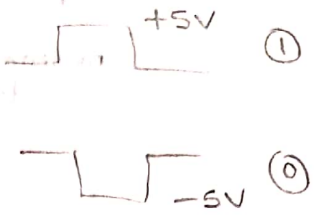
COMPUTERS NETWORKS:-

Protocol: rules for transmitting & formatting of data.
set of rules.

* internet: inspired from ARPANET
1960s

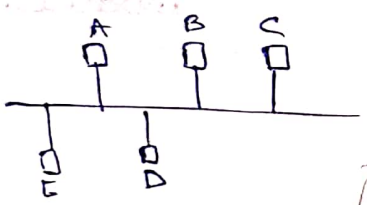
data → bits → signal.
(wire/wireless).



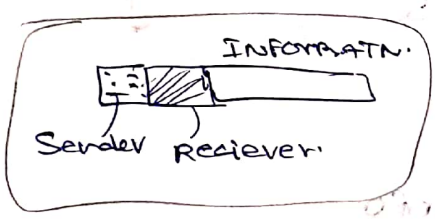
1) **Physical Layer:-**

some headers; to "denoise".

2) **Data Link Layer:-**



medium access.

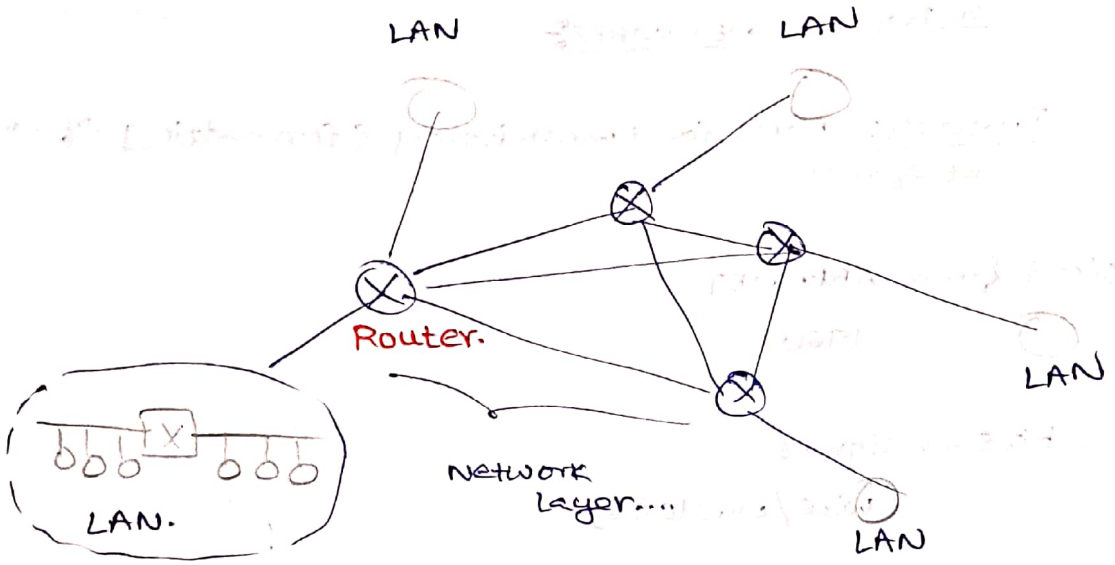


PACKET.

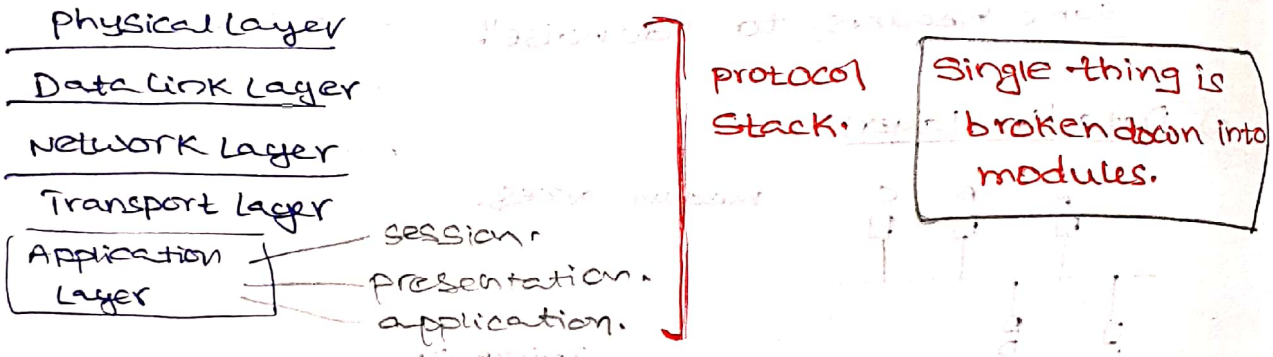
- Also gotta resolve interference.

* **Level-2 switch:-** repeater.
(repeats only those signals which are necessary).

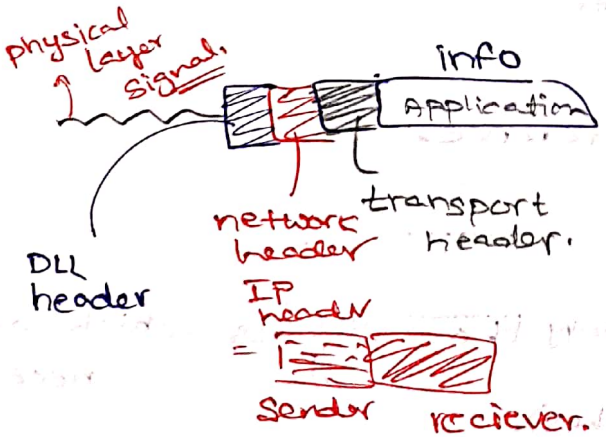
(routers are Level-3 switches).



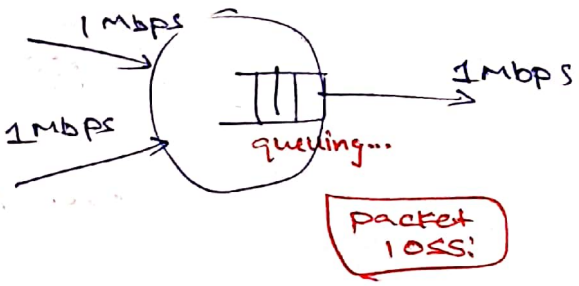
*



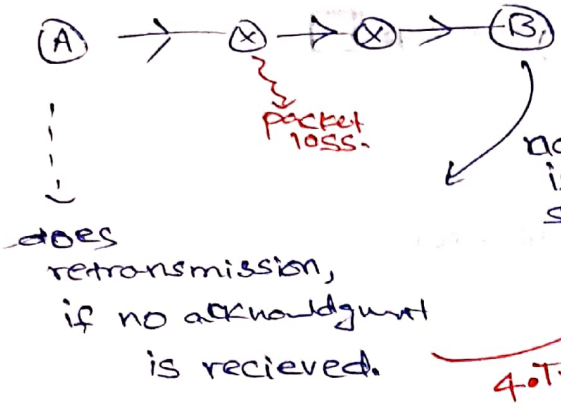
message packet:-



* router



* When A & B talk.
(end devices; with transport layer intact).

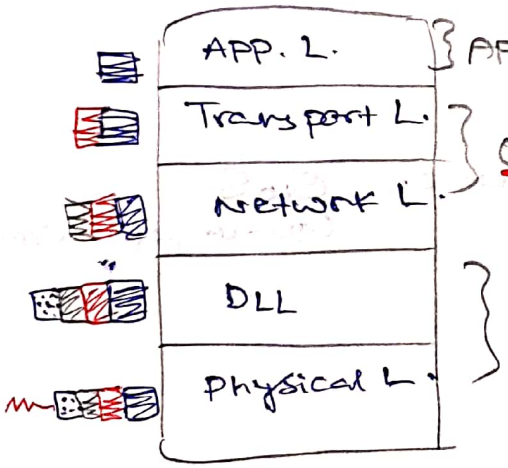


4. Transport Layer.

ubisoft!
TCP protocols
UDP

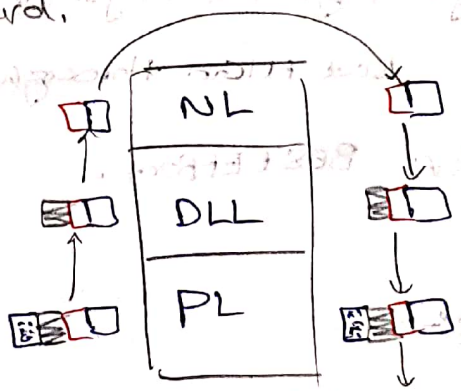
this design; retransmission by source; rather than intermediate is by choice.
(some military history).

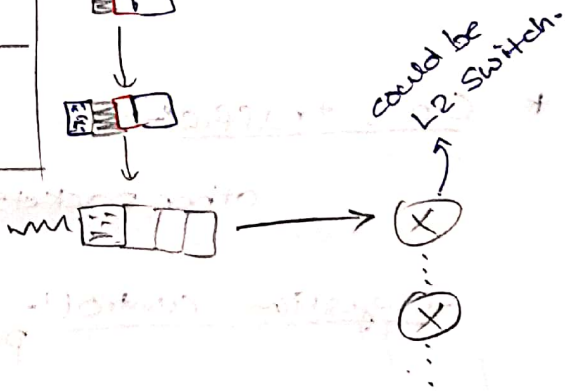
* My mobile:-



Packet Sent to Base Station

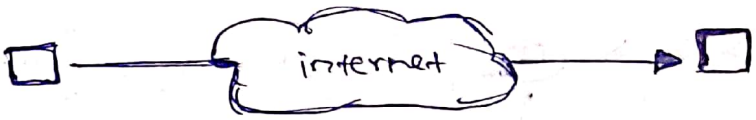
A router.
means -> L3 switch





Again all 5 layers present.

* Performance:-



Throughput:-

data rate transfered

> 100 KBPS

> 1 Mbps ...

Latency:-

Delay of a message across network.

~ 100 mS for one way latency.

> time for reaching.

RTT:-

round trip time.

* different applications have difference QoS (Quality of Service)

Chat: Low latency.

Video streaming: high latency... okay...

But HIGH throughput.

File transfer: Best Effort.

* Cross-traffic:-

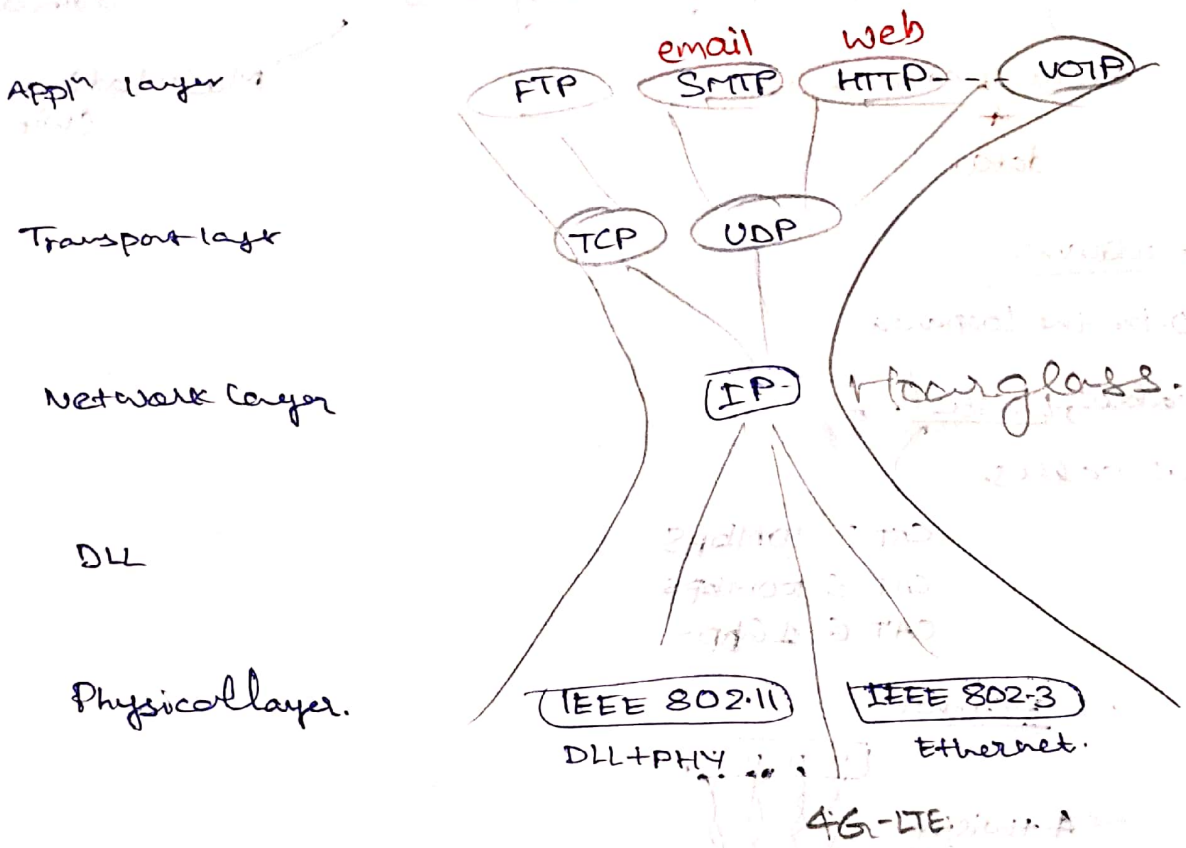
other packets.

Congestion control:- protocols exist...

Queue buildup.

Handeled by TCP. or transport layer

* Hourglass model:-



I) Physical layer:

Converts bits into physical signals → line coding (wired)
→ Modulation (wireless)

bits: Headers of other layers + data.

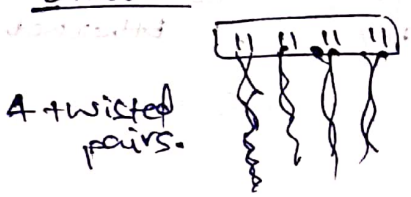
Wired medium:-

Gotta O-ize the looparea.

- wired cables, twisted.
- co-axial cables.

CAT 3 10Mbps
CAT 5 100Mbps
CAT 6 1Gbps

Ethernet:

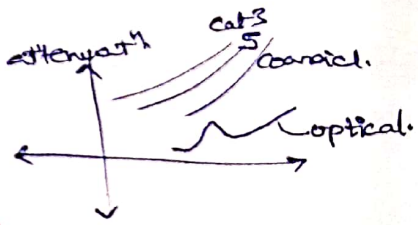


- optic fiber.

→ Attenuation:-

Attenuation in dB = $10 \cdot \log_{10}\left(\frac{P_{in}}{P_{out}}\right)$ dB

$= 20 \log_{10}\left(\frac{A_{in}}{A_{out}}\right)$ amplitude



usually: dB/1000 ft.

3dB → $\frac{1}{2}^{th}$ power transmitted.

* as distance ↑; attenuat^n ↑.

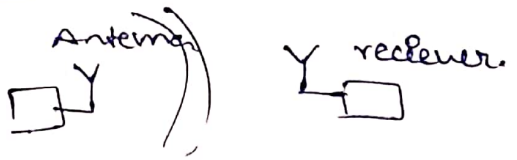
→ Power in dBm, dBW:-

$P_{dBm} = 10 \cdot \log_{10}\left(\frac{Power}{1mW}\right)$

∴ 3dBm → 2mW.

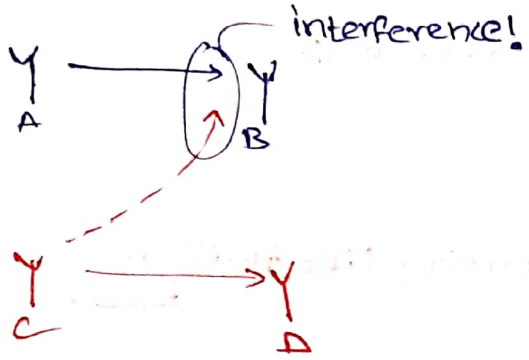
0dBm → 1mW.

Wireless:-



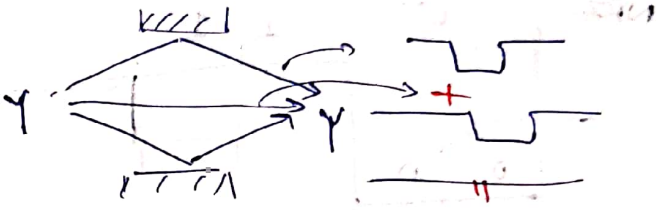
- directional antennas.
- MIMO — massive MIMOS.

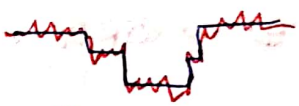
problems in wireless:-

1) 

2) Obstructions

3) Multipath effect.



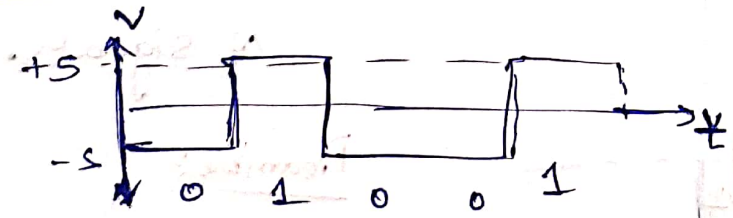


& noise too!

Signalling:- bits to SIGNALS.

1) Wired :- aka **LineCoding**.

i) NRZ: non return to zero:-



Issues:-

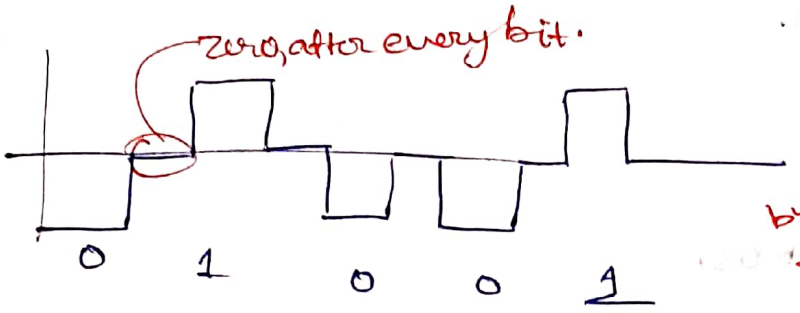
1) clock recovery: (reciever clock may not match sender's)

2) Baseline wander:-

∴ we prefer "changing" signal.

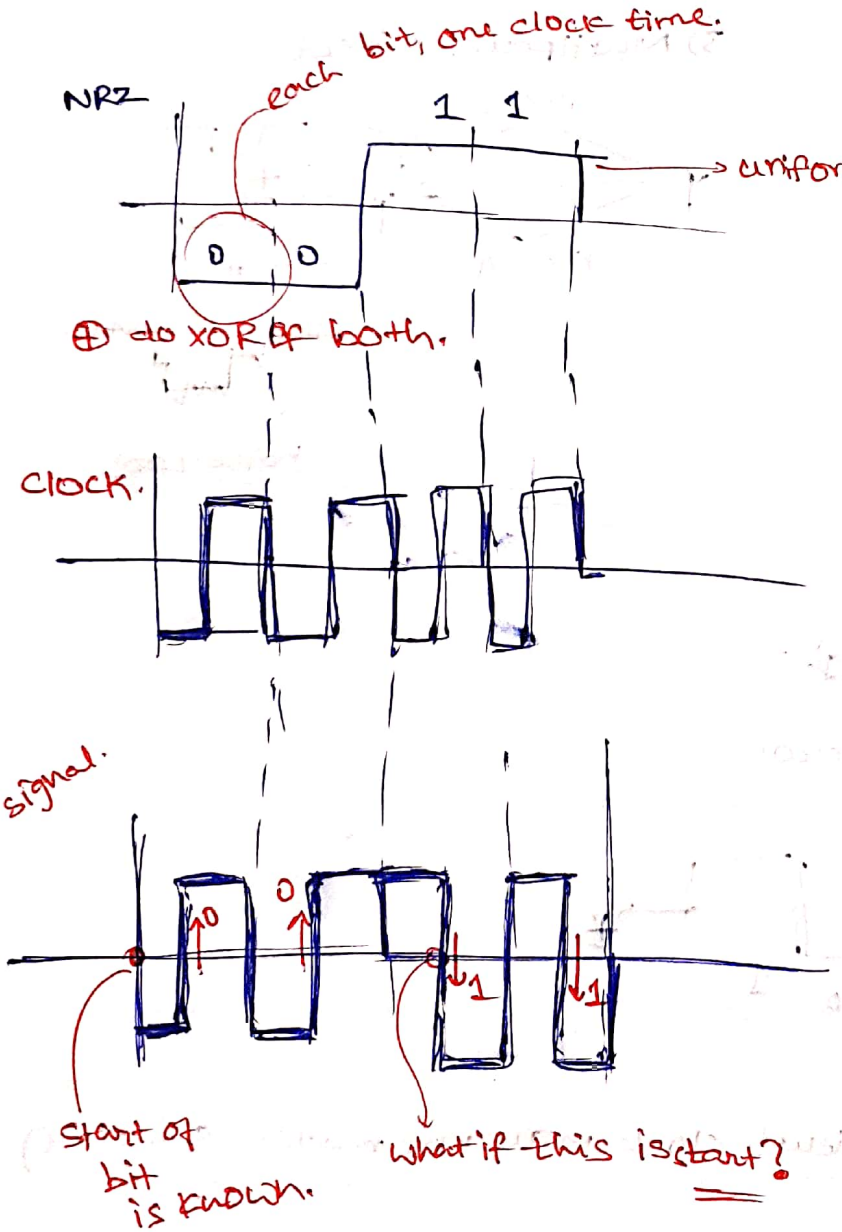
reciever; due to some HPfilter.

2) Return to Zero (RZ):-



but 3 voltage levels!
3 wires needed! X lol.

3) Manchester Coding (IEEE 8023, ethernet):-



issues:-

1) Need to know, when a signal starts

2) Need correct polarity of signals.

Preamble:-

known signal, helps synchronize sender with receiver.

4). Differential Manchester coding.

3) 4B|5B coding:-

4Bits } → written & sent as
2^4 } 5 Bits;

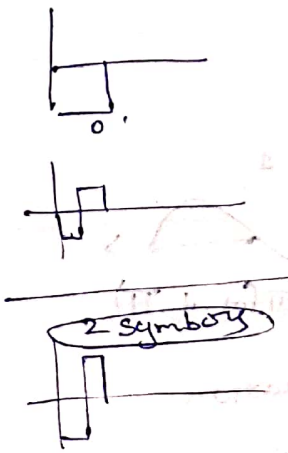
So that we dont have more that 3 consecutive zeros.

→ BAUD RATE:-

bit-rate: no. of bits transfered per sec.

baudrate: no. of distinct symbol changes, made in the medium, per sec.

Eg: In manchester coding:-

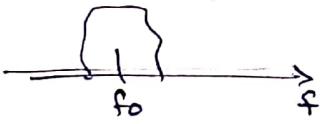


∴ Baud rate = 2x Bit rate.

relates to frequency in the wires.

∴ Baud rate more relevant to physical medium.
- its limitations for attenuation
!
!

→ MODULATION:



(after F.T. of original wave)

each signal carrier is alloted a bandwidth.

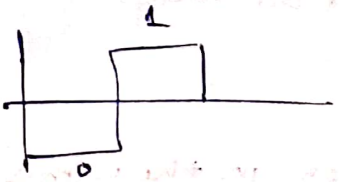
* we can do FM, AM, PM (Amplitude phase modulation.)

FM: old & slow!

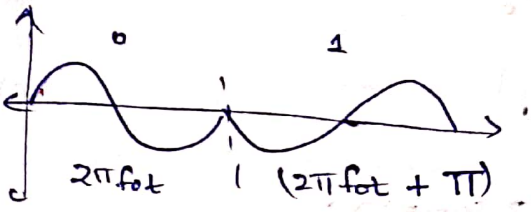
AM, PM: now ✓ for high speed data transfers

$A \cdot \cos(2\pi f_0 t + \theta)$

*

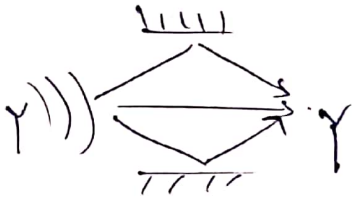


→



phase modulation.

* when s(t) is sent out;

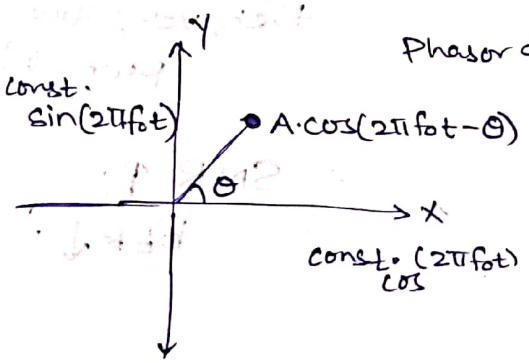


$$r(t) = \sum_{i} a_i \cdot s(t - t_i) + n(t)$$

- $\sum_i$ → multi path
- t_i → phase differences in multipath
- $n(t)$ → white noise from source (Gaussian)

* BER: bit error rate.
  SNR: signal to noise.

→ Signals in vector spaces constellation diagram.



Phasor diag... for freq. fo.
(we are not frequency modulating)

* f(t), g(t)

$$\text{inner product} = \int_0^T f(t) \cdot g(t)\, dt$$

$$\therefore \hat{e}_x = K \cdot \cos(2\pi f_o t)$$

$$\langle \hat{e}_x, \hat{e}_x \rangle = 1$$

$$\therefore K^2 \int_0^{N \cdot 1/f_o} \cos^2(2\pi f_o t)\, dt = 1$$

$$K^2 \cdot \frac{N}{f_o} \cdot \frac{1}{2} = 1$$

$$\therefore K = \sqrt{\frac{2f_o}{N}}$$

$$\therefore \hat{e}_x = \sqrt{\frac{2f_o}{N}} \cdot \cos(2\pi f_o t)$$

$$\hat{e}_y = \sqrt{\frac{2f_o}{N}} \sin(2\pi f_o t)$$

* we have r(t). we implement in circuit itself, that it "computes" $\langle r, e_x \rangle$ & $\langle r, e_y \rangle$

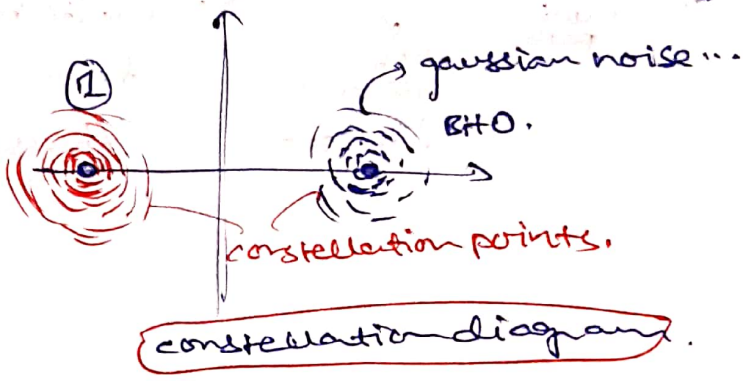
$\therefore r_x = a \cdot \langle s(t), e_x \rangle + \langle n(t), e_x \rangle$

attenuation — noise in x.

Gaussian. white noise

* if $s(t) = A \cdot \cos(2\pi f_0 t)$ Bit 0.

$= A \sin(2\pi f_0 t)$ Bit 1



constellation diagram.

low probability that bit 0's signal goes over to 1's side.

bit error

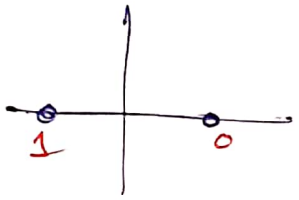
if $s(t) >> n(t)$

then, very low probability

$\therefore$ SNR ↑; BER ↓

Modulation eqs:-

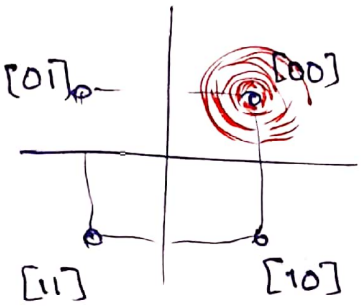
1) Binary phase shift keying (BPSK):



phase modulation.

slowest.

2) Quadrature Phase shift keying (QPSK):-

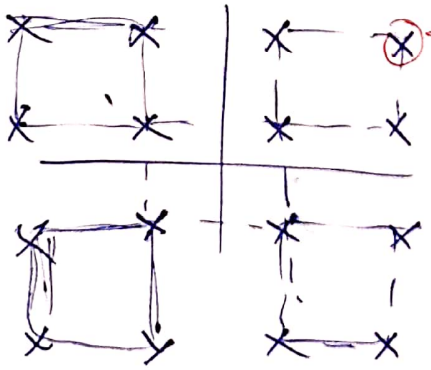


∴ Bit rate = 2x Baudrate.

only phase modulation.

3) QAM-16: Quadrature Amplitude Modulation. — family of modulatn

QAM-8
-16
-128
-32
:



each 4bits msg.

Both Amplitude, phase modulation.

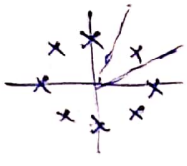
QAM-16,82... offer high speeds;

but require good SNR values.

for detecting (∵ constellation points will be far away).

4) 8-PSK:

only phase modulation.



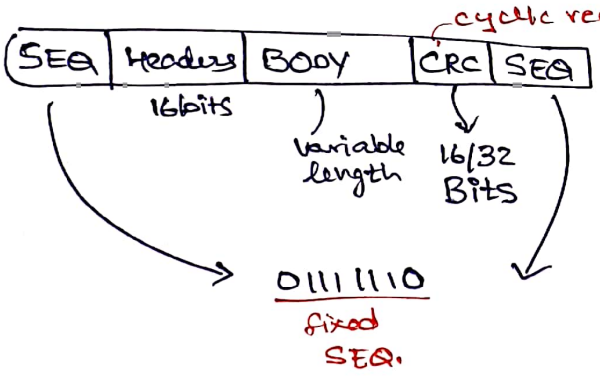
(2) Data link layer:-

protocols for when & when not to send information.

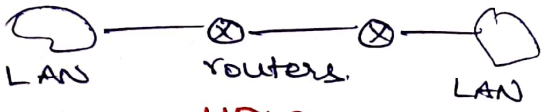
APPLN
↓
TRANSPORT — segment — TCP segments
↓
Network — packet
↓
DLL — Frame.
↓
PHY → SYMBOLS.

→ HDLC:- Highlevel Data link Control. used as layer-2 in WAN.
(wide area network).

FRAME:-



in WAN...



HDLC.

* when ideal; SEQ is sent:-

seq seq seq... (useful for clock synchronization).

NOW;

seq seq seq... |HEADER|BODY|CRC| se....

the SEQ "MAY" come here also na!

do Bit STUFFING:-

Algorithm:- (encoding) for every 5 consecutive 1s; place a zero.

Algorithm:-

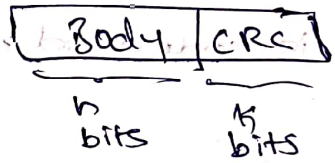
(decoding) if we encounter 5 consecutive 1's;

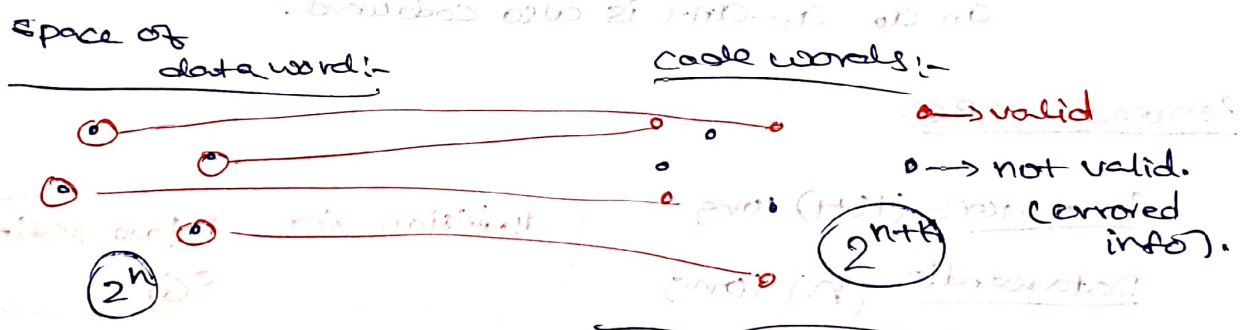
011111 | 0 ⟶ remove this & thus decoded!
| 10 ⟶ SEQ ✓
| 11 ⟶ NO chance!

∴ Bit error! discard the entire frame.

→ Cyclic redundancy Check (CRC):-

By coding theory, we try to somehow "code" the body & give a codeword (= BODY + CRC).



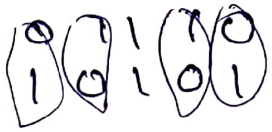


choose these, such that they are "far" apart.

So, if bit error comes, then we shouldn't be able to reach one codeword from another

→ Hamming Distance:-

→ How many bits different b/w two strings.



Hamming distance = 4. ("far")

Hence, very less probable by bit errors.

We'll see **Galois Field GF(2):-**

2 elements 0,1.

Add^n, Subtr^n:- do XOR.

Multiplication:-

$(1011) \times (11)$

$=$

$$\begin{array}{r} 1011 \\ \times\ 11 \\ \hline 1011 \\ 1011\ \times \\ \hline 1110\ 1 \end{array}$$

= xor na!

* CRC; Such that.

a0 a1 -- an is codeword means,

(try to verify).

an a0 a1 -- an-1 is also codeword.

Generating CRC:-

Generator:- (K+1) long.

Dataword:- (n) long

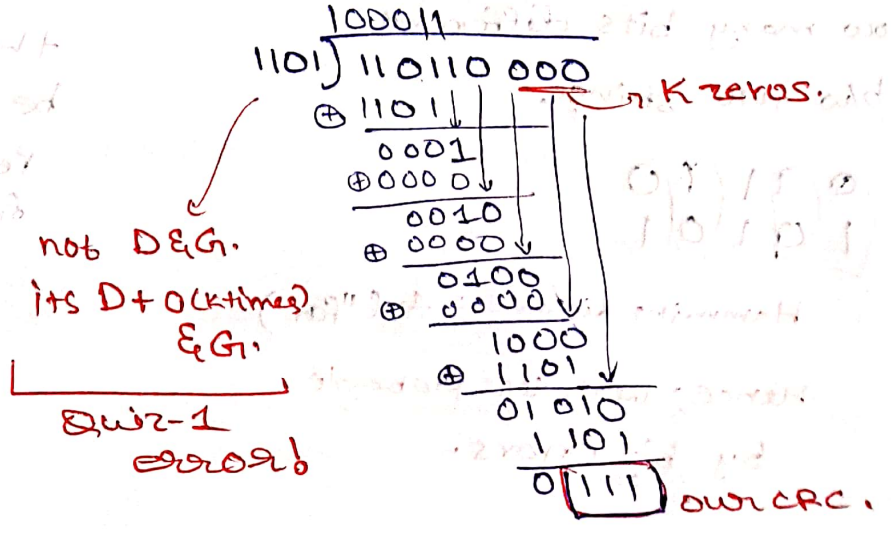
Remindr:- (K) long] our CRC.

division for D|000..(K times) E&G

codeword = Data + CRC (append)

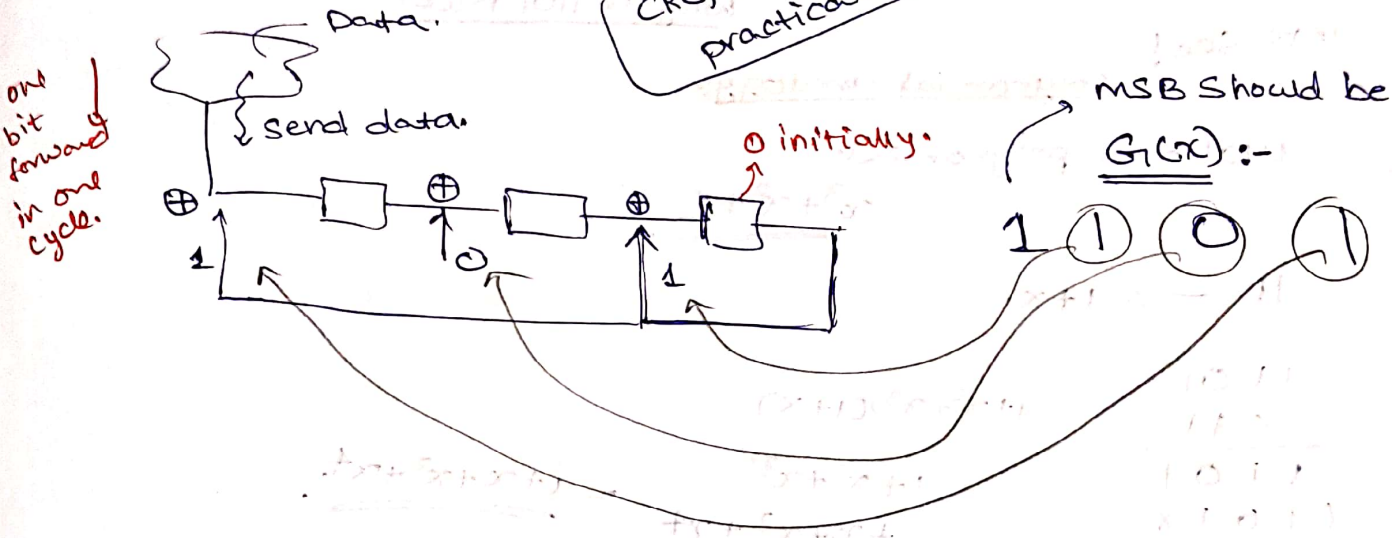
Eg: G = 1101

D = 110110



* easy shift register circuit:-

CRC; easy to generate practically.



(same as division!)

Last 3 bits will be reminder! (from right → MSB to left → LSB).

* So, now the reciever recives the code word.

How to decrypt?

Data/CRC.

Data/000

↓ CRC compute

≡ CRC sent

✓

recalculate CRC for data.

Data/CRC

↓ CRC compute

≡ 000

✓

will get 0; when divided data/CRC.

* What is a good generator for CRC?

Why is 1101 nice?

lets, see!

Polynomial analogy:-

* 1101 ⟶ polynomial:

x^3+x^2+1

11 ⟶ $1+x$.

```
 1 1 0 1
 x 1 1
-------
 1 1 0 1
 1 0 1 x
-------
 1 0 1 1 1
```

$(1+x^2+x^3)(1+x)$

$= 1+x^2+x^3+x+x^3+x^4$

$= 1+x+x^2+x^4$.

add^n XOR

* P(x) for 1001 is x^3+1. $\frac{P(x)}{G(x)} = 0$

codeword. ↳ codeword

let error in transmission be E(x) = 00---010---

for 1 bit error.

Recieved = P(x) + E(x)

↓ in GF(2).

NOW; G(x) → generator.

NOTE! $\frac{P(x)+E(x)}{G(x)} = 0$; then we say, no error.!

even if $E(x) \neq 0$.

∴ for 1-bit error detection;

$E(x) \neq 0$; then $\frac{P(x)+E(x)}{G(x)} \neq 0$.

$\frac{P(x)}{G(x)} = 0$ (∵ P(x) is codeword)

∴ $\frac{E(x)}{G(x)} \neq 0$. ⟶ $\frac{x^k}{?} \neq 0$

Say G(x) is of form 1(.)(.)(.)---(.)1.

then $G(x) = 1+x^k+(\ )$.

never divides (x^k)

P.T.O.

* How to say $1+x^{k+1}+\dots$ never divides x^i?

this is GF(2)!

not intuitive...

By contradiction:

XOR swallows terms.

Say

$$(x^{k+1}+\dots+1)(x^p+\dots+x^q) = x^i$$

$x^{k+1} + x^2 + \dots$ — atleast 2 terms.

Can't equal x^i. (yay!)

∴ 1-bit errors are always detected. for $G(x) = 1\dots1$ model.

→ 2-bit errors:

$$E(x) = x^i + x^j \quad (j>i)$$

$$= x^i(x^{j-i}+1)$$

$$\frac{E(x)}{G(x)} = \frac{g_1(x)\,g_2(x)\,\dots}{f_1(x)\,f_2(x)\,\dots}$$ — can't be divided into smaller parts.

let $G(x) = 1\dots1$.

then x^i → not divisible.

x^{j-i+1} → can be divisible but....

$(1+\dots+x^{k-1})$ divides $(1+x^n)$ for large n.

$E(x) = \dots00\,1000\dots0001\,00\dots$ (large $j-i$)

* Order

Def^n: Smallest n s.t. $G(x)$ divides $1+x^n$ is called order of $G(x)$.

$G(x) = 1+\dots+x^k$; then we CAN have $G(x)$ with order 2^k-1.

(maths exist) Crazy good!

∴ if 16-Bit CRC;

then min. length b/w two erroneous bits $= 2^{16}-1$. for non-detection.

∴ Codeword:-

length $< 2^{16}-1$ then; we can always detect 2-error frames. (with G(x) being discussed as before).

→ handling odd no. of errors:-

E(x) = 1001-1-0-- (odd no. of 1s)

& say G(x) has (1+x) as factor.

then G(x) doesn't divide E(x). Thus error detected!

Since;

in GF(2);

(1+x) (———) = ——— eqn with even no. of 1s.

Eg:

```
  1 0 1 1 0 1
  x       1 1
-------------
  1 0 1 1 0 1
1 0 1 1 0 1 x
-------------
(1 1)(1 0 1)(1 1)
  even!
```

converse:-

if G(x) has even 1s then (1+x) is a factor.

So; if G(x) has even 1s; then we can detect odd-errored frames (and discard them)

* HDLC:-

16 bit CRC:

$$G(x) = x^{16} + x^{15} + x + 1.$$

32 bit CRC:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

(Lord!)

HDLC over. Lets see ARQ:-
(b/w WANs)

damn;

burst of errors:-

$E(x) =$ 00 --1111100-- (l bits.)

$= x^i(1 + x + x^2 + \cdots x^{l-1})$

$G(x) = x^K + \cdots + 1$

$\therefore$ $G(x)$ won't divide x^i

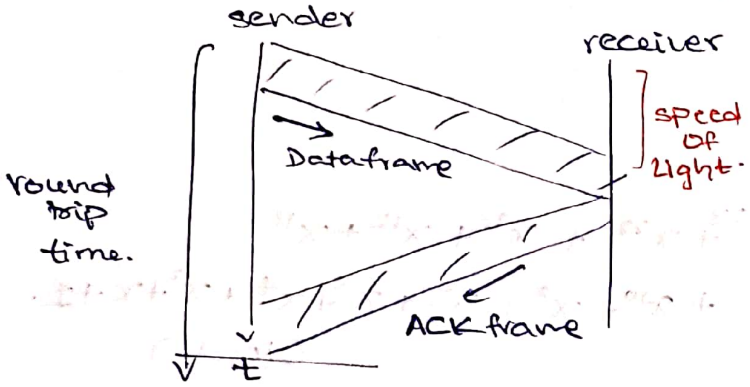
$G(x)$ won't divide $1 + x + \cdots x^{l-1}$ if $l-1 < K$

$\therefore$ bursts of length $l < K+1$ are detected.

Protocol:-

1) ARQ (Automatic Repeat ReQuest). in DLL

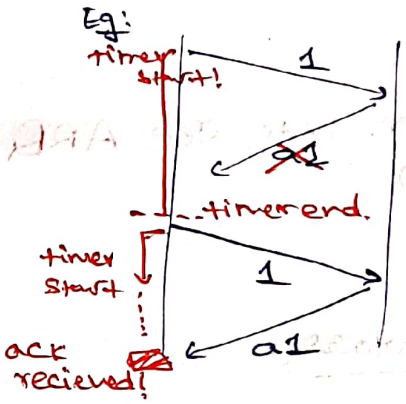
Handles reliability.



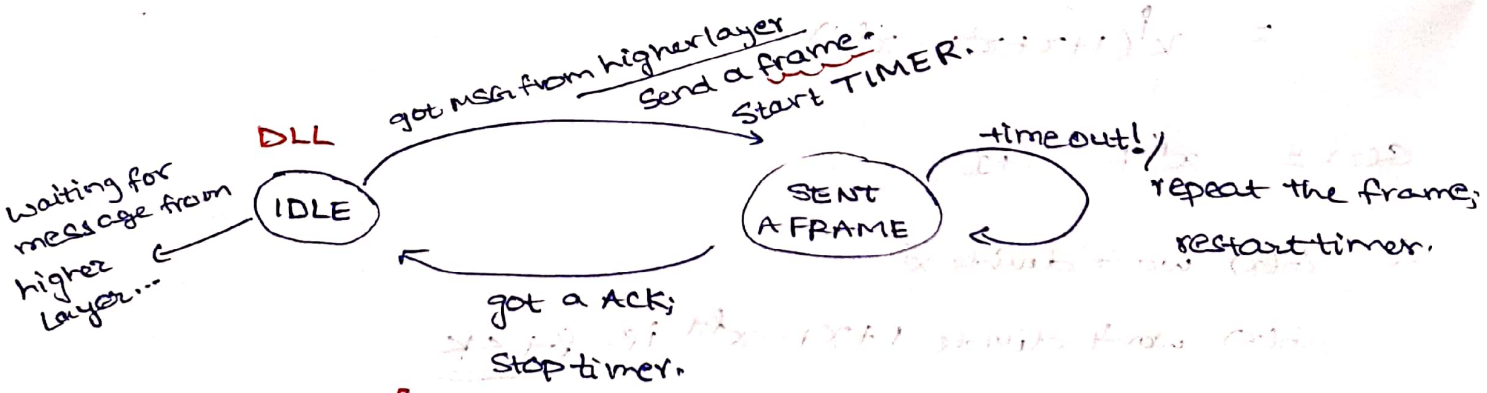
i) what if acks dont come?

- set a time out!

we did a socket coding for this na!



State diagram at sender; for ARQ ('.' receiver; just sends ACKs).



Happens inside NIC. Not Software.....

[Deque the MSG; as it has been acknowledged!

Medium Access:- (Both wired & wireless!)

* what will we do, to prevent interference of ≥2 messages?
  OR how to proceed, after we've two msgs at once?
  we can't!
  we drop that msg.

Our design is to make sure; that at any instant, only one MSG is in the medium. (Medium Access):

- if, by chance, we send a packet & 1. detect collision
  2. Don't recieve ACK;

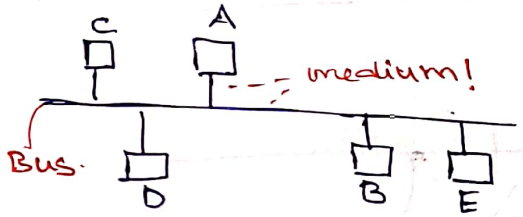
  we simply need to re-transmit that packet!.

<Looks very elementary, crude! But this gives you the 150Mbps>
!
wow!

→ ~~Wireless~~ protocols:-

Ethernet LANs:-

CSMA :-
carrier sensing Multiple Access.



* These are random access protocols:- schedule for transmission; not decided beforehand.

- unicast:- single node is destination
  broadcast:- Every node is destination.

Basic Idea:-

Create a frame, send it! (But do carrier sensing first!)

- If we get a collision; Stop transmission; wait for RANDOM time & repeat!
  How to detect?
  RANDOM BACKOFF.

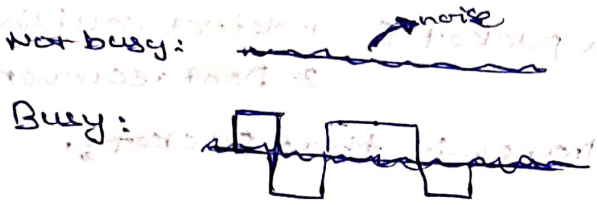
IEEE 802.3:- Ethernet.

CSMA - CD:- No Acknowledgements.

Collision detection. (not possible in wireless; heavy attenuation).

1. carrier sensing:- (is possible in wireless, but "HIDDEN TERMINAL ISSUE").
   is the medium free, or busy?

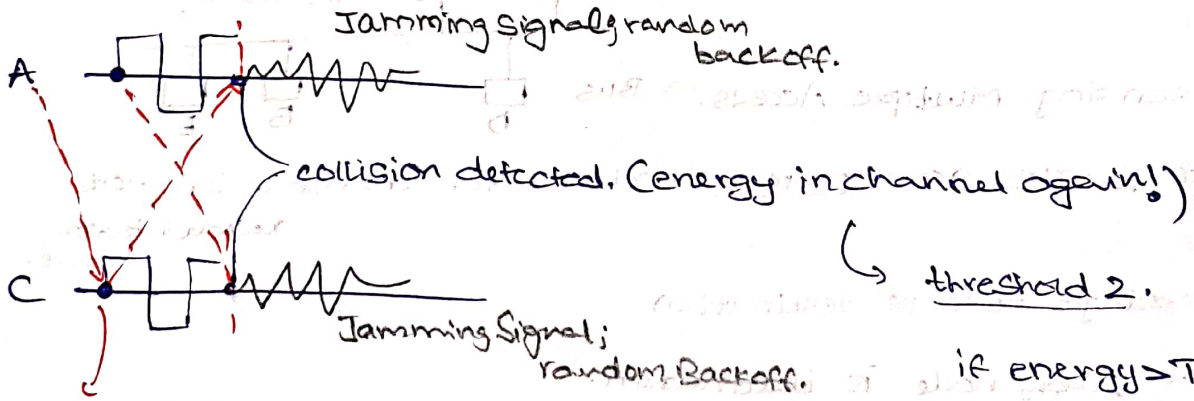
   > Detect energy.



∴ if energy > threshold 1; channel is busy.

just above noise energy.

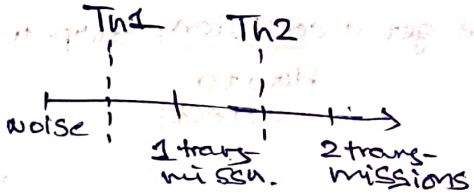
2. Collision Detection:- (not possible in wireless; try Collision Avoidance)



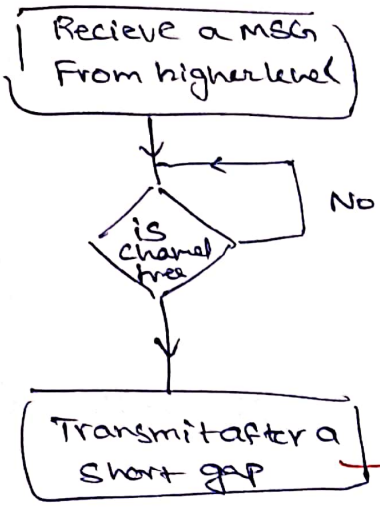
↳ threshold 2.

if energy > Th2; collision detected!

at this instance; when C does carrier sensing; Channel is free. So transmit!

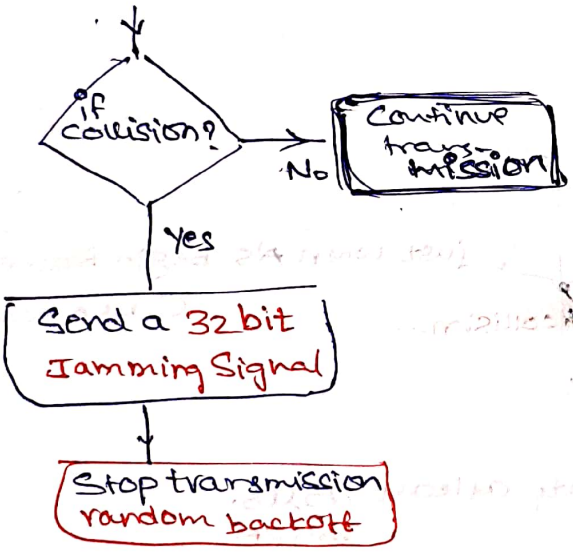


Flow charts:-

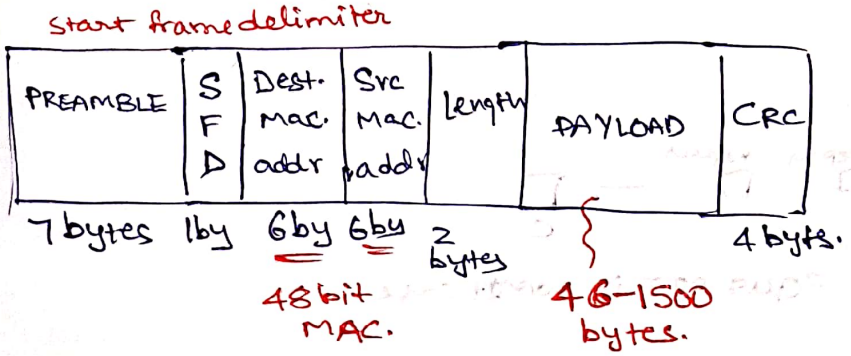


↳ for destination to process the previous packet. CRC check!

Collision detection:-



→ Frame details in 802.3 :-

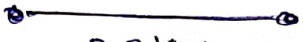


64 - 1518 bytes

Why this min-limit?

Why this max-limit? → to give others also, access to channel
→ even if 1 bit error; we need to discard frame.
memory in NIC goes up!

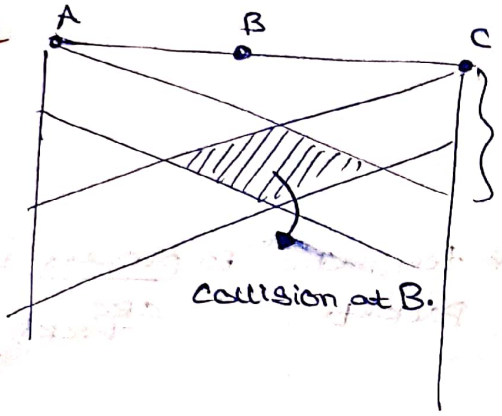
→ Why the min-frame rate?

Ethernet: 

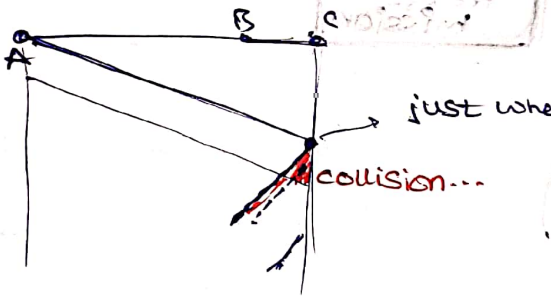
$RTT = \frac{5000}{2 \times 10^8} = 25\mu s.$

✓ not $3 \cdot 10^8$

A must detect collision; before his frame ends. (since there is no ACK in CSMA-CD)



∴ worst case will be



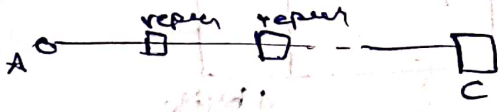
just when A's begin frame comes. Coll. starts!

∴ A's frame must last, atleast 25 μs.

2.5 KM is range!

Hence, minimum size is imposed.

But, Scenario:-



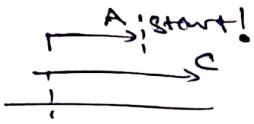
≃ 50 μs RTT in worst case...

& 64 bytes; 10 Mbps → 51.2 μs > 50 μs.

↓ the min. imposed on payload!

So that A can detect collision; before his frame (51.2 μs) ends.

→ random Backoff:-



Δ = min. wait time; 51.2 μs. give enough time to detect val.

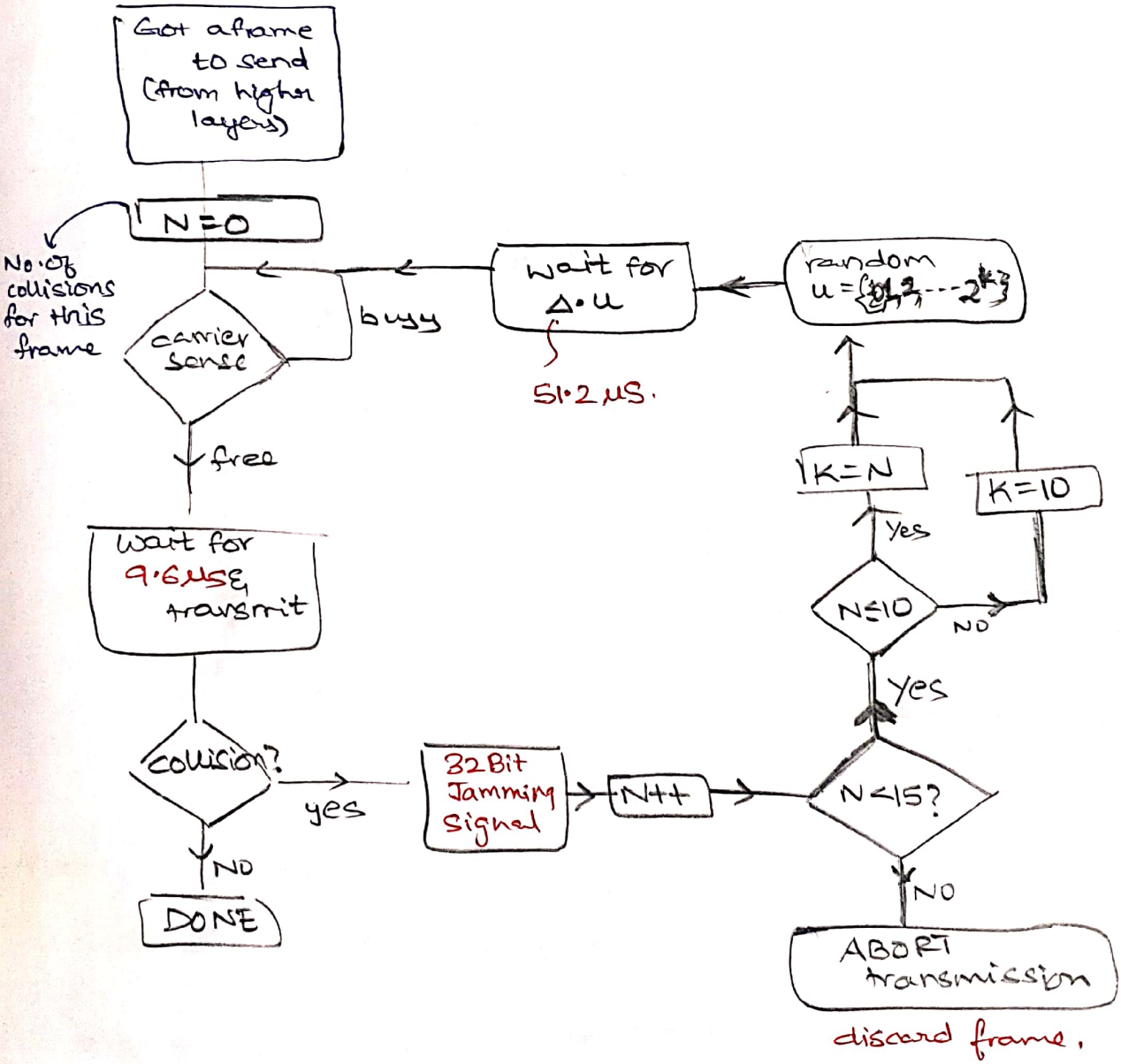
wait time ∈ Δ.u

u ∈ {0, 1, 2, ... 2^K}

K = no of collisions (consecutive)

exponential backoff.

Final, CSMA-CD flowchart:-

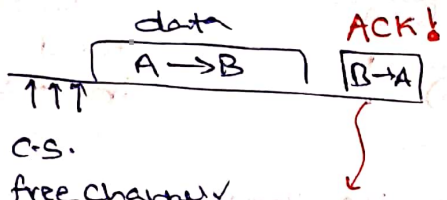


WIRELESS (WiFi):-

high attenuation! maybe CS; but no hopes for CD.

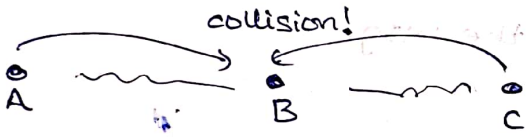
$$\frac{1}{d^{\alpha}}, 2<\alpha<5$$

so, how TO detect collisions?



NO ACK;
assume collision.
& do something.

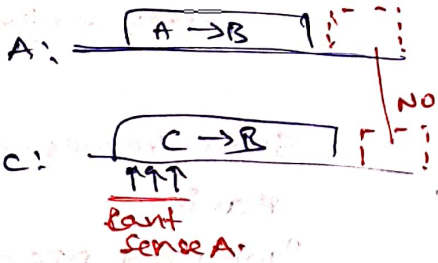
→ Hidden Terminal problem:-



B can hear A & C.

A, C can't hear eachother.

So; consider following scenario:-



NO ACK! So,... COLLISION!

Too long!

In ethernet; we would stop DATA in middle. But here; we are waiting for full transmiss!

do something!

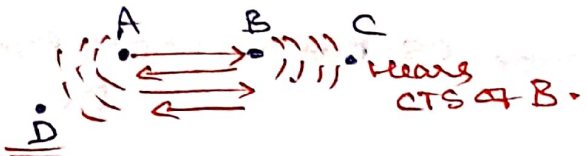
* lets do virtual carrier sensing!

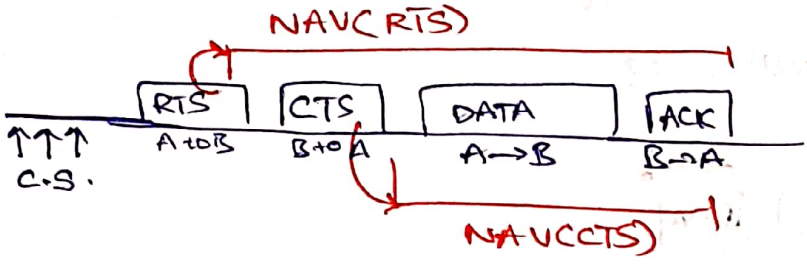
(B) will tell (C), that it is expecting (A) to transmit!

RTS: request to Send. → tells B; that it wanna send pckt.

CTS: Clear to Send. → B tells A, to continue!

short frames





Network allocation vector

stored in CTS, RTS.

Now; 'C' remains silent bcoz it reads NAV(CTS) from B.

Virtual carrier sensing DLL

> **Rule:** Any node hearing an RTS/CTS should remain silent for the NAV in the msg.

To summarize:-

option-1:-

Don't use RTS/CTS:-

when traffic is less.

Just carrier sensing → Send data → lookout for ACK → retransmit, if no ACK.

But; why won't I send RTS, CTS.... they are just short frames na...

Short frames → bits lo!

Kani;

"we need RTS CTS for a wide range of coverage".

∴ we use lower modulation technique (BPSK); which will work with low SNR.

∴ duration is long!

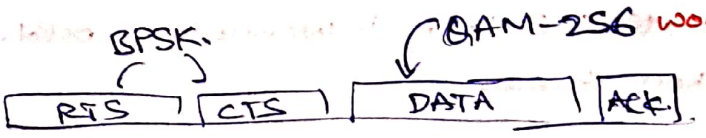
not trivial durations!

option-2:-

use RTS, CTS, carrierSensing and everything.

No CTS → assume collision.

time scale:-



PROTOCOL:-

CSMA-CA:- (WiFi)

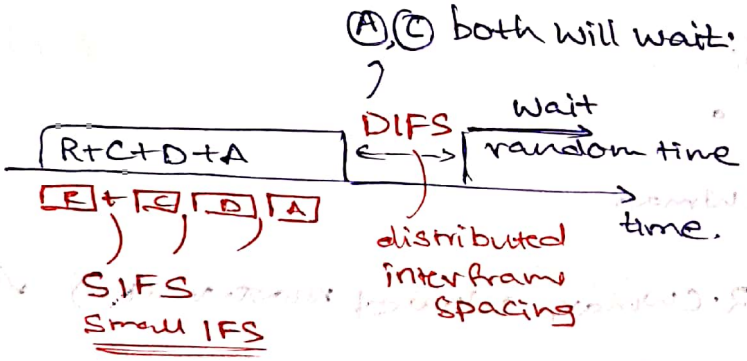
Collision Avoidance.

Exposed terminal Problem:-

In theory; A → D
B → C should be possible simultaneously.

But A → D; B hears the RTS & stays quiet.
or
due to carrier sensing.

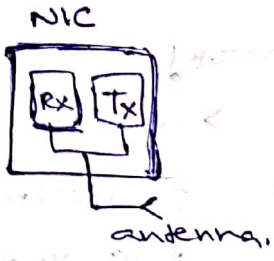
*



DIFS > SIFS

SIFS is needed, bcoz;

When RTS: A → TX, B → RX
CTS: A → RX, B → TX

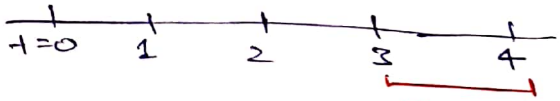


* random wait time!

contention window (CW)

Bad disagreement.

divide time into slots



9 µs.

Should allow C to hear A's RTS

C's slot = 1 + A's slot

propagation delay
+
possible offset
+
time to carrier sense.

→ How long to wait?

initialize $W \in (0, CW_{min})$

& wait for $W \times 9\mu s$.

1. decrease W by 1; for every ideal SLOT.

! 2. freeze W; if the channel is busy.

3. Start from this W only; when again waiting...

the node waiting from before, should preferably transmit.

Hence; Freeze W.

→ collision:-

detected as

- NO CTS;
- NO ACK.

Now; $CW_{min} = 2 \times CW_{min}$

$CW_{min} = \max(2 \cdot CW_{min}, \text{allowed max. value})$ ✓✓

$W = \text{uniff}(0, CW_{min})$.

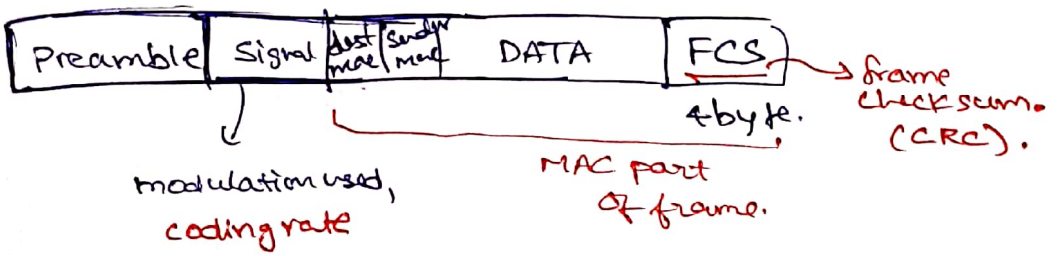
repeat:

1. wait for DIFS
2. decrement W every ideal slot
3. if channel busy; freeze W

Transmit, when W=0.

* Frame of CSMA-CA:-

IEEE 802.11 b,g,n,as,ax



data → code → sent over channel.

rate 3/4 → sent N bits; $\frac{3N}{4}$ are information bits.

5/6 → 5/6 are information bits.

			Bandwidth:-	Speeds:-
802.11g	64-QAM	, rate 3/4	20MH	54MbPs
11n	64QAM	5/6	40MHz	150MbPS
11ac	256QAM	, 3/4, 5/6	160MHz	866Mbps
11ax	1024QAM needs ↑SNR	, 3/4, 5/6	160MHz	1.2Gbps → SISO rates...

* DIFS > SIFS

DIFS = SIFS + 2x slot time (9μs).

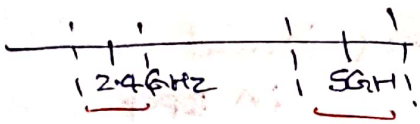
PIFS = SIFS + 1x slot time.

* Regarding QoS,

we can give preferento video call latency; by using smaller CWmax limits

But cross-layer interaction.

*



unlicensed bands

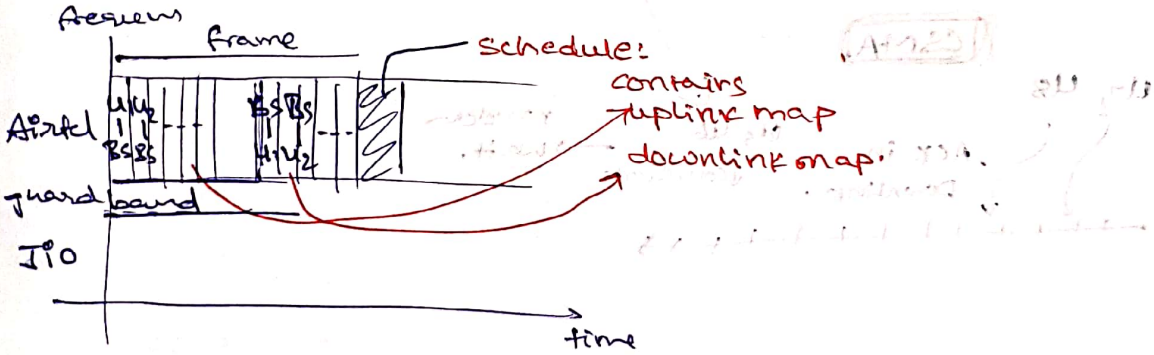
* say I got a licensed bandwidth. Do I still have to do CSMA/CA?

Hi! I'm Airtel

can I save on this? I've gots crores of people... :(.

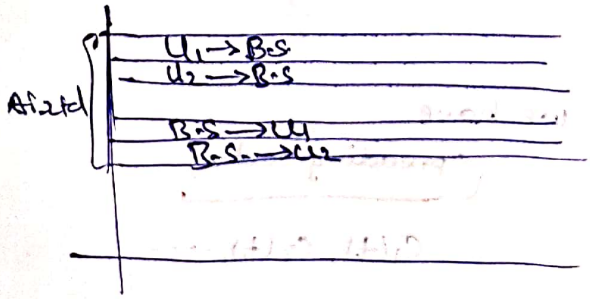
phew! lets see...

→ TDMA: time division multiple access



(or)

→ FDMA: Frequency division Multiple Access

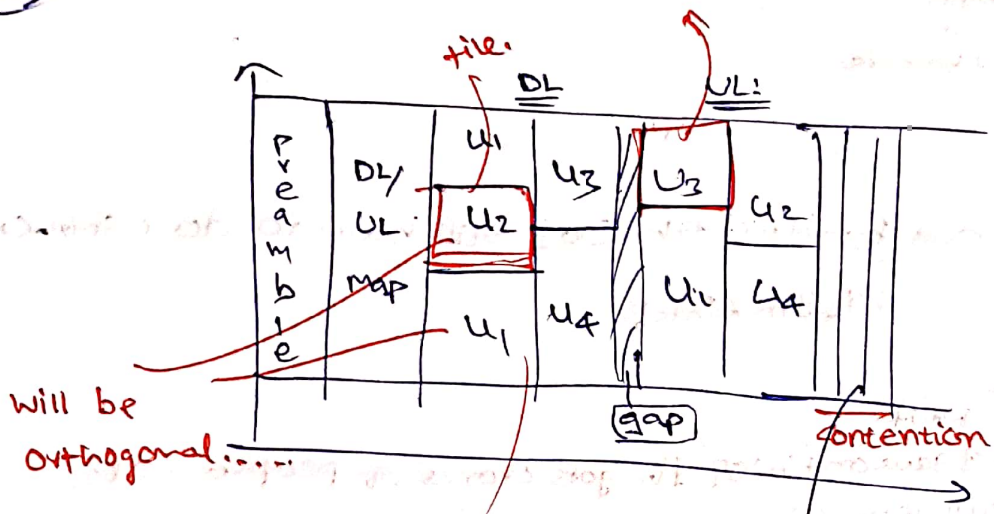


looks nice na? ya...

* 4G uses OFDMA: orthogonal Frequency division Multiple Access.

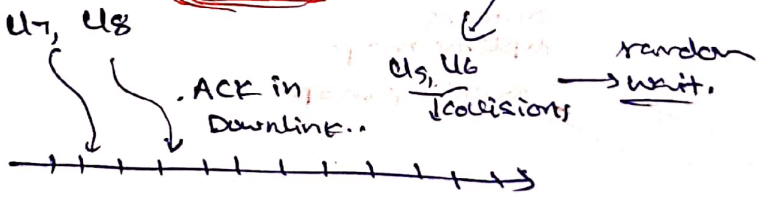
OFDMA

center frequency is used in BPSK QAM !

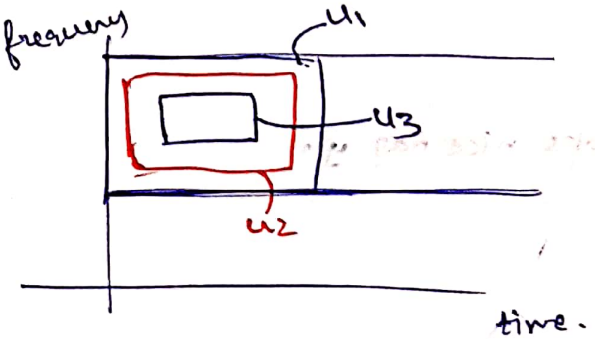


this map is dynamic; caters to demand.

CSMA



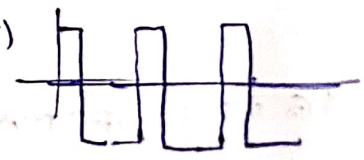
CDMA:- Code Division Multiple Access:-

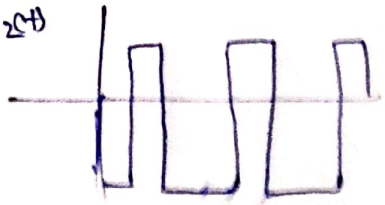


we have Spreading codes

$C_1(t), C_2(t), \cdots$

like $C_1(t)$





All spreading codes are mutually 1st,

$|C_1(t)^2| = 1$

$C_2(t)^2 = 1$

$\int_0^T C_1(t) \cdot C_2(t) = 0$

$u_1 = c_1(t) \cdot A\cos 2\pi f_c t$

$u_2 = c_2(t)\{A(t)\cdot\cos 2\pi f_c t\}$ → different constellation points.

somehow; CDMA also mitigates multipath issues.

CDMA, OFDMA

robust to multipath

* to recover from $r(t) = u_1 + u_2$;

dotproduct with $c_1(t)\cdot\cos 2\pi f_c t$.

$u_1 \cdot c_1(t)\cdot\cos(2\pi f_c t)$

$= \frac{A}{2}(1+\cos 4\pi f_c t)\cdot 1$.

remove by LPF.

$u_2 \cdot c_1(t)\cdot\cos(2\pi f_c t)$

$= \frac{-A}{2} c_1(t)c_2(t)(1+\cos 4\pi f_c t)$

remove by LPF

∴ we have

$\frac{A}{2} + \frac{-A}{2}c_1(t)c_2(t)$

Now! average over time!

$= \frac{A}{2}$. ∴ we recovered $c_1(t)$'s equn.

Switching:-

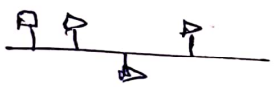
at layer2 (ethernet).
also at layer 3 → routers.

* use MAC addresses to switch, in layer-2. (Amplify only necessary packets)

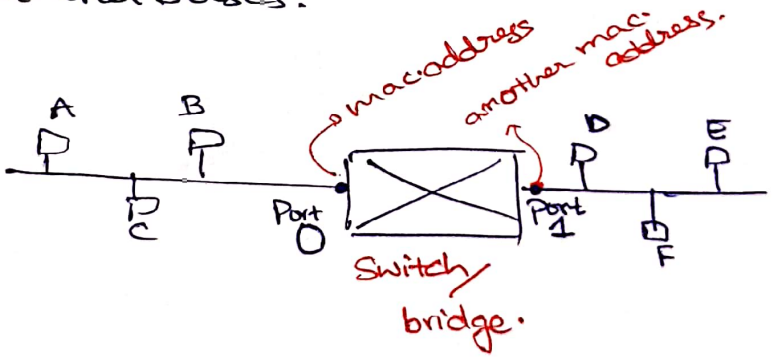
use IP address; for layer-3.

* the lan bus; can have only so many devices connected...

since more devices → Much more collisions.

∴  } not scalable by itself.

*we use bridges / repeaters / L2 switches to connect different ethernet buses.

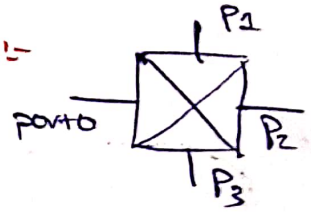


every ethernet module manufactured, must have different MAC-address.

2^{48} possibilities.

* The switch is supposed to send messages from one side source to other side destination, & ignore src-dest. on same bus.

can have more than 2 ports:-



- for the SWITCH to behave in such an intelligent manner, it needs FORWARDING TABLE.

This needs to be dynamic; bcoz, what if device A is removed from one bus, & connected to another bus?

FORWARDING TABLE:-

Dest.	Port#

1. Initially empty.
2. Say a frame has come-in;

	SRC	DEST
	A	B

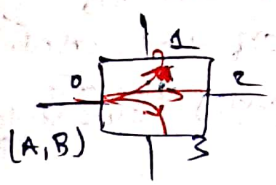
Now, it registers the sender, with the appropiate port.

append (A, 0)

Dest.	Port #
A	0

3. Since B isn't in our forwarding table, we forward it via all other remaining ports.

P# 1, 2, 3. ✓



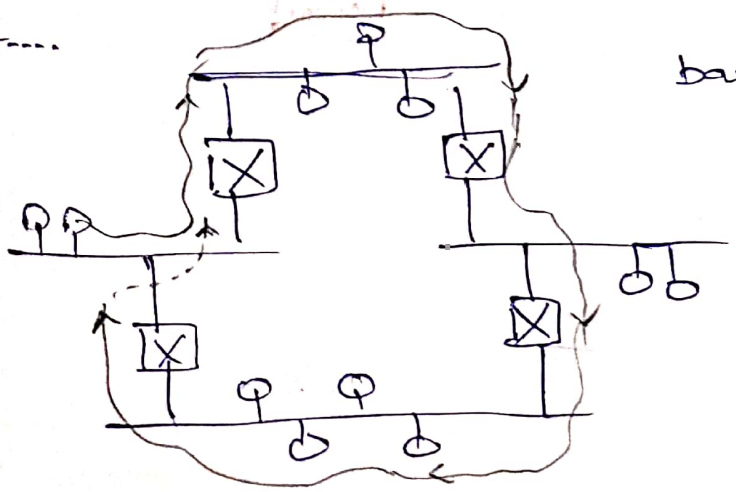
4. Like this, SWITCH builds up its forwarding table.
5. Set an expiryTimer on each entry. reset timer to full whenever we recieve a frame, from the same MAC.

This allows the BRIDGE to delete unexisting MAC; & update port # in case a device is connected to another.

what about collision detection?

* A huge issue:- LOOPS

What if a loop exists,? Some frames will be repeatedly sent....



bandwidth will be exhausted like this.

→ SPANING TREE PROTOCOL by radia perlman.

* every bridge
  ↳ has rootport.

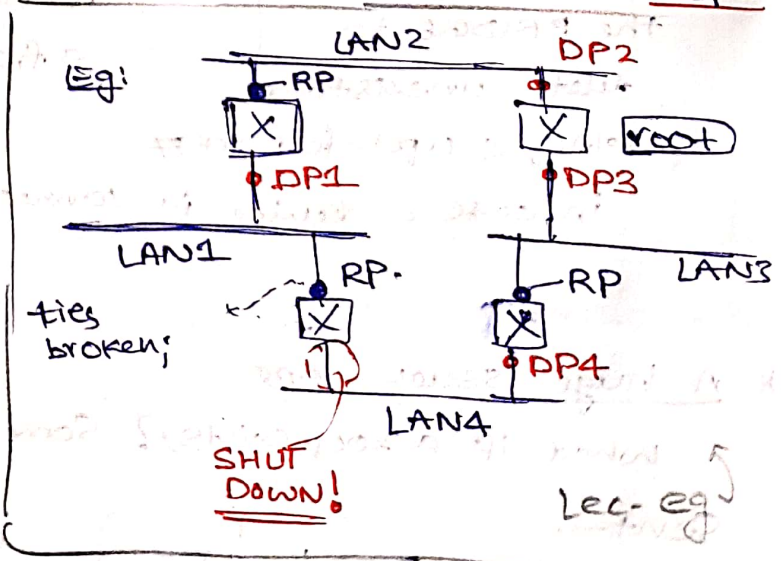
* every LAN
  ↳ has designated port.

∴ NO confusion in deciding the route of frames via BRIDGES.

→ any message; is sent into all LANS initially; & no loops!
eventually; the SWITCHES become better at deciding & then; we'll have efficient repetion of frame.

1. elect a root bridge. (one with lowest Bridge ID).
2. Every other bridge finds which Port is closest to root & make it rootport! (we have lower neighbour switch ID, for tie-breaking)
   bridges have many ports
3. All bridges connected to a LAN, ethernet bus. elect one among them to forward frames of this LAN. (designated port) the one closest to root.
4. Any port; that is not a root port; nor a DP; is shut down.!

In this manner; we avoid loops.



Lec-eg

Details:-

1) Elect root Bridge:-

each bridge has Bridge ID:-

BRIDGE ID:- [configurable | MAC → of one of its ports. (the least mac).]
2 bytes | 6 bytes

- the bridge with the lowest ID becomes root.

default: 32768

values: 0 – 61440
in multiples of 4096. (2^{12})

Set to 0, for some bridge; if you want it to be root.
"configurable".

→ How do bridges talk:-

during the election Algorithm:-

- each bridge broadcasts

(Y, d, X)
- Y: the smallest ID till now
- d: distance from the smallest ID
- X: my ID

∴ initial:-

SW1: (1,0,1)
SW2: (2,0,2)
⋮
SW4: (4,0,4)

→

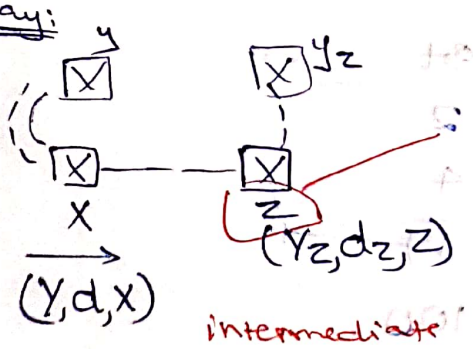
SW1: (1,0,1)
SW2: (1,1,2)
SW3: (1,1,3)
SW4: (2,1,4)

→ ...

converge:-

1,0,1 → distances; also depend on wire speed.
1,1,2
1,1,3
1,2,4

Say:



intermediate step:-

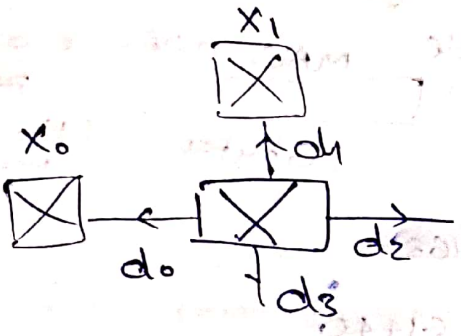
if y < yz:-
then (Yz, dz, z) ← (Y, d + dist(X,Z), Z)

if y = yz & d + dist(X,Z) < dz:-
dz = d + dist(X,Z)

varies; based on throughput rate of ethernet.

2) root port of a bridge:-

Smallest distance to root ⟶ root port of bridge! ✓



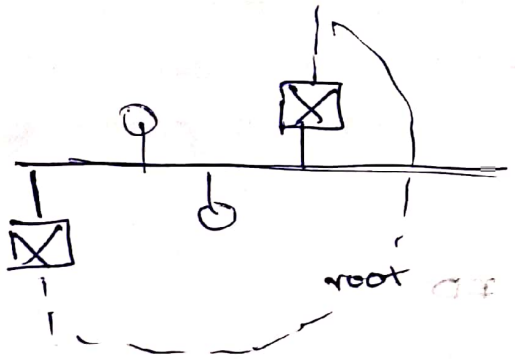
say $d_0 = d_1 < d_2 < d_3$

tie breaking, based on ID of neighbouring switches of ports.

Why not MAC of ports themselves?

maybe, some profit in Spanning tree.

3) Designated port of a LAN:-



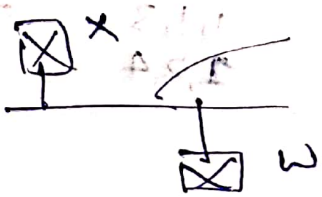
- See the distance of port from root.

tie breaker:-

See the minimum SWITCHID of corresponding ports.

4) Finally; PORTS of bridges which aren't (root port) (wrt a SWITCH) or (designated port) (wrt a LAN) are SHUT DOWN.

* in practicality;



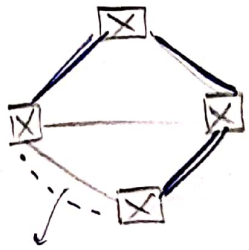
dist (X,W) ≠ 1

depends on speed of wire:-

Speed	Cost
10Gbps	2
1Gbps	4
100Mbps	19
10Mbps	100

. okay! But is this it? Will Layer-2 switching help me interconnect globally?

→ Shortcomings of Ethernet switching:-



direct port, not used!

1) path choosed is not optimal

2) some links are not used at all.

3) Forwarding tables are O(n)
bcoz FLAT ADDRESSING.
no meaning in MAC address.

Layer-3 (IP): Hierarcial Addressing.

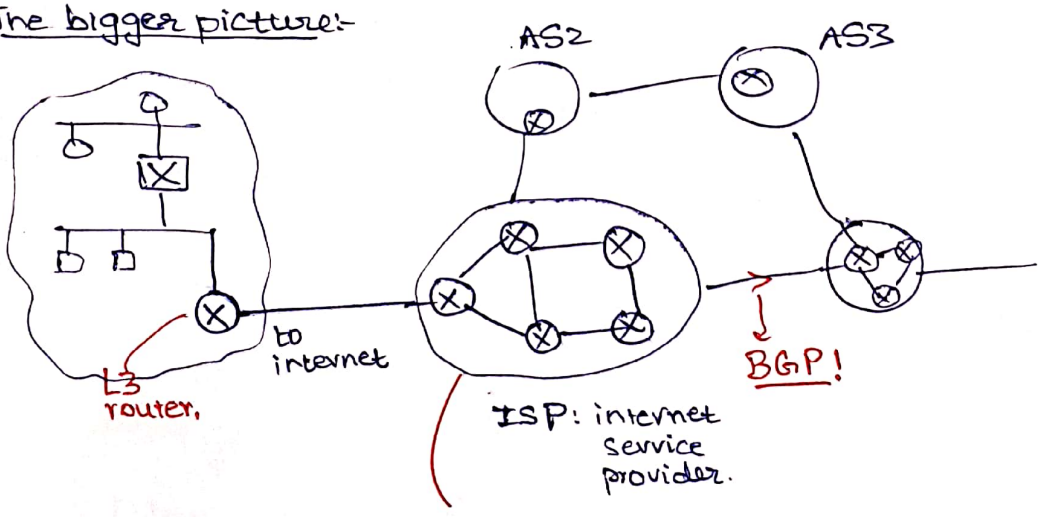
4) Stability is an issue for large SpT.

If some switch, link fails; whole SpT is reconstructed.
detected when 'Hello' messages stop.
large Network → more reconstruction.

Some more issues:-

- No common addressing scheme across globe
- No common communication protocol.

→ The bigger picture:-



Autonomous systems AS.
- can choose any internal routing protocol.
Eg: least hops, least latency...

* Intra-domain routing : within AS.

Inter-domain routing: B/w AS.
BGP: border gateway protocol.

will enable us to go less than O(n) tables.
Hence Global Scalability.

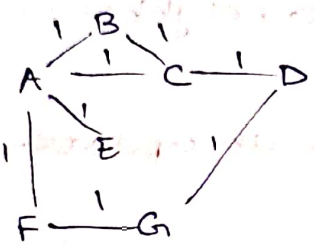
→ INTRA-DOMAIN ROUTING:-

1. Distance Vector based : RIP
   transmit forwarding tables — routing Information protocol

2. Link State Routing: OSPF — open shortest path first.
   transmit neighbouring topology
   IS-IS — intermediate System to intermediate System.

* All these will
  - use shortest path
  - Avoid loops.

Distance-Vector routing:- we just store the next node to be used, to reach dest.



* A sends out its 'table' to neighbours.
NOT BROADCAST.

A:

Dest.	Next hop	distance to dest
A	—	0
B	B	1
C	C	1
E	E	1
F	F	1

} initially will be there.

hears from B:-

B	-	0
A	A	1
C	C	1

from C:-

C	-	0
A	A	1
B	B	1
D	D	1

* Now; after hearing from neighbours, A updates its table:-

A	-	0
B	B	1
C	C	1
E	E	1
F	F	1
G	F	2
D	C	2

} Like this; converges for every node.

Bellman-Ford Algo.

∴ we have the forwarding tables for every node.
(& these will be shortest paths)

→ **if G-F link is down!**

G, F both send

for G:

Dest	next hop	dist
F	F	∞

→ node D will hear this; & update accordingly.

for F:

Dest	dist
G	∞

→ node A hears.

As a chain reaction, all node's tables get updated.

- **Trigger updates:-**

  Some Event triggers a routing update to neighbours

  Eg: F —X— G link failure TRIGGERS an update.

- **Periodic updates:-**

  periodically keep sharing the routing tables.

→ **Count to infinity Problem:-**

* X—A—B after convergence:-

A:

dest	next	cost
X	X	1
B	B	1

B:

X	A	2
A	A	1

Now X—A is broken:-

A: sends trigger update

X	—	∞
B	B	1

B: (hasn't updated yet)

X	A	2
A	A	1

sends periodic update.

A: sent

X	B	3
B	B	1

B:

X	—	∞
A	A	1

A:

X	B	3
B	B	1

B:

X	A	4
A	A	1

3 5 7 ... 4 6 8 ...

count to ∞.

RIP protocol sets 16 as infinity.

i.e. can't reach destination

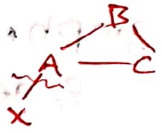
propositions:-

split-horizon:-

Don't advertise information about destination to the neighbour, if it is the next hop for that destination.

So; B doesn't tell A about X.

problem solved! is it? what about 3-member loop



split-horizon with poison reverse:-

Node tell its next hop to the destination, that distance to destination is ∞.

So; B tells (X, ∞) to A.

*

now; 3-membered loop is coming, even with Split-horizons.

→ Routing Information Protocol RIP:-

- Distance vector based.
- cost of all links = 1.
- max distance = 16 → infinite.

Advantages:- Simple to implement. (not in my opinion)

Disadvantages:-

- count to infinity & routing loops
- convergence of routing tables is slow.
- since 16 is max, can't have large networks.

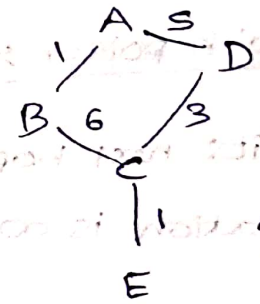
→ Link-State Routing:-

Each node broadcasts information about cost to its neighbours.

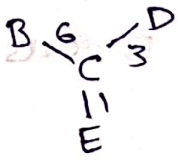
∴ each node has a picture of network topology completly.

* Apply beautiful dijkstras for min paths, from that node to graph.

& decide "next node".



C broadcasts:-



→ On link failure:-

* A ✗ F (2)

A, F broadcast that their link has failed.

- All nodes rerun Dijkstra's.

1) Advantages:-

- NO cost-to-infinity, routing loops.
- convergence of routing tables is quick.

2) Disadvantage:-

- A bit complex than DV. (I personally dont think so)

Finally, what should be the cost on the links?

speed? Latency? Queuing!

ethernet bus; joining two ports.

→ choosing link weights:-

* DV; LSR choose shortest paths.

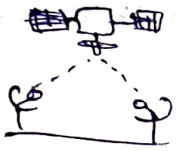
Hence, every Network Engineer has to decide... what are the weights....

Lets ask good a.k.a. ARPANET :(lol)

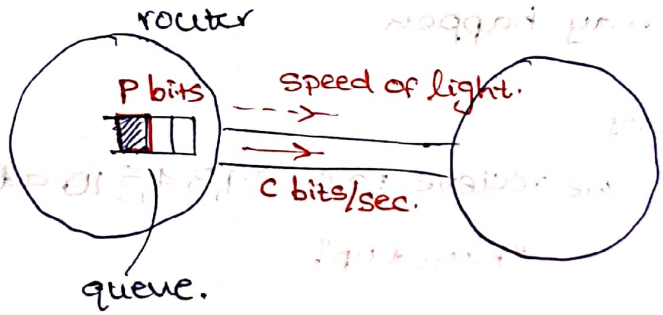
had two basic link speeds

56 Kbps — terrestial link

9.6 Kbps — Satellite link



* A link:-



idea for weights:-

use latency?

lets see.

latency = queuingdelay + transmissiondelay + speed of light delay.

latency: taken time for last bit of message to reach

queuingdelay: demand for the link.

transmissiondelay = P bits / C bits/sec

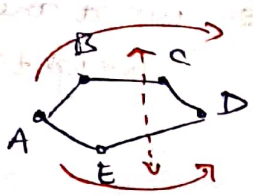
- well, looks perfect analogy for cost on links.

but latency is sooo dynamic.

→ take time window average, latency avg. of all pckts is the link weight.

* Many problems arised, by this analogy...

1. Under heavy load, routing paths oscillate ! (understandable)

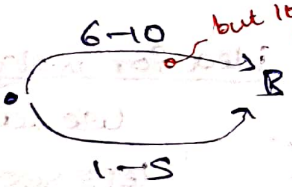


Queing goes up → increase link weights → new shortest path

2. In this manner, end-end latency varies.

May affect performance in application layer.

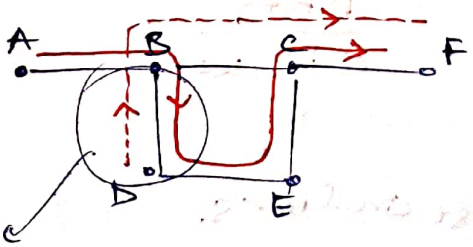
3. Reordering of packets may happen



we recieve 1,2,6,7,8,9,3,4,5,10 at B.

"messed up".

4. Routing loops are possible....



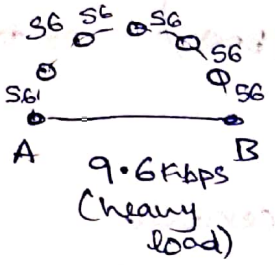
A → F

packets stuck in a loop. } since all nodes won't update simultaneously.

After convergence, we won't have these loops.

5. Range of link weights were too wide.

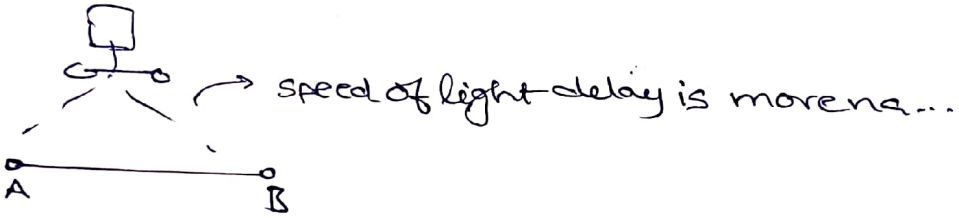
Due to which some links were penalized too much.



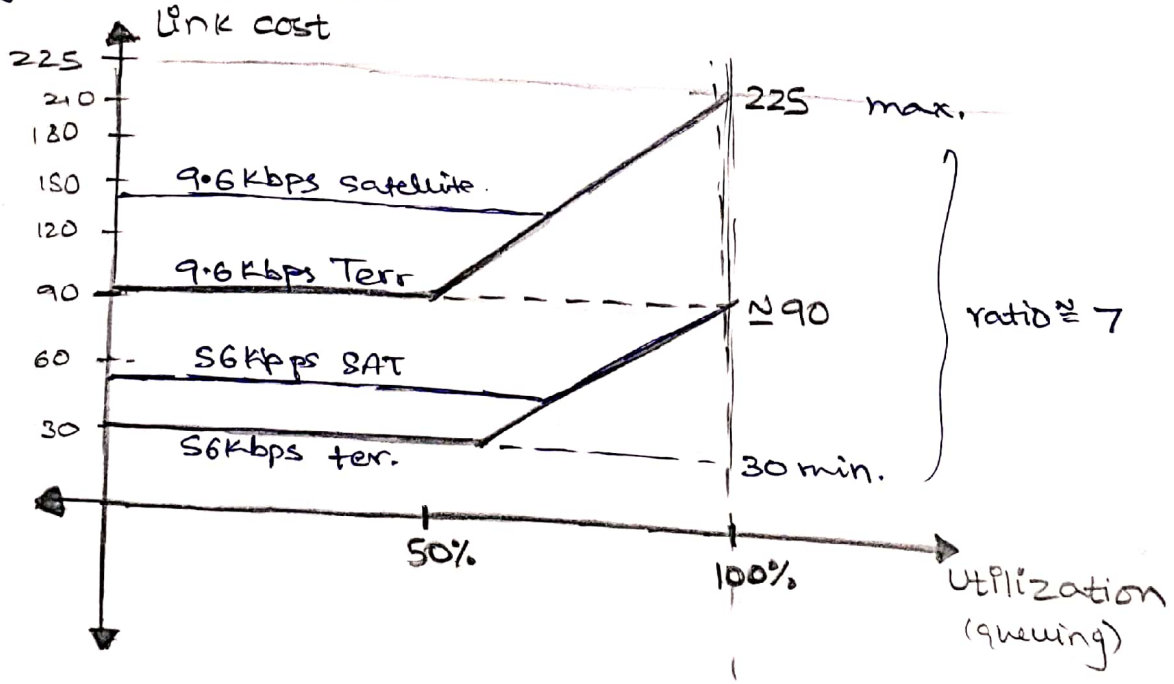
126 links of 56Kbps had same weight as one 9.6Kbps.

Why use 126 entities, when we have a direct edge.

6. Satellites were penalized too much. (low utilization)



Hence, graph was made.



& weights changed infrequently.

Today;

OSPF protocol:-

based on link state routing

$$\text{wt. of link} = \max\left(1, \frac{10^8}{\text{linespeed(bps)}}\right)$$ no consideration of traffic.

then how?

Setup a NOC - network operations center.

change weights due to traffic manually!

L18:- IPAddressing:- Layer-3: Network layer

* MAC address (48 bits) for layer 2.] Built into NIC.

IP address (IPv4: 32; IPv6: 128 bits) for layer 3.] assigned via Software.

2^{32} Not at all sufficient. → We'll use, something called NAT; (network address translation)

* IP address as (8bits).(8bits).( ).( )

Eg: 255.255.255.255 } all 1s; reserved for broadcast.

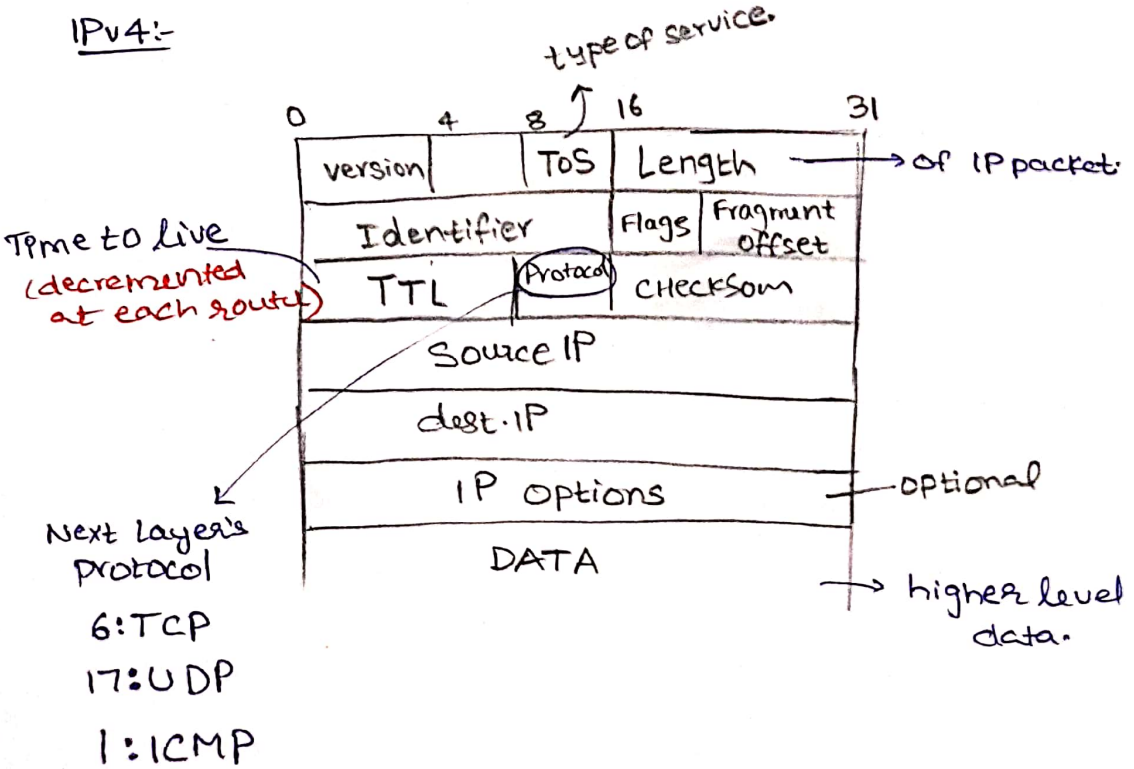
10.*.*.* }
192.*.*.* } private IP address

- blocked by routers.
- works for a network.

public IPaddress } should be unique, only one host must use this IP addr.

→ Layer 3 headers:-

IPv4:-



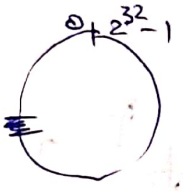
* Routing tables of routers be like,

Dest.	Next hop
730.52.30.*	R_2

Not flat addressing

IP-prefix

Hence, small sizes.



assigned to an organisation. rather than scattered IP Addresses.

Like machine's IPs be like

730.52.30.1

730.52.30.2

Network part | host part

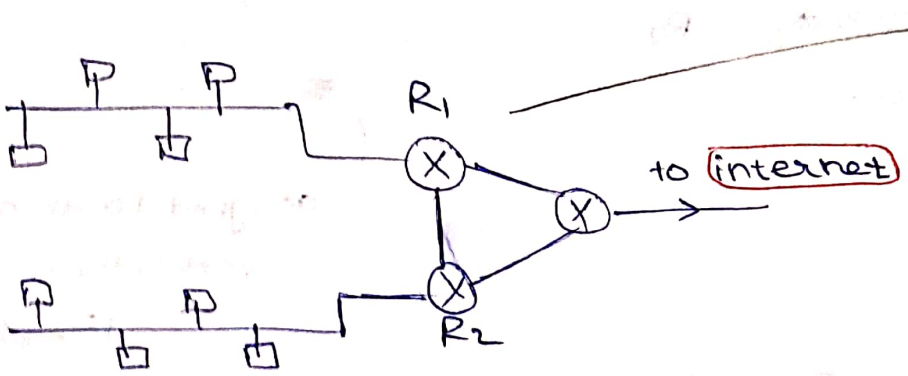
* class A: 8 bits (network) | 24 bits (host) — 2^{24} devices!
  big MNCs

class B: 16 bits | 16 bits

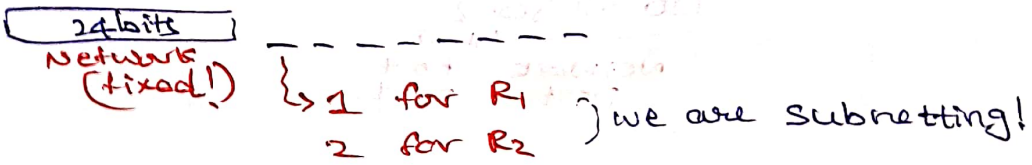
class C: 24 bits (network) | 8 bits (hosts) — 2^8 devices hmm...

→ Subnetting:-

* we are given a 'slice' of IP addresses, how to divide among LANs



maybe this is class C:-



* Subnet mask:-

says which bits in IP address to use,

to decide which LAN to route to.

Eg: to consider first 25 bits:-

MASK = 11111111.11111111.11111111.10000000 (binary)

255.255.255.128 (human readable notation)

* Subnet address:- (Is this unique?)

S_1 for LAN1 (M1 is mask)

S_2 for LAN2

* Routers be like:-

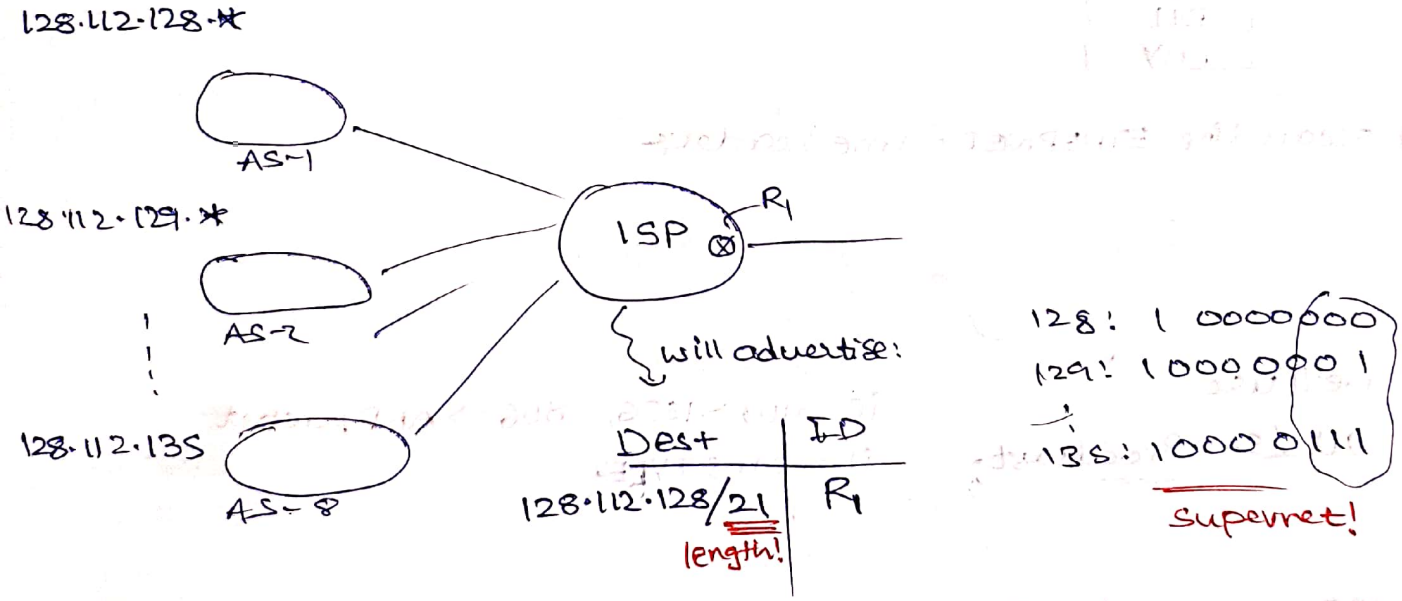
Router R_1:-

dest. IP address is 'D'.

Is (D AND M_1) == S_1?

- No → Is (D AND M_2) == S_2?
  - No → Send to internet.
  - Yes → FWD to Router R_2
- Yes → do nothing

→ Super netting:- helps to reduce size of forwarding tables.



* IP prefix: a.b.c.d/N
  - ↳ first N bits for prefix.

* CIDR:- Classless InterDomain Routing
  - no class A B C...
  - use any prefix length (N)

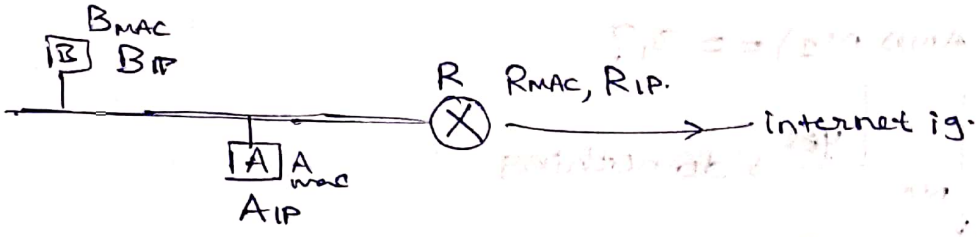
L-19 ARP & DHCP:-

DHCP → Assign IP to host.

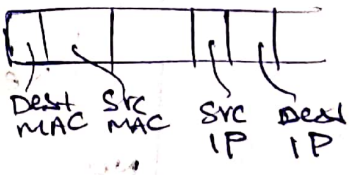
ARP → find MAC of dest.

→ Address Resolution Protocol:- Remember, IP addresses are software assigned.

where as MAC is built in.



frames will be:-



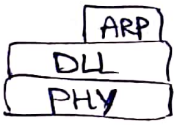
* A wants to send to B:-

A knows B_{IP}; but doesn't know B_{MAC}.

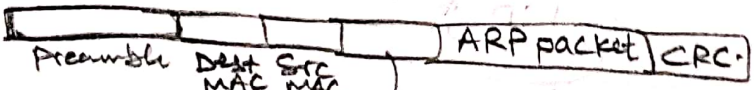
How? server?

→ ARP to the rescue:-

where's ARP in the stack?



* recall the ETHERNET frame header:-



we'll use all 1s → Broadcast.

Type/length.
if value > 1536, it means TYPE.

806 ⇒ ARP packet.

?? < 1536 na?

* ARP request (by A):-

(A → all)

SENDER MAC; A mac

Sender IP; A IP

Target MAC 00000000---

Target IP B IP

* ARP reply

(from B to A)

unicast here.

Sender MAC , Sender IP
Bmac Bip

Target MAC , Target IP
Amac AIP

* Information: B_{MAC} ; Stored in ARP cache of A, with a timeout.

Q1) How does A know, if dest. IP belongs to same network?

A: Using Subnet mask.

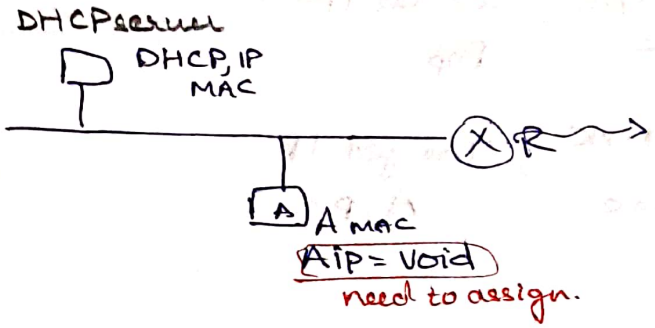
if (DEST IP) AND (MASK) = (AIP) AND (MASK)

then same network.

Q2) If DEST IP is different; we'll need routers R_{IP}, R_{MAC}.
we'll get it from DHCP server

A: R_{MAC}; by using ARP.

→ Dynamic host configuration protocol:-



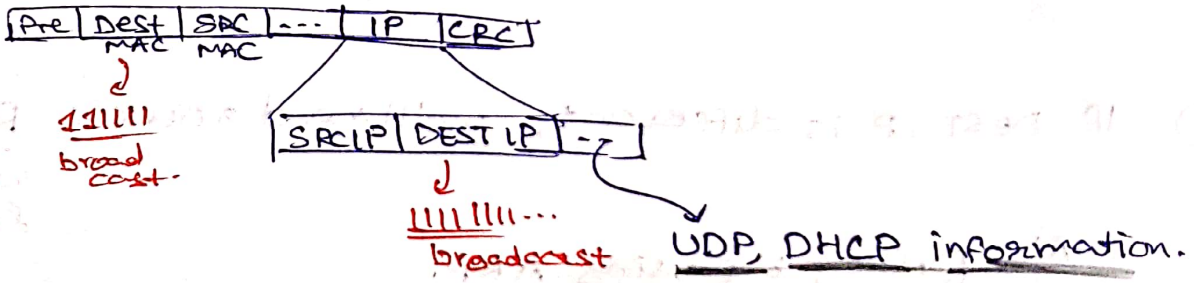
* in protocal stack:-

DHCP ——→ port # is 68: DHCP server
67: DHCP client.

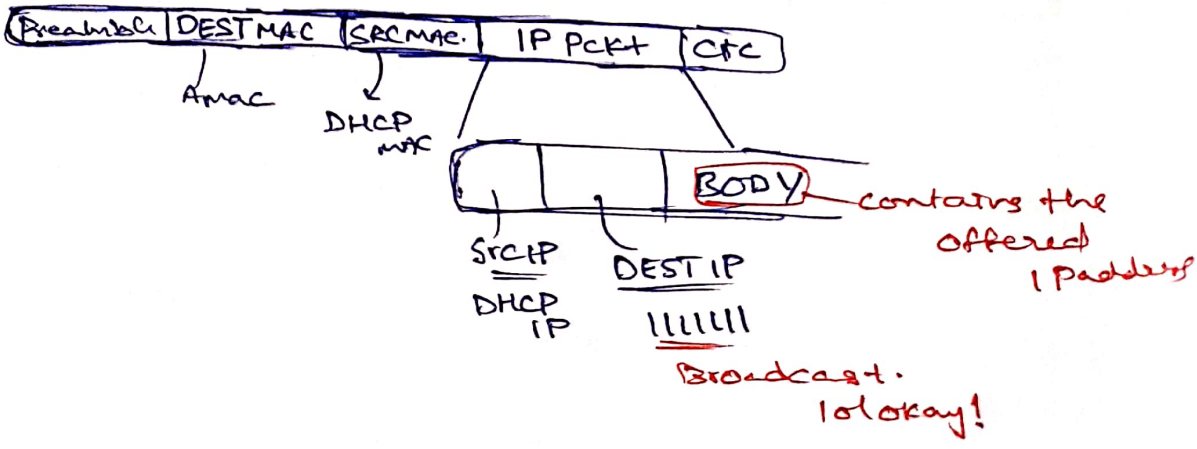
UDP
IP ← broadcast!
DLL → broadcast!
PHY

} by A; during "DISCOVERY" packet.

1. A sends out a 'DISCOVERY' packet:-

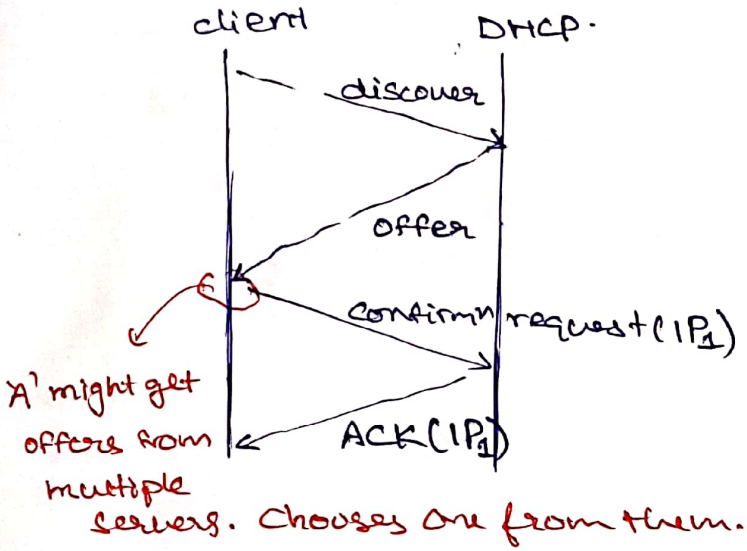


2. DHCP server replies with an offer:-



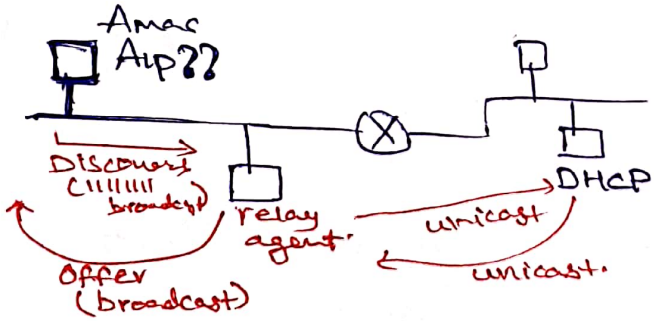
3. Host 'A' checks whether offered IP is taken --- by ARP.

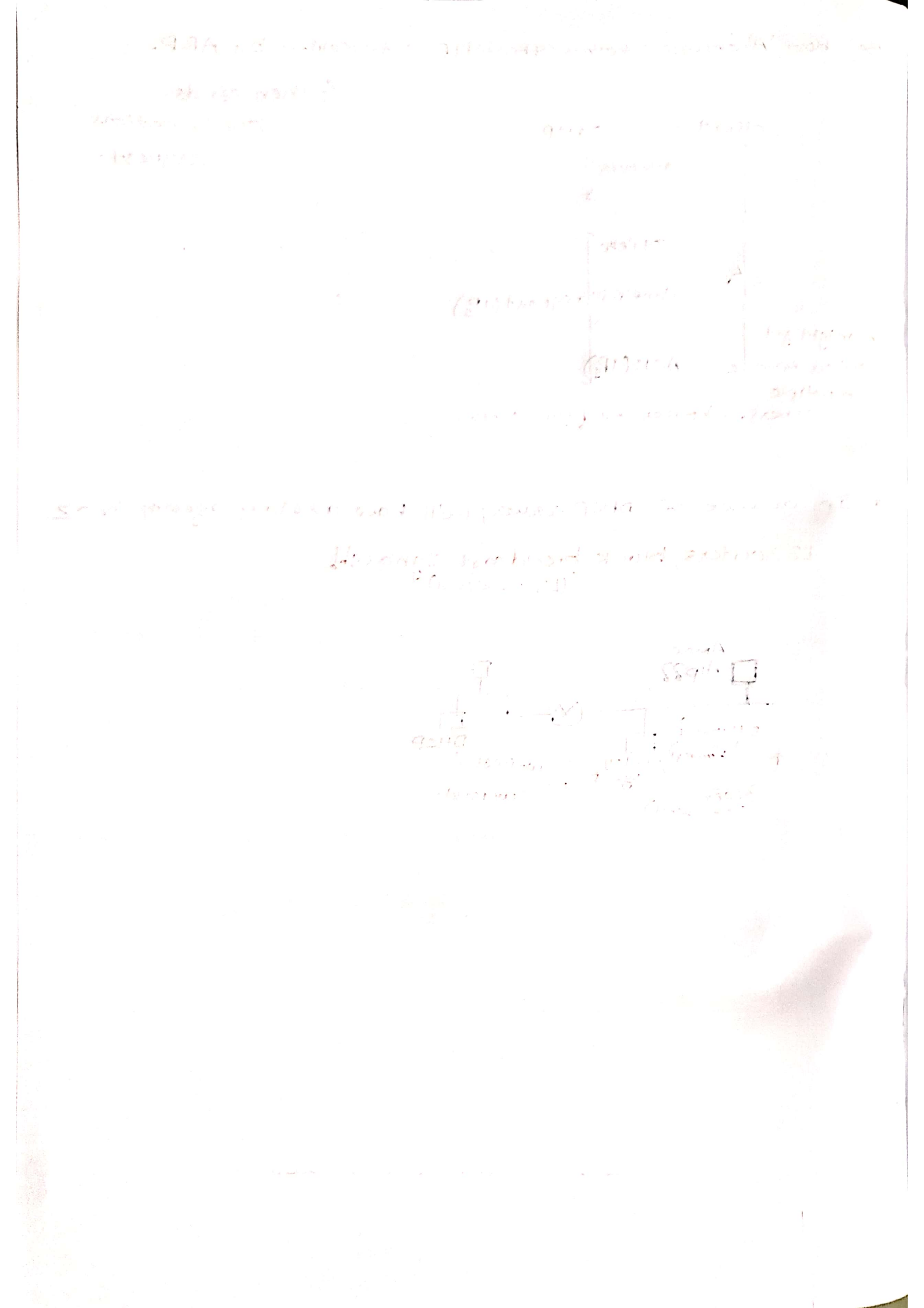
& then sends confirmation request.



* In absence of DHCP server; we'll have a relay agent; bcoz L3 routers, block broadcast signals!!
(L3 broadcast)

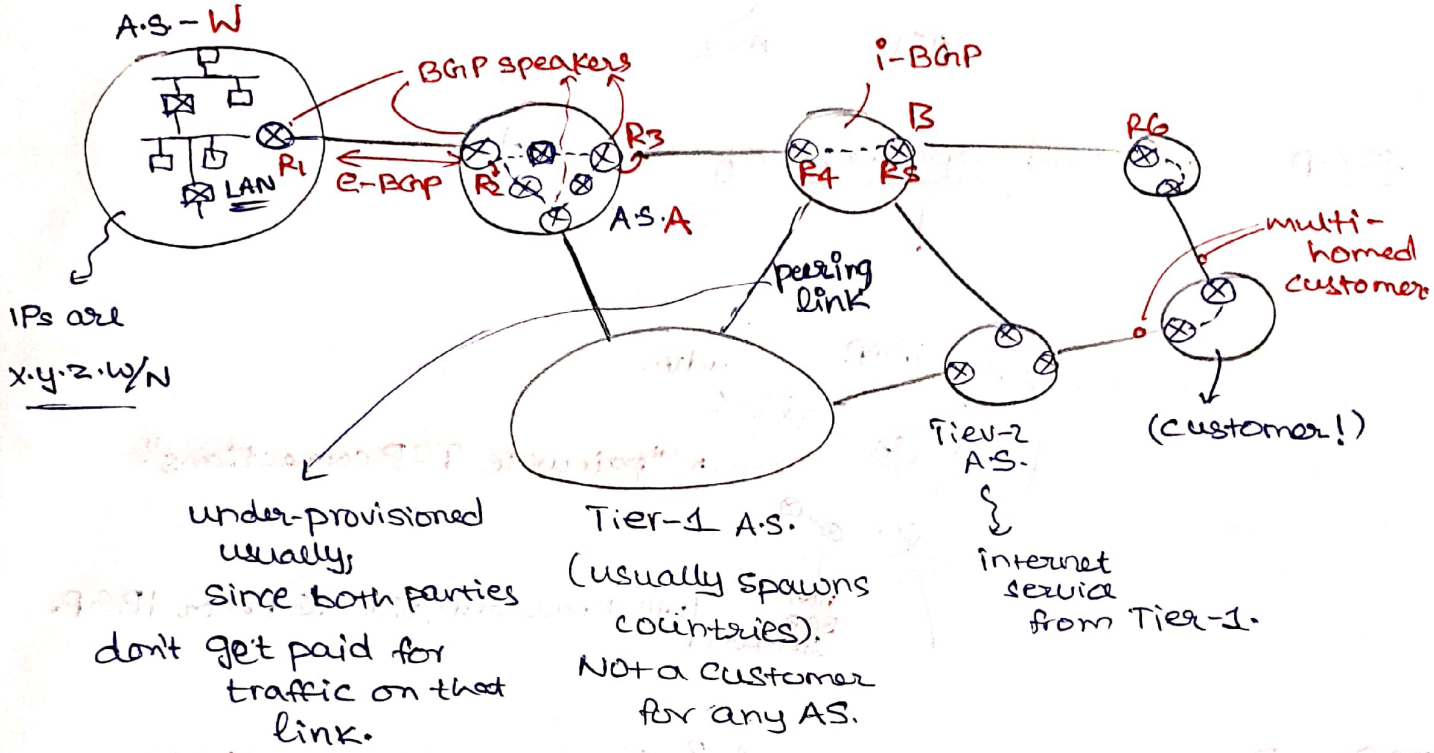
Otherwise, RIP internet.





Lec 20:- Border Gateway Protocol:- also HW3.

> Inter-domain Routing:- Between ASes. the glue that brings the world together.



BGP advertisements:-

eBGP $R_1 \rightarrow R_2$: $x.y.z.w/N$ (prefix) | W (A.S. path) | ____ (other attributes)

iBGP

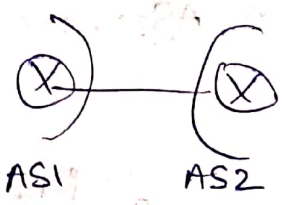
eBGP $R_3 \rightarrow R_4$: $x.y.z.w/N$ (prefix) | $A-W$ (path) | ____ (other attrb.)

$R_5 \rightarrow R_6$: $x.y.z.w/N$ | $B-A-W$ (path) | ____

meaning; if R6 gives a pckt to x.y.z.w; for sure, this path'll be followed.

* Tier-1 ASes; don't need to use peering links for advertising... flexibility is there.✓

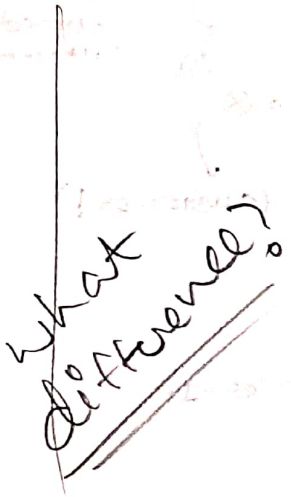
eBGP:- Protocol between BGP Speakers in neighbouring A.S.

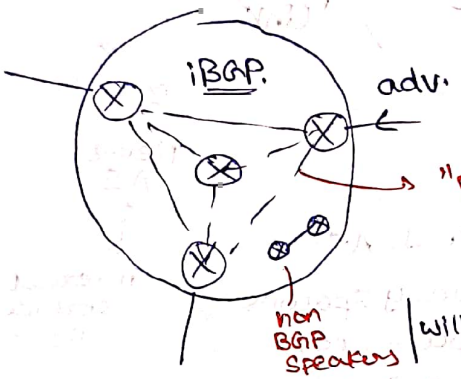


TCP# 179.

iBGP:- b/w BGP speakers of same AS.

!





will know stuff, via IGP. not IBGP.

IGP:- Interior Gateway protocol } intra-domain routing!...

Between every router of an AS. whether speaker or not.

based on DV, LSR

these are concepts. Not protocols. hmmm...

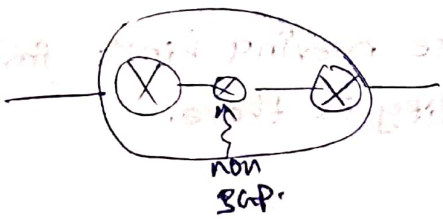
→ Steps to determine BGP paths:- Protocol:-

1) e-BGP speakers learn AS-paths from neighbours.

2) e-BGP nodes share learned information via iBGP with other nodes inside A.S.

3) BGP speakers choose routes to various IP prefixes.

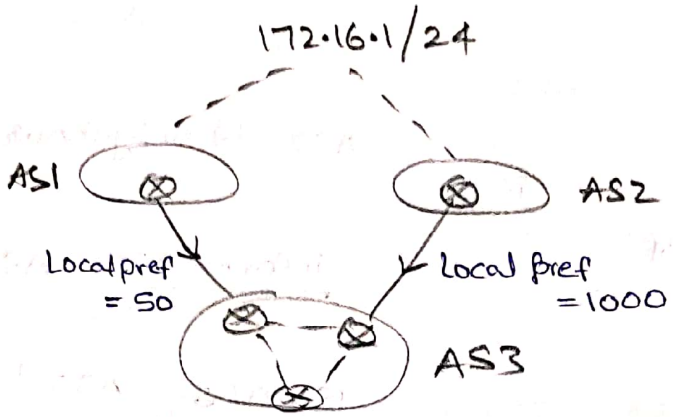
4) The speakers; insert choosen routes into IGP. Since all routers; whether speakers or not; should be able to send pkt to every IP destination.



5) eBGP speakers can advertise newly inserted routes to neighbouring AS.

→ BGP attributes:-

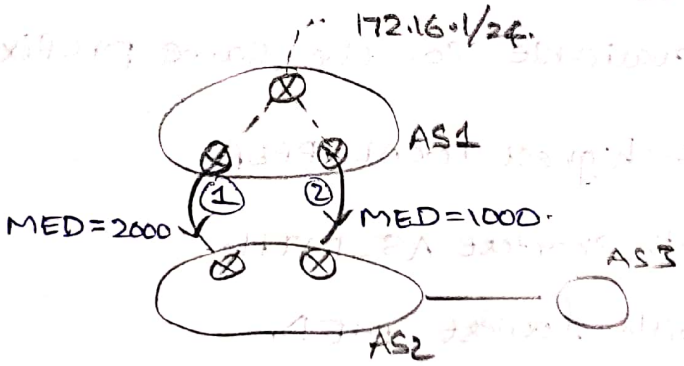
1) LOCAL_PREF:-



> Admin of AS3 adds LOCAL-PREF himself.. Usually left default.

> Higher pref; more preferred path.

2) MULTI-EXIT DISCRIMINATOR:- MED

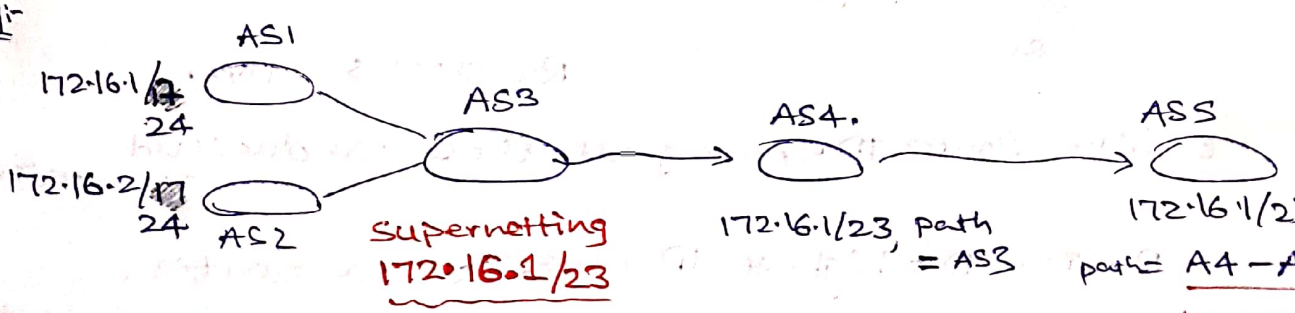


> AS1 tells AS2 that use ② over ①.

> lower MED; more preffered path.

3) AS-PATH:- List of AS; all the way to the destination who advertised that prefix first.

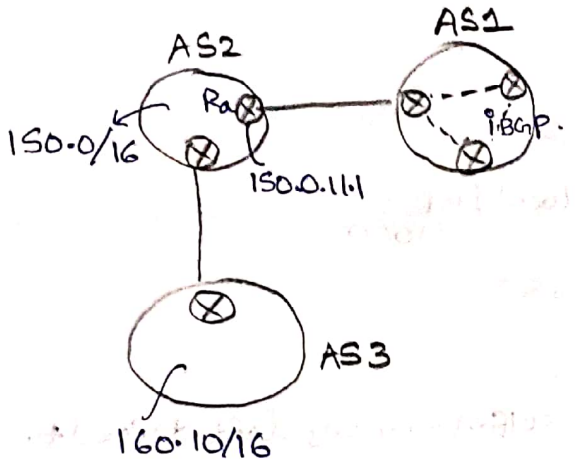
Eg:-



but no A1, A2; Since prefix came at AS3.

4) NEXT_HOP:-

the ip address of the BGP speaker; in the neighbouring AS.



AS2 is neighbour of AS1:-

information AS1 has:- (circulated by iBGP)
(forwarding tables)

150.0/16	AS2	NEXT HOP = 150.0.11.1
160.10/16	AS2-AS3	NEXT HOP = 150.0.11.1

→ Rules to choose routes, in BGP:-

* each BGP speaker decides which AS-route to use, among many available for the same prefix.

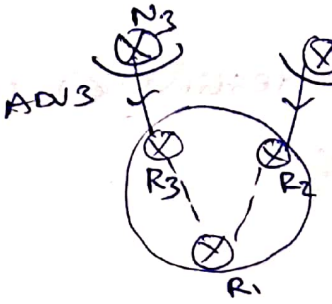
a) use route with highest LOCAL_PREF.

b) choose path with shortest AS_PATH.

c) choose path with lowest MED.

d) choose path learned via eBGP over one learned via iBGP.

e) choose path with lowest IGP metric to NEXT_HOP.



HOT-POTATO ROUTING.

Suppose

dist(R3, N2) > dist(R3, N3)

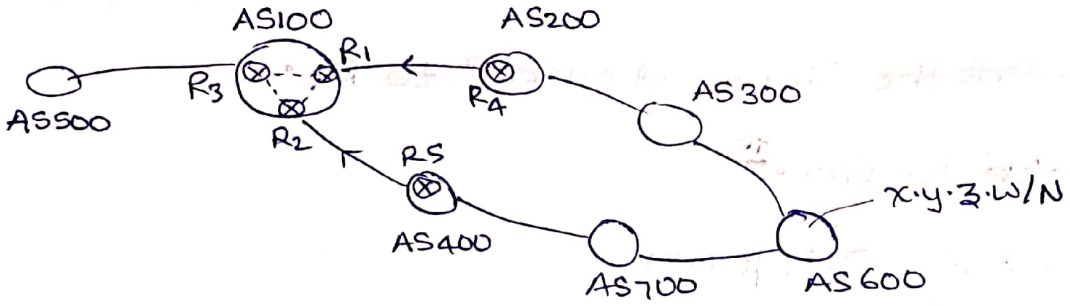
R3 chooses ADV3.

f) use least Router ID. (Among all those who've send advertisements)

Router ID = highest IP address on a router,

can have multiple NICs & multiple IPs.

Eg:



* Look inside AS100:-

Say both paths have same LOCAL-PREF and MED

Path 1: AS200 – AS300 – AS600

Path 2: AS400 – AS700 – AS600

R_1: chooses path 1 : eBGP over iBGP

R_2: chooses Path 2 : " " "

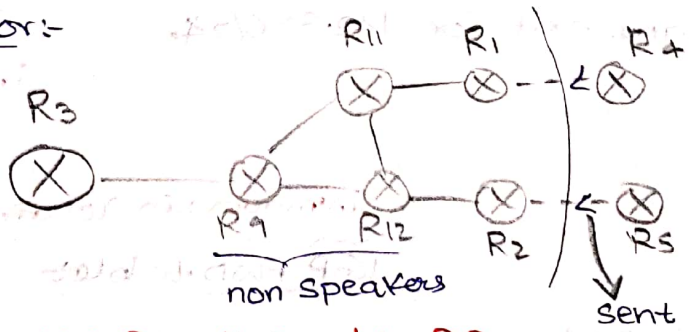
R_3: ? : both paths learned over iBGP.

if dist(R_3, R_1) < dist(R_3, R_2) then R_3 goes for path 1.

Hot potato routing.

AS100 interior:-

Can R9 talk with internet? lets work on it.



BGP interaction with IGP? ! Speaker-speaker talk.

Solutions:-

Also; How R3 will send to R1? Same as below (IGP)!

142.32.6/24

1) Encapsulation:-

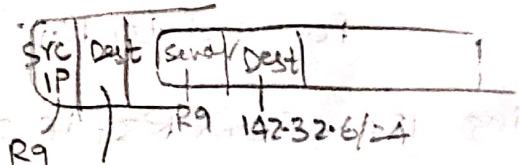
Say R9 has a destIP as 142.32.6/24 for a packet. But it has no BGP information. But, every router gotta send packets to world, not only speakers

Now R9 encapsulates this packet by a new IP layer:-

Ex: R9's FWD table:-

DST	NextHOP
R_1	R_{11}
R_2	R_{12}
R_3	R_3
R_{11}	R_{11}
R_{12}	R_{12}

no 142.32.6/24



R1 ! present in FWD table! (PTO)

might be wrong. R9 is internal. R3 will do this.

- R_1 recieves this packet, strips off outer layer, forwards the "internal packet" to R_4.

R_1's table be like:-

DST	Nxt
R2	
R3	
⋮	
142.32.6/24	R_4

∴ Encapsulation means: Non-speaker routers have no clue about outside IP prefixes, but they do know that, they need to ask BGP-speaker router for help, & thus forwards their packet to speaker router.

2) Pervasive BGP:- All this work; to allow internal routers to talk to internet. ??

All routers are treated as BGP speakers.

* Say theres a unique exit for 142.32.6/24.

& also for interaction of BGP and IGP. // (Speaker–speaker talk).

R9s:-

intradomain routing.

BGP table:-

Prefix	Gateway/Exit
142.32.6/24	R_1
⋮	

IGP FWD table:-

DEST	Next
R2	⋮
R3	⋮
R_1	R_{11} ✓✓
R_4	⋮
R_{12}	⋮

Forward to R_{11}.

- on reciving IPpkt from R3; if Dest.IP= 142.32.6/24; then R_9 will do above thing. (Recursive lookup)

I understood wrong at first.

these solutions are for BGP-IGP interaction.

I thought internal router - BGP interaction...

3) Tagged IGP:-

* Internal routers may not be BGP speakers.

But IGP allows addition of tags.

* R_1 can insert (advertise), into its own IGP. Not a new BGP table. insert into IGP.

142.32.6/24 — Prefix

R_1 → gateway router — TAG

propagated to all routers, by IGP (or LSR)....

Ex: R_9's IGP table be like

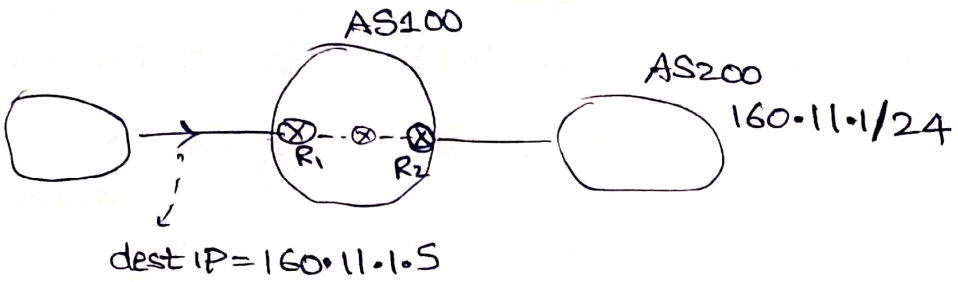
DST	Next	Tag	Cost
R_1	R_{11}	-	5
R_2	R_{12}	-	10
⋮			
announcements 142.32.6/24	-	(R_1)	
142.32.6/24	-	(R_2)	

Now; R_3 will send a pkt: DestIP = 142.32.6/24

then R_9 will lookup; find matches & forward to closest "tag".

Hot potato.

*



NOW; R_1 recieves this packet.

- R_1 looks up BGP table & decides to FWD to R_2.

  But How!

  How this interaction of BGP & IGP?

  - Encapsulation:
    R_1 encapsulates & sends to R_2
  - Pervasive BGP:-
    R_1 sends as it is, intermediate routers look up 2 tables, every time they recieve this pkt; to determine NEXT.
  - Tagged IGP:-
    R_1 sends as it is. only one.
    Internal routers look up IGP table; decide via hot potato & forward to NXT.

(visit HW3.)

→ Network address Translation (NAT) :-

* To access the internet; we need an public IP address. But we can use private IP in our private networks.

* NAT's idea is to allow multiple devices to access the internet; through a single public address.

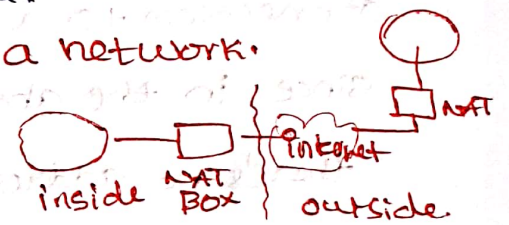
10.*.*.*
192.*.*.*

* NAT is a process in which one or more local address is translated into global address. Also port number is translated

- The corresponding entries of IP address and port number is made, in the NAT table.
- NAT operates on router or firewall.

edge of a network.

* There are 3 ways to configure NAT:-



1) Static NAT:-
   - a private IP mapped to a legally registered IP.
     i.e. public IP.
   - Generally for webhosting.
   - not for organisations, as many devices need internet. costly to afford those many public IP.

2) Dynamic NAT:-
   - a private IP is assigned a public IP, from a pool.
   - if the pool of public IP is not free; then further IP pkts will be dropped.

Ex:-
we got 2 public IPs. Hence we can't afford 3 hosts.

3) Port Address Translation (PAT):-
   - Known as NAT overload.
   - many private IPs to single public IP, port numbers are used to differentiate traffic.
   - most often used, cost effective. Thousands can use single public IP.

→ NAT security:-

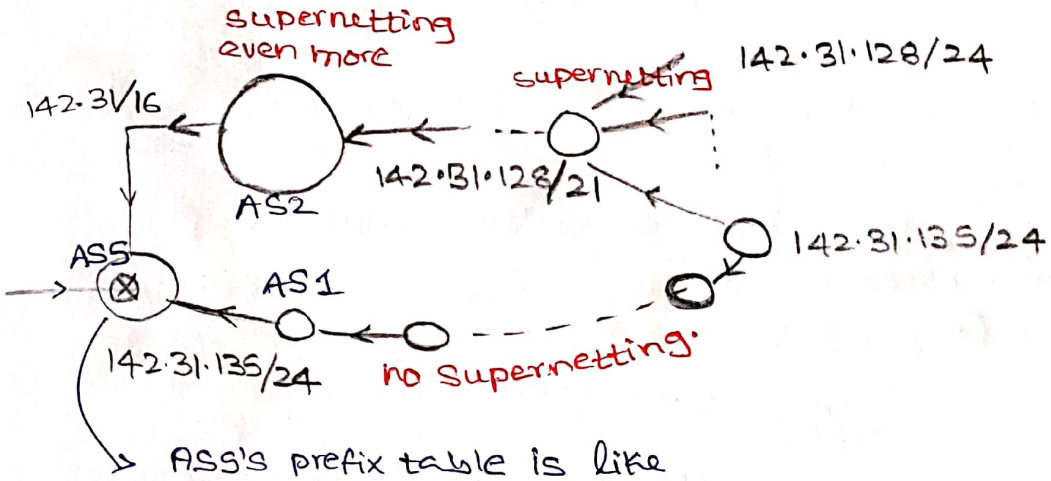
* NAT provides privacy.

- NAT allows you to display a public IP address while on a local network, helping to keep data & user history private.

- NAT transfers packets of data from public to private addresses, it also prevents anything else from accesing the private device.

* NATs make it difficult for an outside device to initiate a connection to a device on the private side of NAT, since in the absence of a mapping in NAT table, there's no address to target.

→ longest prefix matching (BGP mei):-



→ AS3's prefix table is like

prefix	next hop
142.31/16	AS2
✓ 142.31.135/24	AS1

fast lookup possible due to hardware aids;

CAM: content addressable memory.

* Now AS3 gets a pkt with dest IP = 142.31.135.24.

"choose longest prefix matching to decide next hop"!

Hence AS2.

- This'll prevent routing loops. And also we'll have a deterministic path (instead of choosing randomly).
- longer prefix implies we are "closer" to destination.

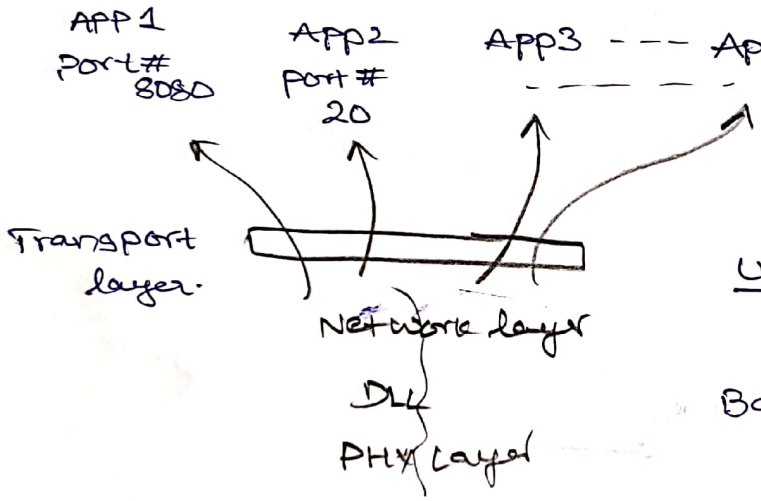
Layer 4: Transport Layer:-

* Many applications are running simultaneously on our PC; bcoz we have port numbers.

Each-application → Allotted a port number.

* The transport layer (implemented in OS) acts as a 'DEMULTIPLEXER'; send pkts to their appropiate applications.

Hence the name 'transport' inside OS.



UDP:- user datagram protocol.

Basic task: demultiplexing.

TCP:- transfer control protocol:-

* Does a-lot- of stuff.

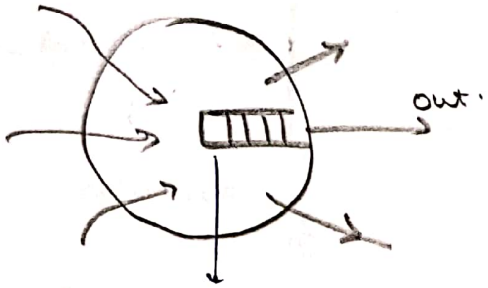
* There's no client TCP & server TCP. Its just TCP. Both behaviours are part of this.

- Reorders recieved segments based on seq#.
- retransmits lost packets. (Maybe lost due to queue full at routers).
- 2-way data transfer.
- Acks are present. for sent data.
- Congestion control, after infering congestion in different mechanisms.
- Flow control.

!! routers don't have L4.

* TCP is purely an end-to end congestion control protocol.
  - This means, it doesnot require any special information from routers or explicit information about other TCP flows.

* Recall the structure of a router:



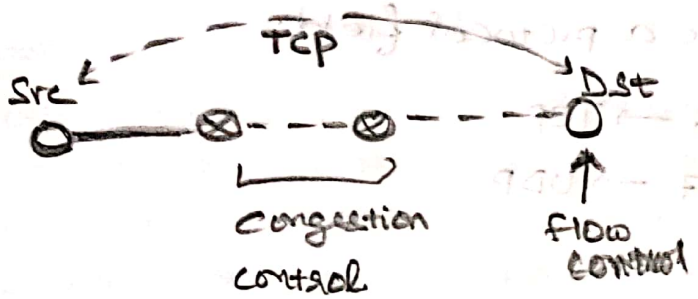
this is usually a FIFO queue.
also "droptail mechanism".
i.e- when it is full it'll drop further incoming packets.

Van Jacobson
Sally Floyd
1980s
TCP.

In case of congestion,
Tcp will infer this
(by pkt loss due to timeout/3DUP.ACK method.)
"packet loss due to congestion"!
& reduce segment sending rate.
(decreases congestion window).

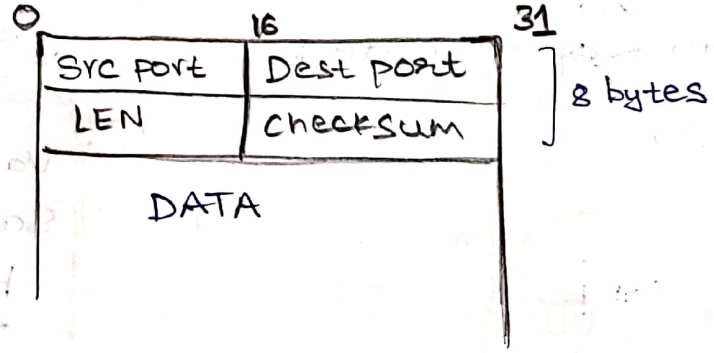
* Flow control:-

When there is congestion at the end host (rather than intermediate router) then the end host can inform the sender. No need of infering this flow congestion.

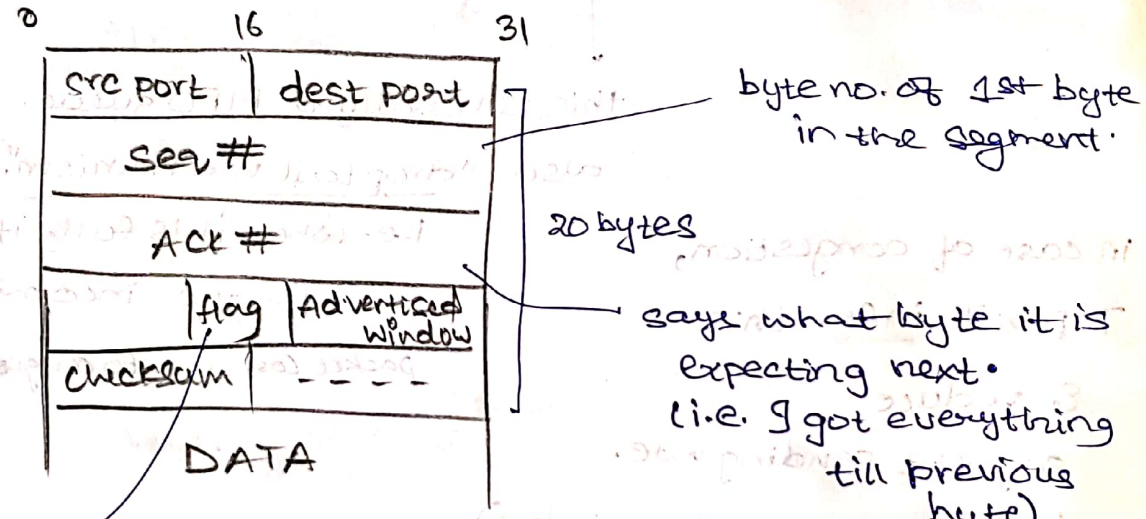


→ Headers of the transport layer:-

1) UDP:- this is a plain protocol.



2) TCP:-



SYN | FIN | RESET | PUSH | URG | ACK

Start connection

end connection.

to say if ACK is there or no.

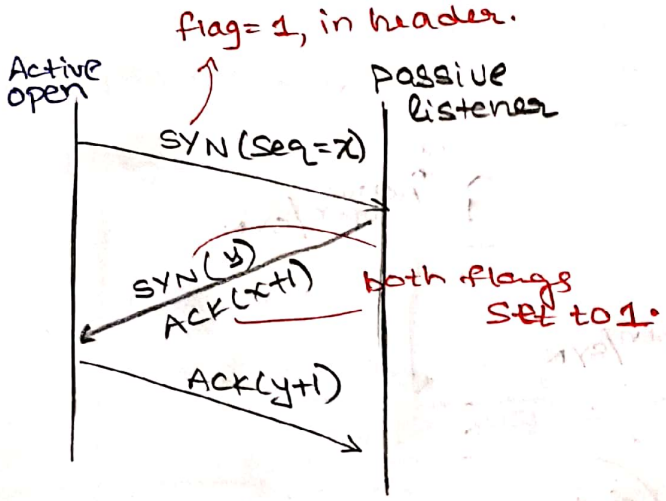
* IP header will have a protocol field:-

6 → TCP

17 → UDP.

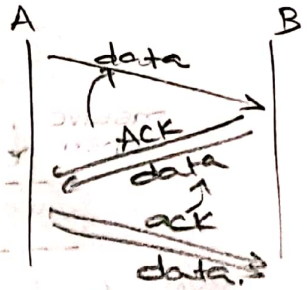
L23:-

→ connection establishment:-



3-way handshake

• Typically:-



Two way data transfer in TCP.

* SYN → synchronization
* x,y → randomly selected not 0.
* The SYN, FIN segments, typically shouldn't have any data. But taken to have 1 byte.

Hence ACK is x+1 bytes!

* if ACK = K; then it means "I have everything till K-1".
* x,y are randomly selected from $0 | 2^{32}-1$.

Bcoz; say if we take in a ordered fashion;
We'll maybe have overlaps...

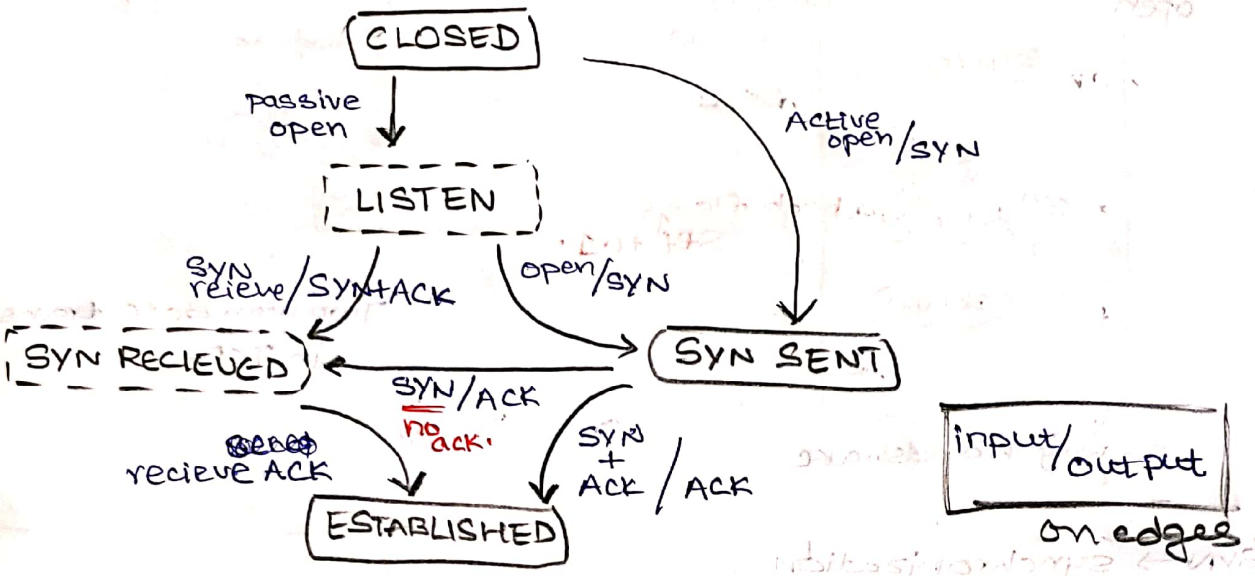
↳ [Not sure here]

verbatim from book:-

TCP specification requires that each side of a connection select an initial starting sequence at random. The reason for this is to protect against two incarnations of the same connection reusing the same sequence numbers too soon - that is, while there is still a chance that a segment from an earlier incarnation of a connection might interfere with a later incarnation of the same connection.

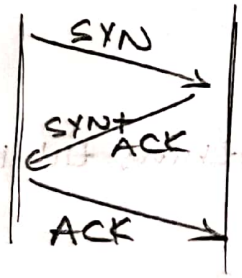
→ State Diagrams:-

1) opening a connection:- same diagram should be used for a client & a server.

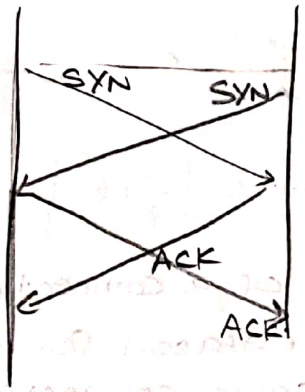


* this State diagram; encompases 2 things :-

(i) 3-way handshake:-



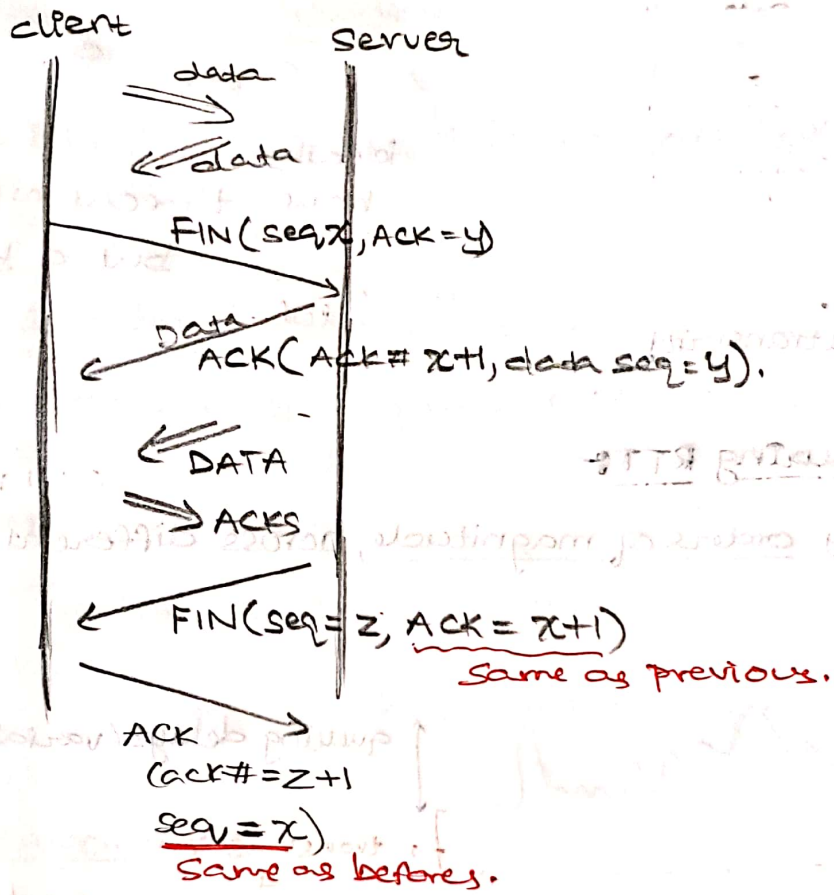
(ii) Mutual active open:-



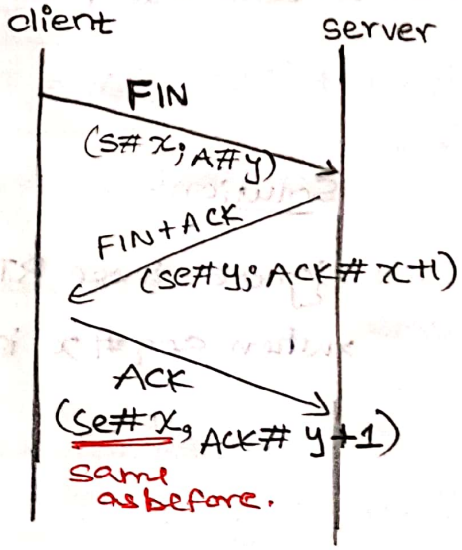
2) terminating a connection:- (use FIN flag).

State diagram becomes convoluted.

1) Half-close:- (only one end closes)



2) 3-way handshake termination:-



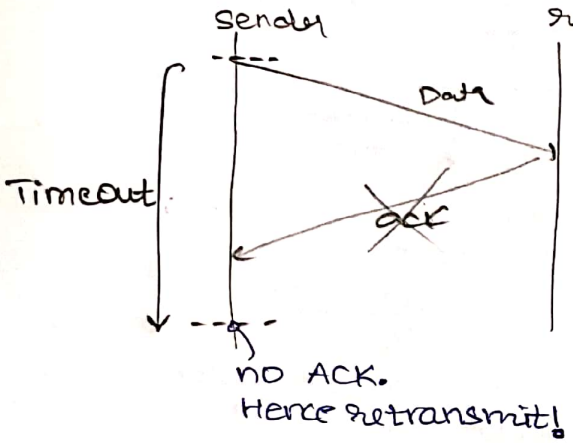
∴ Both sides must close the connection.

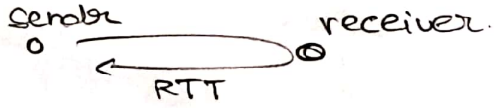
If only one side closes, it means it won't send data; but it will recieve.

L-24 :-

→ TCP timeout :-

* when does the sender retransmit?





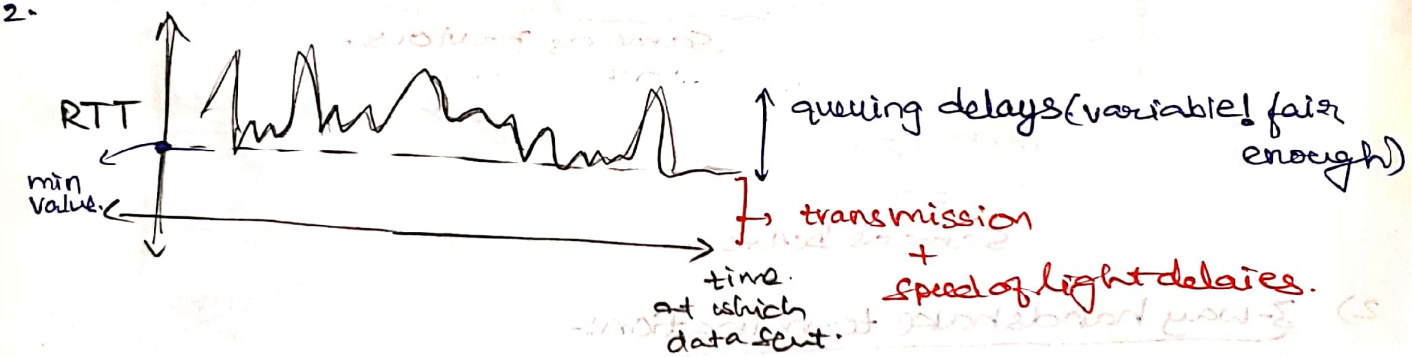
ideally;
have timeout ~ RTT
but a bit more.

lol...

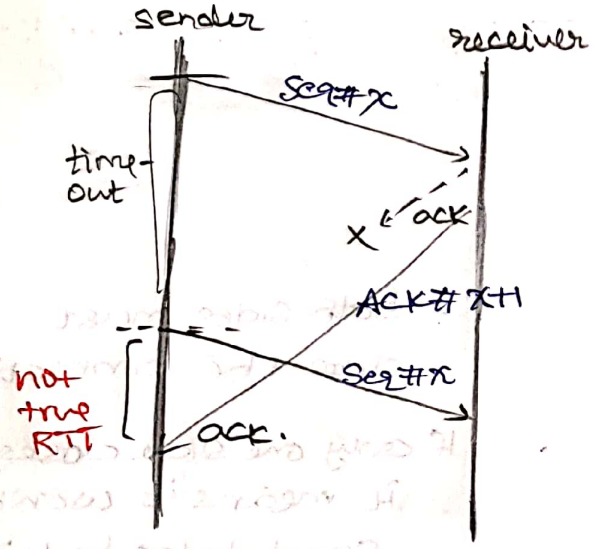
→ issues in measuring RTT :-

1. RTT varies by orders of magnitude, across different paths.

2.



3. Packet loss makes it difficult to estimate RTT.



Solution :-

ignore these RTT measurements
> when seq#x is 'retransmitted'

* so, how DO we measure avg. RTT or something like this:-

1) old algorithm:-

sample RTT ← most recent RTT measured

$$\text{Est.RTT} = \alpha \cdot \text{Est.RTT} + (1-\alpha)\,\text{Sample RTT} \quad \alpha \in (0,1)$$

"Smoothing".

$\alpha \simeq 1 \rightarrow$ less importance to current RTT

$\alpha \simeq 0 \rightarrow$ lot of importance to current RTT.

& then,

$$\text{Timeout} = 2 \times \text{Est RTT}$$

mmm..... has some issues....

2) New Algorithm:-

assume gaussian distribution on RTT.

1. estimate Est.RTT as before.

$$\text{Est.RTT} = \alpha(\text{Est RTT}) + (1-\alpha)(\text{Sample RTT}) \qquad \alpha \to \frac{7}{8}$$

2. estimating deviation:-

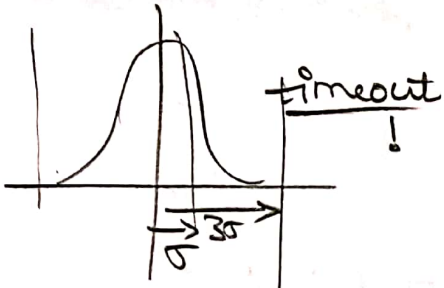
Standard deviation → intricate RMS value....

use mean deviation!

$$= \frac{1}{N} \Sigma |x_i - \bar{x}|$$

$$\text{Dev} = \text{Dev}(1-\beta) + \beta \cdot |\text{diff}| \qquad \beta \to \frac{1}{4}$$

where $\text{diff} = \text{SampleRTT} - \text{Est.RTT}$



Set timeout = mean + mxσ

my midsem approach!

3) calculate Timeout, from RTT & Dev:-

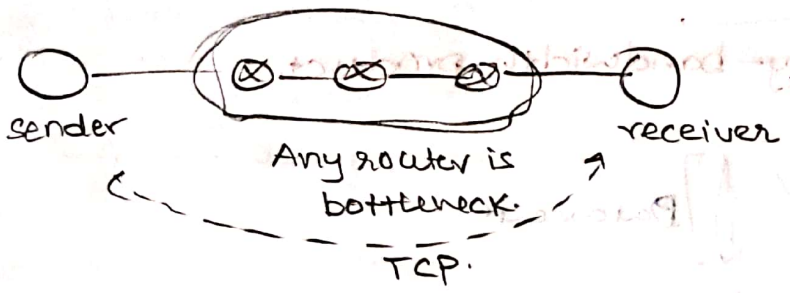
$$timeout = \mu \times EstRTT + \phi \times Dev$$

μ → 1

ϕ → 4

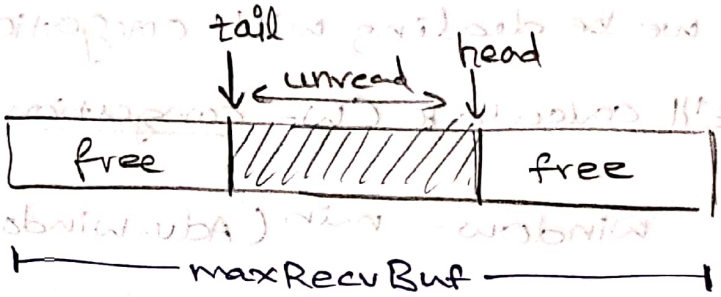
(mean dev^n; not std-dev^n)

→ Congestion control:- hosts need to infer. — packet loss.
congestion. → greater RTTs.



lets see at receivers end:-

* Receiver Buffer (Layer 4)



Flow control:- congestion control at receiver.
- No need to 'infer'.

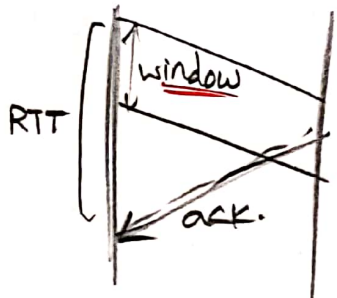
* A field in L4 Header → Adv. window = space left in receiver buffer.
So, the sender knows!

*
Sender has a "window".

Window = Max amount of data (in bytes) which is outstanding, i.e. sent into the network but not acknowledged.

* Say RTT is constant

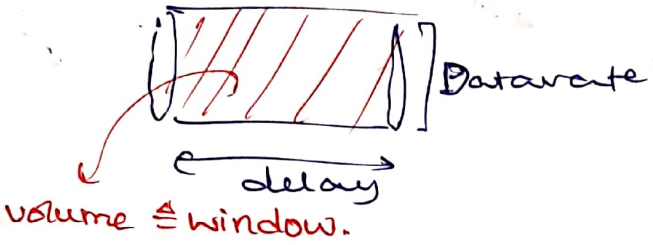
$$\text{Datarate} = \frac{\text{Window}}{\text{RTT}}$$



∴ we'll have window < Adv. window (for flow control).

* window = RTT x datarates

delay-bandwidth product.



volume ≙ window.

* How would we be dealing with congestion control?

we'll calculate CW- congestion window.

∴ window = min(Adv-window, CW)

TCP congestion control:-

* problems with congestion at routers
  - Packets are dropped.
  - queuing delays; more RTT.

$delay = \frac{8 \times 10^6}{C}$ s

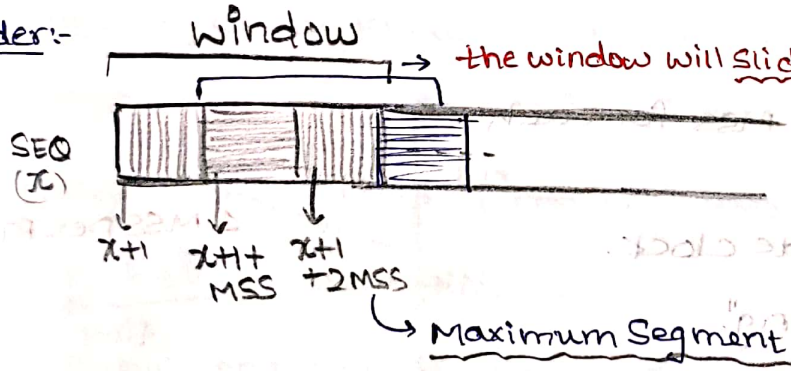
* Router will not distinguish UDP, TCP packets. It is responsibility of TCP to do congestion control.

VoIP → UDP protocol.

* How do we infer Congestion?
  - large RTT values
  - packet loss. How to infer packet loss?
    - Timer expiry
    - 3 Duplicate ACK approach.

→ Detecting Packet loss:-

Sender:-

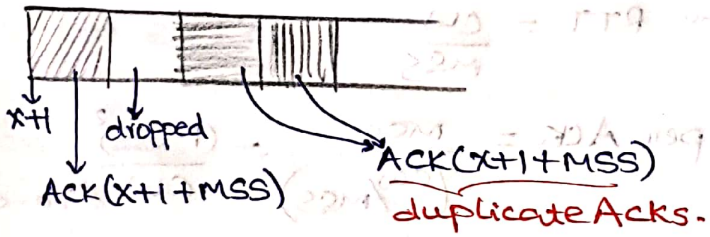


the window will slide; after we recieve the 1st ACK & then immediately transfer 4th pkt.

(sliding window)

If initial window = 3 MSS; we'll send out 3 segments & wait for ACKs.

receiver:-



Cummulative ACKs:-

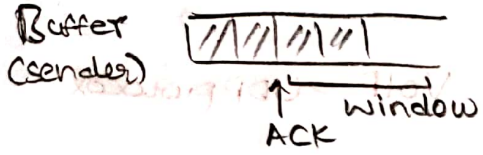
Ack # z (sent) ⇒ means; "I have every byte, till (z-1)".

(PTO)

Duplicate acks:-

- ACK# of some previous ACK is sent again.
- Each dup ACK says "one more segment received".

* Where as; at the sender side; window starts at the latest ACK recieved.



→ Principles of congestion control:- For now, assume window = CW; & take Adv.window is sufficient.

1. If no congestion, increase CW conservatively

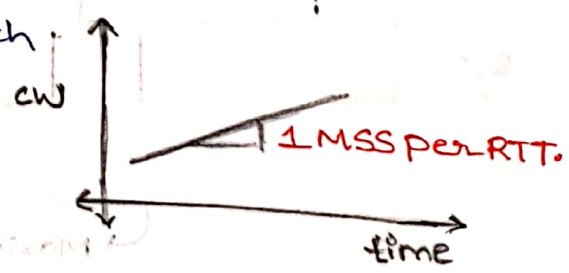
$$\text{Datarate} \propto \frac{CW}{RTT}$$

We'll have something called slow-start too!:

∴ as CW↑; datarate↑. (yay!)

* additive increase:

we increase CW by 1 MSS for each RTT.



→ But, don't use absolute clock.

* TCP uses "self-clocking".

i.e. increase CW on arrival of ACKs.

no. of segment in one RTT = $\frac{CW}{MSS}$

∴ no. of ACKs in one RTT = $\frac{CW}{MSS}$

∴ increase in CW per ACK = $\frac{MSS}{(CW/MSS)} = \frac{(MSS)^2}{CW}$ // Ta-da!

∴ on reciving an ACK;

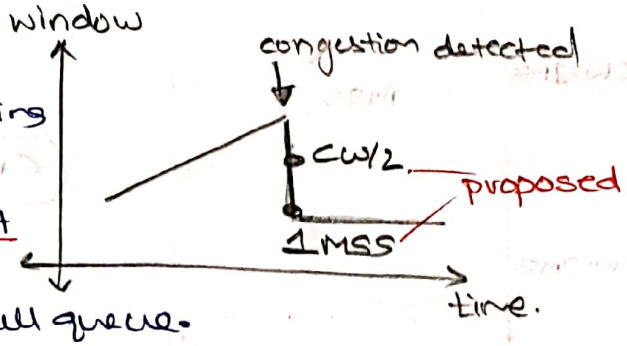
$$CW \mathrel{+}= \frac{(MSS)^2}{CW}$$

; Assuming no congestion.

2. If congestion is detected, decrease cw aggresively.

- aggresively, bcoz already data incoming rate exceeds outgoing at router.

Hence; small decrease won't help, the already full queue.



TCP tahoe ⟶ cw = 1

TCP reno ⟶ $cw = \frac{cw}{2}$ MULTIPLICATIVE DECREASE.

TCP vegas ⟶ & some other mechanisms.

TCP Africa, cubic ⟶ we ain't studying.

linux!

AIMD:- additive increase multiplicative decrease.

---

3. Slow start:- (Multiplicative increase)

* Initially, we donot know; what the optimal datarate/cw is.

It might be 10 Gbps ($\sim 10^3$ MSS).

Starting from 1MSS & Additive increase ⟶ very slow!

* Hence, be agressive initially.

- double the cw, per every RTT.

again! using self-clocking.

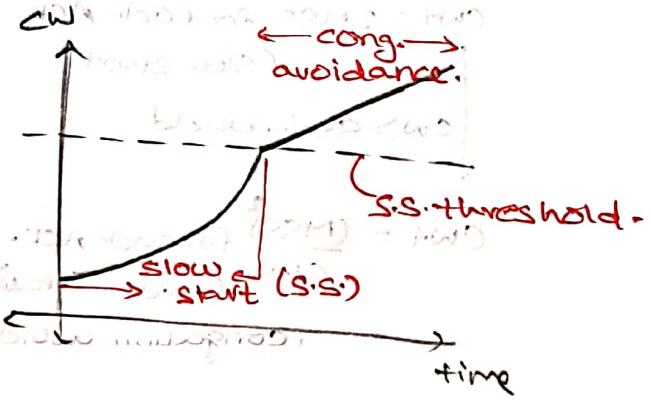
in each RTT:-

ACKs# = $\frac{cw}{MSS}$

increase per ack = $\frac{cw}{\text{since, doubling!}}$

$= \frac{cw}{\left(\frac{cw}{MSS}\right)}$

$= 1\,MSS$

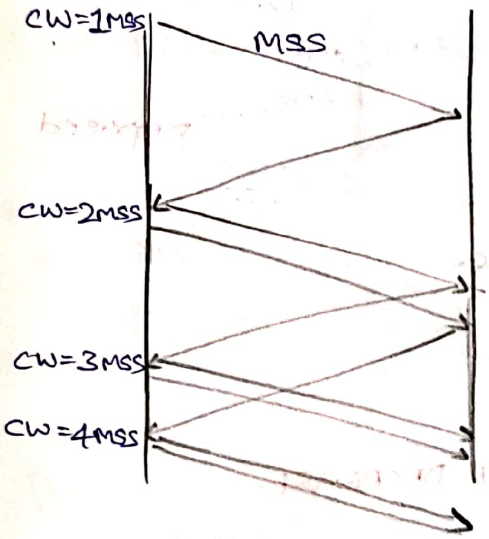


$cwt = 1MSS$

per ACK. } self-clocking!

→ irony! high growth, but slow start.

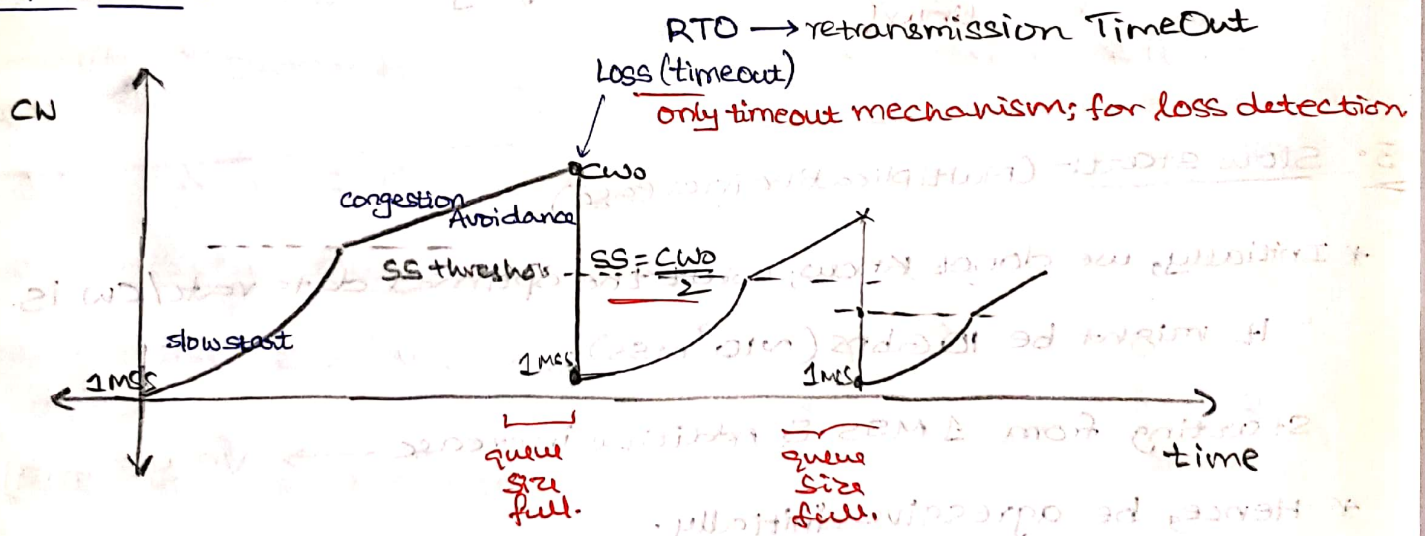
* in Slow Start phase;



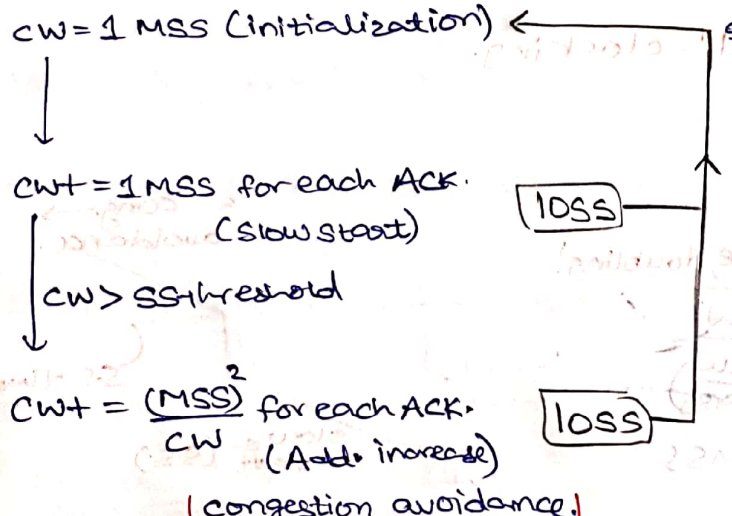
CWt = 1MSS per ACK recieved.

→ TCP TAHOE:-

RTO → retransmission TimeOut



rules:-

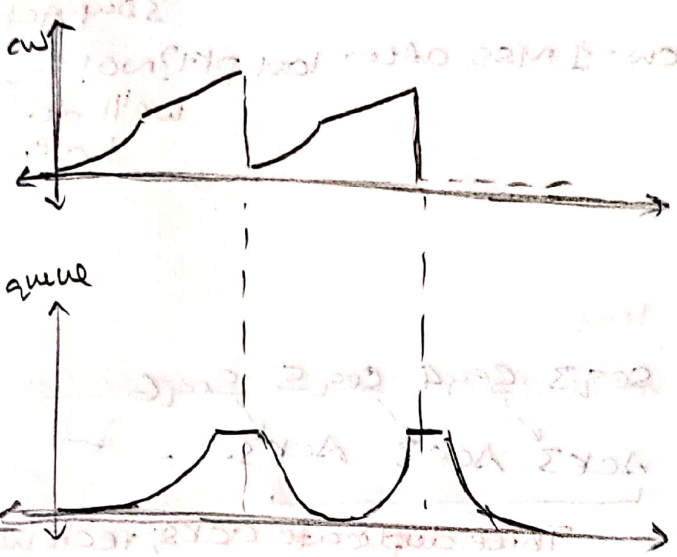


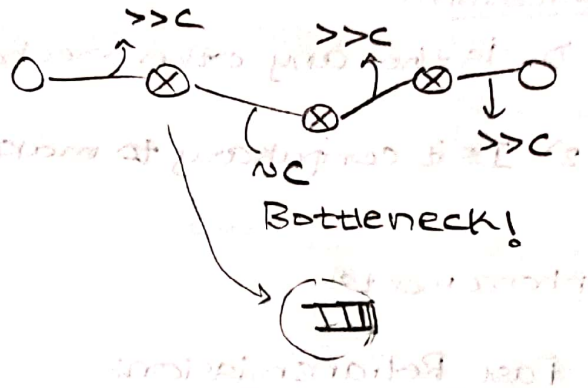
SS_threshold = CW/2

min. SS_threshold = 2 MSS

gave in a RFC (request for comments)

* Plot the queue size





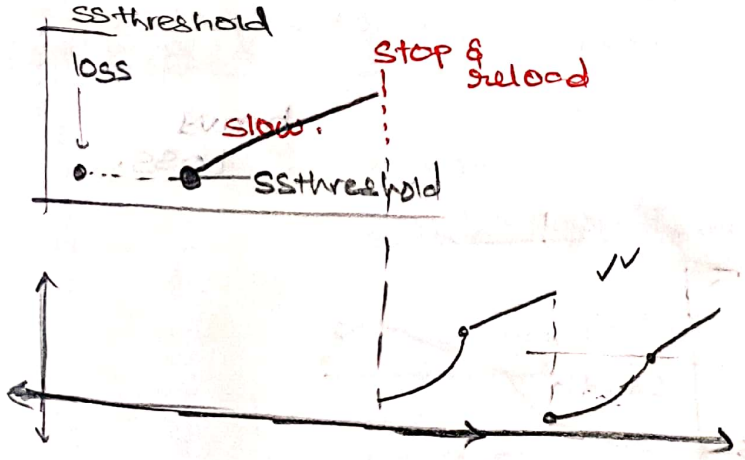
* Two TCP Tahoe flows... okay! (See in slides; nothing special graphs).

→ Stop & Reload Phenomenon:-

→ We observe that sometimes, downloading is slow; so we reload the page, now its fast!

- Because; due to bad luck, at the start itself, we might detect packet loss.

  & so; we'll be in additive increment mode later.



fine after reloading.

Questions:-

1) Is the any other mechanism, to detect losses? Yes! 3DupAck way.

2) Is it compulsory to make CW = 1MSS, after lost pkt? NO! we'll do half.

Enhancements:-

1) Fast Retransmission:-

lost...

we sent seq 1 seq 2 seq 3 seq 4 seq 5 seq 6 ...

we got Ack 2 Ack 3 Ack 3 Ack 3 Ack 3 ...

Three duplicate acks; recieved before RTO expiry.

new way to find lost seqs.

Declare seq 3 lost.

reset RTO.

2) Fast Recovery:-

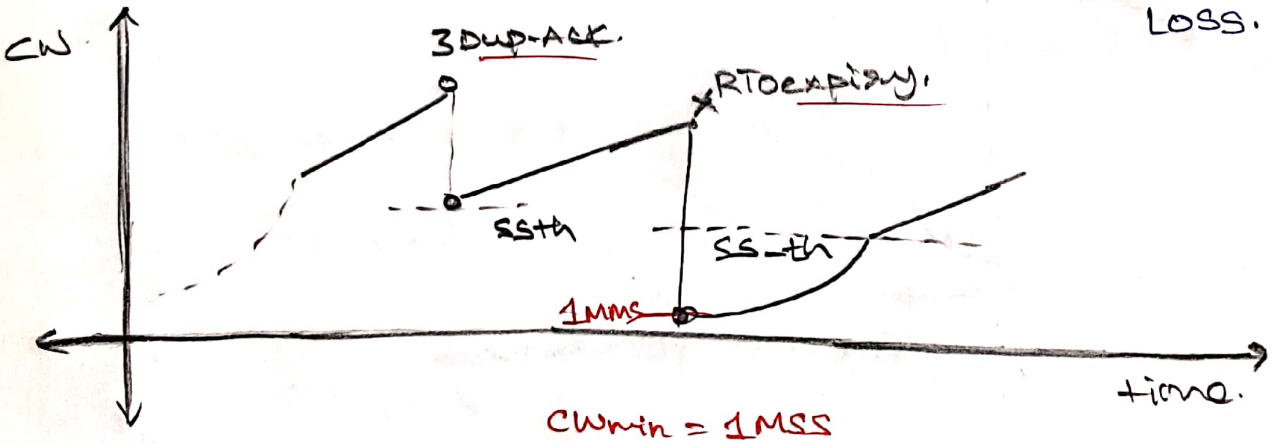
when loss is detected by 3DUPACKS; $CW = \frac{CW}{2}$

& the usual rule:- when loss detected by RTO expiry; CW = 1MSS (minimum.)

Also $SS\_th = \frac{CW}{2}$, so we'll have Add. incr.

meaning the other ACKs are also absent na!

heavy loss.



$CW_{min} = 1MSS$

$SS\_th_{min} = 2MSS$

* TCP Reno; is TCP Tahoe + Fast retransmission + Fast Recovery.

$cw = \frac{cw}{2}$, if 3DupAck

$cw = 1$; if RTO expiry

retransmission TimeOut.

* some miscellaneous points from RFCs;

(1.) In slow start; SS

if the ACK acknowledges 'N' bytes of new data; rather than 1MSS, then

$cw += \min(1MSS, N)$

In Congestion Avoidance; CA

every ACK, do

$cw += \frac{(MSS)^2}{cw}$; but in one RTT, increase by 1MSS.

that means, take this caution.

(2.) Initial RTO = 1 sec or more

Loss by Timeout:-

$ssthresh = \max(2MSS, \frac{cw}{2})$

$RTO = \min(2 * RTO, Max\ RTO)$

cw is made 1MSS only.

You confused before!

this is RTO for the same seq# after sender retransmits.

Hardcoded value

→ TCP vegas:- (1994 n 95)

uses RTT values also; to infer congestion. Nice!

* The previous two TCPs; try to achieve the best possible rates. They would increase packet loss for themselves & also, maybe for other VoIP- UDP packets.

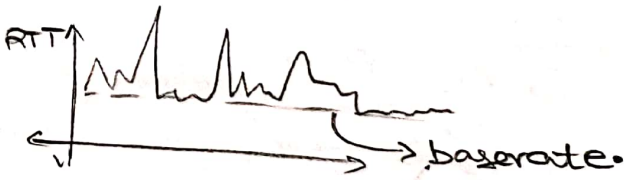
* TCP-vegas is RTT sensitive. It will try to use optimal bandwidth & also allow transfer for other packets. Less pkt drops.

* Detect loss & modify window, ss_th as in TCP reno. Slow start rules are the same.

* Model path as a queue:-

> queue is empty:-

then RTT = baseRTT



$$\text{Expected Rate} = \frac{CW}{\text{baseRTT}}$$

$$\text{Actual Rate} = \frac{CW}{RTT} \leq \text{Expt-Rate}$$

↳ "smoothened out" remember!

$$\text{Diff} = \text{exptrate} - \text{Actualrate}.$$

& Hence; during CA :- congestion avoidance.

Diff < α :-	Diff > β :-	α < Diff < β :-
do A.I. (additive increment)	do additive decrease.	CW is not modified!
+1 MSS per RTT (or) $+\frac{(MSS)^2}{CW}$ per ACK	−1 MSS per RTT	* we are optimal enough!

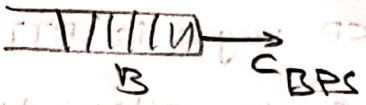
data rate comparision

Diff = expt-rate - Actual rate

$$= \frac{CW}{baseRTT} - \frac{CW}{RTT}$$

$$= \frac{CW}{(baseRTT)(RTT)}(RTT - baseRTT)$$

Say queue length B



$$\approx \frac{B}{BaseRTT}$$

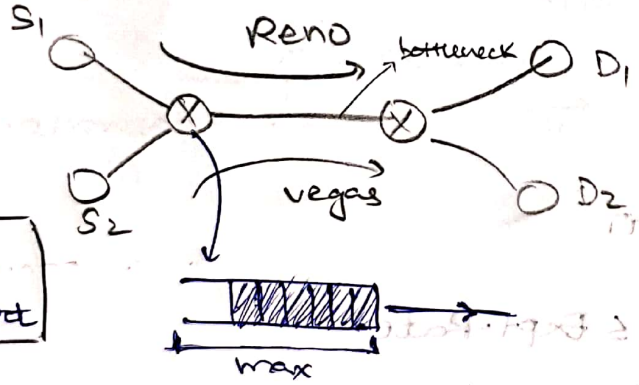
∴ this difference is $\propto$ to queue length.

Q1) What if TCP vegas flows compete with TCP Reno flows?

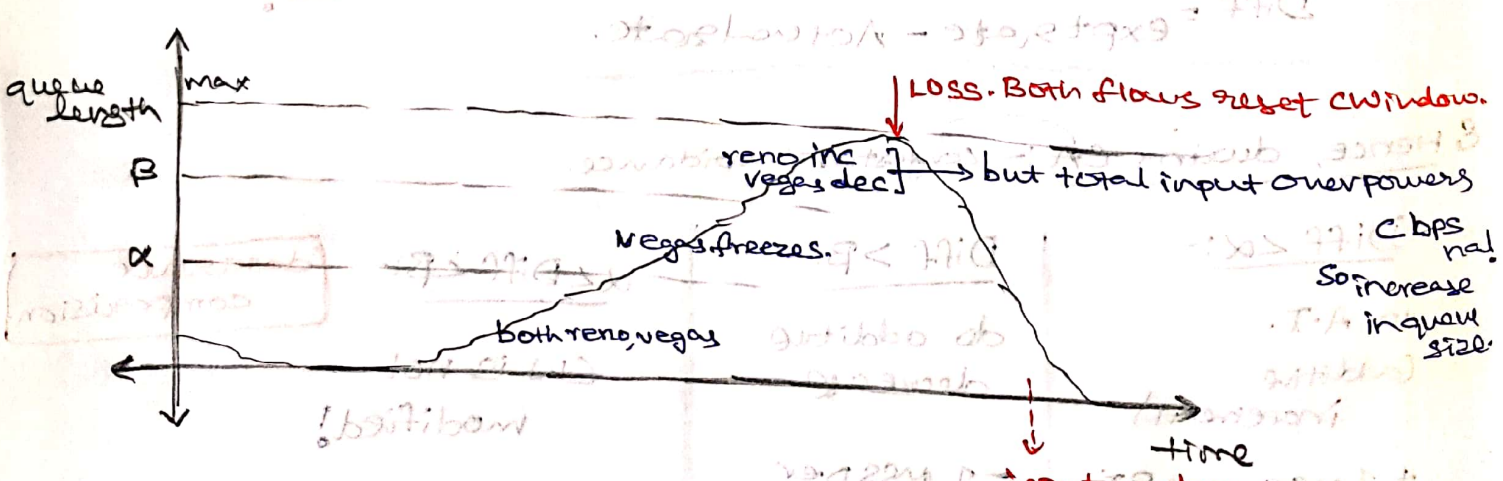
vegas → gets satisfied

Reno → Pushes till extreme of packet loss situation.
aggressive!

this results in pkt loss for all other flows of the common router.
evil Reno!.



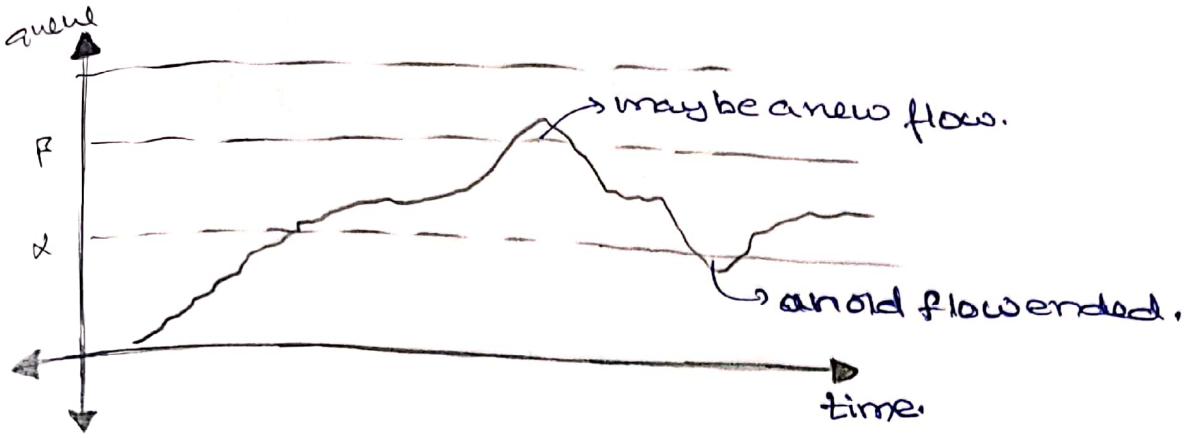
Ignoring slow start

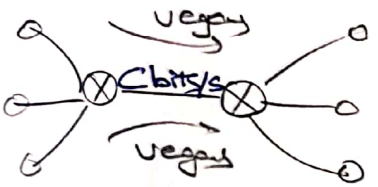


∴ average datarate < C bps.

not nice

Q2) All vegan flows; then what?
lol





Assume all N same RTT

A TCP flow is a 5-tuple

$(SRC_{IP}, DST_{IP}, Src_{port}, DST_{port}, Protocol\ field)$

∴ two devices may have multiple flows b/w themselves
Awesome!

benefits:-

a) No pkt losses. (TCP & VoIP & ... any flow)

b) queuing delays are predictable.
b/w $\frac{q_1}{C}(\alpha's)$ & $\frac{q_2}{C}(\beta's)$.] some Appln need this.

c) Throughput can be higher than all Reno case bcoz the queues never empty here.
i.e. datarate ~ C bits/sec.
max. data rate

Claim:- All vegas is 50% better than all reno.

L28. Application Layer:-

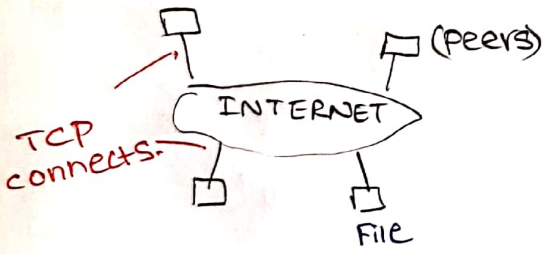
* We have unlimited flexibility in application layer. We can define our own protocols here, if needed.

* various protocols here include SMTP (Simple mail transfer protocol), HTTP, HTTPS, FTP, SSH, IMAP, LDAP, POP, TLS/SSL.....

→ Peer to peer networks:-

* Usual, starting internet was used on a client-server basis. (peer-peer interaction way like mails; which needed server to store database). ∴ not TRUE P2P scenario.

* A P2P should like look:-

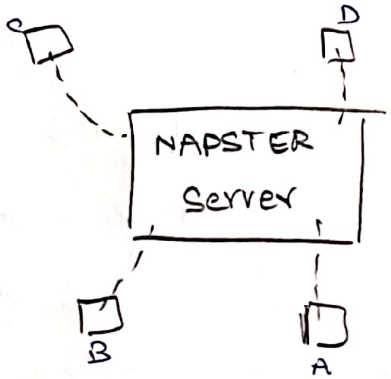


1) Bootstrapping
- How does one user 'get into' this web. How does he start.

2) What sort of service?

1990's:- NAPSTER. (digital content sharing platform).
(lot of music was pirated! so shutdown main server.)

*



Server:-
- Stores list of files; available with each user.

Eg: f1 | IPA
f3 | IPB

- Whichever peer requests for a file, gets a reply from server containing IP of file's host.
- C directly connects to A over TCP.

* Bootstrapping:-

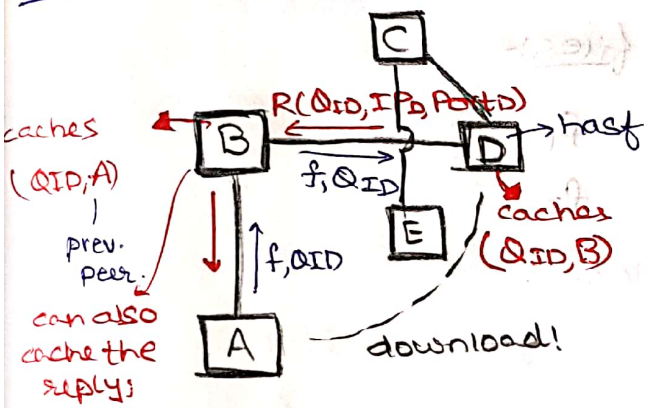
- A logs in
- A gives details of files it has.

* Drawbacks:-

centeralised ⟶ single point of failure.
(not fault tolerant)

↳ legally, easy to takedown.

GNUTELLA:- (Next gen P2P idea)



* Bootstrapping:-

(1) Application might have IPs of some other peers.

(2) Lookup IPs from one or more websites.

& connect to them. (TCP!)

* Suppose A wants file 'f'.

Can't search complete internet. Gotta do limited search.

LIMITED BROADCAST (TTL=n)

↳ this is Application TTL. not L3TTL.

maybe suppose n=2.

First method:-

Query from A has "f", QID (unique ID for query).

Reverse reply; over the path which query travelled.

2nd method:-

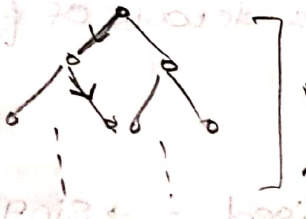
Query has IP_A as well. D will reply directly to A.

V0.4 — TTL=7

V0.6 ⟶ TTL=4 } since network became dense.

Next Idea:-

* If N nodes in the network, can I query only log(N) nodes to find where data is stored?

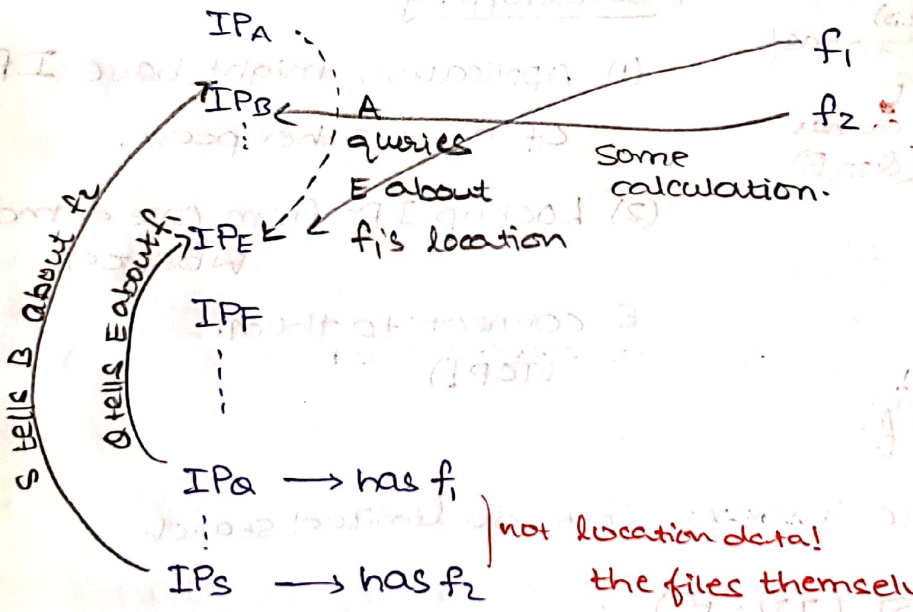


N nodes
logN levels.
there's some hope...

* lets store (location data) of files, at some peer whose identity can be calculated based on file name.

then, every peer can calculate this specific peer & query him.

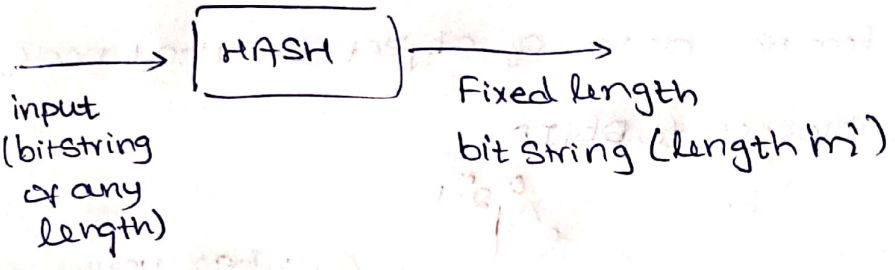
Eg: IP addresses:- files:-



* Here note that every peer should have reasonable equal load as others. Our mapping of f → node should be as such!. Even though in IP wise bunch of nodes are close or far apart, they gotta have equal chances for being responsible for 'f'.

* Better use hash functions.

→ hash function:-



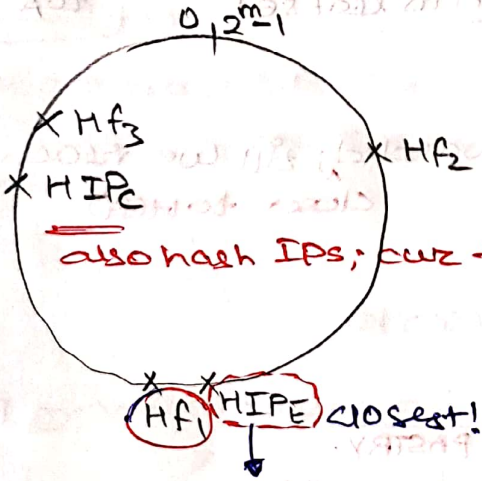
* properties of hash:

- uniformity: map inputs uniformly to 2^m space.
- Avalanche effect: small change in input, gives huge change in output.
- collision: very less probability for hash collision.

* Now; $f_1 \longrightarrow [H] \longrightarrow h(f_1)$

$f_2 \longrightarrow h(f_2)$

m = hash output length.



also hash IPs; cuz the initial IP addresses might not be uniformly spread.

now; this means that Peer-E will store all data regarding f_1.

Q ⟶ to E tells about f_1

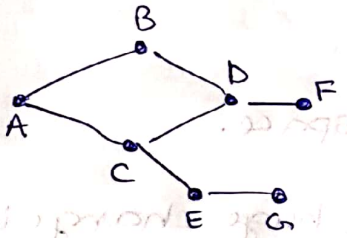
A ⟶ E. queries about f_1

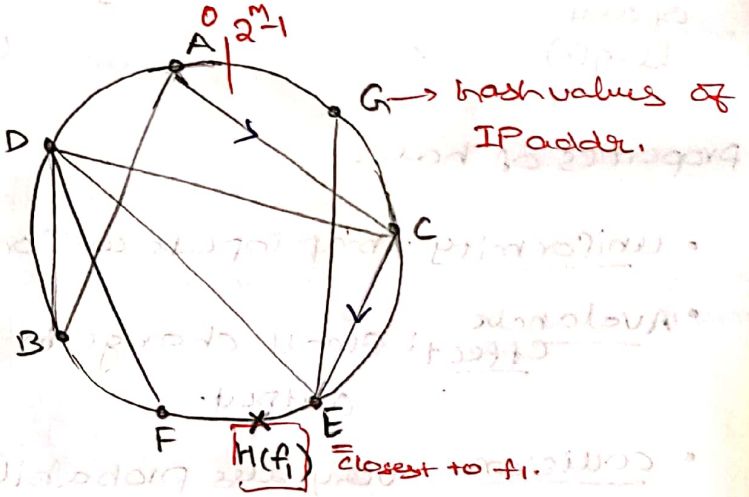
L29:

* nodeID = Hash(IP)

  objID = Hash(f)

  - we want to store location-data of object on that node, whose nodeID is closest to objID.





* our idea; route query message closer to H(f1) in this virtual space till we reach E.

- Suppose A; wants to find out where file f_1 is present.
- A sends query to the node in its leafset that is closest to H(f1).
- This query is recursively forwarded; till we reach the "node closest to H(f1)".

leafset: of A; peers connected to A

closer to H(f1)

But this need not always work.

We'll mandate a special structure among peer network; for this algo to work.

PASTRY.

* 1st query forwarding from A to E.

  then, E connects to A.

  has IP_A.

PASTRY:-

The leaf set of a node x includes $\frac{L}{2}$ closest nodes on either side of node x in the virtual space.

2^m Hash space.

Eg: L=4;

for node A. Like this, for EVERY node.

* This PASTRY ensures that our "closer to H(f)" routing protocol will reach E - the closest node to H(f).

* PASTRY ensures that leaf set of Any node is L+O(logN).

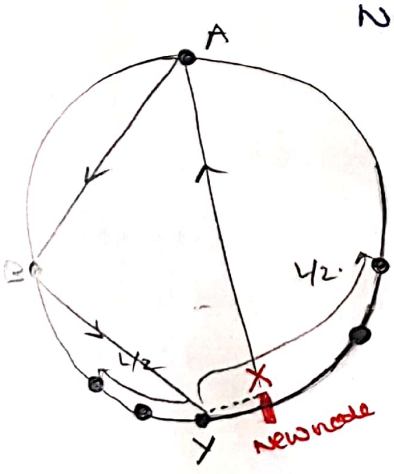
its also betta to have some more naf.

* Now; how am I going to construct pastry?

- Recursively

Suppose P2P network already has the property PASTRY.

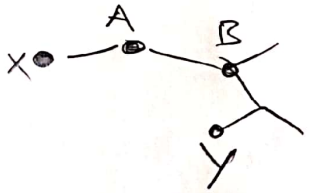
Node X is added.



1) node X routes a message to find node, whose nodeID is closer to Hash(IP$_X$) = Y.

2) From Y, X determines it L/2 neighbours on either side.

Say; X; via bootstrap, is connected to A:

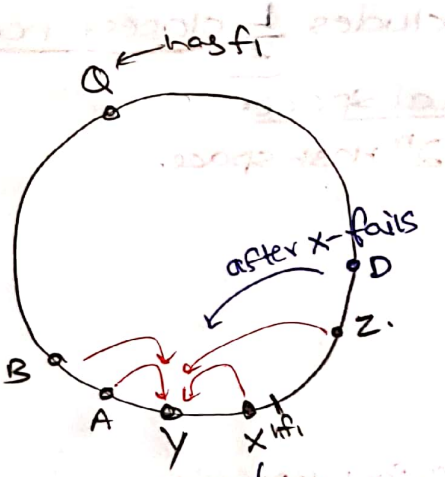


This way of storing data 'everywhere' is

DHT.

Distributed Hash Table.

→ Fault tolerance, what if a node fails...



↳ contains whereabouts of f1.

Fails!

* Lets be a bit redundant!. Every node stores the file info of 2 other nodes, on its either side!

Eg: Y stores not only of itself's; but also of B, A, X, Z.

* All peers send [keepalive] messages to peers of their leafset.

if any node within 2 distance fails, (X here)

then Y finds about a further node - W & stores its info.

L30: Domain Naming System:-

DNS.

* www.google.com — 192.168.1.13

user friendly! — bah!

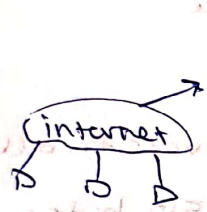
we know! — we need!

www.cse.iitb.ac.in → DNS → IPaddress.

gotta be robust.

Importance is obv.

* idea-1:-



single server

DNS ← well known IP Addr. Hardcoded.

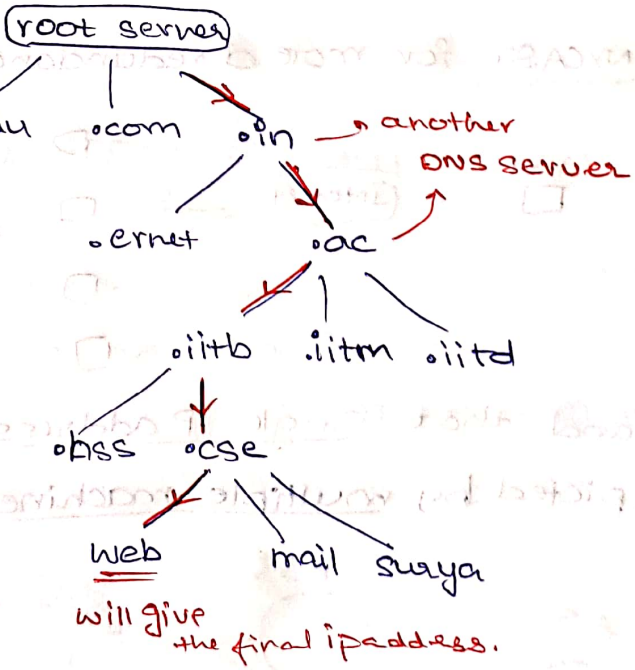
1) Single pt. of failure → lets use multiple

2) Extreme overload. → lets maintain Heirarchy.

* idea-2:- (what runs now-a-days)

URL:-

www.cse.iitb.ac.in



Hierarchy!

will give the final ipaddress.

→ Root server:-

1. well provisioned - bandwidth, CPU... cooling!

*to prevent DoS (denial of service) attacks. To prevent overload.

2. Redundancy - we've got multiple root servers.

* 13 root servers globally.

A. root-servers.net — 10 machines
B. - - - — 10 machines
⋮
M. root-servers.net

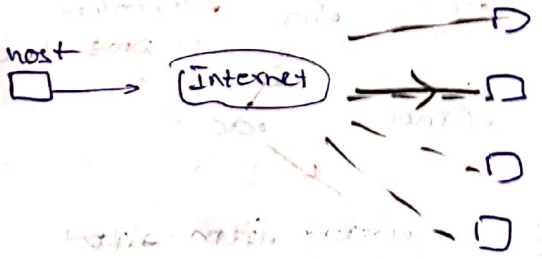
many servers finally!.

* we stopped at 13, bcoz DNS pkt size = 512 bytes.

has space for 13 addresses at most.

not convinced!

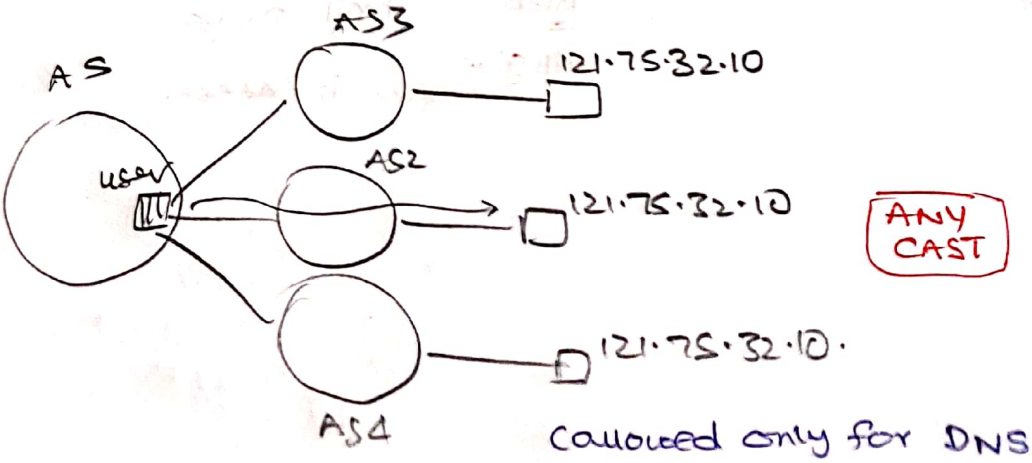
* use ANYCAST for more redundancy.



choose anyone server.

• understand that single IP address <A. root-server.net> is depicted by multiple machines. even more redundancy!

Nice!



allowed only for DNS servers) (not part of a AS).

→ Resource Record:- Inside any DNS server:-

<Name, Value, Type, Class, TTL>

- Name → URL
- Value → value
- Type → says how 'Value' is to be interpretted
- Class → IN (meaning, internet)
- TTL → to enable caching in host systems.

* TYPE :- Meaning of value:-

A	IP address
NS	Name Server (the host, running the DNS service in domain corresponding to 'Name' field)
CNAME	Canonical Name (alias of host, specified in 'Name' field)
MX	(name of host, running mail server in domain specified in 'Name')

Note: Linux command - $ dig www.google.com a few times.

what it does?

used to gather DNS info.

Domain Information Groper.

DNS runs on UDP (port 53 for server)

Eg: root server has Resource Records

<edu, a3.nstld.com, NS>

name of DNS server of <edu domain. .edu

< a3.nstld.com, 192.5.6.32, A >

server a3.nstld.com has

<princeton.edu, dns.princeton.edu, NS>

<dns.princeton.edu, 128.112.129.5, A>

next level server has

<cs.princeton.edu, ---, A>
dns

dns.cs.princeton.edu has

<penguin.cs.princeton.edu, ---, A>

<www.cs.princeton.edu, corewals.cs.--, CNAME>

<cs.princeton.edu, mail.cs.princeton.edu, MX>

<mail.cs.princeton.edu, 128.12.1.5, A>

* Look up example:-

~~DNS root server~~ IP, manually config or) DHCP.

Local DNS Server