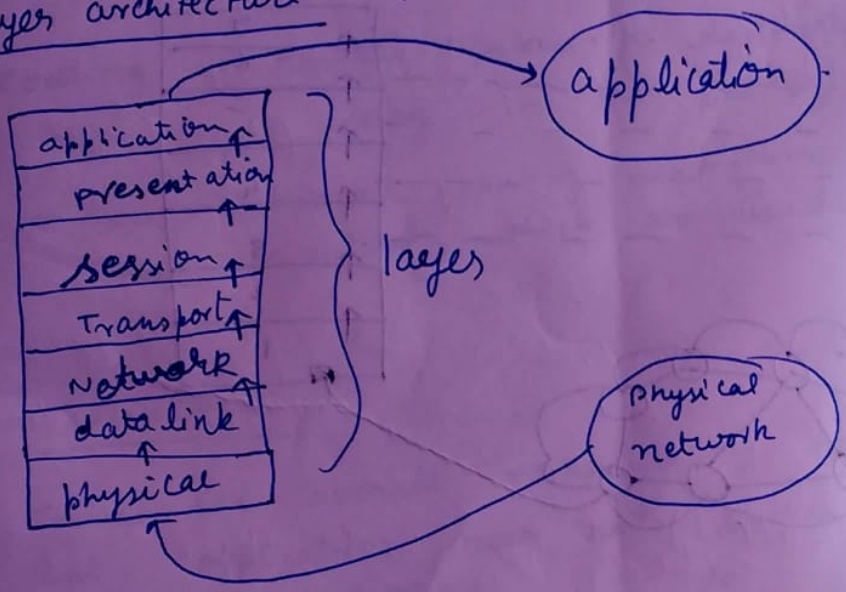
19/7/17

Layer architecture → ISO-OSI 7 layer architecture



Physical layer : Takes the responsibility of sending bits with an agreed amount of error. 1 should be received as 1 and 0 should be received as 0 with ~~probab~~ probability of error $< \epsilon$.

Data Link layer : It takes the responsibility of

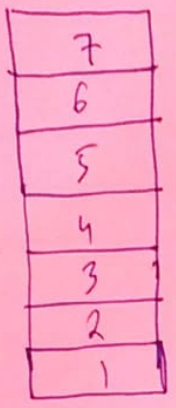
i) Demarking the beginning and end of a unit of data known as frame.

frame = user data + all other headers including the headers of data link layer

2) If required it should promise error free communication

3) Sharing common medium of transmission (Medium Access control) if required

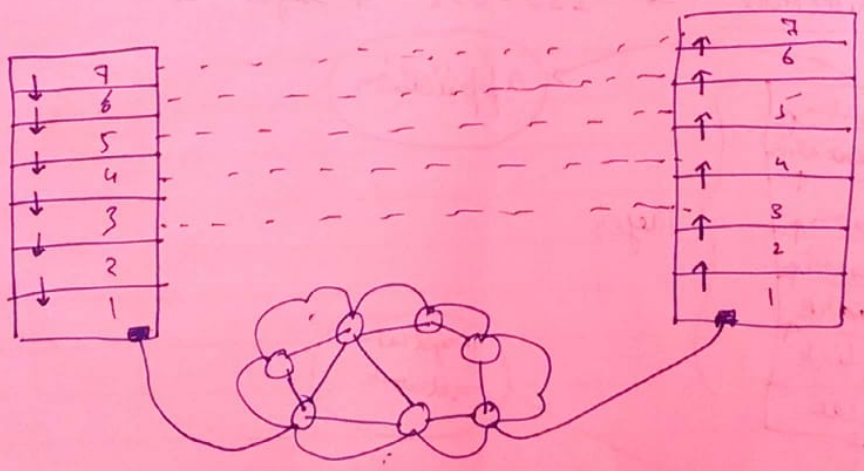
1/7/17



Application
Presentation
Session
Transport
Network
Data link
Physical.

7 layer

n^{th} layer receives service from $n-1^{th}$ layer and provides service to $n+1^{th}$ layer. while sending data to network.



- Segment, Packet, frames ??

Transport layer converts the file to segments. Segments are sent to the network layer which transforms it into a packet.

A packet is transformed into a frame by the data link.

The physical layer interprets the entire frame as a stream of bits.

The data link layer adds ~~head~~ ~~has~~ headers to the packet.
→ start & end of frames
→ error handling headers

Network Layer

1) Packet forwarding to the destination device

A device having n layer is a n layer device.
computer is 7 layer, routers / switches are 3 layers
(1, 2, 3)

- Tables are used for forwarding a packet to its destination. Destinations are given addresses, look up in the table occurs on the basis of the address.

dest	Node
001	5
002	3
⋮	⋮

form of the Routing table

ii) Routing (Packets taken to destination with minimum delay in the paths)

Runs distributed routing algorithms and creates routing table as output, which are used by forwarding process.

Transport layer

i) Takes the overall responsibility for communication so that application need not bother about the details of communication.

- Type of Responsibility →
  a) error free (reliable or unreliable) ~~as it is or~~ or not
  b) Connection oriented surface (ordered) or connectionless (ordered delivery is not guaranteed.

TCP → Transport layer provides error free connection oriented services. (Less efficient)

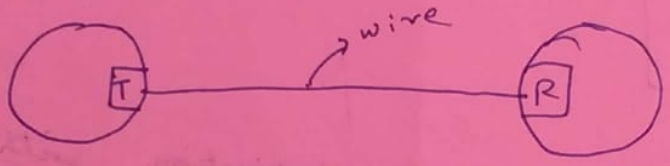
UDP → Transport layers provides unreliable and ~~connectionless~~ connectionless serives. (more efficient)

If application wants order free (error free delivery) then TCP we will be used.

Till network layer, there should be NDP for all nodes. But from transport ~~netwo~~ layers to top, direct link is there. They are ~~peer~~ peer to peer protocols.

- We create socket from layer 1 to 4, ~~we~~ allocate a port and thus run an application ~~and~~ which uses that port.

26/7/17



- Transmision wire is frequency selective. So some freq component suffers less loss, some suffer more loss. hence shape of signal gets distorted.
- To rectify the above problem we find diff freq component, we use modulation technique.

If Initially, in $\frac{\text{freq(highest)}}{\text{freq lowe}}$ is high then freq. selectivity is impact is highest.

Thus we use signals of very high ~~frequ~~ frequencies such that $\frac{\text{freq (highest)}}{\text{freq lowe}}$ is ~~less~~ less and thus impact is less.

Thus, on us. That's why we use modulation ($f_c + f_a$)
to increase freq & reduce selectivity.

modulate carrier with the msg signal

$f_L = 1KH_3$ $\quad f_H = 100KH_z$

$\therefore \frac{f_H}{f_L} = 100$

Now, if $f_L = 1MH_3 + 1KH_3$ $\quad f_H = 1MH_3 + 100KH_3$

$= 1.001\ MH_3$ $\quad = 1.1\ MH_3$

$\therefore \frac{f_H}{f_L} = \frac{1.100}{1.001} \simeq 1$

Modulations

$A\cos\omega t$

- $A = f(x(t))$ → amplitude modulation
- $\omega = f(x(t))$ → angle modulation
  - freq modulation
  - phase modulation.

- In AM we just change the position of the spectrum in freq domain.
- In FM we change the shape of the spectrum and position as well. The BW can be increased arbitrarily.
- In PM we get the similar effects like FM but BW variation is limited within a range.

$C = B \log_2 \left(\frac{S}{N}\right)$ — S: signal power, N: Noise power, B: B/W

Shenoni's Channel capacity equation.

↓ info carrying capability

We can transport more info if we have large signal to noise ratio.

i.e all have to send more power (S) to increase SN ratio.

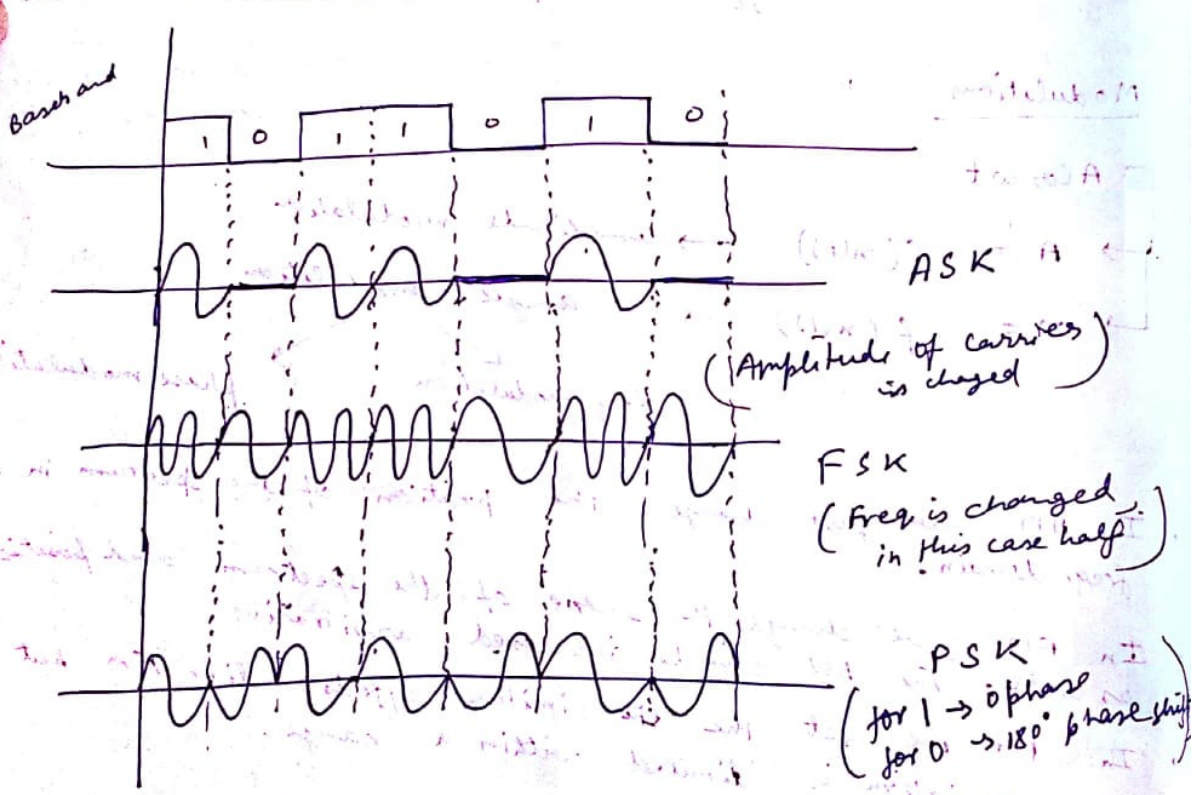
Also, increasing B/W increases the channel's carrying capacity.

Point to note : We can exchange powers for B/w to send more signal i.e Power is not superior to B/W. (B/W much more impacting)

- In AM, only powers could be changed.
- In angle modulation, it came to existance that B/w can also be same.

Refer Base band digital communication.

Digitally modulated analog ~~signals~~ carriers :



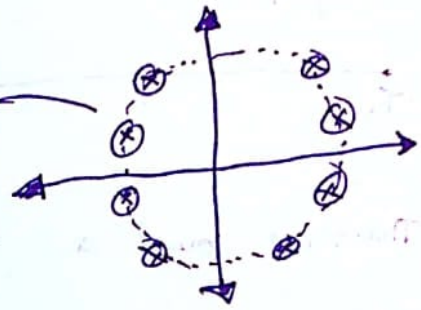
31/7/17

- ASK → BER = $erfc\sqrt{E_b/\eta}$ E_b is energy per bit, η is noise density
- FSK → BER = $erfc\sqrt{E_b/\eta}$
- PSK → BER = $erfc\sqrt{2E_b/\eta}$ (Lowest)

where $erfc = \int_{\sqrt{-E_b/\eta}}^{\infty} e^{-x}dx$

BER wise, PSK is better ...
it gives lowest bit error.

- Phase modulation → 4 array pm , 8 array p.m.



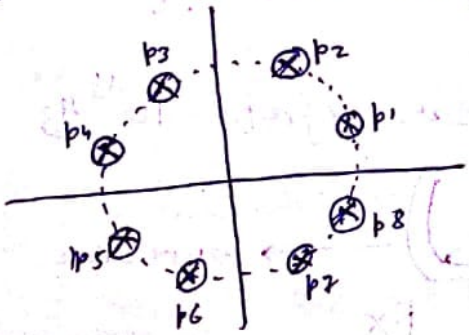
8 array pm ←

- with carrier of low freq, we can send more info.

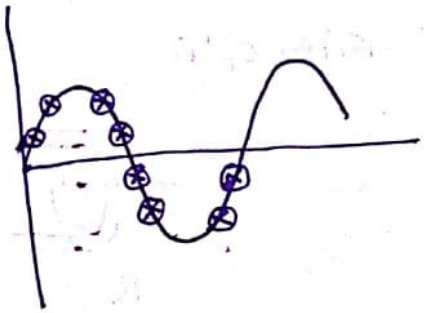
- Amplitude Phase m array ?? (32 array modulated)
- constellation diagram ??
- Any mistake (voltage) above threshold will cause a visible error
- Our motto is less noise system.

$$BPS = SPS \times \log_2 M$$

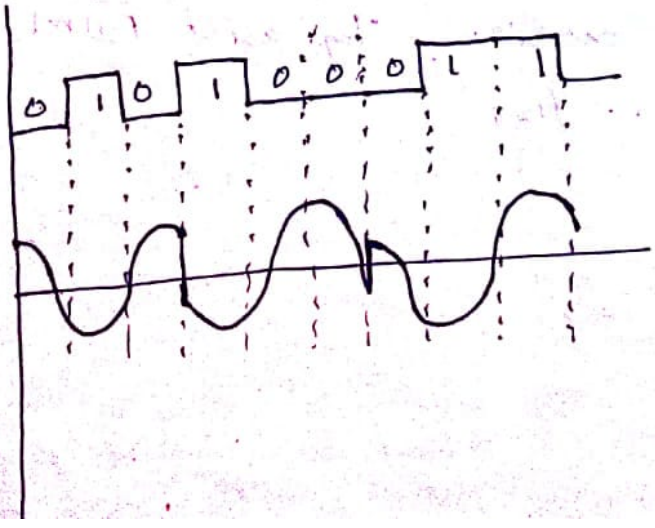
2/8/17



000 → p1
001 → p2
010 → p3
011 → p4
...
111 → p8



In this case amplitude is constant.



Transmission medium

- Telegraph line

→ Freq of 10 MHz or higher cannot be transmitted

(low freq thick skin & vice versa)

- Thus we found a conductor of type → Co-axial cable

Signal sent through this wire

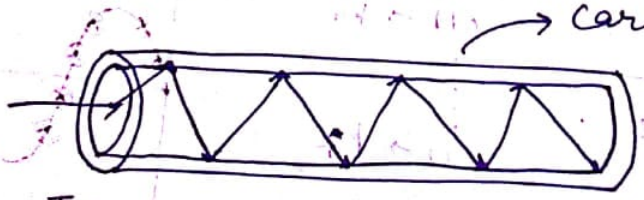
Thick Conductor

(used for long dist.)

Carrying capacity > 10 GHz but < 20 GHz
(Obvio can send less)

- ~~There~~ also we used twisted pair wire. (Can carry 10 GHz over short dist)
  If distance increases then pulses broadens.

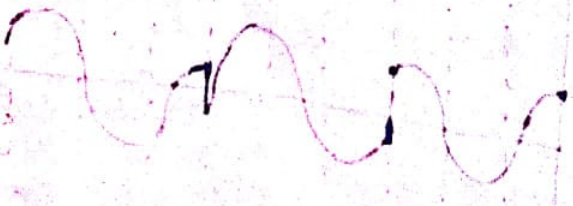
- ~~Fiber optic~~
- LASER & APD → Transmit & receive optical wave. ??
- Fiber optics



→ carrier freq 10^{18} Hz

→ WDM link

If we can use multiple carriers by using FDM (WDM) like $f_1, f_2 \dots f_{128}$

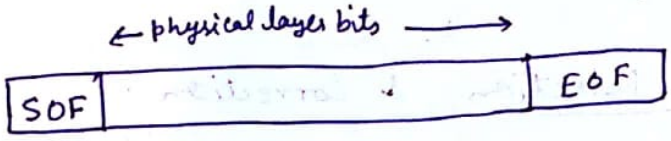


1) Frame construction (Frame delimiters)

2) Error Handling

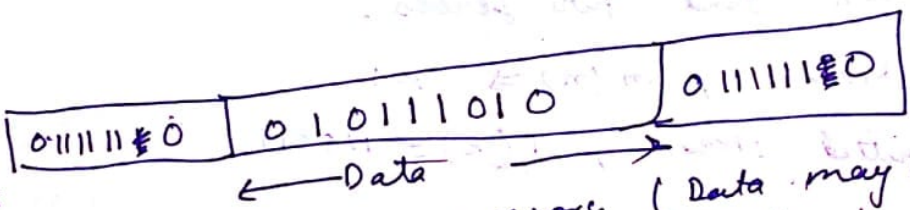
3) Medium Access control

1)



- User emits long stream of bits. These bits are divided in packets or frames.
- Frame delimitors → SOF & EOF. ~~They are~~ This type of delimition is known as Bit pattern based delimition.

Say for SOF and EOF we use the same bit pattern for delimiting a frame, the standard pattern used is '01111110'.



There will be many errors. (Data may contain EOF)

So, Before sending we can make sure that EOF/SOF doesnot exist in the data. To do this we use Bit Stuffing (Adding additional stuffed bit)

Instead of sending '1 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 0 1'

we send this '1 0 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 1' (Stuffed Bit)

↑ Stuffed Bit

i.e If we find 5 consecutive 1's after a 0 then insert an additional 0 after the fifth 1. ~~Add~~ Apply this to the whole data stream. This is done at sender side.

At receiver, If we receive '0' followed by five consecutive 1's and a '0', then take away the zero. i.e Destuff the zero. Doing so, we will receive the original data.

16/8/17

Error Detection & Correction

Basic Principle is to attain reliability through redundancy.

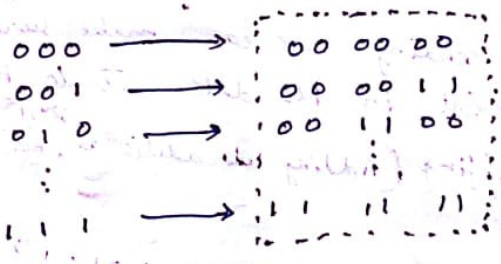
Ex: two symbols 1 & 0 are to be transmitted. Say the message stream is 101. While transmitting the transmitter adds some redundant bits.

Rule ⇒ If the input msg stream has a symbol 1 then send 2 ones and if the input stream has 0 then send two zeroes.

∴ msg stream, m(x) ⇒ 101

transmitted stream, T(x) = 110011

The valid streams at the receiver assuming 3 bits msg →



These are the valid msg

(Only 8 out of 64 possible msgs)

Transmitter sends T(x) & Receiver receives R(x)

Thus, • if no error ⇒ T(x) = R(x)

• if there is error ⇒ R(x) ≠ T(x)

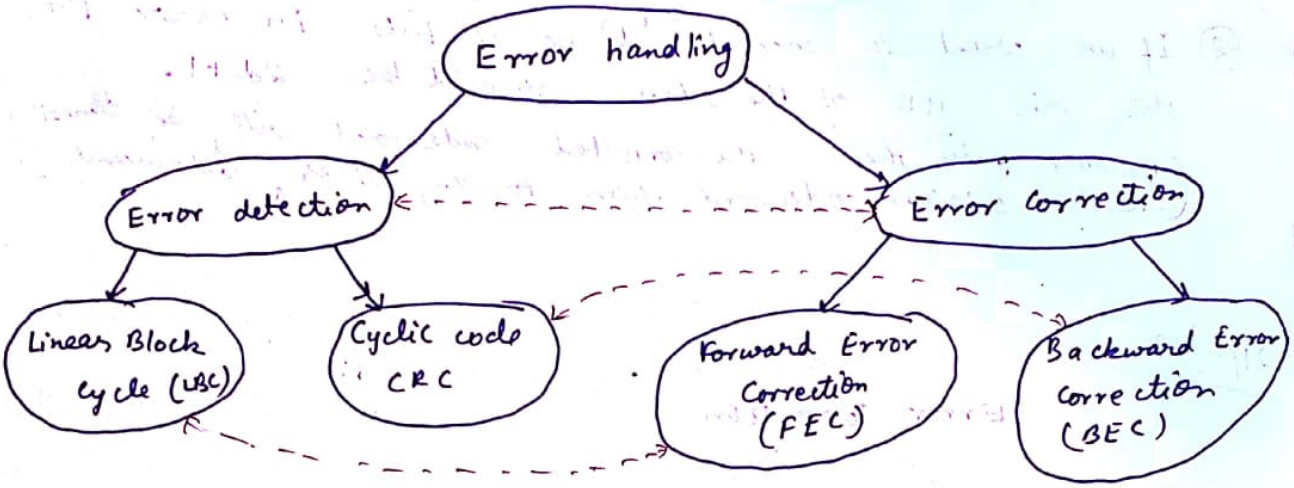
Say T(x) was 00 00 00. Due to noise in the channel it may become 01 00 00. Receiver understands that error has occured and it can detect it. (8 valid msgs). (for pairwise error, receiver cannot detect error properly)

We change our rule to 3 ones & zeroes.

So,
000 → 000 000 000
001 → 000 000 111
⋮
111 → 111 111 111.

Say T(x) = 000 000 000 is passed through a noisy channel and receive R(x)

Say, R(x) = 010 000 000



Hamming Distance

Let me have a coding system where the following words are valid codewords.

Say w_1 = 11 00 11
w_2 = 10 10 01
w_3 = 01 10 00
w_4 = 01 11 11

H.D bet. w_1 & w_2 = 3
H.D bet. w_2 & w_3 = 3
H.D betw. w_3 & w_4 = 3

Similary we can find H.D between any 2 w's.
Thus if we get ${}^4C_2$ H. Distances & for each pair of valid code words, then we can determine the min hamming distance of the coding scheme.

Let d = min HD of the coding scheme
Then d no of errors occuring in one codeword can transform it into another valid code words.

Thus, detection capabilities will be d-1 since if d errors occur, then codeword changes to another valid codeword.

Thus, The power of a coding scheme in terms of no of errors it can detect and correct is determined by the minimum H.D of the scheme.

① If we want to detect 'd' no of bits in error then minimum HD of the scheme should be (d+1)

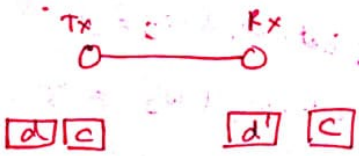
② If we want to correct 'd' no of bits in error then the min HD of the scheme should be 2d+1.
(Because only then the corrected codeword will be closest to the original codeword than to any other valid codeword)

21/8/17

Error Correction

Forward correction (for real time apps)

Backward correction. (Not suitable for real time applications)



$c = f(d)$

∴ At receiver, we find c' in terms of d'
i.e. $c' = f(d')$

If $c' == c$ then no error.

General Error detection Principle

- most popular method in linear block coding.
- Let n be the no of information bits in a unit block of data
- The transmitter adds up 'c' no of check bits along with the data bits. These chat bits are func of the

data bits. In effect, the check bits increases the hamming distance of the coding scheme. Hence with the help of added check bits we can differentiate among valid and invalid code words. Thus we can detect error and with sufficient no of check bits even we can correct error.

n ⟶ no of information bits

c ⟶ no of check bits

Hence size of the message is $n + c$

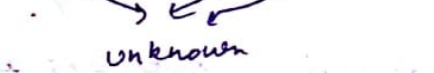
Let $m = n + c$

Thus, we define (m, n) code as a coding system where m is the total msg size and n is the no of info bits

For eg: $(7, 4)$ code → $m = 7$, $n = 4$, $c = 3$

Each check bit is derived from a suitable no of info bits.

D_3	D_2	D_1	C_2	D_0	C_1	C_0
say 1	0	1	—	1	—	—



$$C_2 = D_3 \oplus D_2 \oplus D_1$$
$$C_1 = D_3 \oplus D_2 \oplus D_0$$
$$C_0 = D_3 \oplus D_1 \oplus D_0$$

} func for deriving check bits

In this case, $C_0 = 1$, $C_1 = 0$, $C_2 = 0$

Now, At the receiver, it separates the bits

received signal ⟶ 1010101

→ 001 check bits [say C_R]

→ 1011 Data bits [say D_R]

The receiver genarates the check bits from the received data bits D_R

If there is no error then the generated check bits will be same as the received check bits

Now, if we calculate $C_g \oplus C_R$ then it should be (0 0 0) if there have been error in both D_R and C_R.

This $E = C_g \oplus C_R$ is called error syndrome vector & $E = (000)$ is represents correct reception

Actually, value of E determines the position of error.

(7,4) code has capability of detecting atmost 2 bit error and correcting atmost 1 bit error.

If the no of check bits is c, then the error syndrome can generate 2^c no of patterns.

7 diff position of errors may occur. Therefore 1 errors occur in $^7C_1$ ways.

Thus in general, k errors can occur in $^m C_k$ way.

i.e Now, if we are interested in correcting error up to k no of errors in coded message of size m bits then c should be chosen in such a way that

$$2^c \geq \sum_{i=0}^{k} {}^m C_i$$

or $$2^c \geq \sum_{i=0}^{k} {}^{n+c} C_i$$

This is a computational problem to find c. (rather min c)

we need $\min c\left[2^c \geq \sum_{i=0}^{k} \binom{n+c}{i}\right]$

100101

$C_2 = 1$ 0
$C_1 = 0$ 0
$C_0 = 0$ 1

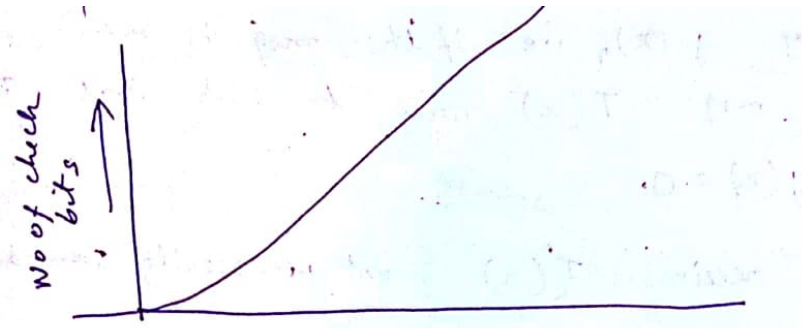
101 = 5th bit error

10101**0**1

C_2	C_1	C_0
0	0	1
0	1	1
0	1	0

10

$^m C_k \rightarrow$ k errors to be corrected



Thus, this method becomes inefficient with ~~no~~ increase in no of information bits. as more error. will require more no of check bits.

23/8/17

Backward Error Correction (BEC)

1) The Tx transmits the message after adding some ~~dont~~ check bits ~~as~~ so that error can be detected at the receiver. [not corrected]
2) The receiver examines the correctness by using the received and ~~gen~~ generated channels. If error is detected then it requests for retransmission

Efficient Error Detection Codes

Generally occuring errors are ~~busty~~ bursty in nature

say error in → 1 0 1 0 0 [* * 0 * * 0 ⊠] 1 0 0 1 1 1 0

Burst error of length 7

We need error detecting codes that can efficiently detect burst error. One such scheme is cyclic redundancy coding (CRC)

Principles

1) Both sender and receiver agree to use a generating function $g(x)$.
2) The sender adds up some redundant bits with the info bits so that the transmitted msg is exactly

divisible by $g(x)$, i.e if the msg is $m(x)$, then the transmitted msg $T(x)$ will be such that

$T(x) \,\%\, g(x) = 0$

3) Receiver receives $T_R(x)$ { not necessarily same as $T(x)$ } and then perform $T_R(x) \,\%\, g(x)$.

If the value is 0 then the msg has been received correctly, Else it requests for retransmission

How to generate $T(x)$?

Let $m(x)$ be the msg polynomial and $g(x)$ be the generator polynomial of degree k.

We find $m(x) \cdot 2^k$. Then find $R(x) = \{ m(x) \cdot 2^k \} \,\%\, g(x)$

The transmitted msg will be

or $T(x) = m(x)\,2^k - R(x)$ [+ also correct]

For eg:

Let $m(x) = 10101$ & $g(x) = 1001$

$(x^4 + x^2 + 1)$ $(x^3 + 1)$

Here degree of $g(x)$ i.e $k = 3$.

So, $m(x) \cdot 2^k = m(x) \cdot 2^3 = 10101000$

So, $R(x) = 10101000 \,\%\, 1001$

$= 111$ [modulo 2 system]

$\therefore T(x) = 10101000 + 111$

$= 10101111$

Refer modulo 2 arithmatic

9)168(1
9
78
72
6

110

10111
1001)10101000
1001
01110
1001
1110
1001
111
100
11

modulo 2 division (type of

BEC Protocols

i) Stop and wait

ii) Sliding window Protocol

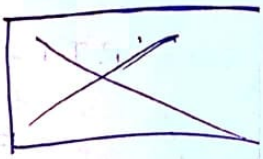
- **Stop & Wait:**

i) Sender sends the msg^ (frame) & waits for acknowledgement

ii) receiver receives the msg^ (frame). If no error, then it sends Acknowledgement but if error is detected, it does not send acknowledgement.

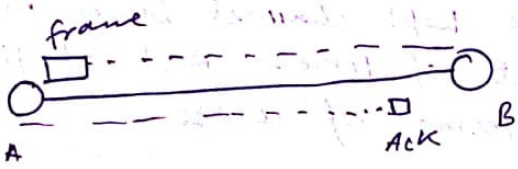
iii) The sender waits for a predetermined amount of time and if ack... is not received within that time then retransmits the frame

iv) If Ack is received properly then next frame is sent by the sender



Note: It works fine if we can differentiate between two consecutive frames

- we need only 1 bit (seq. bit) to differentiate as we send the next frame only when ack. of previous frame is received.



- **Transmission time of a frame:**

It is the time required to transmit (upload) the frame to the communication channel.

Let $C \rightarrow$ Capacity of channel in bps

$F \rightarrow$ frame size in bits (packet size)

Then transmission time $t_t = F/C$

Eg: Say $F = 10,000$ bits $= 10^4$

& $C = 100$ Mbps $= 10^8$ bps

$\therefore t_t = 100$ μsec

Propagation Time

Time required by the frame front to reach from the sender to the receiver.

$$t_p = \frac{\text{distance}}{\text{velocity of propagation}}$$

Say, $d = 1000$ Km

$v = 2 \times 10^5$ km/sec

$\therefore t_p = 10^3 / 2 \times 10^5 = 0.5 \times 10^{-2}$

$\Rightarrow \boxed{t_p = 5 \text{ msec}}$

Round Trip Time

$RTT = 2t_p + t_t$ as we consider ack ~~trans~~ transmission time is ~~small~~ as ack is small

$= 10 \text{ ms} + 0.1 \text{ ms}$

$\Rightarrow RTT = 10.1 \text{ ms}$ → time taken to receive ack (approx)

Here we kept channel busy only for t_t amount of time. Rest of time was a waste in terms of usage.

25/8/17

If we use simple stop and wait protocol, then ~~uti~~ utilization of the channel,

$\eta = \frac{t_t}{2t_p + t_t}$ ~~$\frac{t_t}{2t_p + t_t}$~~

$\eta = \frac{t_t}{2t_p + t_t} \times 100\ \%$

$\eta = \frac{0.1}{10.1} \times 100 \simeq 1\ \%$.

Inefficient if $t_p >> t_t$

Sliding Window Protocols

Instead of sending 1 packet and waiting for the ACK before sending the next packet, we send W no of packets in sequence without receiving the ACK (for atmost least W packet).

The optimum size of W is such that the whole link is exactly filled up with packets. This value of W is W_{opt} and delivers 100% efficiency of the links.

So, $1 = \frac{W_{opt} \times t_t}{RTT}$ $\left[\eta = \frac{W_{opt} \cdot t_t}{RTT}\right]$

As efficiency in such ~~schem~~ scenarios will be given by $\eta = \frac{W_{opt} \times t_t}{RTT} \times 100$

So, $W_{opt} = \frac{RTT}{t_t} = \frac{11}{1} = 11$ [when we use $c = 10$ mbps]

But if window size $W < W_{opt}$

then $\eta = \frac{W}{W_{opt}} \times 100\%$.

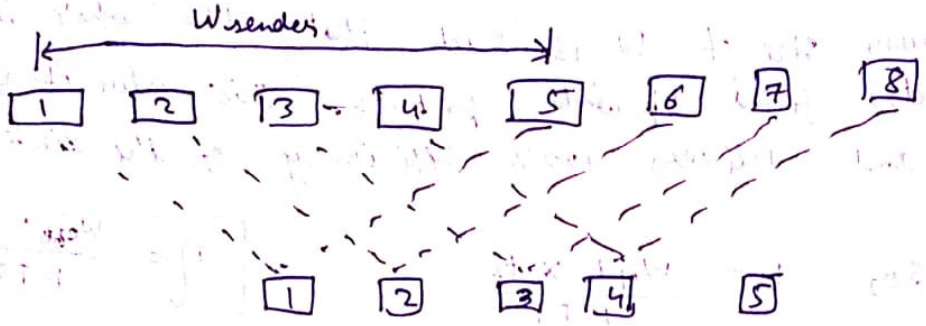
But now as we are sending W no of packets so we need to identify each of the packets with sequence numbers. Hence each frame must have a sequence number header.

Apart from Data part, others are known as DLL headers.

The size of the sequence no ~~shoud~~ should be $2W_{opt}$. Thus in this example we need 22 distinct no's, and this requires $\lceil \log W_{opt} \rceil$ no of bits.

• Go back & pro → Go back to last not-transmitted packet & continue onwards

• Selective repeat/reject → just send the non transmitted packet and continue the rest from the where it packet to be sent " was left.



Say 3 was not sent. Then as slider reaches 3 - 7 window, it knows there is error & thus any 1 of the above 2 protocols will be followed. (Error detected at 7)

28/8/17

selective repeat/reject → Receiver will temporarily store all the frames received after 3 in link layers buffer. and will send a negetive ack acknowledgement for frame 3 to the transmitter. Then transmitter will send only frame(3) and resume transmission from where it were.

stored in buffer bcz because frames are not in order and hence cannot be sent to the network layer.

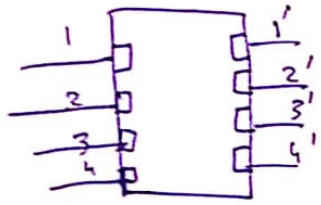
(Used earlier) Go Back N

- Bandwidth inefficient [more no of retransmission]
- Memory efficient (need not store)

presently used Selective repeat

- Bandwidth efficient
- memory inefficient (Hungry) (requires buffer storage)

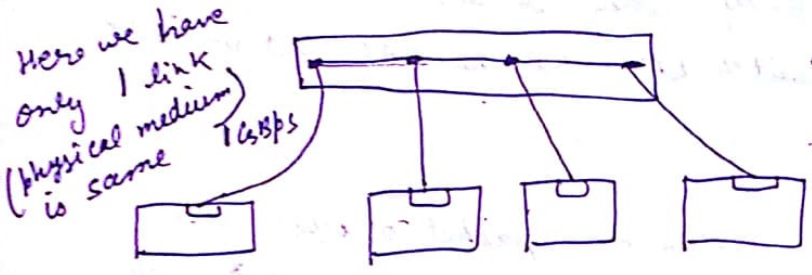
Medium Access Control



This has 4 Gbps throughput.
in general n Gbps

1 & 1' are physically same
1 is for input & 1' is for output,
& 1 cannot send to 1'

All links are physically different. Thus ⇒
Each port can utilize complete 1Gbps bandwidth.

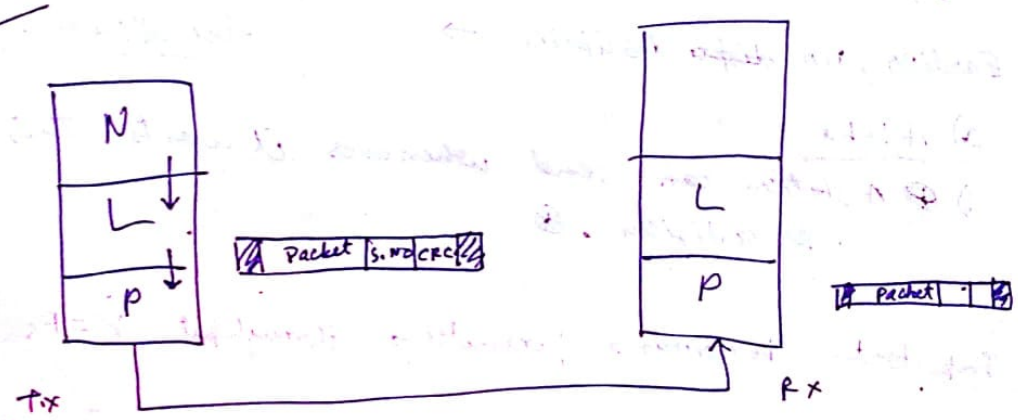


- Used alone, gets 1 gbps
- All working together, each gets $\frac{1}{4}$ Gbps

B/W available to individual node sharing the hub

$$\text{hub} = \frac{1}{N} \times \text{link speed}$$

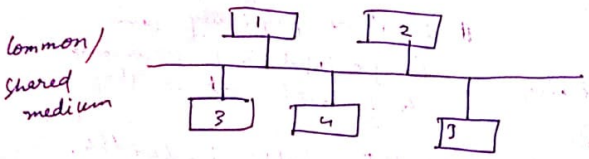
30/8/17



The DLL in Rx delemits the frame by firstly removing the headers.
Then correctness of frame is checked. If correct then CRC & S.no is shredded.
Then the packet is sent to Network layer which has its own headers within the packet.
It will check for correctness of its headers & so on.

In Hubs, some machines are connectly near to each other, some are far.

Thus connection is represented as



MAC is required only when we are using shared medium .. Eg → Switch LAN has no MAC but wifi has MAC

~~In LAN~~ If 1 sends, then the packet occupies the whole channel and others cannot send. If other ~~host~~ station sends simultaneously then data is corrupted. This is called collision.

~~Protocol~~

Earlier, No ~~disp~~ discipline →

we cannot send another packet within $2t_t$ else collision.

a) Aloha

1) A station can send whenever it wants. This leads to collision. ⊗

Important performance parameter → throughput $\gamma = P_s G$

P_s ↙ Probability of success

G ↓ Amt of load put into network

If more stations are greedy of sending data, then P_s will be low and thus γ is low.

But if we are less greedy then G is less and thus we are not using ~~total~~ the complete capacity.

So γ can also be said as successfull Data transmission in the network.

⊛ and the throughput is bounded above to only 18%.

We can prove that 0.18 throughput achieved at a load

To crack AES, total energy of solar system is required.

~~I~~

~~of~~ Key is changed frequently

(a) Privacy

(b) Tamper detection (message authentication)

(c) Non - Repudiation (Non deniability)

⊛ B will know the encryption algo of A and the value of K

Another type of substitution :

length of the key grows and becomes 26.
a → p
b → m
c → z
#

Every letter has its own probability. So, Substitution is not robust and easy to break as we can view the occurrance probability of letters and map it with same probability in english.

Say we take a long length word. We permute the letters 100 times and each time we apply substitution. This makes the key longer and thus encryption is robust.

Standard Algorithms that make use of both permutation and Substitution repeatedly

1) DES → Data encryption Standard with key 64 bits, increased till 128 bits

2) 3DES → Triple DES ~~with key 64 bits till DES, after which~~

These are now obsolete as they were attack as its weakness was found.

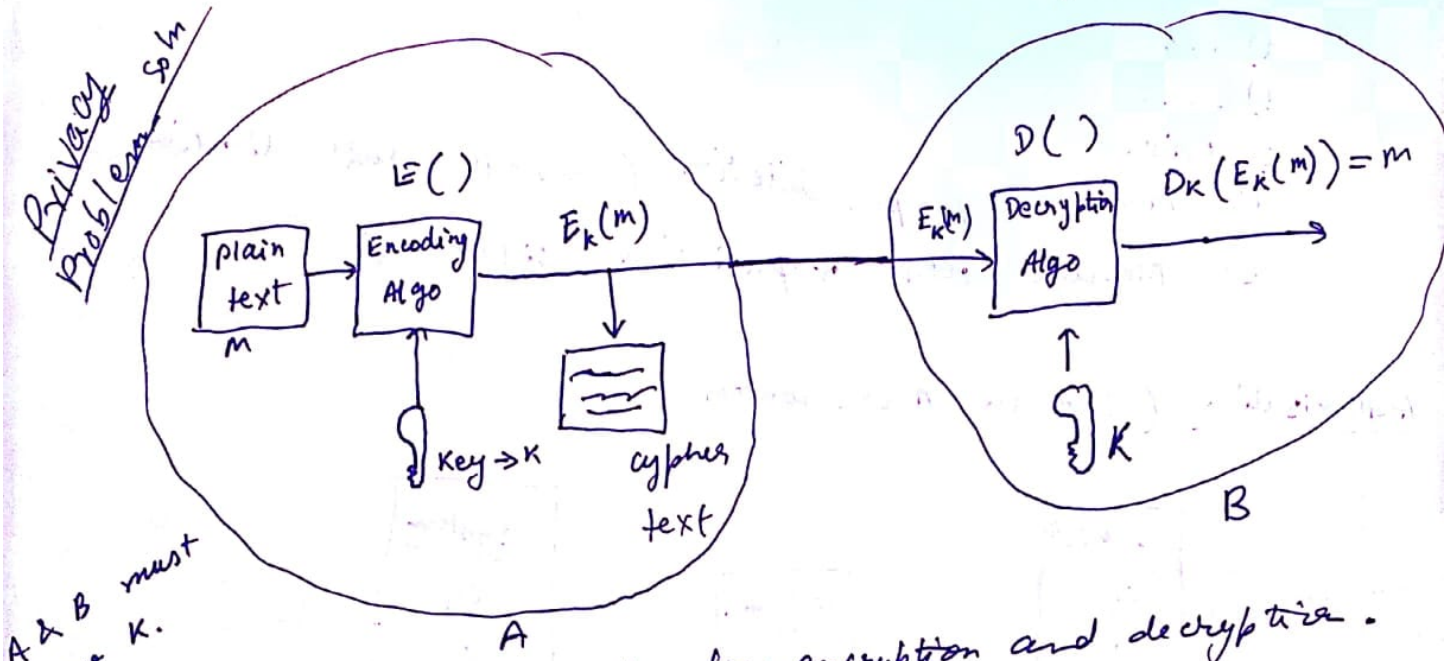
3) Advanced Encryption Standard (AES) ✓ used presently. (Key 512 bits)

Algo Based on difficult problem like prime factors of large no's, elliptic curve etc

This type of Access control is Discretionary Access control (DAC) (not good)

The list is created ~~by the~~ the admin and discretionary powers lies with him.

(MAC)
* Mandatory Access control → Not in course (Not good)
* Role based access control (RBAC)



Same key is used for encryption and decryption.
Hence it is symmetric

Basic Principle of ~~encryption~~ encryption →

① Substitution

Substitute one character by another character.

Shift 3 place forward

Here $K = 3$

a → d
b → e
c → f
:

② Permutation.

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

: we jumble the position randomly.

• I have to specify the position for the alphabet
This becomes the key.

1/11/19

Security

i) Before network

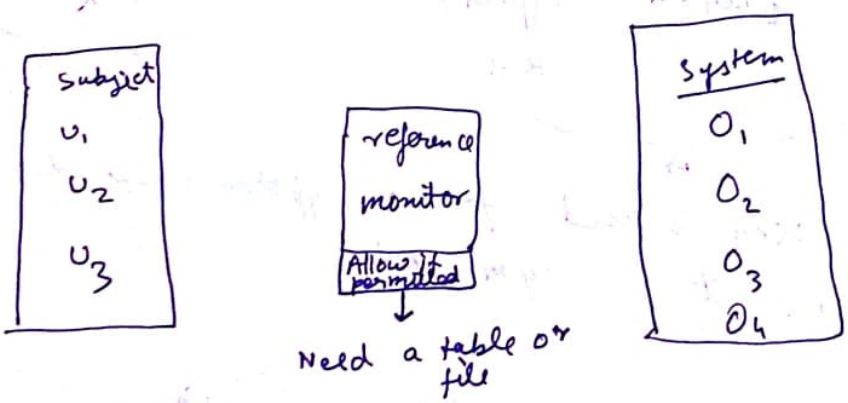
- Authorisation - Implemented through access control.
- Authentication - was initially implemented with passwords

what you can do → who you are.

ii) After network

a) ...
b) ....
c) Privacy
d) message tamper detection (message authentication
e) Message non repudiation. (signature)

Authorisation (through Access control)



Need a table or file

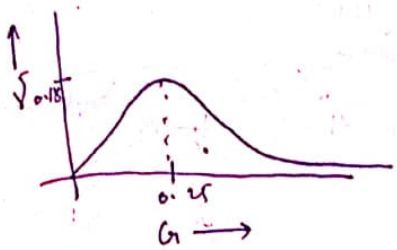
Various users have diff permissions for different object

The table has a format →

Object	Subject	Rights
O_1	U_1, U_2	(r-x), (r--)

Access control list.

for more users, list will be large Hence we group it by object.

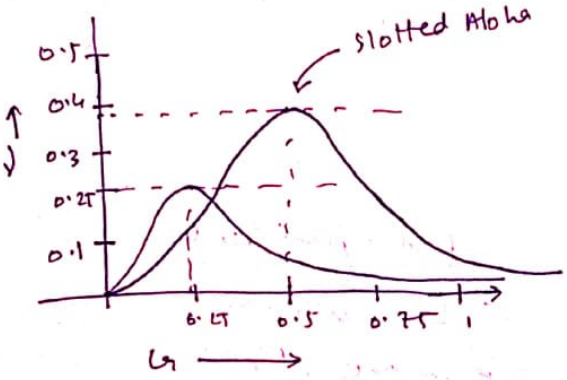


We can prove that 0.18 throughput can be achieved at a load of 0.25

This is the graph for ALOHA.

b) Slotted Aloha (S - Aloha)

- If a station wants to ~~send~~ send, then it should send in certain time ~~slots~~ slots. (unlike Aloha where we can send at any time).
- They can ~~send~~ start sending at the beginning of a time slot only.
- This ~~improves the aloha in successful~~ gives improvement over aloha and the maximum achievement throughput of 0.36



Assumption ⇒ Frames are of same size for all stations.

As each frame is sent in a ~~time fixed time~~ slots of fixed time.

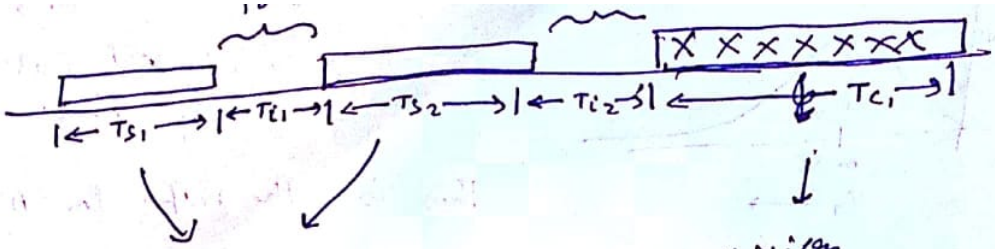
Carries Sense Multiple Access Protocol (CSMA)

When station has data to send,

- It listens to the carries (it detects carrier - it knows that some station is sending). If the channel is busy it doesnot send, but keeps on sensing the channel to find it free.
- When ever the channel is free it sends immediately.

State of channels → ① Successfully transmitting
② Under collision
③ Idle

This is persistently ~~keeps a track~~ ~~collision~~ problem



Sec

1) A

what

Utilization or throughput,

$$\rho = \frac{t_s}{t_s + t_i + t_c}$$

where $t_s = \sum_{j=0}^{\infty} t_{sj}$

$t_i = \sum_{j=0}^{\infty} t_{ij}$

$t_c = \sum_{k=0}^{\infty} t_{ck}$

ii)

$t_s + t_i + t_c \rightarrow$ total time

- whenever the channel is found ~~for~~ free ~~face~~, it waits for a random amount of time and if it is still free, then it sends. This is Non-persistant.

Aut

Collision problem ?? Refer.

Persistent	Non - persistant.
i) more collision	• Less collision
ii) Less idle time	• more idle time

Q) Which one is better? Can we modify further?

More load → more ~~persist~~ collision.

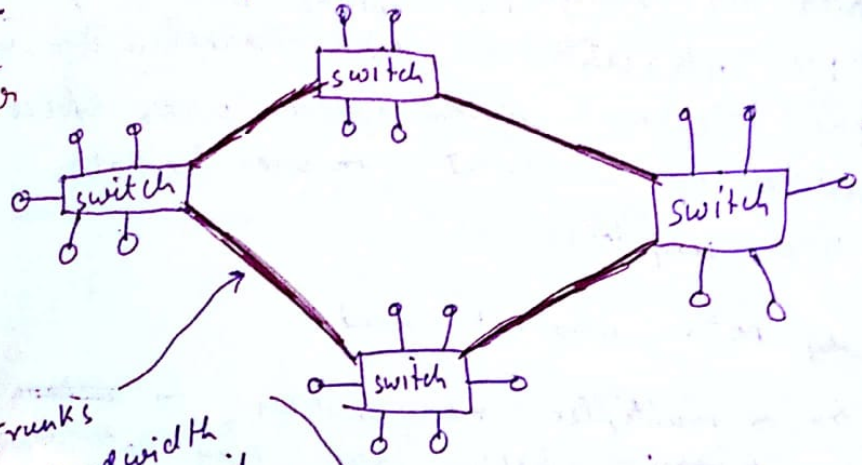
thus, for less load, we use persistant.

with increase in load, we move to dynamically persistant technique. (p-persistant)

4/9/17 Class in ⊛ TOC copy

18/9/17

Network layer

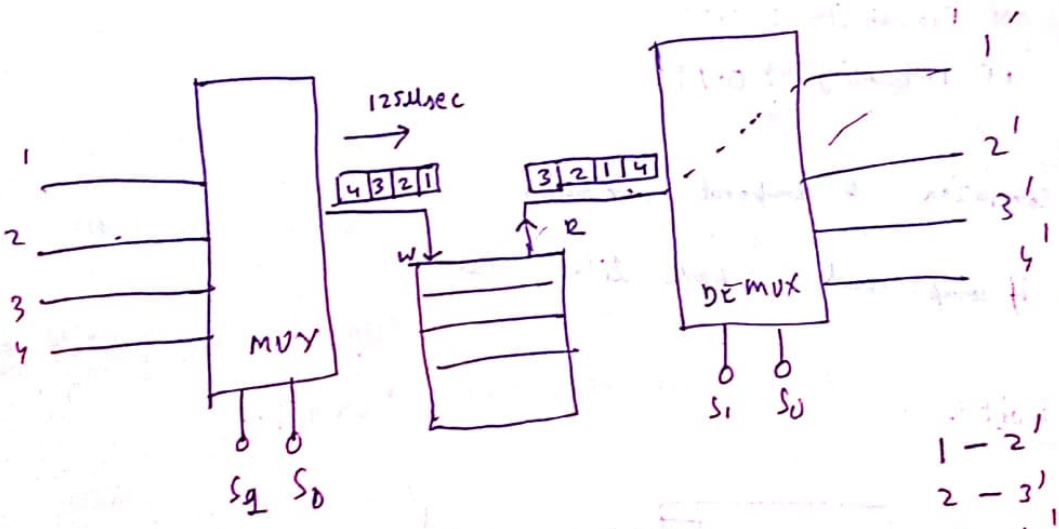


Trunk's

(High bandwidth as they transmit more traffic ~~sss~~ simultaneously)

Q) Why ~~isis~~ such telephone network (existing ones) were not used for data transmission.

switches, sampling (space & time switch.)

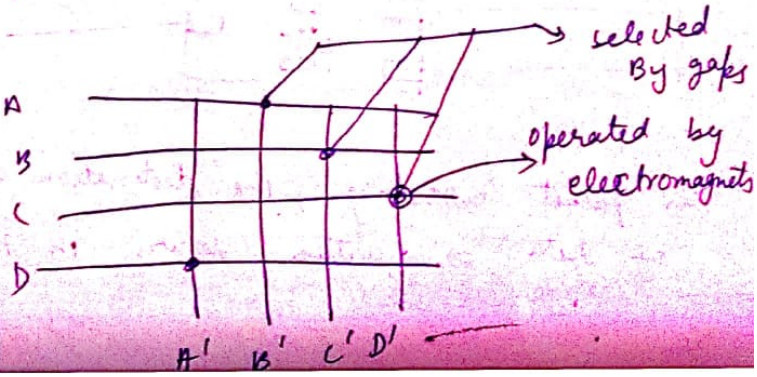


1 – 2'
2 – 3'
3 – 4'
4 – 1'

This is ~~the~~ time swich.

(Links ~~Data~~ are mapped as per requirements)

Space switch →



⊛ Data sources are busty in nature with only around 1% utilization of the channel. This is why the telephone system which uses space time switches is inconvinient to use for data transfer b/w computers.

Thats why POTS was not used.

If we try to multiplex many sources to ~~a channel~~ time frame it may happen they may not want to send at the same place.

20/9/17

- characteristic of voice source:

ON probablity : 0.4
OFF probablity : 0.6

- Characteristic of data source:

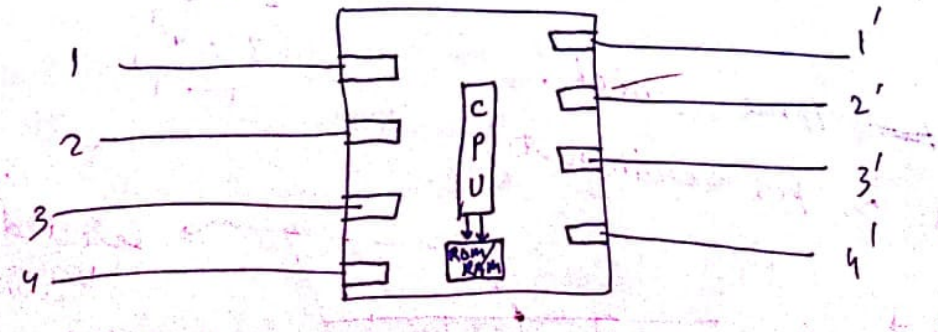
ON Probablity : 0.01
OFF Probablity : 0.99

- Spacial Corelation & temporal correlation

Media if compressed behave like data

Packet Switch

4 x 4 p.s



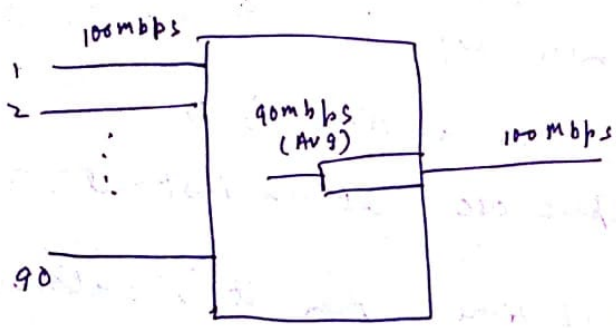
In packet switching, the destination of a unit (block) of data is written down in the data block itself.

Packet gets stored in buffer → generates Interrupt to CPU.

Through DMA ← ISR does something ??

the packet is put in destination Buffer (How??)

- If another packet arrives with the transmission time of previous packet, then the new packet is stored in buffer.
- These are ASIC designs for high speed switches.



~~Buffer~~ There will be growth & shrink of buffer size

So, how much time, a packet spends in buffer is random. It depends on size of buffer at that time. (delay)

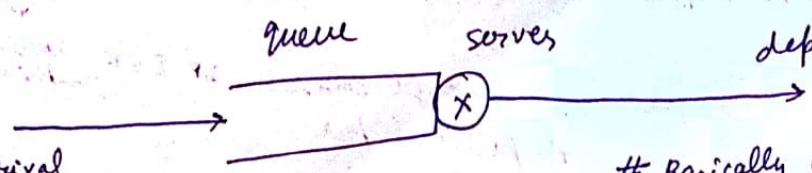
Packet switch provides statistical multiplexing at the cost of random delay & space.

But channel utilization is much more

We dissociates time from space by using Packet switch.

In this diagram, prob of selecting a channel of 100mbps is say 0.01. So & overall is less than output channel capacity

9/10/17



1) # Basically we have to find avg length of queue.

arr process / departure process / no of server / length of queue

2) # We can calculate moment of a statistical process by $\sum_{i=1}^{N} (\bar{x} - x_i)^K$ where K denotes K^{th} moment

Random Ergodic process : Process where time avg is equal to ensemble avg.

Time avg → Avg no of osc over a time is 0

Ensemble avg → Avg of multiple osc at an instant.

If all moment are independent of time, then it is stationary process.

Stockastic system → Processes originating & coming from ensemble source.

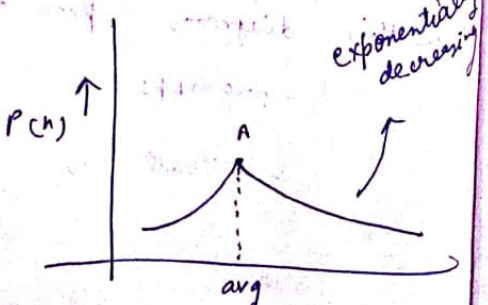
$$P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}$$

Prob that there will be n no of arrival in t duration of time

If we can get such a process which has probability this process, then the process is poisson process.

Prob of finding particle in avg state is highest at A.

Prob that value of random variable will be negligibly less if the value is far away from the avg (very less than the avg).



Such process is Poisson's process

Thus arrival process is poisson's process. (Departure process also)

process → Poissons (M) [Notation]

Representation of queue $M/M/1/\alpha$ (AP: M, DP: M, no of servers: 1, α → queue size)

This queue will be analysed further. So, we have

Avg Arrival Rate = λ

Avg dep. rate = μ

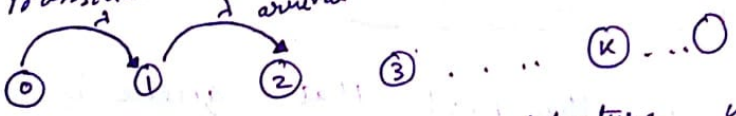
Let the state of the queue is defined as the no of packets in the queue and let P_K denotes the probablity that the queue is in state K.

Hence avg queue length $N = \sum_{K=0}^{\alpha} K P_K$

Now, for P_K, we make some assumption.

Markov chain Representation of the queue.

Note: (Not in Markov) If packet arrives in state i, then there will be transition to state i+1.



Just like arrival, there are departure which brings back to the previous state

If queue is stationary, Rate of transision of arrivals is eq to that of departure, i.e from i^{th} state to J^{th} state while arrival & vice versa.

No of transition from i to j can be found if we have prob for queue is at 0 state → P_0

$P_0 \lambda$ → expected (rate) no of transition from 0 to 1 (say)

Rate of transition from 1^{th} state to 0^{th} state is $P_1 \mu$

Applying the condition of equilibrium we get

$$P_0 \lambda = P_1 \mu$$

$$\Rightarrow P_1 = P_0 \lambda / \mu$$

$$P_K = P_0 \left(\frac{\lambda}{\mu}\right)^K$$

$P_2 \mu = P_1 \lambda$

$P_2 \mu = \frac{P_0 \lambda^2}{\mu^2}$

Let $\lambda/\mu = \rho$ $\quad \therefore P_1 = P_0 \rho$

Thus, $\boxed{P_K = P_0 \rho^K}$

We know that $P_0 + P_1 + P_2 + \dots + \alpha = 1$

$$\therefore P_0 (1 + \rho + \rho^2 + \dots) = 1$$

$$\Rightarrow P_0 \frac{1}{1-\rho} = 1$$

$$\Rightarrow \boxed{P_0 = 1 - \rho}$$

$\rho = 1 - P_0$ is the prob that queue is not empty.

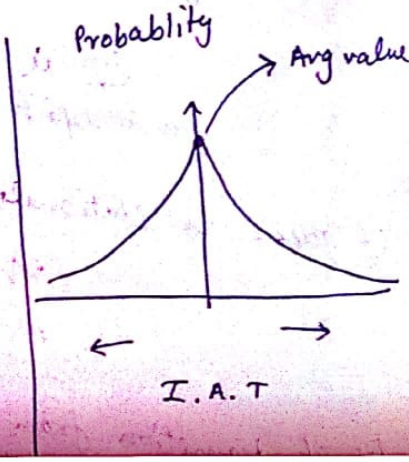
(As P_0 is prob that queue is ~~not~~ empty)

↓

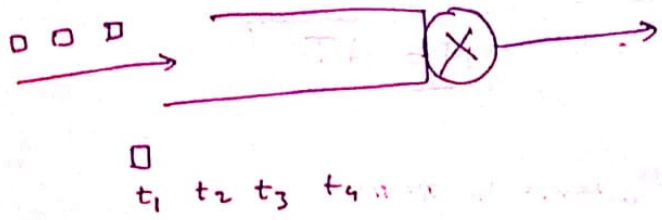
load

$$\therefore N = \sum_{K=0}^{\alpha} K P_0 \rho^K$$

$$= \sum_{K=0}^{\alpha} K \rho^K (1-\rho)$$



11/10/17



t_1 t_2 t_3 t_4

Interarrival times → $(t_2 - t_1)$, $(t_3 - t_2)$, $(t_4 - t_3)$

IT varies randomly.

If the interarrival time distribution is exponential, then the arrival process is poisson's process.

$$P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}$$

Avg no of customer in queue = State * prob. of finding that state.

$$\therefore N = \sum_{K=0}^{\infty} K P_0 \rho^K \qquad (K P_K)$$

$$= P_0 \sum_{K=0}^{\infty} K \rho^K$$

$$= P_0 \left(\rho + 2\rho^2 + 3\rho^3 + \dots \right)$$

$$= P_0 \frac{\rho}{(1-\rho)^2}$$

$$\therefore N = (1-\rho) \frac{\rho}{(1-\rho)^2}$$

$$= \frac{\rho}{1-\rho}$$

$\rho = \lambda/\mu$, $\therefore N = \dfrac{\frac{\lambda}{\mu}}{1 - \frac{\lambda}{\mu}} = \dfrac{\lambda}{\mu - \lambda}$

If $\mu > \lambda$ then also we have a queue due to diff interarrival time. If IAT is fixed in distribution, then queue will be empty, ie distribution is deterministic will cause $N = 0$

Expected delay in the queue → We will take help of little's formula, i.e $N = \lambda T$

where,

N = avg no of customers in queue

T = Avg ~~waiting~~ time of a customer in the queue (Dept - Arriv

λ = Avg arrival rate

~~Then little's form~~ So, $T = \frac{1}{\mu - \lambda}$ [Equate $N_1 \cong N$]

This T includes both waiting time and service time of the customer (T_w & T_s)

$T_s = \frac{1}{\mu}$ (service time) [μ customer per sec ∴ 1 customer → $1/\mu$]

Avg wt time for a customer in the queue →

$T_{w1} = \frac{1}{\mu - \lambda} - \frac{1}{\mu}$ [Time in queue - Service time]

Ex1 A bank teller can serve 12 customers per hr. Customers arrive at the teller at avg rate of 8 customers per hr. If arrival and service process are Poisson's, find

1) Expected no of customers in queue

2) Total time spent by a customer in queue

3) wt time of customers.

$\mu = 12$

$\lambda = 8$

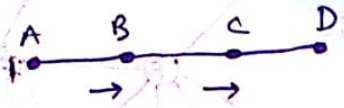
∴ 1) $N = \frac{8}{4} = 2$

ii) $T = \frac{1}{4} = 0.25$ hr

iii) $T_w = \frac{1}{4} - \frac{1}{12}$

$= \frac{3-1}{12} = \frac{1}{6}$ hr

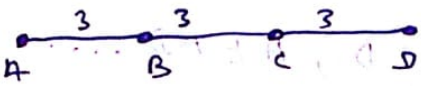
Now,



	A	B	C	D
A	X	10	7	3
B	10	X	8	5
C	7	8	X	7
D	3	5	7	X

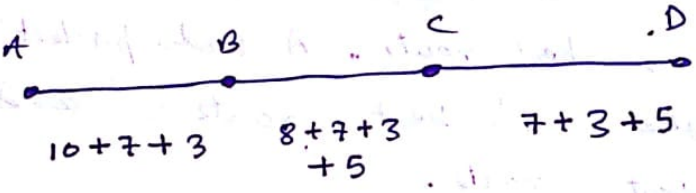
Traffic matrix (No of packets sent)

Say for A → D



3 packet will be there in each link

We find traffic in each link →



λ_{AB} = Traffic at link AB = 20

$\lambda_{BC} = 23$

$\lambda_{CD} = 15$

Say $\mu_{AB} = 25$ (capacity)

$\mu_{BC} = 25$

$\mu_{CD} = 20$

With this data we can find the delay in each link, i.e

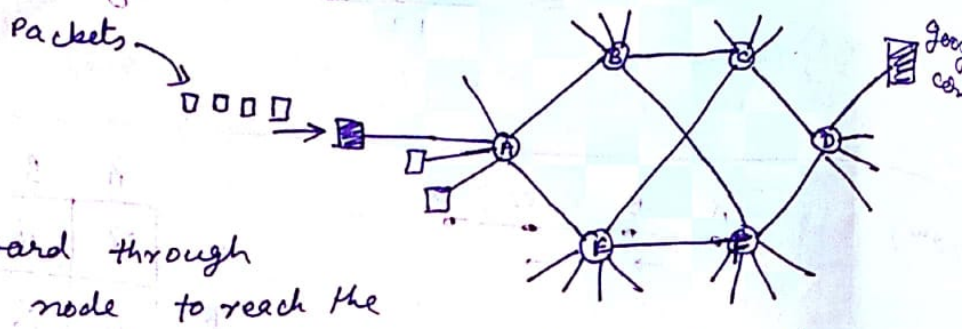
T_{AB}, T_{BC}, T_{CD}

Thus, $T_{AD} = T_{AB} + T_{BC} + T_{CD}$

Each node has to run 2 processes -

i) Packet forwarding process

ii) Routing Process



A has to forward through intermediate node to reach the dest D. Thus A must know the best route.

Routes → ABCD, ABFD, AECD, AEFD .....

A uses dijkstra to find shortest path where the weights of links are the random delay

Table where routes are written is called routing table

Say A finds ABCD as best route. A sends packet to B. B doesn't know what A's best route is and thus it finds its own best route.

If we let A find the best route & follow it throughout, it is source routing. It is generally not a good thing except in wireless adhok routing.

Thus every node basically ~~does~~ finds the best route and thus sends it to the next node in the route. This type of routing is per hop routing. Here each intermediate node finds its own route.

This is Packet forwarding. In above routing, packet arrives at node, destination is checked and then the routing table is looked up to find the next best node and thus packet is forwarded.

Routing process changes the R.T according to the delays in the channel

16/10/17

1) Routing (According to adaptibility)

- Static Routing
- Dynamic Routing

Criteria for routing → 1) Min hop routing
2) Min delay routing (Better)

We need to set the weights of the links with average delay and find out the shortest path using Dijstra Algorithm

In real time, delays are changing. Thus accordingly the shortest path change with time. This is dynamic routing.

2) Where the decision should be made →

Routing

- Source routing
  - Path decided by dijkstra and followed throughout [Quasi-static]
- hop by hop routing (per hop)
  - Each node does dijkstra and thus gets the path to the next node required. Thus we get more better result

3) Another way —

Routing

- Proactive
  - used for large wired networks
  - A node has route to all nodes through routing table. Thus by look up, packet is forwarded quickly
- Reactive (No routing table)
  - Good for Adhoc network (less infrastructure)
  - route is learnt and packet is sent. To find route time is required thus its slow. Route is determined as the packet arrives.

Thus the best one is ~~Dynamic Hop by~~ in each type is dynamic, hop by hop, proactive routings.

Routing information, update interval ??

Two classes of routing algorithm:

① Distance vector routing

② Link state routing

DVR → Every node ~~knows the~~ measures its min delay (or no of hops) from every other node and share this information with the neighbours only

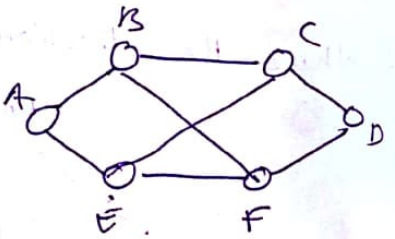
Say nodes are A B C D E F

A ⟶ (.0, 5, 13, 18, 6, a) (say)

0 1 2 3 1 2 → hops

B ⟶ (-, 0, -, -, -, -) ...

A node ~~has~~ sends its vector to its neighbours and according the neighbours will update its vector according to shortest path.



Then the neighbours forwards its vector to its neighbours and so on.

This process is done at regular interval of time.

The relation is of type A knows C through B etc. This takes large time and even accuracy is less. [large convergence time]

⭐ Asp info of all is sent to the neighbours.

An alternative is → We sent the info of the neighbours only to all the nodes. Thus we send few bits of info to all the nodes. This reduces convergence time. [See appropriate ans]

⊛ Previously we may have a problem → (count to infinity prob)

Say, C gets to know from B that 2 hops are required to reach A. When this info reaches B (being C's neighbour), B may assume C takes 2 hops excluding B and thus B may store its path for its neighbours as 2+1 hops.

LSR → ① each node creates a link state packet of following format

Seq no → 1	time stamp → T
Node	Delay
A	10
C	15
F	5

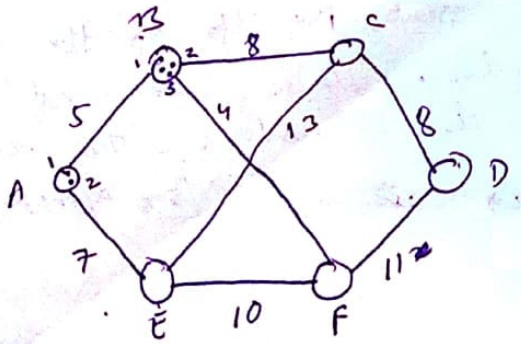
Say for node E ⇒

E sends an Eco packet ↓

A packet that doesn't wait in buffer as it has high priority

The packet is broadcasted to E's neighbours and they do it to their neighbours as they receive it. Info of sender is not there. The packet expires if a max count for hops is achieved or the time span has expired (time stamp is to old)

18/10/17

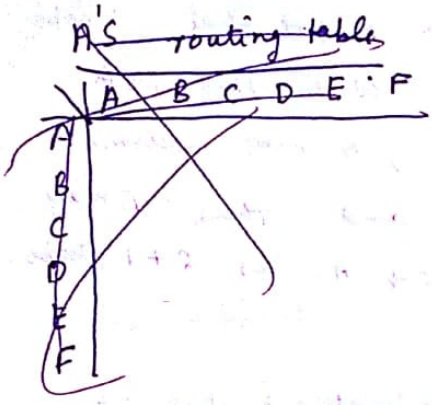


D's LS advertisement

C	8
F	12

A's routing table

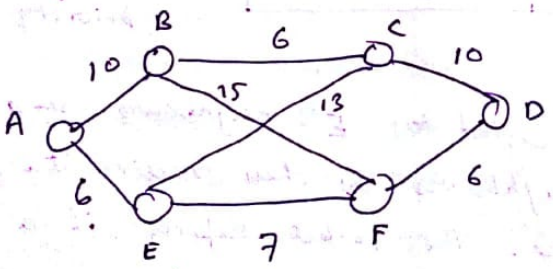
	Next hop	delay	Interface
A	NA	NA	NA
B	B	5	1
C	B	13	1
D	B	20	1
E	E	7	2
F	B	9	1



No of interfaces is actually the no of ports in the routers

R.T is created using LSP

After some time, delay may change.



then the routing table values will change depending on link state info from other nodes

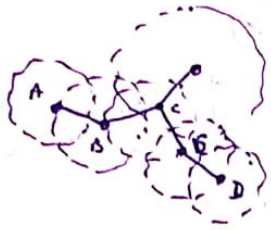
If a link is broken, then the delay is infinity in the routing table.

Routing algo → max utilization of global bandwidth

- All optical switching } check. ??
- WDM routers
- AODV ??
- MANET ??

AODV

- We send a RREQ broadcast packet with senders nodes address.
- The receiver appends its address and as it knows the route of receiving the packet, it sends back the packet through same route. Thus sender gets to know the receiving nodes path.
- sends packets within sending radius only



23/10/17

- Routing process creates routing table
- Forwarding process forwards the packet to the destination by a series of intermediate hops according to the routing table (every hop is the smallest path)
- for efficient packet forwarding we use heiarchical addressing.

IP → ~~network layer protocol~~

① Packets may reach out of order to the destination
② IP layer doesnot deploy error detection or correction.

① → Connectionless (delivery out of order)
② → unreliable

25/10/17

Establishing a TCP connection

- TCP is connection oriented and reliable.
  [Sequence Number and error correction headers will be essentially part of TCP headers]

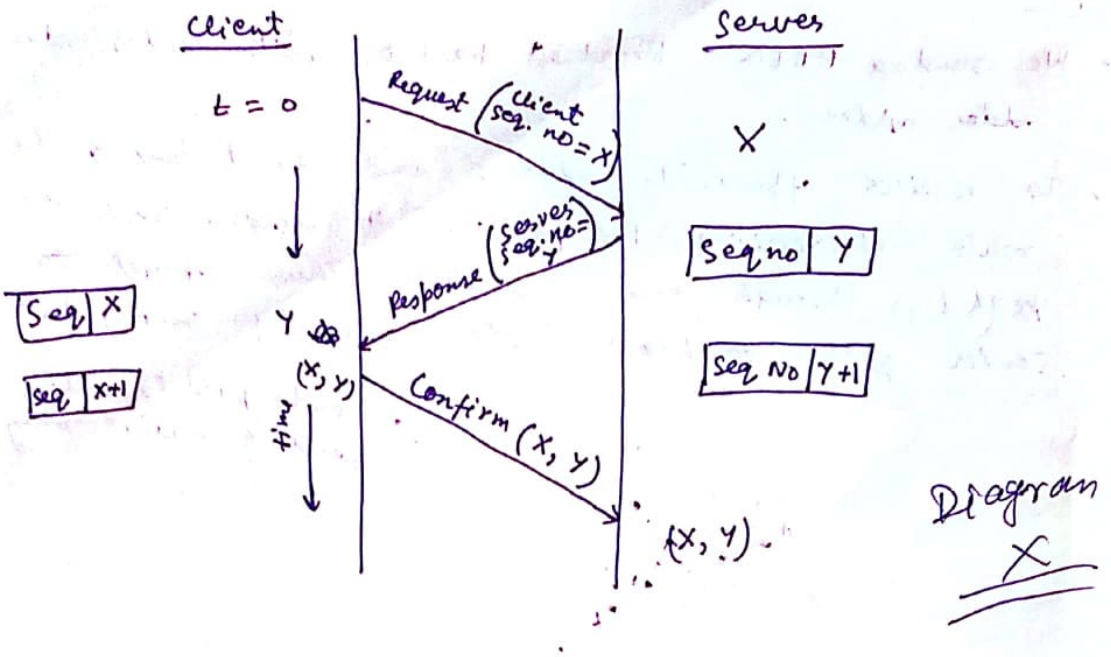
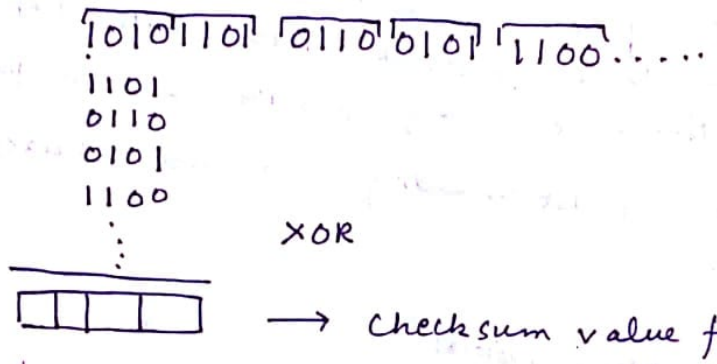
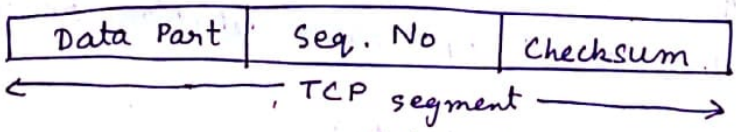
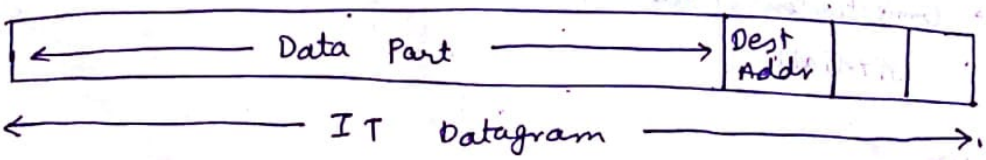


Diagram X

Rehability (Error handling)





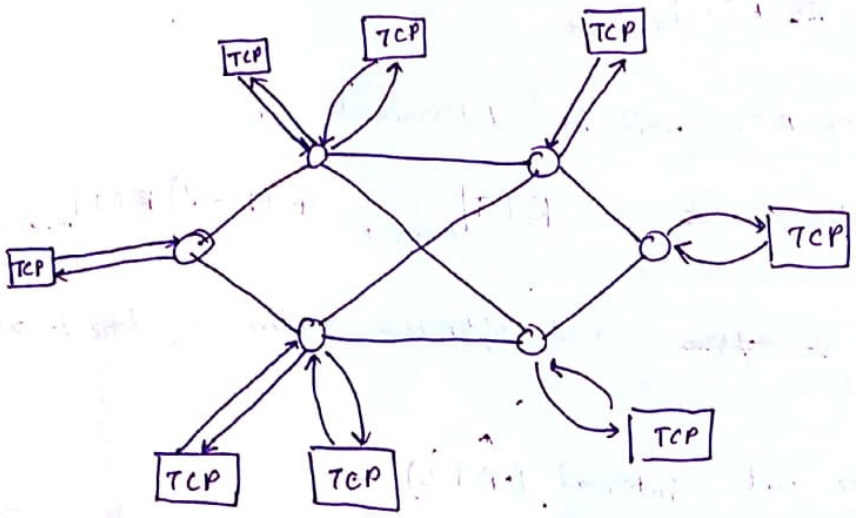


* unique
A TCP connection can be represented as vector

C_1 = (Source-addr, source port, destination-addr, dest-port)

	X	N	Y	M
say google	100.5.3.7	2135	200.5.1.2	80
say Amazon	same	2136	16.2.1.3	80
say again google	same	2137	200.5.1.2	80

If any of the 4 is varied, it represents a new connection.



If traffic generated by TCP could not be carried by the network, buffers fill up & extra packets are lost, then TCP will ask for retransmission.

Congestion ⇒ TCP ~~to~~ sends traffic at a higher rate than network layer can handle.

~~For bot~~ TCP should learn at what rate it should send. But this learn and improve leads to congestion. Thus it should be greedy but should not be too greedy.

Congestion ~~Control~~ Control in TCP

∴ t=0, S=x

at t=t_1, ACK for x is received

RTT = t_1

TCP keeps on measuring the RTT

Let the most recently measured value ~~RTT~~ be $RTT|_{measured}$

Also assume that the existing value of RTT for your current is $RTT|_{current}$

The new value of RTT will be determined as

$$RTT|_{new} = \alpha\, RTT|_{current} + (1-\alpha)\, RTT|_{measured}$$

where $0 < \alpha < 1$ and optimum value is close to 0.7-0.8

Retransmission time out interval (RTO) →

$$RTO = RTT + 4 * Var(RTT)$$

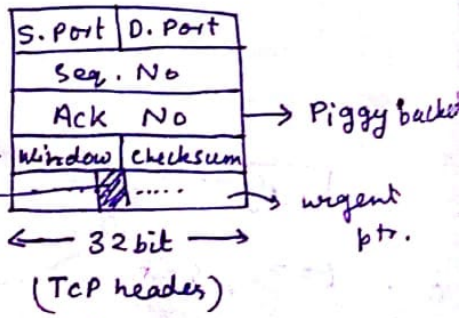
where Var (RTT) is the variation in RTT between previous & new value.

Now, for congestion control →

TCP uses three windows to ensure window based flow and congestion control

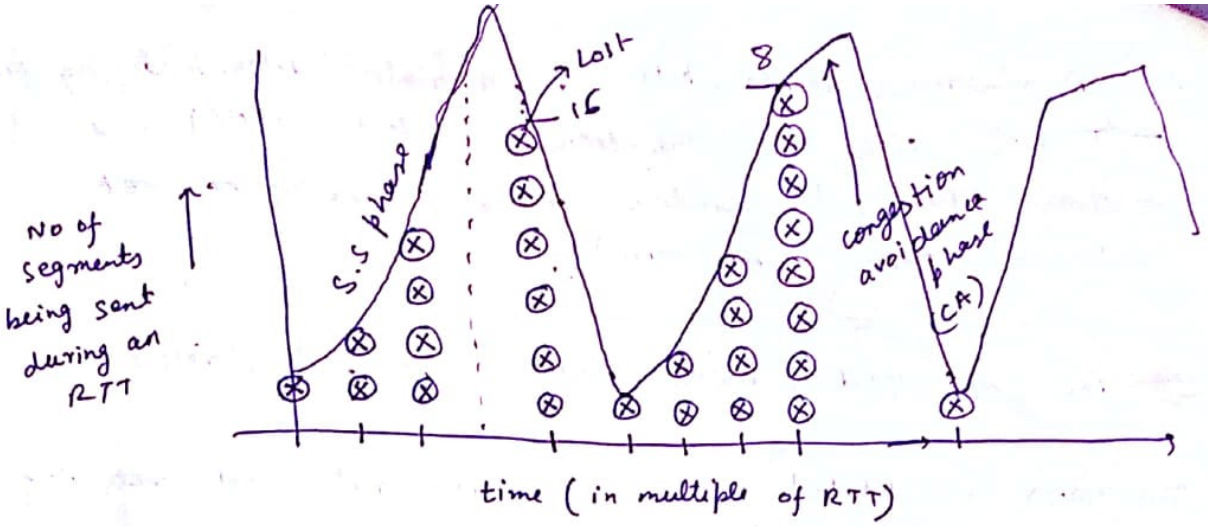
finite size of its buffer (Space in buffer)

6 bits were kept reserved so as for further modification They remained unused for 30 years.



Thus TCP cannot send data more than buffer size

i) Advertised Receiving Window → WND
ii) Congestion Window → CWND
iii) Threshold window → THWND



Concides window size 1 KB & RTT as 10msec. ∴ 1 segment per RTT

In next RTT we send 2 segments.

If we follow that for each ack. received back, we increase no of segments by 2 times.

When a packet is lost, we know that the limit ~~is~~ has exceeded (~~Here~~ Intense congestion). In such case window is reset and comes back to 1.

Say packet was lost at size 16. Thus we know that $16/2 = 8$ is optimum. [CWND = 8|16, THWND = CWND/2]

Now after reset when we reach 8, each time we keep of increasing no of segments ~~by 1~~ ~~a 16 loss value~~ by say 1. Then next time we suffer ~~same~~ loss, we come back to 1 but we get a better view of the limit ∞ after which error occurs. This process continues till limit is ~~achieved~~ obtained.

Nowadays, the TCP is not reset in case of loss. ~~It~~ ∞ The rate is reduced to half the value at which error occurred, instead of setting it to 1.

A TCP always sends data at a rate determined by the ~~rate of~~ size of congestion window (CWND) and receives advertised window using sending ~~rate~~.

rate = min (CWND, WND).

for IOT kind of devices, WND is ∞, thus rate = CWND

The CWND is updated using following rules after receiving each acknowledgement

CWND = 1 initially

i) In SS-phase (Slow start): CWND ← CWND + 1 ~~(window)~~

ii) In CA-phase : CWND ← CWND + $\frac{1}{CWND}$

In (i) → window is growing exponentially

(ii) → window grows linearly.

Amount of traffic controlled by the network will not exceed CWND and thus controlling the rate is adaptive. This is done so as to prevent loss of packets & finally prevent blasting of servers.

30/10/17

i) How to establish connection

Diagram X (Previously)

(1)

CWND = 1

WND = 2

THWND = 2

(Initially)

(2)

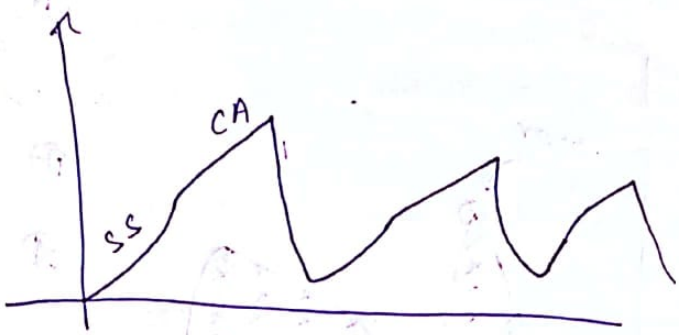
CWND

WND

THWND

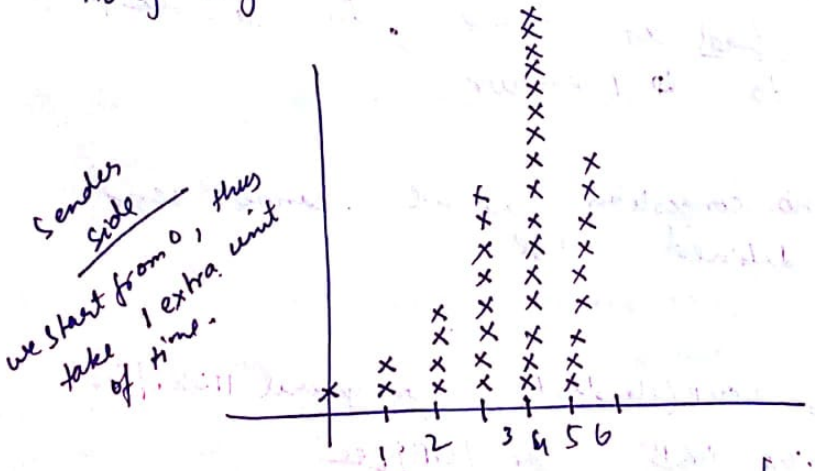
The one who sends uses their respective CWND

Initially THWND = ∝. After error occurs, THWND is updated as CWND/2.



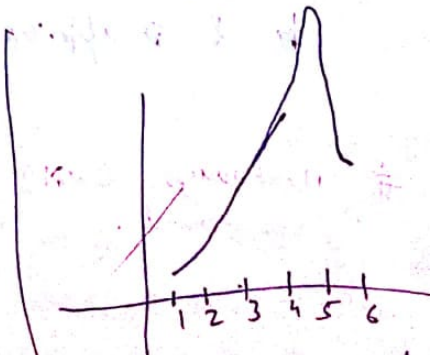
Ex Let a file of size 60KB is to be transmitted using a TCP connection with RTT = 10 msec. How long will it take by the receivers to receive the file completely if no segment suffers any error. Say the maximum size of the link layers frame is 1.5 KB.

No of segment to be transmitted = $\frac{60}{1.5}$ = 40



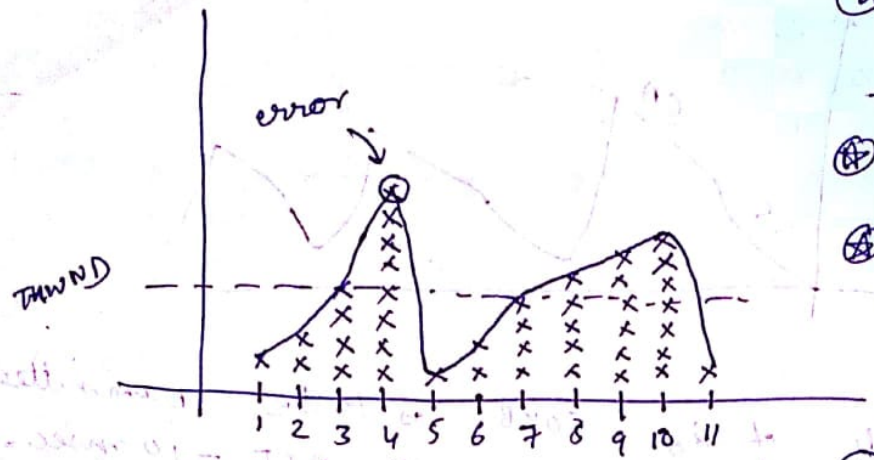
Thus finally receiver will receive at 6 and acknowledgement is sent

∴ time = 6 × RTT
= 6 × 10 = 60ms



It may start at 1 as well.

Ex In prev. ques, An error occurs at 5th Segment and no error afterwords. Then find the time to receive.



⊛ When error occurs CWND is set to 1

⊛ Previous size was 8

⊛ Thus THWND is now 4.

⊛ Thus, we send exponentially Till THWND. After that we send linearly

∴ Time = 11 × RTT
= 110 msec. (Ans)

Here we see that a bad TCP connection is really bad as time gets almost doubled due to 1 error.

TCP assures no congestion as we cannot send more than defined size.

frame size is 1.5KB (standard) for general 110KB/sec & approx 8KB for 1GB/sec

Nowadays CWND is set to half and then THWND is followed.

8/11/1

Tamper Detection

It Is implemented with hash function –

h → hash func

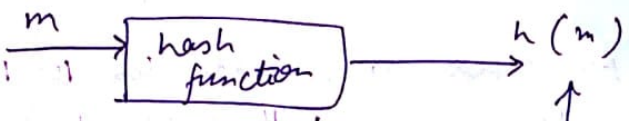
m → message

Sender → Sender →

- The h(m) is the hash or digest of the message is calculated.
- send both the message m and hash of the message h(m) to the receiver.

Receiver →

- It receives both m_R and $h(m)_R$
- Recomputes $h(m)_c$ from received m_R using the same hash function h.



MD & SH

digest or hash of msg which generally has a short fixed size.

- If $h(m)_R \neq h(m)_c$, then the message has been tempered by a third part.

But the hash function is many to 1 as the message length has no limit but digest length is fixed.

Thus more than 1 message can create the same digest.

→ Then how do we detect tamper.

Non repudiation

Putting a signature on the message $A \xrightarrow[K]{} B$

$E_K(m)$ to B can be accepted as a signed message because except B only A ~~message~~ knows the value of K

$E_K(m)$ should be stored by B at a save place before decryption.

All above was done using symmetric key. But there is fundamental limitation or deficiency that we cant communicate with an unknown person as we have not yet established a secret key with him/her.

To solve the problem, we came to the concept of

Public key Cryptography

i) Everyone will have a public key which is available in the public domain

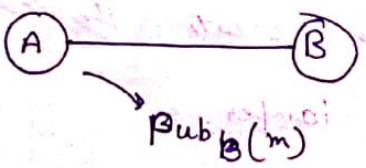
ii) Each individual will have the corresponding private key which is ~~secret to~~ known to him only.

iii) If A wants to send a message ~~to~~ to B, then A encrypts the msg with public key ~~of~~ of B, i.e A sends $Pub_B(m)$

iv) The private public key-pair satisfies the following condition :-

a) $Pri_B\{Pub_B(m)\} = m$

b) $Pub_B\{Pri_B(m)\} = m$



When B receives $Pub_B(m)$ from A, it decrypts the message using (a)

How to generate public private key pair ?

→ The most popular soln is RSA (Rivest, Shasmin, Addleman)

RSA Algorithm → (Used for establishing secure connection because too much work)

i) Take two big prime no p & q

compute $n = p \times q$

ii) Determine ~~$z = (p-1)(q-1)$~~ $z = \text{Lcm}\{(p-1), (q-1)\}$

iii) Now find a member 'e' relatively prime to z. (e is not a factor of z). This e will be used as ~~encryption~~ public key.

iv) Now find a no d such that $e \cdot d = 1 \pmod z$

then d is the private key of the pair

We have generated e and d

Say $p = 11$, $q = 3$

$\therefore n = 33$, $z = 20$

Say $e = 7$ & $d = 3$

$\therefore e \cdot d = 21$

So, $21 \% 20 = 1$

Thus, $e = 7$ & $d = 3$ is a valid key pair

How do we encrypt and decrypt??

$l = \underbrace{010}_{m_1} \underbrace{101}_{m_2} \underbrace{101}_{m_3} 1 \ldots$

Encryption = $(m_i)^e \bmod z$

(Check)

Decryption = $(m_i^e)^d \bmod z = m_1^{ed} \bmod z$

$= m$

Non - Repudiation, sending → secret msg encrypt by my private key then with his other public key. (Double encryption so that secure proof is there)