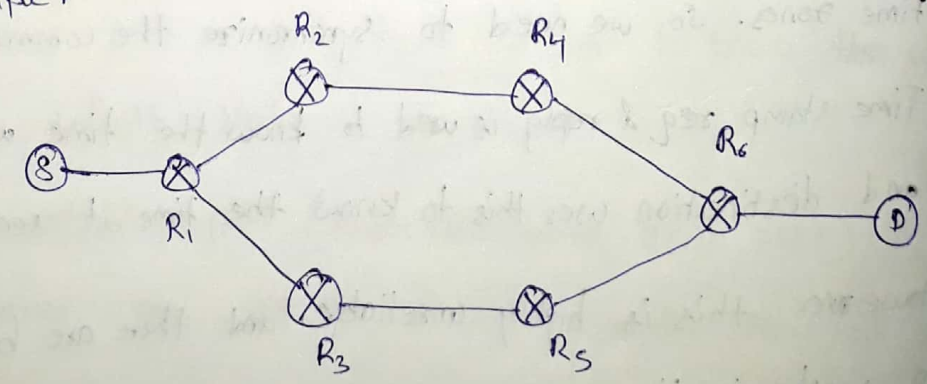
→ The option record route is used to know the path ~~by~~ for destination.
But trace route is used to know the path for sender.

Implementation:

→ Here we send a pckt with TTL = 1.
At ~~router~~ first router TTL becomes 0 and we can know the first router. (∵ router sends TTL exceeded ICMP)
Similarly we do ~~that~~ the same with TTL=2 and know the 2nd router in the path.
We continue this process until we get ICMP pckt from the destination. (To get ICMP pckt from destination we intentionally ~~intentionally~~ put unreachable port number in the pckt)

→ However in some ~~cases~~ case route obtained may be wrong.

For example.



TTL = 1 ⇒ R_1
TTL = 2 ⇒ R_2
TTL = 3 ⇒ R_5
TTL = 4 ⇒ R_6
TTL = 5 ⇒ D

∴ $R_1 - R_2 - R_5 - R_6 - D$ is not a valid path
(But this is generally least probable case cuz routes don't change that fast)

PMTUD (Path MTU Discovery):

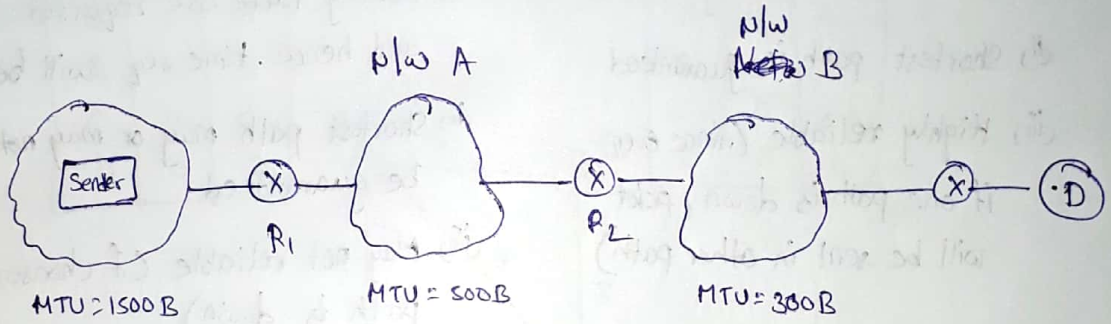
→ It is another application of ICMP

→ It is used to know MTU along the path.

→ Thus, using this, we can know least MTU in the path and we can send datagram of that size and thus avoid overhead of fragmentation.

Implementation:

Consider below scenario



→ Initially we send pckt with maximum allowed size in sender network with DF = 1.

Once it reaches N/w A, destination unreachable ICMP pack is sent by Router R_1 to sender. After this sender reduces the size and sends again.

This process is continued until pckt reach destination.

(To know that pckt has reached destination we put invalid port num in destination port and thus destination sends destination port unreachable message to the sender.)

→ However here also there is a chance that path followed for each packet may be different.

Routing

→ Routing is process of preparing routing table and this routing table is used for switching

→ If there is no routing then we use flooding.

→ Flooding is process of sending obtained pckt to all the links except to the link the packet has arrived on.

Advantages of flooding:

(i) No need of routing

(ii) Shortest path is guaranteed

(iii) Highly reliable (since even if one path is down, pckt will be sent in other path)

Disadv of flooding:

(i) Many duplicate pckts are received at destination

(ii) More traffic

Disadvantages of Routing

(i) Routing table are required and hence time req will be more.

(ii) Shortest path may or may not be guaranteed

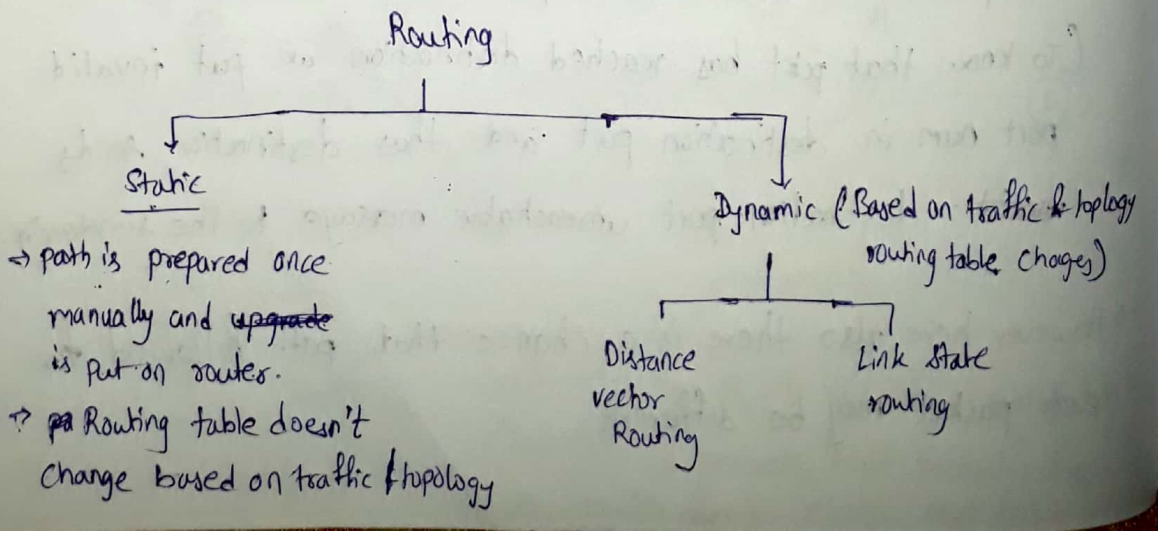
(iii) May not reliable (if choosen path is down)

Advantage of Routing

(i) No duplicate pckts will be received.

(ii) Low traffic.

Types of Routing Algorithms:



⇒ path is prepared once manually and is put on router.

⇒ Routing table doesn't change based on traffic & topology

Distance Vector Routing (DVR)

Every router has a routing table with below structure :

Destination	Distance	Next hop
⋮	⋮	⋮

Distance vector

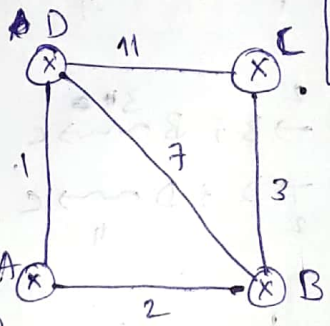
If a router is not directly connected then distance is ∞.

Step 1)

Eg: Based on the local information every router builds routing table as shown below

Des	Dis	NH
A	1	A
B	7	B
C	11	C
D	0	D

Des	Dis	NH
A	∞	-
B	3	B
C	0	C
D	11	D



Des	Dis	NH
A	0	A
B	2	B
C	∞	-
D	1	D

Des	Dis	NH
A	2	A
B	0	B
C	3	C
D	7	D

Step 2 : Every router sends its distance vector (DV) to adjacent routers.

At A : DV received from B, D

At B : DV received from A, C, D

At C : DV received from B, D

At D : DV received from A, B, C

All this sharing will happen at once in parallel (∵ the channels are full duplex)

Using these distance vectors of neighbouring routers, each each router builds new routing table.

At A:

From B

2
0
3
7

From D

1
7
11
0

New routing table. (while building new routing table old table shouldn't be used)

A	0	A
B	2	B
C	5	B
D	1	D

$$A \rightsquigarrow B = \min \begin{cases} \overset{2}{A \to B} + \overset{0}{B \rightsquigarrow B} \\ \underset{1}{A \to D} + \underset{7}{D \rightsquigarrow B} \end{cases}$$

$\to$: edge

$\rightsquigarrow$: path

$$A \rightsquigarrow C = \min \begin{cases} \overset{2}{A \to B} + \overset{3}{B \rightsquigarrow C} \\ \underset{2}{A \to D} + \underset{11}{D \rightsquigarrow C} \end{cases}$$

$$A \rightsquigarrow D = \min \begin{cases} \overset{1}{A \to D} + \overset{0}{D \rightsquigarrow D} \\ \underset{2}{A \to B} + \underset{7}{B \rightsquigarrow D} \end{cases}$$

Shortcut

B:

A	B	C	D
2	0	3	7

↓ AB (i.e., 2)

4	2	5	9

D:

A	B	C	D
1	7	11	0

↓ AD (i.e., 1)

2	8	12	1

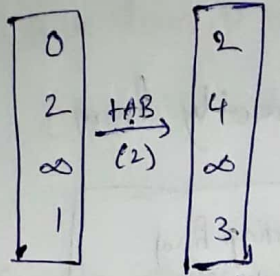
pick Minimum

A	0	A
B	2	B
C	5	B
D	1	D

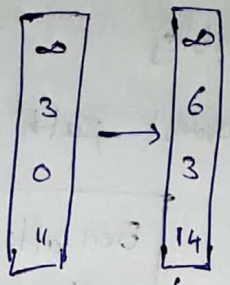
(since it A to A it is 0 but not min of above two)

At B :

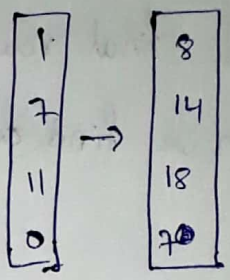
From A



From C



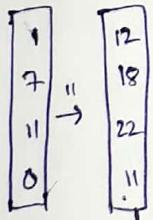
From D



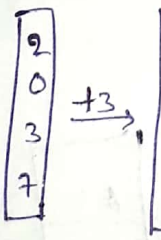
A	2	A
B	0	B
C	3	C
D	3	A

At C :

from D



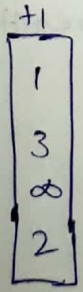
from B



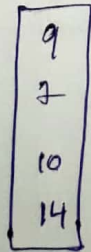
A	5	B
B	3	B
C	0	C
D	10	B

At D :

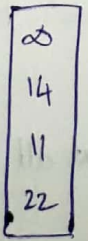
A



B



C



A	1	A
B	3	A
C	10	B
D	0	D

At the end of 2nd round we get shortest paths containing atmost 2 edges.

Since we have 4 routers, we need to go for 3rd round also for the final routing tables.

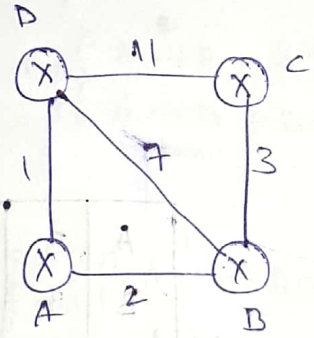
(But in exam we find out shortest path directly from the graph)

So the final routing tables are

Even after getting final routing table, exchanging DVs will continue forever periodically.

A	1	A
B	3	A
C	6	A
D	0	D

A	5	B
B	3	B
C	0	C
D	6	B



A	0	A
B	2	B
C	5	B
D	1	D

A	2	A
B	0	B
C	3	C
D	2	A

→ In routing table of D, C is not present as next hop for any entry. Similarly in routing table of C, D is not present as next hop for any entry

This means the link DC is not at all used

Similarly the link BD is also not used.

Shortcut for finding unused links:

A link PQ will be unused if there is ~~a~~ better path from P to Q other than the direct path b/w P & Q.

(Try this method for previous example)

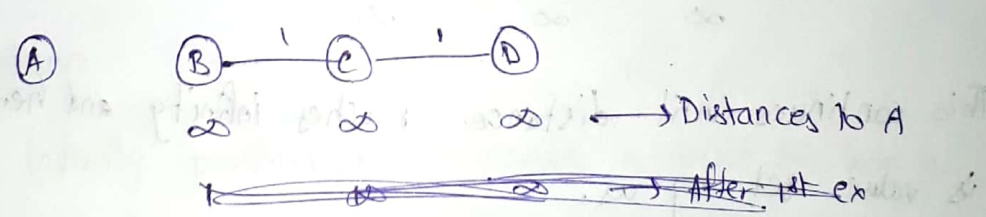
Count to infinity problem

→ This is a disadvantage of DVR

→ the problem is bad news ~~spear~~ spreads slow
good news ~~spe~~ spreads fast.

Good news (A new router is on):

Assume Router A is initially ~~dow~~ down and now it is up



∞ ∞ ∞ → Distances to A

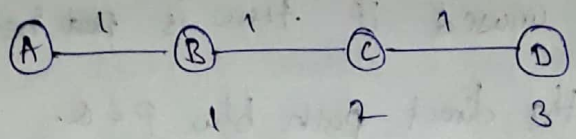
1 ∞ ∞ → After 1st exchange

1 2 ∞ → " 2nd "

1 2 3 → " 3rd "

Thus good news ~~spe~~ spreads fast.

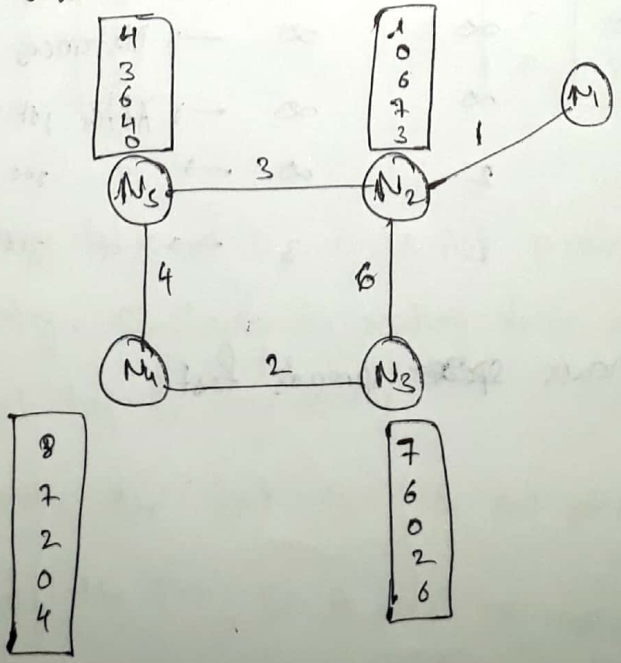
Bad news: (when a router is down)



A	B	C	D	
	1	2	3	
	∞	2	3	
	3	2	3	(Here B thinks that C has some other path to A)
	3	4	3	
	5	4	5	
	5	6	5	
	7	6	7	
	7	8	7	
	9	8	9	
	⋮	⋮	⋮	
	∞	∞	∞	

This continues until distance reaches infinity and here infinity is value set by us.

Q) Consider below network and ~~redi~~ ~~Here~~ their distance vectors



If somehow distance btw N_2 & N_3 fall to 2, then what will the new routing table at N_3 after one exachange.

eg:

From N_2

1	3
0	2
6	8
7	9
3	5

+2

From N_3

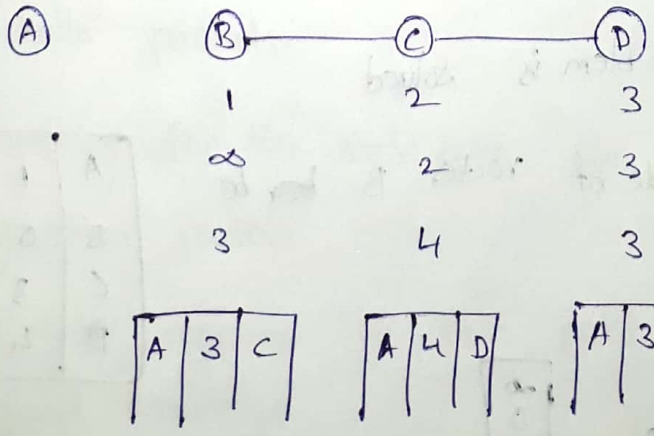
8	10
7	9
2	4
0	2
4	6

+2

3
2
0
2
5

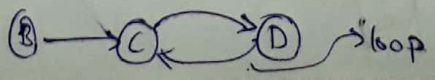
~~Split Horizon~~

→ Count to infinity problem also causes a pckt to loop in the network. For example consider



Now consider B wants to send a pckt to A.

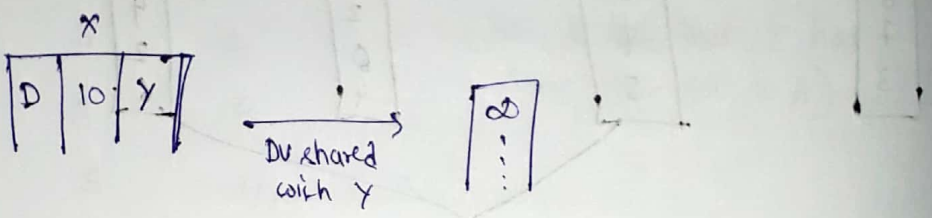
B sends it to C; C sends to D; D sends to C;
C send to D; D sends to C; ....



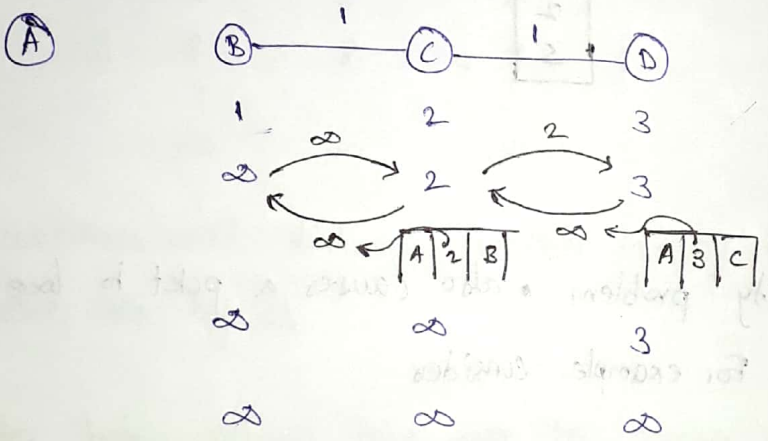
Split Horizon:

→ This is solution for count to infinity problem.

→ Split Horizon says if a router X is depending on router Y (i.e., next hop entry is Y) to go to some destination D, then when X sends DV to Y it should send ∞ in the entry corresponding to D.



Now consider the previous example



An thus the problem is solved

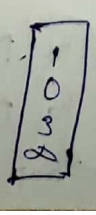
Eg: Let routing table of router B be

A	1	A
B	0	B
C	3	A
D	4	D

Then

DV sent to A

DV sent to D



Disadv without split horizon:

(i) Count to infinity

(ii) loops

(iii) Convergence is slow

i.e., new updates or modifications are not known quickly to routers that are far away from the point where the modification is done

Disadv with split Horizon

→ Convergence is slow.

Link State Routing (LSR):

→ DVR is based on local knowledge where as LSR is based on global knowledge.

→ Here every station prepare link state pckts.

A Link state pckt of a router contain distances of its neighbouring routers.

The neighbours and the corresponding distances are found using 'hello' packets.

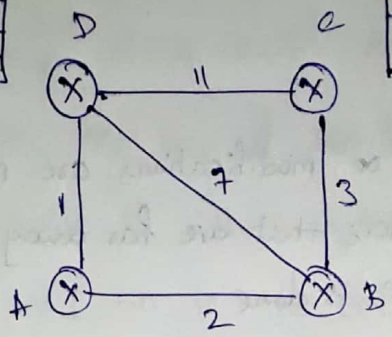
→ After creating the link state pckt, the pckt is flooded to every other router.

So every router receives link state pckt of every other router i.e., global knowledge

→ From this information, a router can build entire graph of the network.

D	
Seq	
TTL	
D	11
B	7
A	1

C	
Seq	
TTL	
D	11
B	3



A	
Seq	
TTL	
B	2
D	1

B	
Seq	
TTL	
A	2
D	7
C	3

→ Now each router applies Dijkstra's algorithm and knows shortest path ~~dtg~~ to each router.

∴ Here convergence is faster.

→ Need of seq num:

Since flooding is used there is a chance that a router receives same pckt twice. ~~If~~ ~~ob~~ Assume by the time 2nd copy arrives, the router has already obtained latest packet (from same router). But still this accepts & uses old pckt. To over come this we use sequence number field.

Every router maintains latest seq num of each router's link state pckt

Router	latest seq num	life time
R_1	10	60 min
R_2	9	10 min
⋮	⋮	⋮

Thus using seq num old pckts are discarded and are no more flooded and thus seq num field even helps in decrease flooding.

→ Need of TTL field:

Since flooding is being used, there is a chance the pckt may loop infinitely in the network. To avoid this TTL is used.

→ Need of lifetime field:

~~Assume a case where seq num is 15.~~

Assume a case where a link state packet with seq num 15 (00001111) is being transmitted and say due to a single bit error it is changed to 143 (10001111).

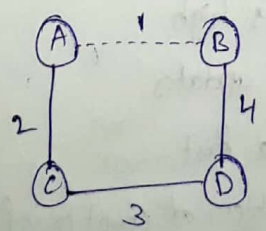
Now all the link state pckts upto seq num 143 will be discarded and ≥144 will be accepted.

To overcome this problem every entry has lifetime/validity after which we accept any link state pckt that arrives.

Problems in LSR:

The problems in LSR are ~~not persi~~ transient (non-persistant)

i) Black hole problem:



If the link AB is down then A & B will know after this some duration of time.

So any packets sent by A to B will not reach B during this time.

After some time A will get to know that the link is down and will send pckts in different path

(ii) Looping:

Here looping is present but for only some duration.

→ Consider the previous case where the link AB is down.

After A knows the link is down, A chooses the path A-C-D-B.

But C knows that the link is down after some duration. So for C the path to B is still C-A-B.

→ So: if C needs to send a pckt to B then it sends to A and A sends it to C will send it back to A again and this process continues until C discovers that the link AB is down.

DVR	LSR
1) used in 1980's	1) used in 1990's
2) BW req is less	2) BW req is more
3) Based on local knowledge	3) Based on global knowledge
4) Similar to bellmanford	4) Dijkstra's algo
5) Traffic is less	5) Traffic is high
6) Periodic updates	6) periodic updates
7) Convergence is slow	7) convergence fast
8) Count to inf is present	8) No problem of count to inf
9) persistent looping	9) Transient looping
10) Implementation: RIP	10) Implementation: OSPF

Routing Information Protocol (RIP):

→ This is implementation of DVR.

→ Here metric is hop count

∴ Every edge has weight 1

→ Here ∞ is equivalent to 16

→ Computation is simple (compared to OSPF)

Open Shortest Path First (OSPF):

→ Computation is complex.

→ Avoiding traffic of flooding:

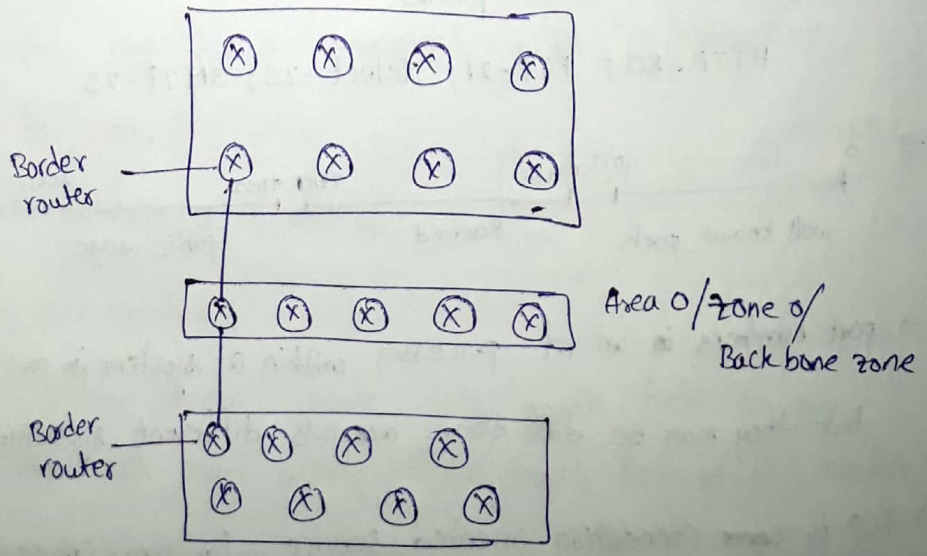
→ Here routers are divided into several regions and each region has a border router.

→ All the border routers are connected to area 0/zone 0/ back bone

→ A border router summarizes the flooded information and sends it to area 0 and from there this pckt will be forwarded to other regions.

Thus using this process traffic due to flooding is reduced.

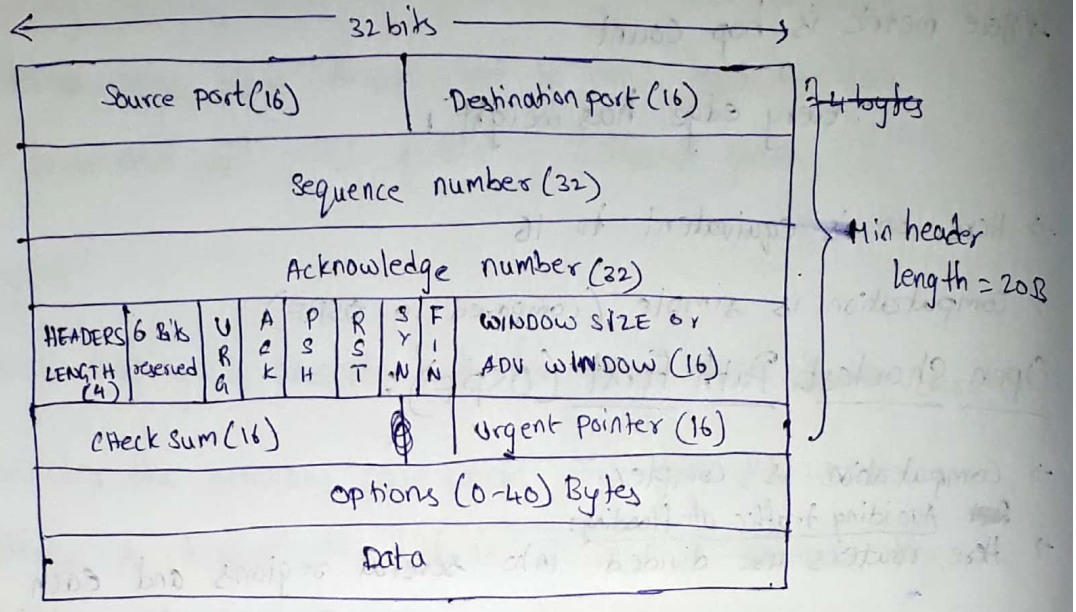
→ Metric is weight set by network administrator.



12/12/20

TCP

TCP Header:



Max header length = 20 + 40 = 60B

→ TCP is a transport layer protocol that ensures end to end communication.

→ So, TCP is an end to end protocal.

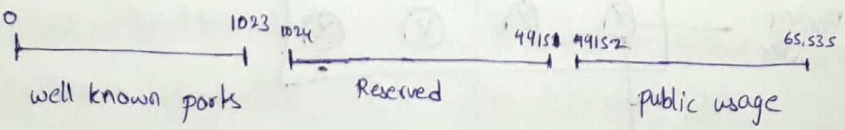
Source port & Destination port

→ Using port numbers TCP does the job multiplexing & demultiplexing.

→ port is a 16-bit number.

→ Example of well known ports

HTTP - 80; FTP - 21; Telnet - 23; SMTP - 25

→ 

→ port numbers of processes within a system are unique but they may be same acoross different systems.

→ TCP is connection oriented service. For any connection oriented service, resources (Bw, buffers etc.) are reserved.

→ Identifying a connection uniquely is not possible using only port or only IP. ~~is not possible~~.

→ So we use combination of IP add & port num called Socket to identify a connection uniquely.

∴ Socket = IP + port

∴ Socket is 48 bits

Sequence Number:

→ TCP is a byte stream protocol

i.e., Every Byte is counted

IP is packet stream protocol

It means logically we assume every byte is numbered.

→ The seq num field contains number given to the first byte in the segment

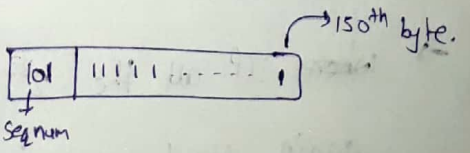
Acknowledge Number:

→ It is byte number of next expected byte.

This number is sent as ack by receiver.

For example consider a segement with 50 bytes and seq num is 101.

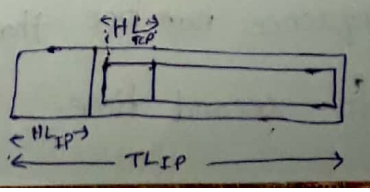
Then ack number is 151



→ How to calculate ack number?

TCP header doesn't have total length field. So we calculate ack num using total length field of IP header

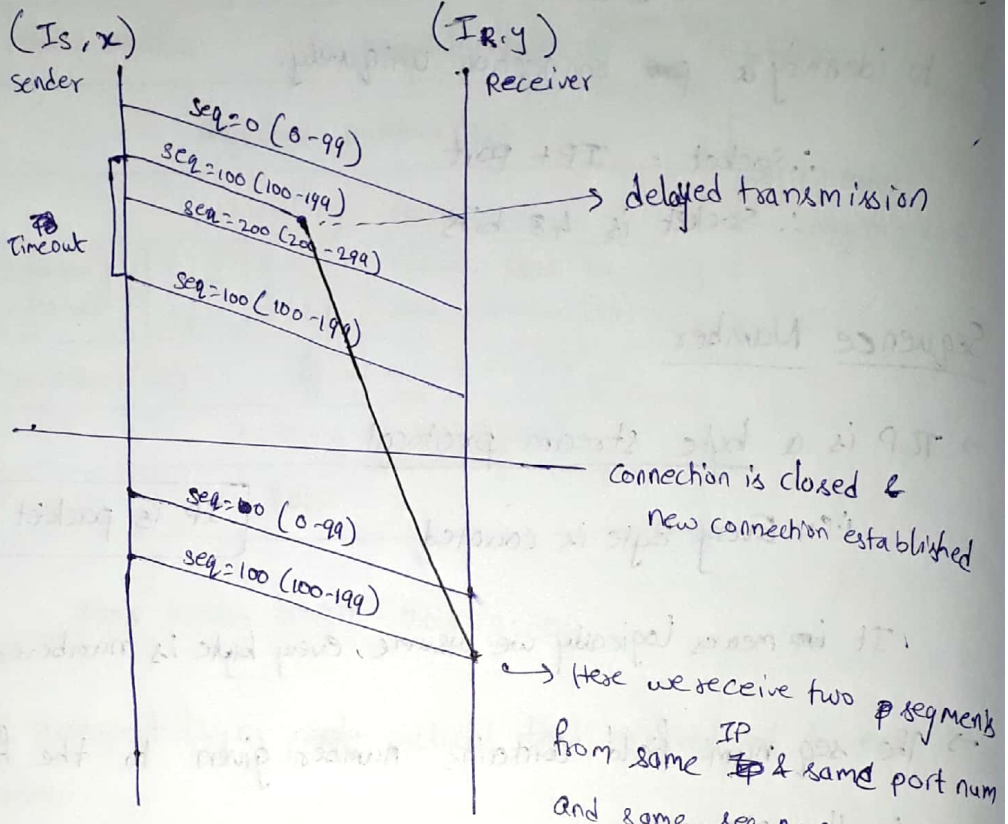
$TL_{IP} - HL_{IP} - HI_{TCP}$ gives no of bytes in segment and from that we calculate ack number.



Note:

Starting a seq num always from 0 has a problem.

Consider the below case.



→ To overcome this, TCP uses randomly generated numbers as starting sequence numbers.

However here also there is a chance of conflict but it is very negligable.

→ If all the 2^{32} seq numbers are assigned then we again start numbering from 0. This is known as wrapping around.

Wrap around time: The amount of time take to use the sequence number that we started with (need not to be 0) for second time.

wrap around time depends on bandwidth.

Eg: If BW = 1MBPS

then

10^6 seq num ——— 1 sec
are used

$\Rightarrow 2^{32}$ seq num ——— $\frac{2^{32}}{10^6}$ sec

i.e., $\cong$ 4296 sec

Since wrap around time is this big, conflicts won't occur.

Life time: Maximum amount of time for which a packet can be in network.

Generally it is 3 min.

→ For no conflicts to occur,

wrap around time > Life time

For example consider, let BW = 1GBPS

$\Rightarrow$ wrap around time = $\frac{2^{32}}{10^9} \cong 4.29$ sec

Here wrap around time < lifetime

The problem for this soln is to increase bits in seq num field.

∴ given life time = 180 sec & BW = 1GBPS we find no of seq num req.

1 s ⟶ 1 G seq num

180 s ⟶ 180 G seq num

=) no of seq num req = 180 Gi

=) no of bits in seq num field = $\lceil \log_2 180 \times 10^9 \rceil$ = 42 bits

However we have only 32 bits in the seq num field.

→ So the extra bits seq are kept in the options field.

These bits in the options field is called time stamp

Header Length:

→ It is a 4-bit field

But max header length possible is 60

∴ Content in 4-bit field = $\frac{\text{original header length}}{4}$

Q) Consider IP datagram with TL = 1000 Δ HL = 5 and consider segmend within IP datagram with HL = 5 and seq = 100. Find acknowledgment number.

Sol:

~~no size of data bytes in segment~~

Data size of IP datagram = 1000 − 5×4

= 980

Data size of TCP segment = 980 − 5×4

= 960

∴ seq num of last byte in segment

= 100 + 960 − 1 = 1059

=) ack num = 100 + 960 = 1060

Flags:

→ SYN - Synchronization Flag
Ack - Acknowledgement flag

→ TCP is connection oriented and it has 3 phases

(i) Connection Establishement
(ii) Data transfer
(iii) Connection Termination

SYN: This flag is set, when we need to tell the other host that our starting sequence number is so and so. The packet with this information is called SYN packet.

SYN pckt consumes a seq num.

→ SYN is used to synchronize the sequence numbers.

Acknowledgement Flag:

In TCP, Acknowledgements are sent in two ways

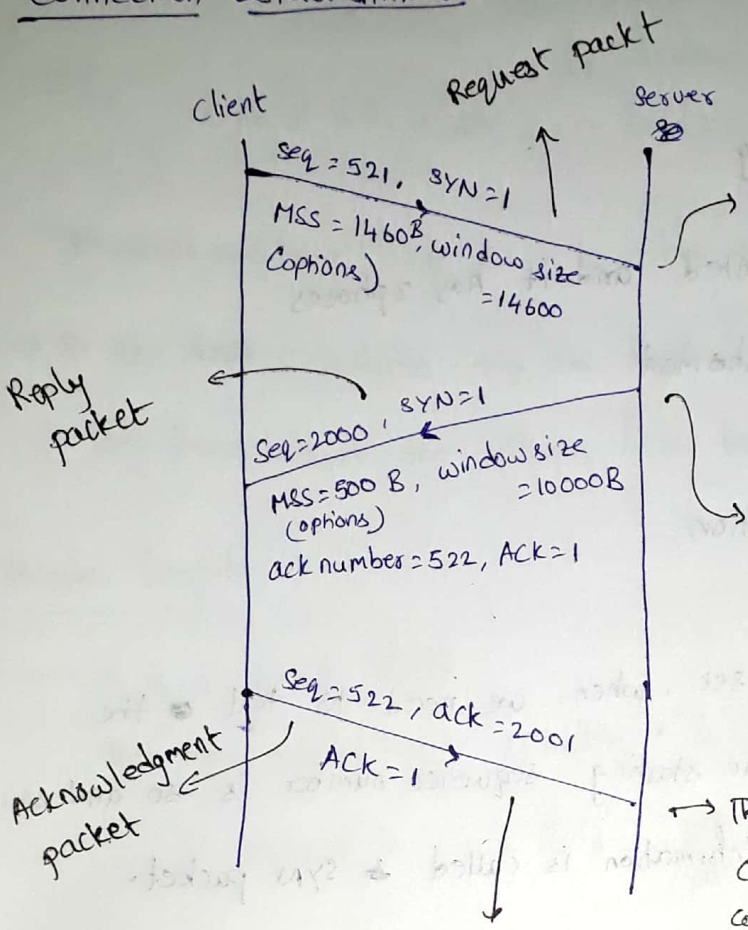
(i) piggy backing
(ii) pure acknowledgement

For both acknowledgements the ack number is put in ack number field and ACK flag is set to 1. If Ack flag is 0 then no matter what value is present in ack number field, the pckt is considered not to have any acknowledgent.

When sending pure acknowledgement only header is sent.

→ when pure ack is sent, no seq num is consumed.

Connection Establishment:



MSS: Maximum Segment Size is maximum DATA size in segment (i.e., excluding headers)

→ The seq num 521 from client & seq num 2000 from server are generated randomly

→ Both server and client knows MSS of each other and thus sends datagrams of the size which is minimum of the 2 MSS's

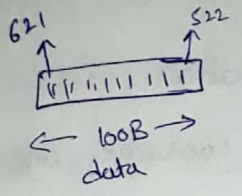
→ This connection establishment is called 3-way handshake.

→ As soon as connection is established resources (BW, buffers etc.) are allocated & reserved at both client & server.

Note:

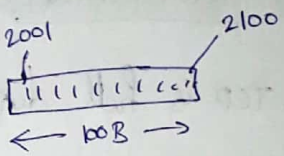
→ TCP is a full duplex connection.

Data Transfer:



522 is not consumed by Ack. So we start from 522.

Even though ack was sent we send it again since the field is anyhow available. We do this because incase if the previous ack is lost then this would be helpful. So whenever there is no new data received we send ack for previous data only.



seq = 522, Ack = 1
ack num = 2001

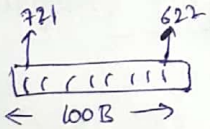
seq = 2001, ACK = 1
ack num = 622

If data is not ready then we have to send pure ack because there is a chance that timeout occurs at receiver and it resends the data

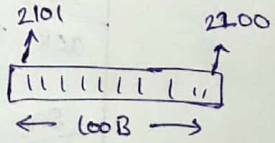
Pure Ack

seq = 622, Ack = 1
ack num = 2101

Observe the seq num & ack num in the packets that are sent



seq = 622, Ack = 1
ack num = 2101



seq = 2101, ACK = 1
ack = 722

Pure ack

seq = 722, Ack = 1
ack num = 2201

Note:

SYN	ACK	
1	0	→ 1st segment / Request packet
1	1	→ 2nd segment / Reply packet
0	1	→ Ack is present in header
0	0	→ Invalid Combination

FIN:

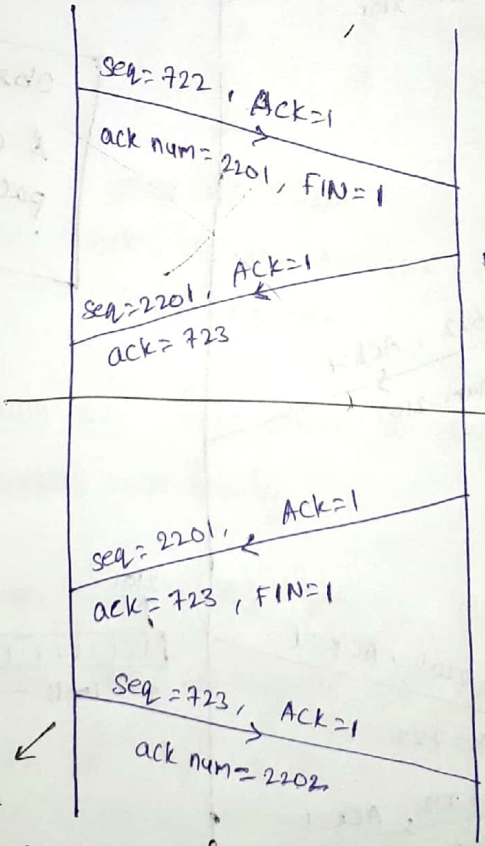
When this flag is set, it means that this is last segment and the sender wants to close the connection.

i.e., FIN flag is ~~request~~ for connection termination

→ A FIN ~~segment~~ packet (i.e., pckt with FIN=1) consumes one sequence number.

Connection Termination:

→ Since TCP is full duplex we need to terminate both connections (one from client to server & other from server to client)



→ Thus connection termination requires 4 packets in the worst case. However by setting FIN in the 2nd pckt itself, we can terminate the connection ~~is~~ with 3 pckts.

PSH (push flag):

Generally TL waits until it receives large enough data (i.e., MSS) and sends all the data at once. Until all the data is obtained from application layer, the data is stored in buffers. This is done for efficiency.

→ But for some interactive application this technique won't be good.

→ In this case AL sends PSH=1 to TL saying that transmit the data as soon as it is received.

Also on receive side when PSH=1, data is sent to AL as soon as it is received.

URG (Urgent) Flag and Urgent pointer:

Sometimes it may be needed that some data has to be sent first (than other pckts sent by same sender) to the destination.

For this purpose URG flag is given. When URG=1 then the pckt is treated as urgent pckt.

However router doesn't have TL and hence router cannot look into URG flag.

So when URG=1 we set priority=7 in Type of Service field.

→ whether URG has to be set or not is decided by AL.

→ when urgent pckt is received, this pckt is sent to AL first.

Urgent pointer:

→ This is used when only some part of segment is urgent, but not all

→ Urgent pointer, indicates till what part the data is urgent.

Urgent pointer tells the offset

> i.e., last byte of urgent data = Seq num + Urgent pointer
>
> ∴ no of bytes in urgent data = urgent pointer + 1

Note:

→ When PSH = 1, data is sent to AL as soon as it is received without buffering.

So the order in which pcks are received will not be changed

→ when URG = 1, data is sent to AL even before the pckts that are received earlier.

So the order in which pckts are received changes.

RST Flag (Reset):

This is used when there is anything wrong in connection and thus it is used to make other host cautious. This is used whenever something out of normal occurs.

→ One situation where this is used is if a receiver is expecting some seq num pckt and if a third pdty sends some other pckt, then receiver assumes something is wrong and sends a pckt to sender with RST = 1

15/12/20

Window size (or) Advertisement window

→ Used for flow control

→ At the connection establishment stage, window size field is used by receiver to tell its window size to sender.

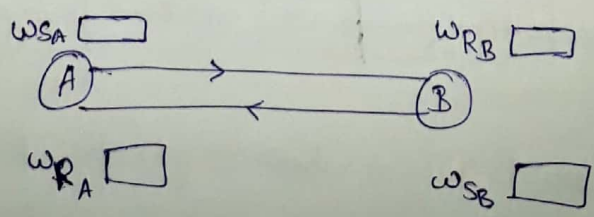
→ As the communication proceeds, this field is used to tell the sender the available amount of space in receiving window.

→ If the sender sends data more than the available space, then the data will be discarded.

→ If window size field is found to be 0 by sender then sender starts a persistent timer and waits until the timer is over or until it receives a message from receiver saying its window is free.

→ If sender receives message saying that receiver window is free, then sender sender data.
Otherwise, it waits under persistent timer finisher and sends exactly 1 byte data. If sender gets reply with adv as non-zero then it continues with communication, else it will wait again.

→ This persistant timer will be useful in the cases where the message (saying window size is available) sent by receiver is lost

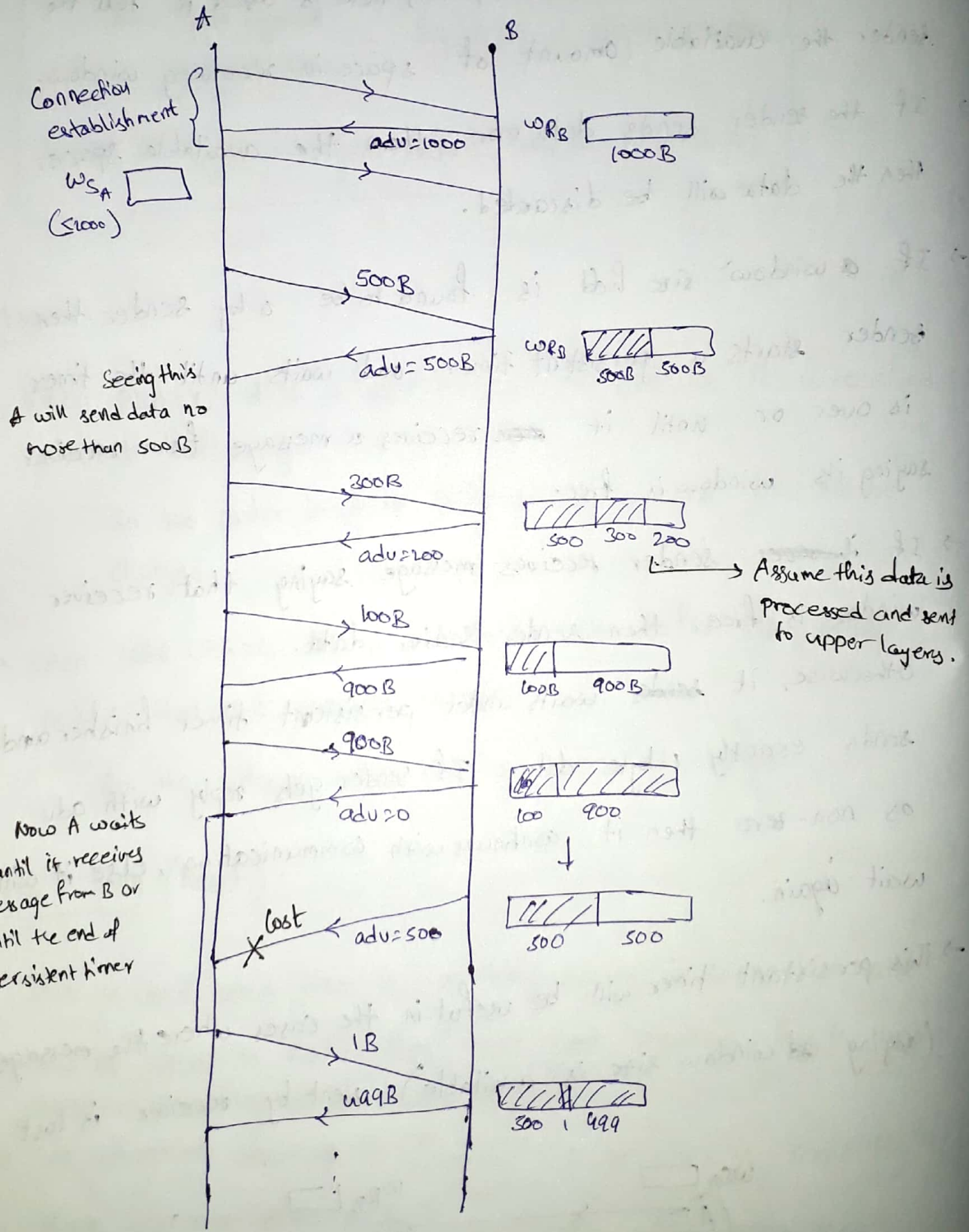


WR_A is known to B and WR_B is known to A at the time of connection establishment.

$WS_A \leq WR_B$ & $WS_B \leq WR_A$

Based on the values of WR_A and WR_B, the station B & A, respectively, sets their sender window sizes.

Consider below case where A sends data to B



Note:

With 16 bit window size field,

the max size that can be advertised is $2^{16}-1$ Bytes

To overcome this we use extra bits from options.

For example if 1GB window is available

i.e., 2^{30}

then we take 14 bits from options field.

Checksum:

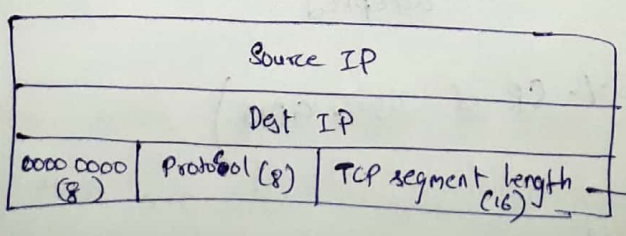
→ The checksum in IP header is computed only on IP header.

→ The checksum is TCP header is computed on IP header, TCP header, data.

But we calculate the checksum only on some fields of IP header as many fields change while the pckt is being transmitted.

→ So in checksum computation we include only those fields of IP header that doesn't change.

Checksum is calculated on below on below fields of IP header (i.e., Pseudo IP header)



Total length − header length

→ The reason behind including IP header in TCP checksum is that we just need to double check that pckt has been delived to correct host.

Options :

(i) Timestamp : used to increase sequence number bits

It is used went wrap around time ≤ life time

The MSB bits of seq num are stored in the timestamp

(ii) window size extension : used when size of window is higher than $2^{16}-1$

(iii) Parameter Negotiation : At the time of connection establishment, there may be a need to exchange some information. In that case we use this option.

Eg : MSS

(iv) Padding : Used to make header size a multiple of 4.

Retransmissions in TCP

→ TCP uses some characteristics of SR and some of GBN

- SR → $w_S = w_R$ → Out of order pckts are accepted
- GBN → Cumulative ack are used

$\therefore$ (75% SR & 25% GBN)

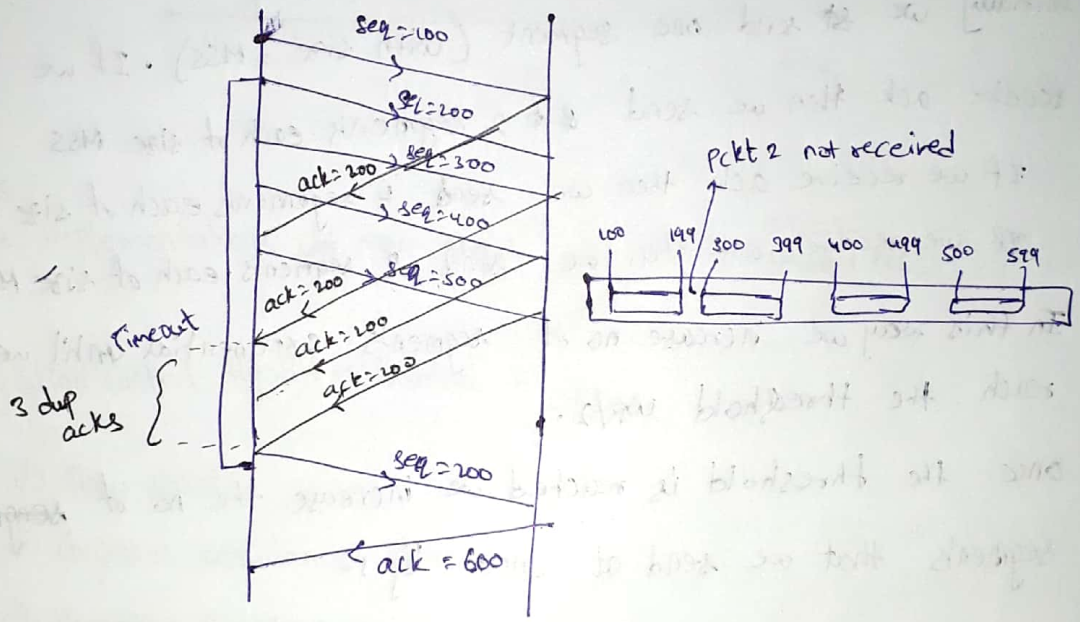
→ In TCP, if ack number received is 'x' then sender assumes that all the data before seq num 'x' is received correctly.

→ TCP uses retransmission after 3 duplicate acknowledgments.

This is also known as Early retransmission.

It means that retransmission done after receiving either 3 duplicate acknowledgements within the timeout ↑ or ~~after timeout~~ retransmission is done after timeout.

For example consider transmitting 5 pckts and say 2nd pckt is lost
Assume size of each segment is 100



→ Retransmission after timeout indicates that, the duplicate acks are also lost and hence congestion is high.

Congestion is also indicated by source quench ICMP message.

Congestion Control:

→ If a sender sends pckts ~~more~~ more that the capacity of network then congestion occurs.

→ If capacity of Nlw is W_c and receiver window size is W_R then sender window size, $W_S \leq \min(W_c, W_R)$

However there is no chance that we can know the value of $w_c \cdot a$

→ So to meet this problem we use congestion control protocols

Working:

→ let receiver window side be w_R

we define threshold = $\frac{w_R}{2}$

→ Initially we send one segment (with size = MSS). If we receive ack then we send 2 segments each of size MSS

If we receive ack then we send 4 segements each of size MSS

If we receive ack then we send 8 segments each of size MSS.

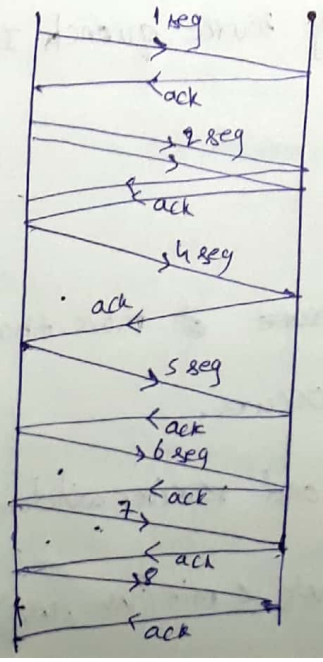
In this way we increase no of segments exponential until we reach the threshold $w_R/2$.

→ once the threshold is reached we increase the no of segments that we send at once by 1.

Eg: Assume $w_R = 8$ segments and $w_s = 8$ segments

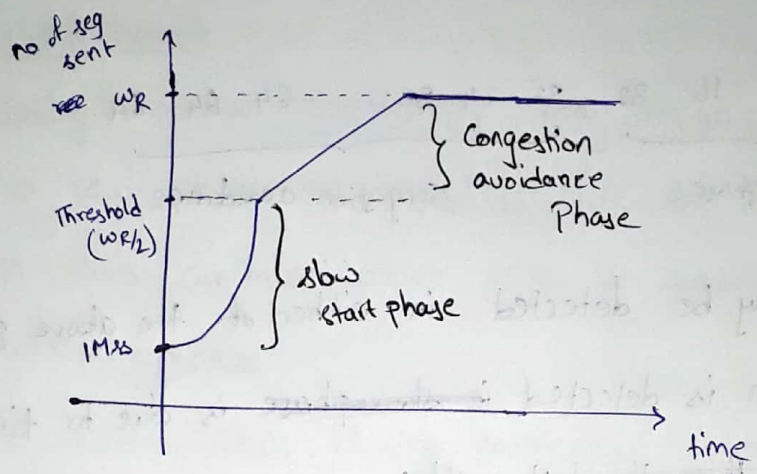
So initially we assume $w_c = 1$ and then increase it

Threshold = $8/2 = 4$



Q: After how many round trip times did we reach ~~reach~~ maximum capacity of ~~sende~~ received

Ans: 6



Note:
Some implementations ~~wi~~ may start with initial $w_c = 2$ ~~w~~ MSS

→ Congestion Control Algorithm involves 3 steps:

(i) Slow start

(ii) Congestion avaidance phase

(iii) Congestion ~~Dection~~ ~~Dection~~ Detection:

Congestion may be detected in slow start phase or congestion avoidance phase.

It is detected in 3 ways

a) Time out (congestion is severe)

b) 3 duplicate acks (mild congestion)

c) ICMP source quench

But this source quench talks only about one host but not about entire network. So this may not be considered • congestion.

Eg 2: $w_R = 64$ kB

MSS = 1 kB

$w_R = 64$ MSS

Threshold = 32 MSS

w_c: 1 2 8 16 32 33 34 35 .... 64 64 64

Slow phase — Congestion avoidance

Congestion may be detected in either of the above phases.

If congestion is detected ~~in slow phase~~ is due to timeout

~~then threshold = old~~

then threshold = $w_c/2$ and enter slow start phase.

If congestion is detected is due to 3 dup ack then

threshold = $w_c/2$ and enter congestion avoidance phase

Eg 2:

1 2 4 8 16 32 33 34 ↑

congestion detected by timeout

∴ new threshold = $34/2 = 17$ & enter slow start phase

1 2 4 8 16 17 18 19 20 ↑

congestion detected by 3 dup pckts

∴ new threshold = $20/2 = 10$ & enter congestion avoidance phase

10 11 12 ↑

congestion detected by timeout

⇒ Threshold = $12/2 = 6$ & enter slow start phase

1 2 (4 6) 7 8 9 10 11 ..... 64 64 64 ....

64 64 ↑ 1 2 4 8 16 32 ↑ 16 17 ↑ 1 2 4 8 9 10 11 ....

To Th=32 — 3 dup Th=16 — To (Th=17/2=8)

TCP timer management:

There are 5 types of timers

(i) Time-wait timer:

- Closing connection immediately may assign the port number to a new process. Due to this the delayed pckts of previous communication may be delivered to this new process.

So rather than closing connection immediately, we wait for time-wait timer.

time wait timer = 2 × life-time.

→ So time wait timer handles delayed pckts.

(ii) keep alive timer:

This used to close idle connections

↳ These are connections that are opened but no communication is done.

Everytime a pckt is received, the system restarts keep alive timer. If no pckt is received until keep alive timer ends, then the station sends 10 probe messages. If no reply is received then we close the connection.

(iii) Persistent timer:

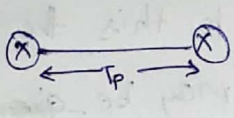
This is used when a station receives, window size of 0 from other host. After this timer we sender 1 byte to test if the window is free.

(iv) Acknowledgment timer:

This used to send cumulative acknowledgments.

Also using this we can wait until data is ready and thus implement piggybacking.

(v) Timeout timer

At DLL 

Round trip time = 2 × Tp

timeout = 2 × RTT

But at transport layer deriving and finding 2×RTT in this way won't be easy. The path, traffic may change dynamically. Hence the timeout timer at TCP also has to be dynamic.

→ Having a static TO timer may lead to congestion.

Consider a case where traffic is high but there is congestion. If TO is small and fixed (static) then we will keep on retransmitting again and again causing congestion.

→ when TO is too high, we wait a lot of time for retransmission and it is a time waste.

There are 2 algorithms for TO timer

i) Basic algorithm
ii) Jackobson's algorithm

Basic Algorithm:

Initially we predicet RTT (say IRTT) and set TO = 2×IRTT.

Assume we receieved ack after some time and we say this is actual RTT (say ARTT). Using IRTT and ARTT we predict NRTT (next RTT)

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

- $\alpha = 1 \Rightarrow$ algorithm is static
  $\alpha = 0 \Rightarrow$ algorithm is fully dynamic

α is called smoothning factor.

Eg: let IRTT = 10 ms & $\alpha = 0.5$

$\Rightarrow TO = 2 \times IRTT$

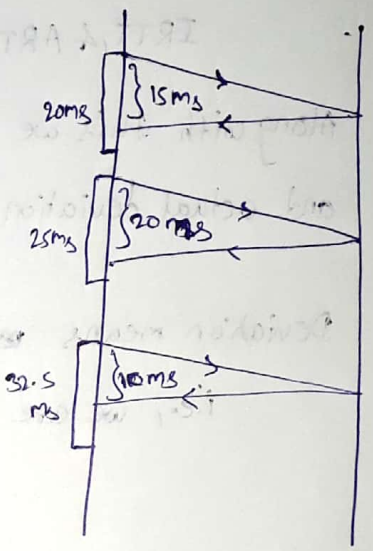
$= 20$ ms

let ARTT = 15 ms

$\Rightarrow NRTT = 0.5(10) + 0.5(15)$

$= 25/2$

$\Rightarrow$ new TO time = 25 ms



Now for 2nd transmission

IRTT = 25/2 ms

ARTT = 20 ms

$\Rightarrow NRTT = 0.5(25/2) + 0.5(20)$

$= 6.25 + 10 = 16.25$

$\Rightarrow TO = 32.5$ ms

Now for 3rd transmission

IRTT = 16.25 ms; ARTT = 10 ms;

NRTT = 13.125 ms

$\Rightarrow TO = 26.25$ ms

Note:

→ Here NRTT depends on entire history of RTTs but previous RTTs have lesser weights.

Disadv: The disadv with this algorithm is that TO = 2 × RTT i.e., we always multiply RTT with a fixed number (2)

Jackobson's Algorithm:

→ This algorithm overcomes the dis adv of basic algorithm

i.e., TO = 2 × RTT

↓

Using this has no logic & reason.

→ Here also we have

IRTT, & ARTT

Along with this we define 2 new parameters initial deviation (ID) and actual deviation (AD) and next deviation (ND).

Deviation means error in our guess of RTT.

i.e., we are expecting ARTT to be in range of

[IRTT - ID, IRTT + ID)

AD is actual observed deviation

ND is predicted deviation for next transfer

ND = α(ID) + (1-α) ND & AD = |IRTT - ARTT|

→ Here timeout is given by

TO = (4 × D) + RTT

(4 × D) → deviation

RTT → This is experimentally found to give better results.

Eg: Let IRTT = 10 ms ID = 5 ms, α = 0.5

for 1st transmission:

TO = 4 × D + RTT = 30 ms

Let ARTT = 20 ms

AD = |IRTT - ARTT| = 10 ms

$\Rightarrow$ NRTT = 0.5(10) + 0.5(20)

= 15 ms

ND = 0.5(5) + 0.5(10)

= 7.5

for 2nd transmission:

IRTT = 15 ms ; ID = 7.5 ms.

$\therefore$ TO = 4(7.5) + 15

= 45 ms

Let ~~AD = 5~~ ARTT = 30 ms

$\Rightarrow$ AD = |15 - 30| = 15 ms

$\Rightarrow$ NRTT = 0.5(15) + 0.5(30)

= 22.5

ND = 0.5(7.5) + 0.5(15)

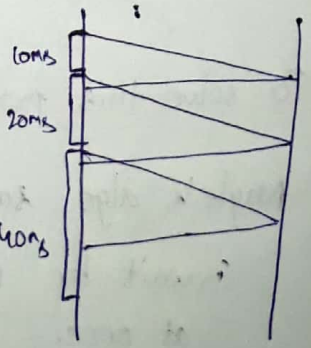
= 11.25

Note (Karn's Modification):

→ Neither of ~~basic~~ the two algorithms talk about the case where ack is not sent within the timer.

i.e., These algorithms don't tell what should be the initial prediction of timeout for retransmission.

→ Karn's modification to Jackson's algorithms says, whenever retransmission is required set TO to double of the previous TO. whenever you receive ack in time, you can continue with normal jacksons algorithm



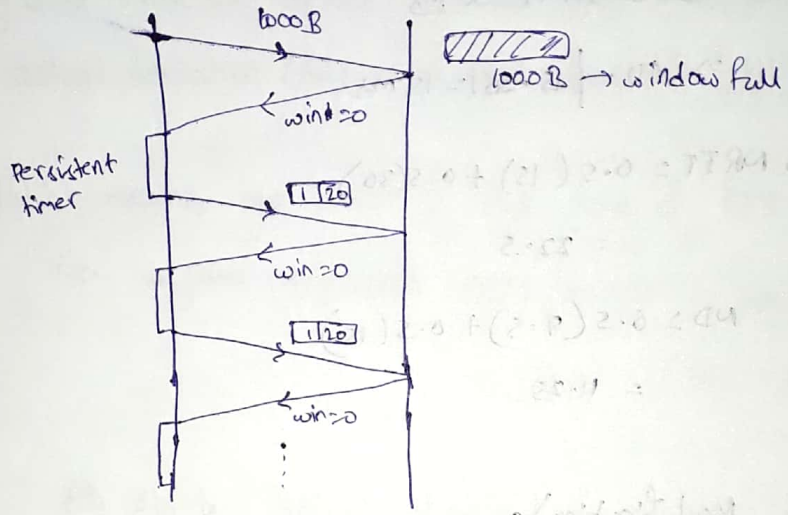
16/12/20

Silly Window Syndrome:

→ We say silly window syndrome problem occured in 3 cases

(i) Receiver ~~always~~ advertises that its window size is 0 for certain amount of time.

But this problem is temporary and will be solved once window is available.



In this case, we use RST flag and restart the connection.

(ii) Second case where this problem occurs is when sender is slow and transmitting less amount of data (say 1 byte) at once. This means for 1 byte, (1+20+20+20+20) 81 byte have to be transmitted

TCP, IP headers on sender side

header on receiver side for ack.

To solve this problem we use Nagle's algorithm

Nagle's algo says

(i) wait for 1 RTT and send the data produced in 1 RTT at once.

(ii) If data sufficient for 1 Mss is produced in less than 1 RTT then transmit it.

(iii) Another case in which this problem occurs is that when receiver accepts only one byte of data repeatedly (i.e., receiver advertises its window size as 1)

Clarck's solution for this problem:

Receiver should not advertise until it gets freed ~~window~~ half of the window or until it gets freed 1 MSS size of buffer

Traffic Shaping:

→ Another method of congestion control is to shape the traffic before it enters network.

→ Traffic shaping control the rate at which the pckts are sent.

→ During connection establishment, the sender and carrier negotiate a traffic pattern (shape)

Traffic shaping is done using 2 algorithms:

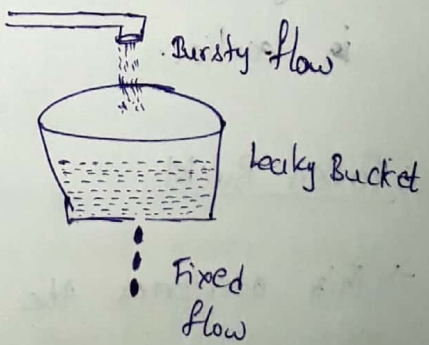
(i) Leaky bucket
(ii) Token bucket

(i) Leaky Bucket:

Here bucket corresponds to Queue.

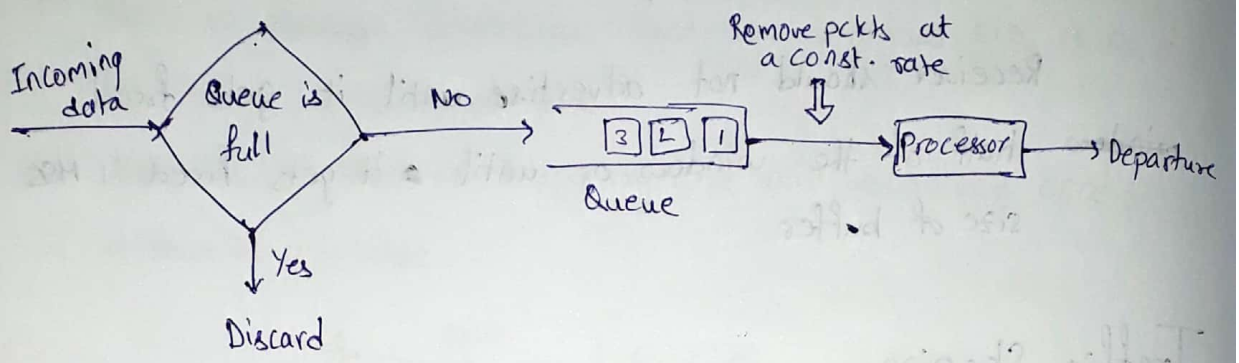
~~This Queue~~

Data may come into queue at any rate But it is taken out and put on the network at a constant rate.



~~However the to Queue (buck~~

However the bucket (queue) has some capacity.
If ~~data~~ bucket is full and new data comes in, the the new data is discarded



→ Here required outflow is x bytes per second and if pckt size is of variable length, then we use a counter to count no of bytes in queue and transmit req no of bytes.

Disadv :

→ If a station doesn't transmit data for long time and if it needs to transmit after sometime, still the station will have same outflow rate and no extra rate will be given for the reason that the station didn't transmit earlier.

(ii) Token Bucket

→ This overcomes the disadv of ~~token~~ leaky bucket.

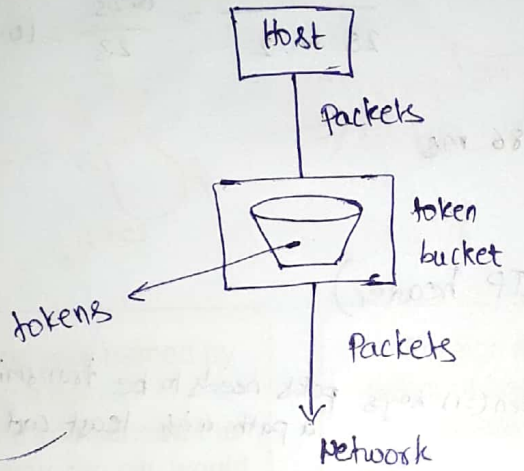
→ Each pckt needs a token to get on the network.

→ Here we maintain a token bucket to which tokens are

added at a fixed rate.

→ In this way even if a system doesn't transmit for long time and need to transmit later, it can use the tokens and transmit packets (one per each token)

→ However, token bucket also has a capacity. Once token bucket is full, no more tokens will be added.



→ If capacity of token bucket is c tokens and token enter bucket at rate of γ token/sec then find max no of pckts that can enter network during time interval t is _____

Max no of packets = $c + \gamma t$

$\Rightarrow$ Max avg rate = $\frac{c+\gamma t}{t}$ pckts/sec

For large value of t, max avg rate = γ

6) Token bucket capacity = 250 kB; tokens arriving rate = 2MB/s
If maximum output rate is 25MB/sec, then find burst time.

Sol:

since given token capacity = 250 kB

⇒ each byte is a token.

Burst time is duration in which we send 25 MB

$$\therefore \frac{C+\gamma t}{t} = M$$

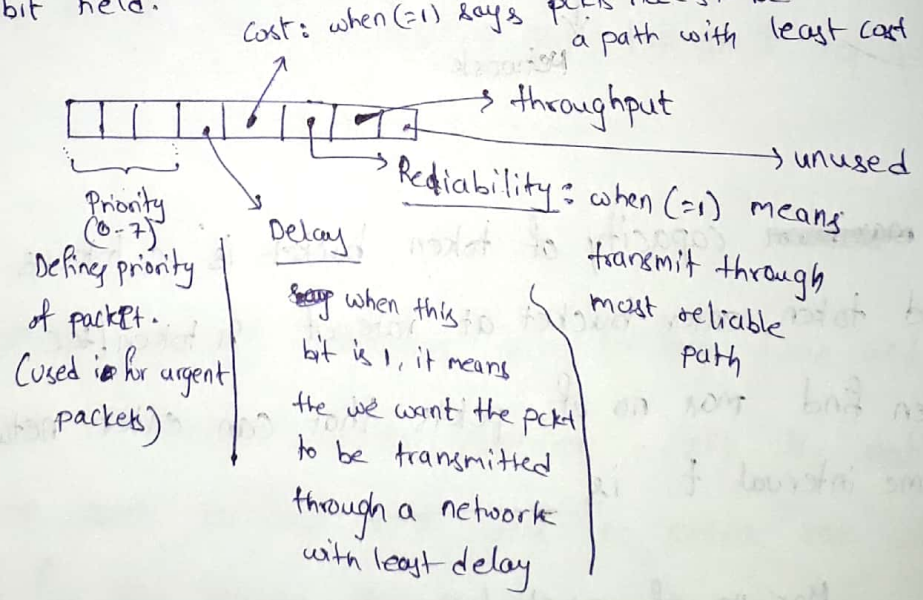
$$\Rightarrow C+\gamma t = Mt \Rightarrow t = \frac{C}{M-\gamma}$$

$$t = \frac{0.25}{25-2} = \frac{0.25}{23} = 10.86 \text{ ms}$$

∴ Burst time = 10.86 ms

Type of Service (field in IP header)

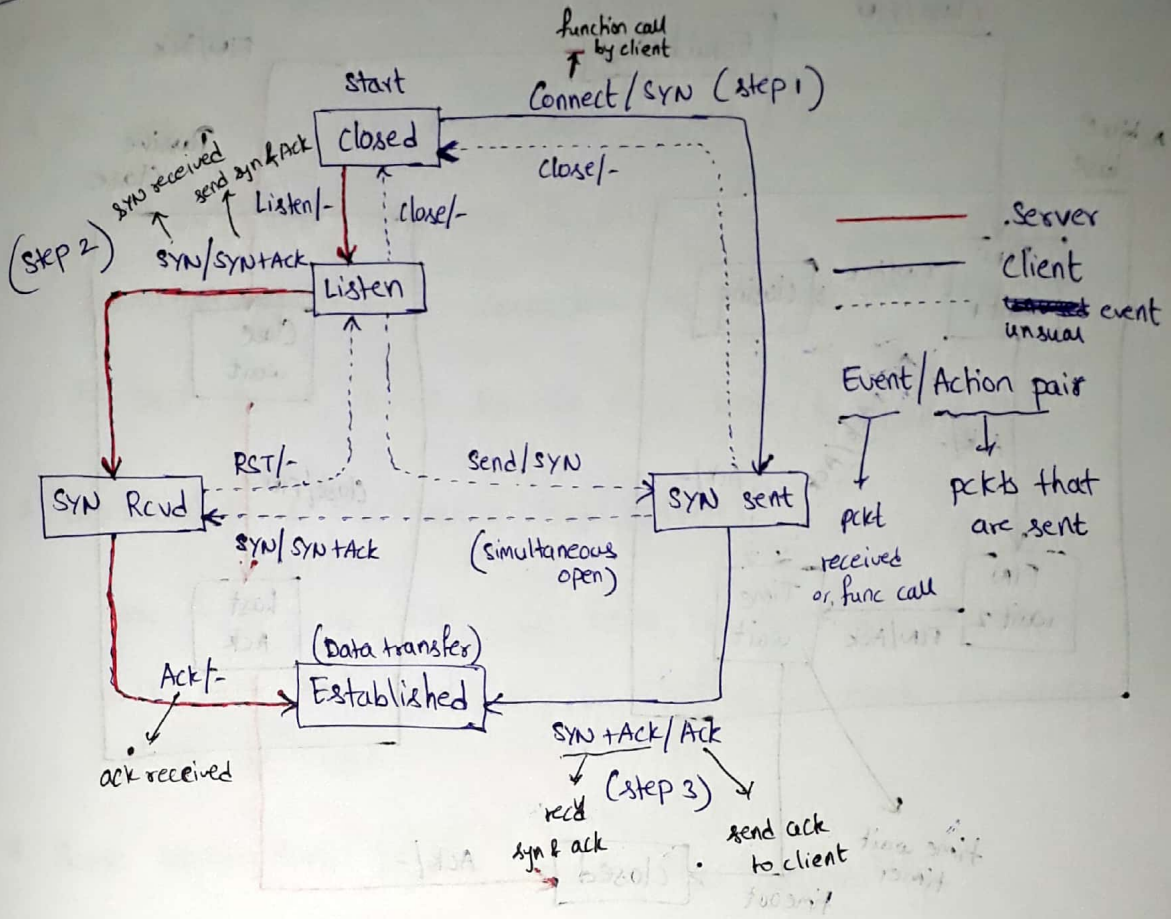
→ It is 8-bit field.



→ when throughput = 1, it means that pckt has to be transmitted through a network where throughput is high

16/12/20

TCP State transition diagram: Connection Establishment



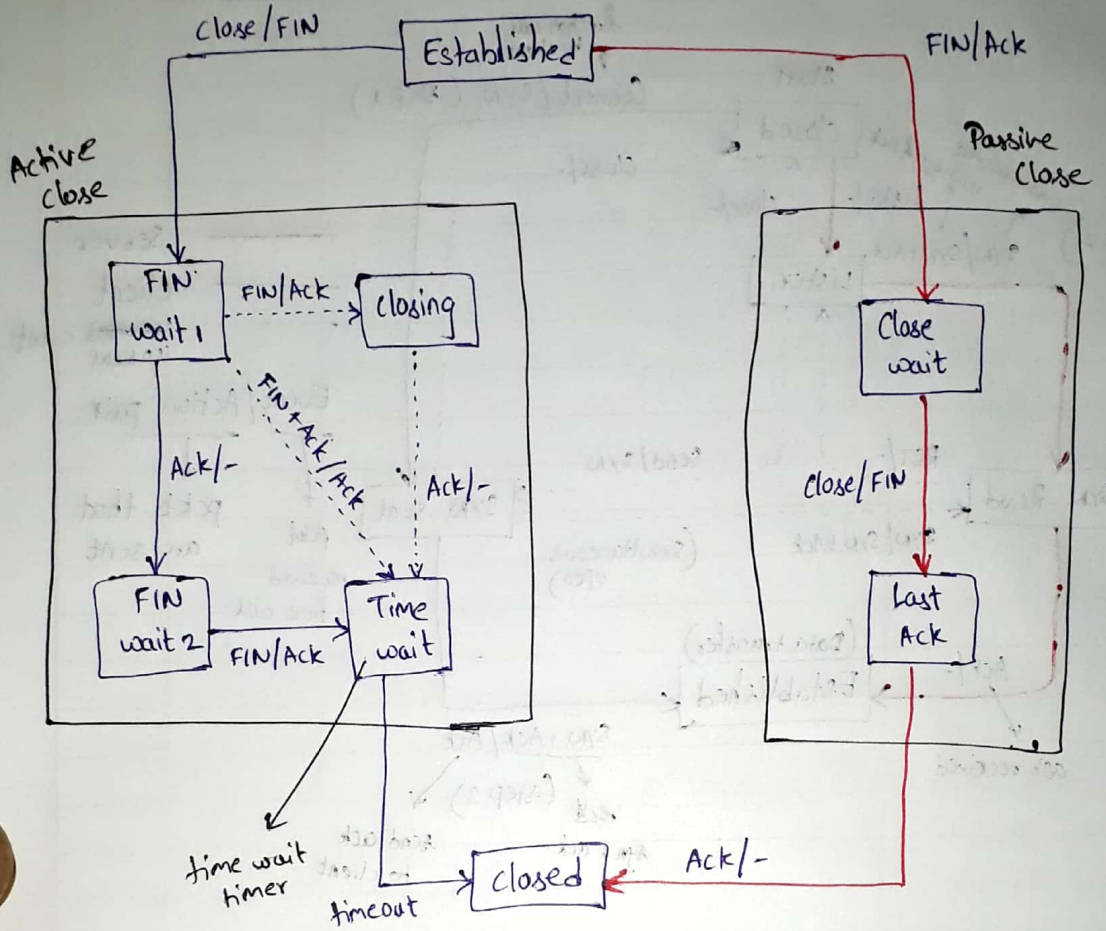
→ At "syn sent" state, if client doesn't receive (SYN+ACK) for long enough time, then client execute close system call and moves to "closed" state.

→ At "Listen" state, if server doesn't receive any connection requests then server executed close and moves to "closed" state.

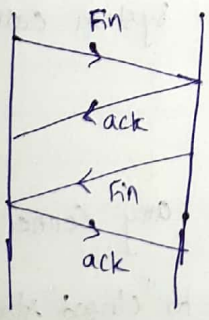
→ At "SYN RCVD", if server finds any problem, it ~~call~~ calls rest and terminates the connection.

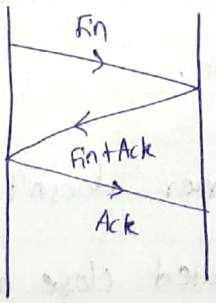
→ Sometime server may initiate connection. In this case server at "Listen" state sends SYN. Then client at "SYN sent" sends (SYN+ACK) on receiving SYN from server. This is called Simultaneous open as both server & client wants to open connection at a time.

TCP state transition diagram: Connection Termination

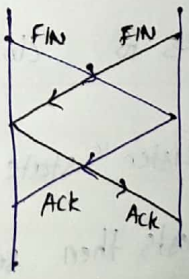


Termination of connection may happen in any of below way





FIN & ACK are sent at a time by server



Both client & server sends FIN at a time

All the above cases are shown in the transition diagram.

UDP (Null Protocol)

→ Need for UDP:

* In applications that need one request & one reply.

  when we need only one request & one reply, the connection establishment, connection termination are overheads

  Eg: DNS, BOOTP, DHCP, Routing algorithms (for sharing DV)

* Broadcasting & Multicasting applications.

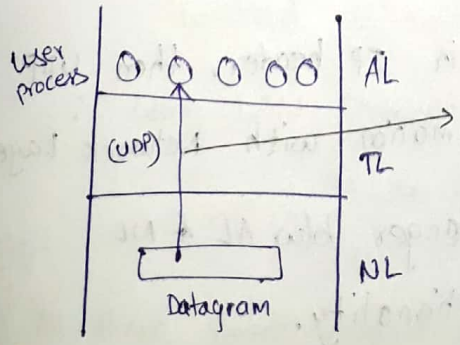
  Here if we use TCP, we need to establish connection with each host and reserve buffers for each connection.

  ∴ we use UDP.

* Some applications require speed than reliability.

  ~~Using~~ TCP applies end to end congestion control due to which speed fluctuates.

  Eg: Multimedia applications (watching videos)
  Online Gaming etc..

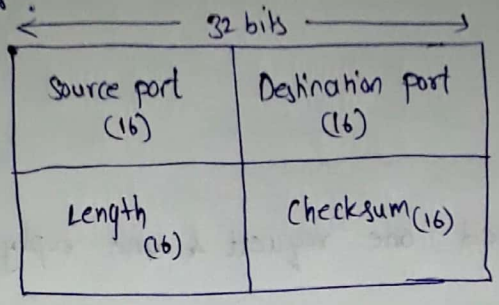


→ The received datagram is directly given to processes in AL bypassing the services of transport layer.

→ UDP header doesn't need seq num, ack, window size, flags, urgent, pointer, options.

∵ no flow ~~flow~~ control

UDP header:



Header size = 8 bytes

UDP has fixed header size

length: length of datagram including header

Checksum: Checksum is computed on UDP header, UDP data, psuedo header of IP.

→ Since UDP is unreliable, checksum may not be used. when checksum is not used it is set to (000...0)

→ However, if the computed checksum itself is 000..0 then we store 111...1 in checksum.

* Checksum is stored in 1's complement form

∴ If 0 is represented as 000..0 => checksum not used

0 is represented as 111...1 => checksum is used

Other functionalities of UDP:

→ If AL wants to set options in IP header, then UDP has to communicate this information with Network Layer.

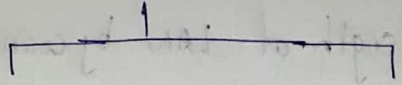
i.e., UDP has to act as messenger b/w AL & NL

TCP also has this functionality.

→ ICMP pckts received at IP, has to be informed to AL through UDP

Hardware and other devices used in Networking

→ 2 types of channels



Base band	Broad band
→ only one station (i.e., 1 signal) can be transmitted at once	→ Multiple hosts can transmit at a time
∴ No multiplexing	Multiplexing is possible

→ The wires come in different types. Few are shown below.

* 10 Base T (10 Mbps, No multiplexing, 100 m)
* 10 Base 2 (10 Mbps, " , 200 m)
* 10 Base 5 (" " 500 m)
* 100 Base T (100 Mbps, " 100 m)

→ The third field is distance for which the wire can run.
i.e., It is max distance for which signal can be transmitted without attenuation.

∴ LAN Segment with 10 Base T can range only for 100 m

LAN Segment → LAN connected to one wire

Note:

(i) All these wires operate at PL.

(ii) Attenuation is possible for all.

(iii) Collisions are possible.

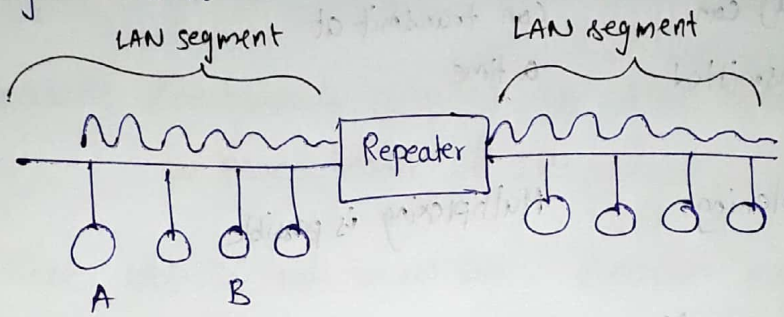
Collision Domain is n.
i.e., no of stations involving in collision

Here length of LAN depends on the type of wire.

Repeater:

→ This is used to increase the length of LAN by connecting 2 LAN segments.

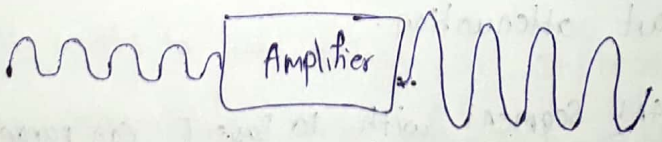
→ Repeater takes in signal which is about to dies and regenerated it



- Even if A want to send data to B, repeater still sends data to other side.

→ The LAN segments connected to repeater must be of same type. i.e., both should be ethernet or both should be token ring.

→ Repeater is different from amplifier. Amplier increases the amplitude whereas repeater just regenerates the original signal



→ Repeater runs at PL
i.e., repeater is purely hardware.

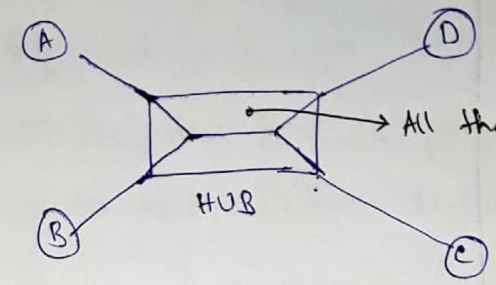
→ Collisions are possible in repeater and hence all the stations in the network are in one collision domain.

∴ Repeater doesn't reduce collision domain. i.e., on

Hub:

→ Hub is a multi-port repeater.

→ For example, a 4-port hub can connect 4 stations



→ All the system are connected to each other, within hub.

Here if B needs to transmit a pckt to A, the pckt will be transmitted to every station

∴ Traffic is very high

→ HUB has only PL.

→ Collision are possible inside hub.

Collision Domain = n

∴ Hub doesn't decrease Collision Domain.

Adv: Hub is cheaper.

Bridge:

→ Bridge is used to connect 2 LANs

(Repeater connects 2 Lan segments of same LAN)

→ If a bridge has n ports, then it can connect n LANs.

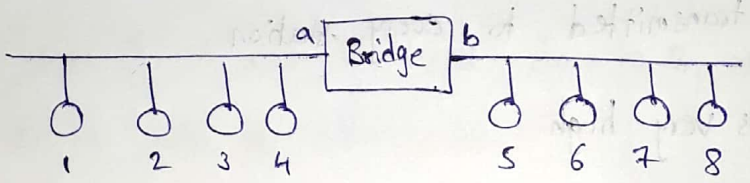
→ Also the 2 LANs connected by bridge may be different i.e., one may be ethernet and other may be token ring.

→ Bridge operates at PL & DLL

→ Since bridge works at DLL, it can look in MAC addresses.

→ Every bridge has a forwarding table.

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



Bridges

- Static Bridgs
- Dynamic / Learning / Transparent bridge.

Static Bridge :

→ Here the forwarding table is static and whenever we need to make a change in the network we need to change the table manually.

Dynamic Bridge :

→ Here bridge will learn the MAC & port of a station.

→ Whenever a station transmits data, the data reaches bridge and thus bridge will understand the MAC of the station and puts an entry in the forwarding table

→ But dynamic bridge never knows the MAC address of a host which has never transmitted any data.

→ Bridge is capable of filtering (∵ Bridge has DLL)

i.e., It sends a pckts to LAN in which the req host is present.

If the dest host is present in the N/w sender then bridge discards the pckt.

→ Bridge is capable of Forwarding.

i.e., sending data from one N/w to other N/w.

→ If a bridge doesn't know where the destination is present, then it floods the pckt.

∴ Bridge is capable of Flooding.

→ Bridge is capable of store & Forward.

Since it can store, collisions blw hosts of diff N/w are not possible.

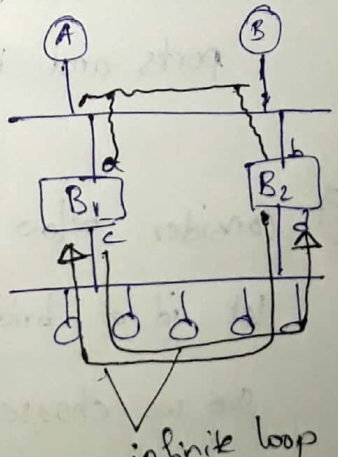
∴ Collision Domain is reduced.

Problems with bridges

→ If we connect two lans using 2 bridges, and assume initially forwarding table is empty.

Now if A send data to B, this sent pckt is fall in infinite loop.

Another problem is that port of A keeps on changing from a to c & c to a.



→ If MTU of LANs connected is different, then bridges cannot do the job of fragmentation.

So the fact that bridges can connect different types of n/w is theoritical but not much practical.

17/12/20

Spanning Tree Algorithm:

→ This algorithm is used to avoid infinite loop that pckts fall into, when two LANs are connected by more than one bridge.

Algo:

(i) Every bridge has a built-in ID. The one with smallest ID is taken as root bridge.

(ii) Mark one port of each bridge which is closest to root bridge as root port.

(iii) Every LAN chooses a bridge closest to root bridge as a designated bridge for that LAN. make the corresponding port as designated port.

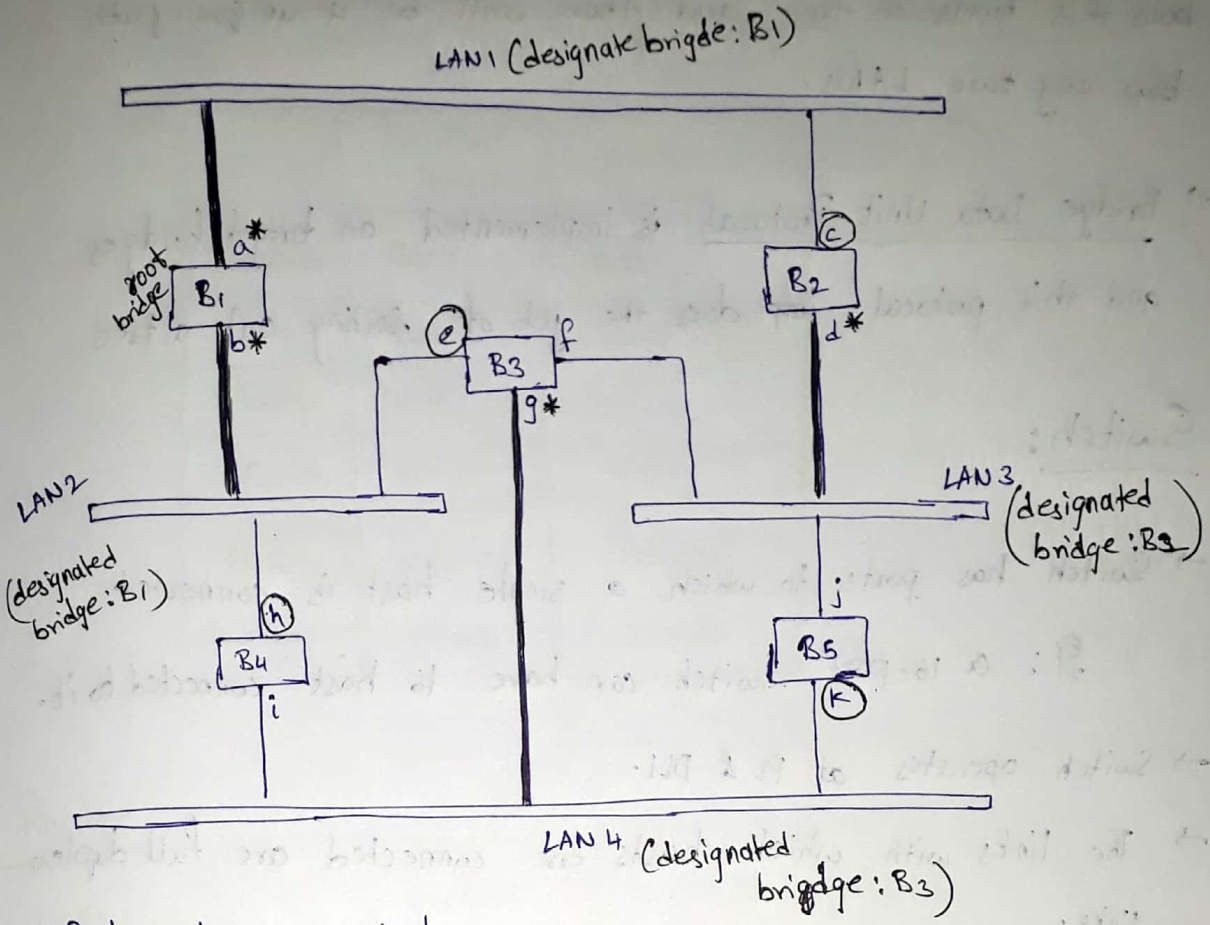
(iv) Mark the root port and designated port as forwarding ports and block remaining.

Eg: Consider below N/w with 4 LANs and 5 bridges.

let id of bridge B_i be i.

So we choose B_1 as root bridge.

→ Here closest is defined with some cost parameters.
Here we consider no of hops

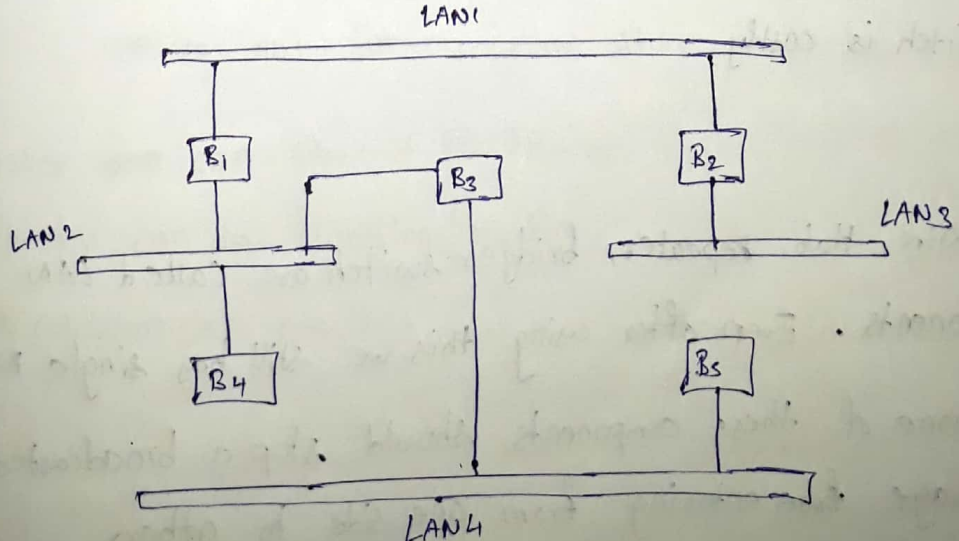


→ Root ports are encircle.

→ when ~~two~~ there is a conflict in choosing designated brigde, choose the bridge with least id.

(thick line) - line that ~~two~~ connects LAN & designated port (*)

→ Now we use root port & designated ports as forwarding port and block remaining



now this forms a tree and there will be a unique path blw any two LANs.

→ Bridge Data Unit Protocal is implemented on ~~brig~~ bridges and this protocal ~~imp~~ does the job of sorting out a tree

Switch:

→ Switch has ports to which a single host is connected.

Eg: a 16-port switch can have 16 host connected to it.

→ Switch operates at PL & DLL.

→ The links with which hosts are connected are full duplex links.

→ Switch is designed in such a way that it can allow more than one communication occur at a time. Switch can forward pckt to crct host. (∵ it has DLL)

→ ∴ No collisions are possible using a switch.

Collision Domain = 0.

→ Traffic is less with switch.

Disadvantage:

→ Switch is costly.

Note:

→ ~~use~~ wire, Hub, repeater, bridge, switch are called LAN components. Even after using this we still have single N/w. So none of these components should stop a broadcasted message ~~from~~ entering from one side to other

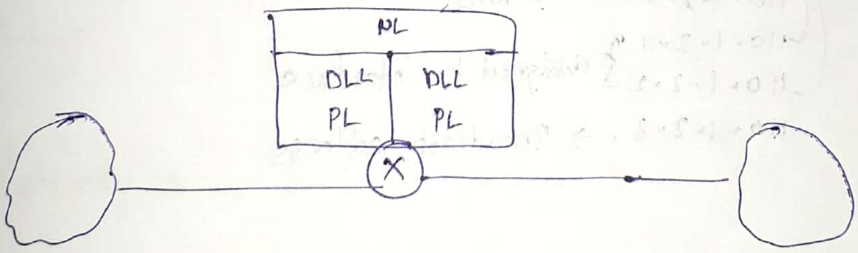
∴ These components doesn't change broadcast domain.

	Broadcast Domain	Collision Domain
Repeater	same	same
Hub	same	same
Bridge	same	reduces
Switch	same	reduces
Routers	reduces	reduces
Gateway	reduces	reduces

Router:

→ Router is a device used to connect 2 networks

→ Router has PL, DLL, NL



→ Router does the job of filtering

i.e., broadcast, BootP, DHCP, ARP etc. are discarded and are not let to cross other networkes.

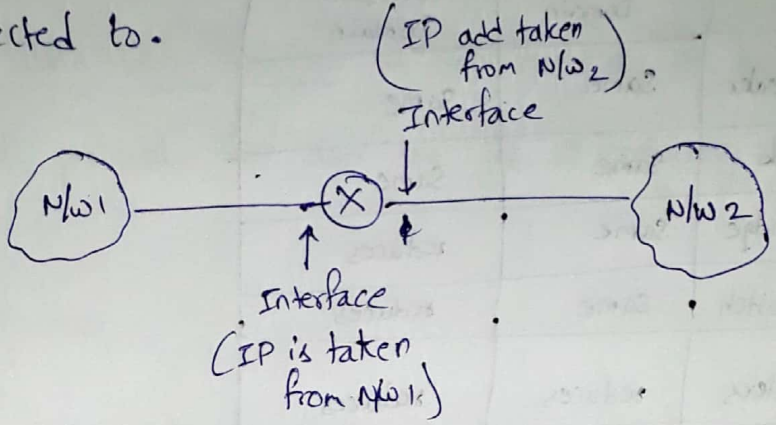
→ Router can store & Forward.

→ Router can do flooding / routing.

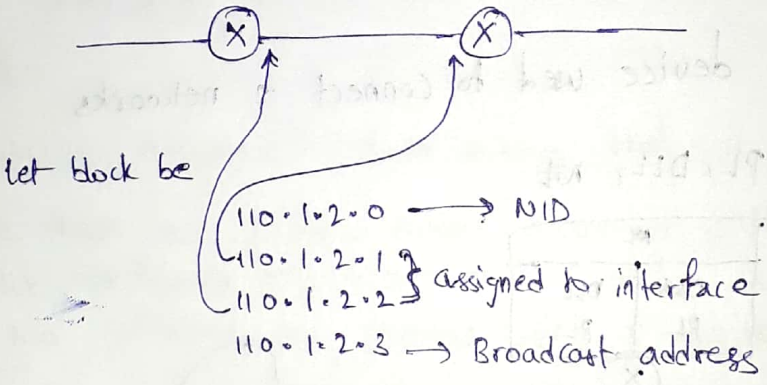
→ No collision are possible within a router

→ Every interface of router has an IP address.

The IP address is taken from ~~one~~ the N/w that router is connected to.



→ However if two routers are connected, then we need to ~~buy~~ buy a block of IP add of size 4.



Gateways

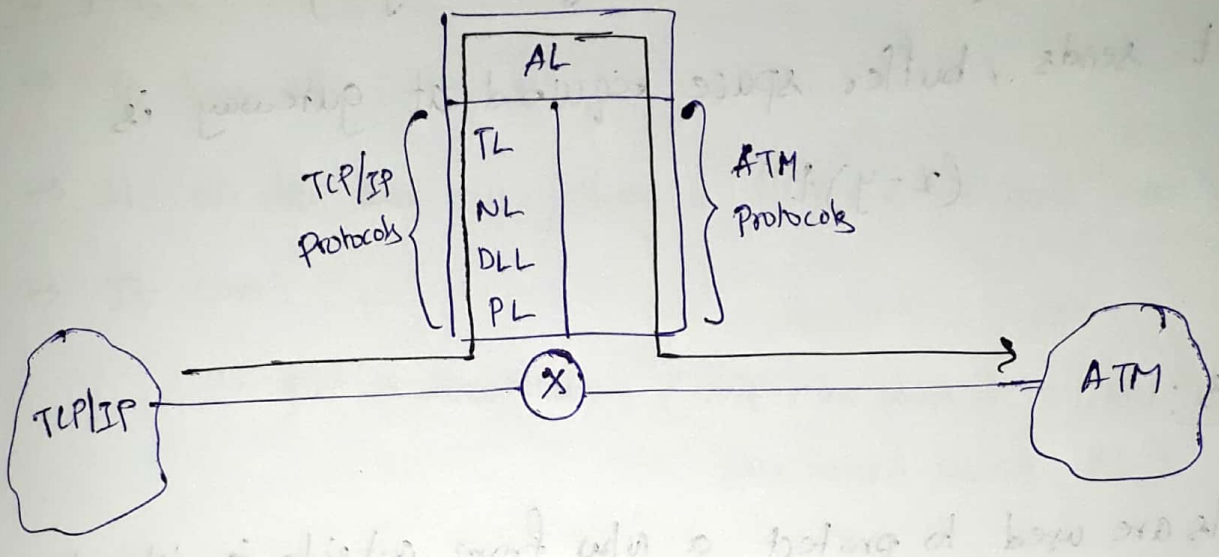
Router is capable of connecting similar type of N/ws.

i.e., two TCP/IP N/w or two ATM N/ws

but not a TCP/IP and an ATM N/w

→ ~~In~~ For this need we use Gateways.

→ Gateway is used as protocol converter.



→ Gateway is is used as proxy.

* i.e. A proxy can monitor every pckt that is going out of the network.

* Proxy also does the job of buffering / caching
  ↓
  i.e., recent accesses are stored.

→ Gateway is use as NAT server
  ↳ used to conserve IP addresses.

→ Gateway is also used as Firewall.

Firewall monitors every incoming & outgoing pckts and can impose some restrictions.

→ Another use of Gateway is Deep Packet Inspection (DPI)

DPI helps look into application layer data and thus know what data is being transmited and can also impose some restrictions.

→ Gateway also does buffer management.

If data coming from other end is too fast, then gateway can buffer the data and sends us data at required rate.

* If x is incoming rate of data & y is outgoing rate then for t sends, buffer space required at gateway is

$$(x-y)t$$

Firewalls:

→ Firewalls are used to protect a n/w from outside internet

→ Any incoming/outgoing pckts has to go through firewall.

Firewalls are of 3 types

(i) Layer 3 Firewall
(ii) Layer 4 Firewall
(iii) Layer 5 Firewall / Proxy

Layer 3 Firewall:

→ It has PL, DLL, NL

This firewall can

(i) Block hosts (using SIP, DIP)
(ii) Block certain protocols (TCP, UDP, ICMP, IGMP)
Block ICMP can save from ICMP attacks.

(iii) A protocal from particular host can be blocked.

Layer 4 Firewall:

→ It has PL, DLL, NL, TL

→ It can do everything a Layer 3 Firewall can do

→ It can:

(i) Block a service (by looking into port number)

(ii) Block a particular service from particular host.

Layer 5 Firewall / Proxy Firewall:

→ It has all 5 layers.

→ It can do ~~any~~ every thing a layer 4 Firewall can do.

→ It can:

(i) give authentication (∵ it can look into username & password using AL)

→ Here a rule table is maintained.

A pckt is discarded if it matches ~~which~~ with atleast one rule.

18/12/20

Application Layer Protocols

→ DNS, HTTP, FTP, SMTP, POP

DNS (Domain Name Service)

→ Port No: 53

→ Given a domain name, DNS gives its IP address.

Even if we remember IP add, there is no guarantee that the IP address of a website doesn't change

Types of domains:

(i) Generic Domain:

.com, .edu, .mil, .org, .net

↓ used by commercial organizations

↓ used by non-profit organization

(ii) Country Domain:

.in, .us, .uk .....

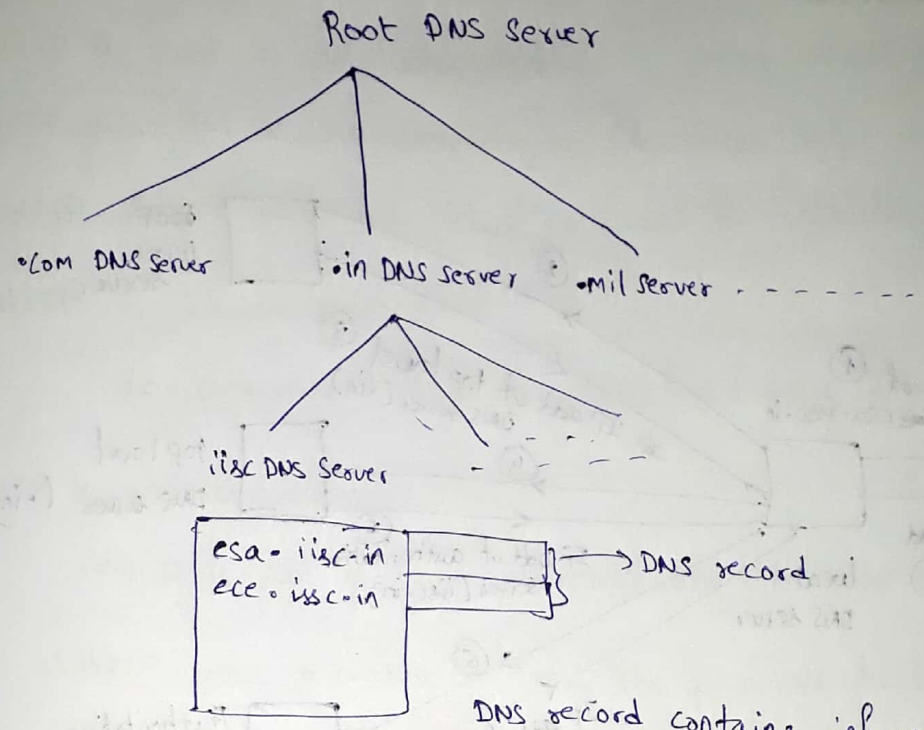
(iii) Inverse Domain:

Given an IP address, DNS can also give domain name.

→ DNS also does load balancing.

i.e., if a website have more than one IP add (more than one server) then each time we access the website it will take us to a different server.

Data organization in DNS



DNS record contains information like how long the IP address will be assigned to that domain, etc.

→ Data of DNS is distributed.

→ There are 13 root DNS server

So that even if one server doesn't work there will be no problem

Note:

→ However, we don't need to go to Root DNS server everytime.

→ ISP provides local DNS which has most frequently used websites. If local DNS can't help then we go to root DNS server.

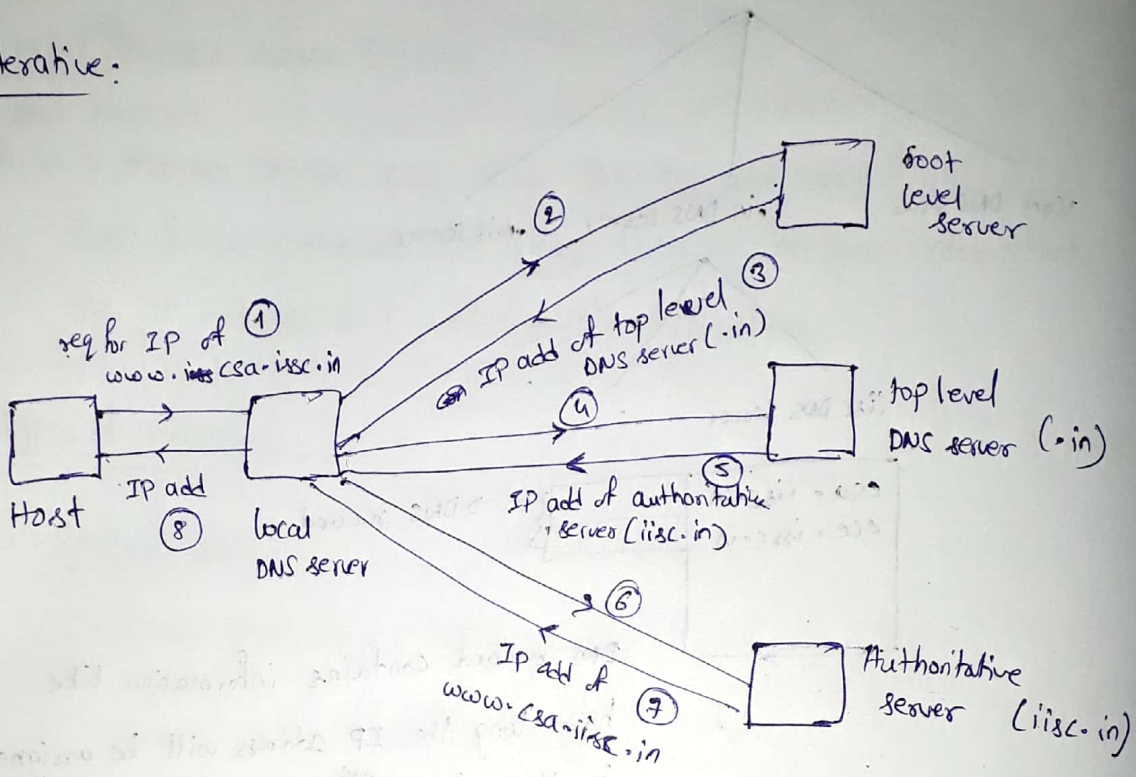
→ The record of local DNS has a field Time to Live, after which entry is deleted.

There are 2 ways through which local DNS contats root DNS server.

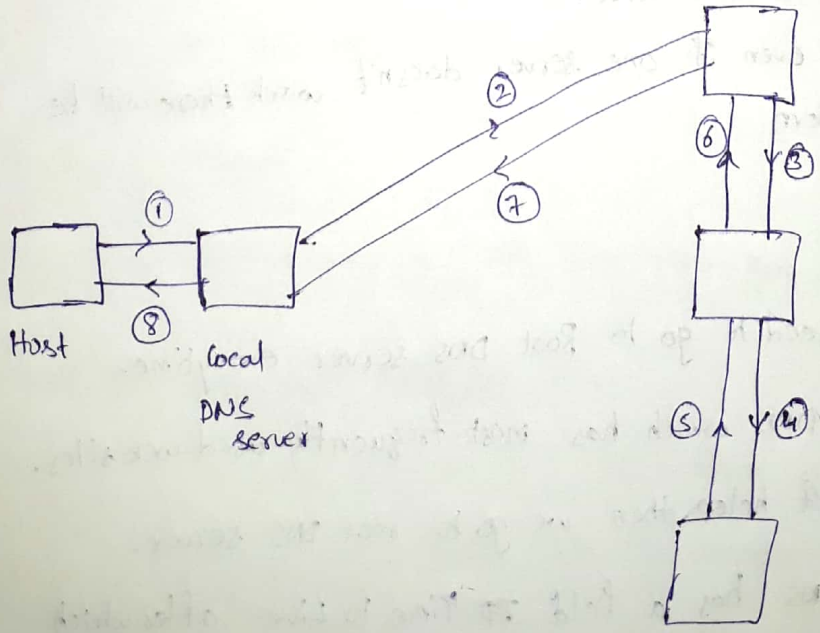
i) iterative

ii) recursive.

Iterative:



Recursive:



→ ~~All the~~ DNS uses UDP (since all the messages are single req & reply)

HTTP:

→ port number: 80

→ HTTP is used to get web pages.

→ HTTP uses TCP at transport layer (for reliability)

HTTP doesn't have any inbuilt mechanism for reliability.

→ HTTP is inband protocol.

i.e., commands & data are transfered using same connection.

→ HTTP is stateless protocol.

i.e., It is not going to maintain any information abt users

∴ HTTP uses cookies to know the previous history of communication. These cookies are stored at users end.

→ HTTP 1.0 uses non-persistant connection

i.e., a seperate connection is used to fetch each object.

For example if a web page has 10 image, we used 11 connections (one for each image & one for web page)

Here server doesn't need to hold connection for long time.

→ HTTP 1.1 uses persistant connection.

i.e., a single connection is used to fetch all the data.

Here server holds connection for long time. So here congestion window grows and thus we have high Bw.

Methods used in HTTP:

(i) Head: used to obtain header (meta data) of a webpage.

Usage: Gateway caches frequently used data.

So by getting head, gateway knows whether the data is modified or not. If not modified, data from gateway cache can be sent.

There are many other uses

(ii) Get: Used to get the webpage

(iii) post: post is used to send data when we fill forms

(iv) Put: used to upload something

(v) Delete: used to delete an object.

(vi) Trace: Used to know the servers involved in sending data.

(vii) options: used to know whether the above functionalities are provided by a server or not.

(viii) Connect: used for security (Authentication).

FTP (File Transfer Protocol)

→ port number: 21

→ Example applications: Techia, Filezilla
→ FTP uses TCP
→ FTP uses out of band connection

i.e., seperate connections are established for commands & data.

Control Connection (used for commands) (port: 21) (Persistant)
Data Connection (used for data) (port: 20) (non-persistant)

∴ seperate connection is opened for each data transfer

FTP server: data of FTP server is used by FTP client.

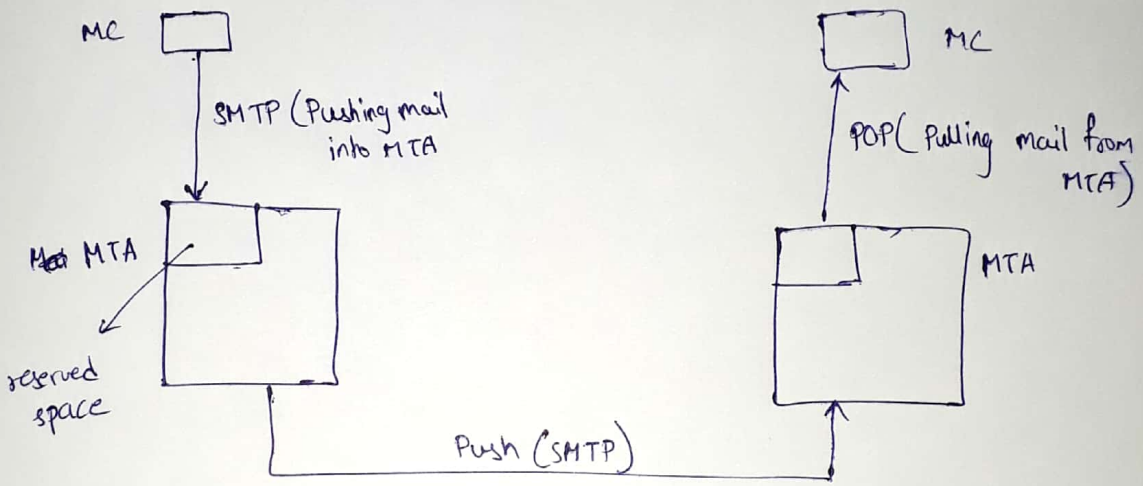
→ FTP is stateful protocol.

∴ FTP is going to log every activity.

SMTP (Simple Mail Transfer Protocol) and POP (Post Office Protocol)

→ FTP requires both the stations to be online while file transfer.

→ But SMTP doesn't need this. So we don't use FTP for emails.



MTA: Mail Transfer Agent

MC: Mail Client

→ SMTP & POP uses TCP for reliability.

→ To send data other than text, this non-text data has to converted into text format on sender side and it has to converted back into non-text format on receiver side.

The protocol used for this conversion is MIME

MIME: Multipurpose Internet Mail Extension.

→ SMTP & POP are inband protocols.