UNIT - II

Data Link Layer

* Error Detection & Correction:

DDL uses some error control mechanism to ensures that frames (data bit streams) are transmitted with certain level of accuracy. But to understand how error is controlled, it is essential to know what types of error may occur.

Types of Errors:

i) Single bit error: In a frame, there is only one bit, anywhere, which is corrupted.

ii) Multiple bits error: frame is received with more than one bits are corrupted.

iii) Burst error: frame contain more than 1 consecutive bits corrupted.

Error Detection Techniques:

1) Single Parity Check:

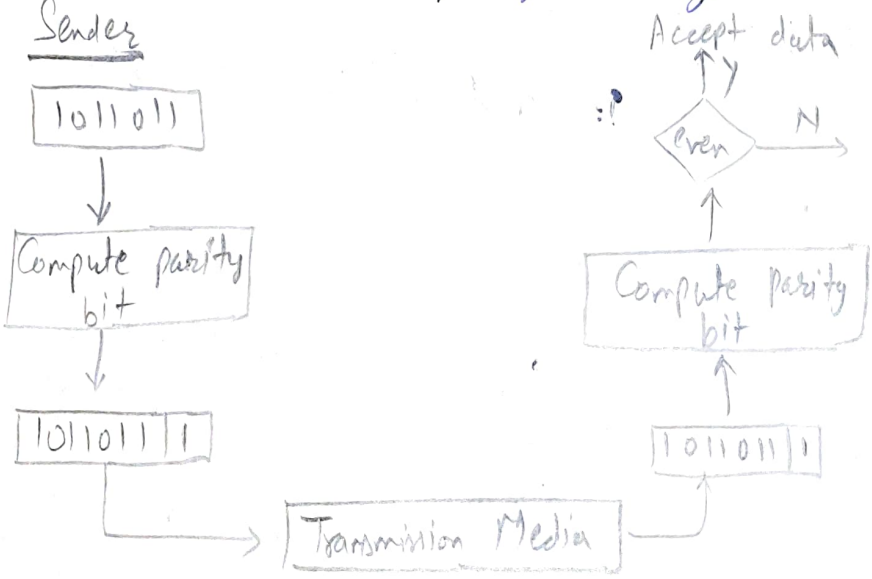
→ Simple & inexpensive to detect the errors.

→ A redundant bit is also known as parity bit which is appended at the end of the data unit so that the no. of 1s become even.

→ If the no. of 1s bits is odd, then parity bit 1 is appended and if the no. of 1s bits is even, then parity bit 0 is appended at the end of the data unit.

→ At the receiving end, the parity bit is calculated from the received data bits and compared with the received parity bit.

→ This technique generates the total no. of 1s even, so it is known as even-parity checking.



Drawbacks:

→ detect only single bit errors

→ if two bits are interchanged, then it cannot detect the errors

ii) Two-Dimensional Parity Checks

→ data is in the form of tables so we can improve the performance.

→ parity check bits are computed for each row, which is equivalent to the single-parity check.

→ A block of bits is divided into rows and redundant row of bits is added to the whole block.

→ At the receiving end, the parity bits are compared with the parity bits computed from the received data.

Original Data: 11001110 10111010 01110010 01010010

1	1	0	0	1	1	1	0	1
1	0	1	1	1	0	1	0	1
0	1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	0	1	0	1	0	0	1
Column parity								

even → 0
odd → 1

Row parity

01010100 1

Drawback:

Sent data: 11001110 1 10111010 1 01110010 0 01010010 1

→ If two bits in one data unit are corrupted and two bits exactly the same position in another data unit are also corrupted, then 2D parity checker can not detect the errors.

→ Can not be detect 4 bit errors.

iii) Checksum:

→ Based on Concept of Redundancy.

Checksum

↓ Checksum generator — Checksum checker

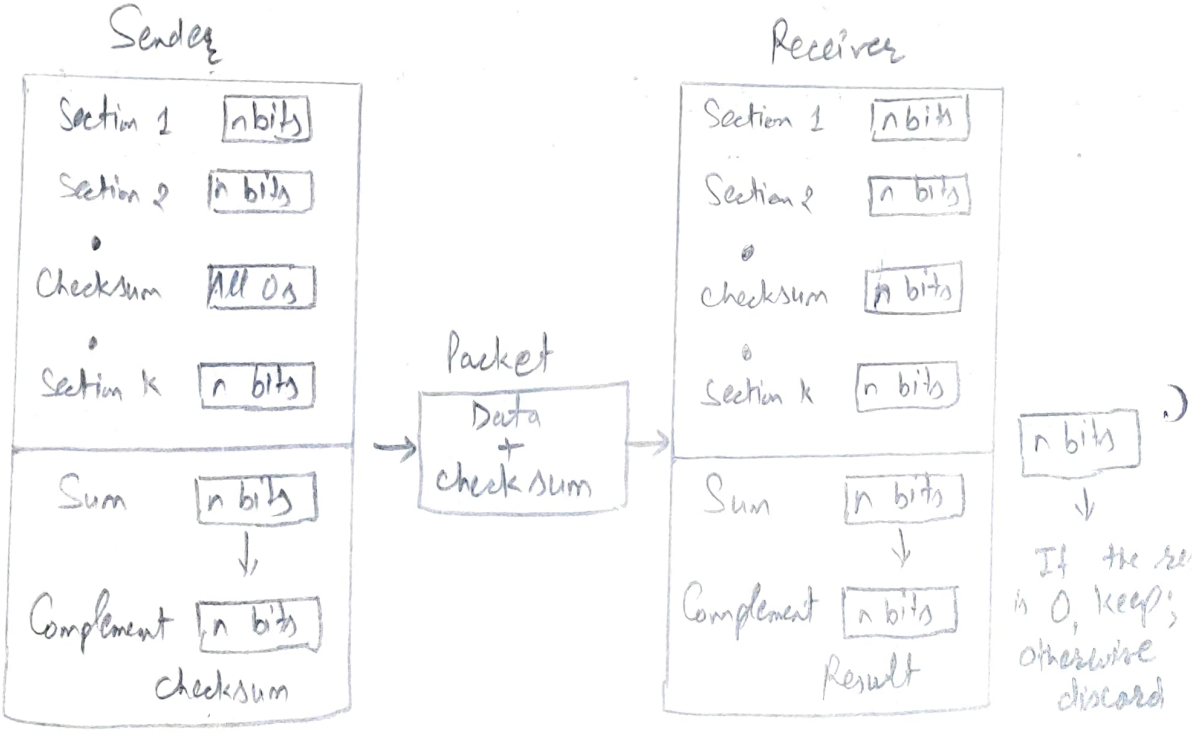
Checksum generator:

→ A checksum is generated at the sending side.

→ Checksum generator subdivides the data into equal segments of n bits each and all these segments are added together by using one's compliment arithmetic.

→ The sum is complemented and appended to the original data, known as checksum field.

→ The extended data is transmitted across the n/w.



~~The sender follows~~

Checksum Checker:

→ A checksum is varified at the receiving side.

→ The receiver subdivides the incoming data into equal segments of n bits each, and all these segments are added together, then this sum is complemented.

→ If the complement of sum is zero, then the data is accepted otherwise data is rejected.

iv) Cyclic Redundancy Check (CRC):

→ a string of n 0s is appended to the data unit, and this n no. is less than the no. of bits in a predetermined no., known as division which is n+1 bits.

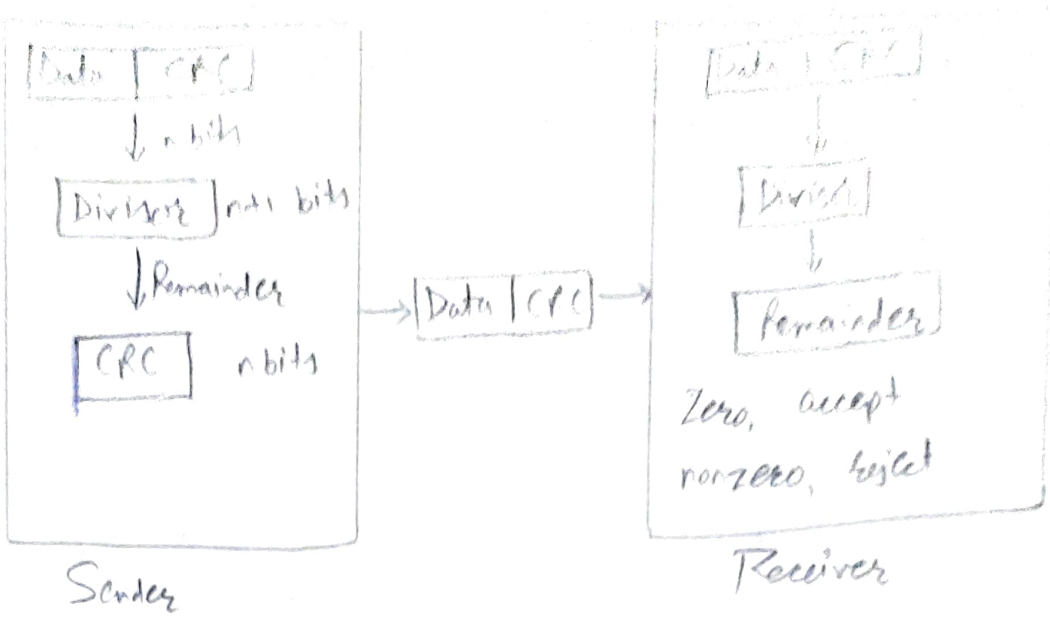
→ The newly extended data is divided by a divisor using a process is known as binary division. The remainder generated from this division is known as CRC Remainder.

→ The CRC Remainder replaces the appended 0s at the end of the original data. This newly generated unit is sent to the receiver.

→ The receiver receives the data followed by the CRC remainder.

→ The Receiver will treat the whole unit as a single unit and it is divided by the same divisor that was used to find the CRC Remainder.

- If the resultant of this division is zero which means that it has no errors, and data is accepted.
- If the resultant of this division is not zero which means it has errors and data is discarded.



Eg: Suppose the original data is 11100 and divisor is 1001.

CRC Generator:

→ It uses modulo-2 division.

→ Three zeros are appended at the end of the data as the length of the divisor is 4 and we know that length of the string 0s to be appended is always 1 less than the length of the divisor.

→ Now, string become 11100000, this divided by the 1001.

→ The remainder generated from the binary division is known as CRC remainder. i.e. 111.

→ CRC remainder replaces the appended string of 0s at the end of data unit, and the final string would be 11100111, which is sent accross the n/w.

CRC Checker:

→ similar to CRC generator.

→ string 11100111 received, perform modulo-2 division.

→ divided by 1001 divisor.

→ if remainder is zero then data is accepted.

```
          1 1 1 1 1
        ---------------
1 0 0 1 ) 1 1 1 0 0 0 0 0
          1 0 0 1
          ---------
            1 1 1 0
            1 0 0 1
            -------
              1 1 1 0
              1 0 0 1
              -------
                1 1 1 0
                1 0 0 1
                -------
                  1 1 1 0
                  1 0 0 1
                  -------
                    1 1 1   ---> CRC Remainder
```

XOR operation

0 0	0
0 1	1
1 0	1
1 1	0

```
          1 1 1 1 1
        ---------------
1 0 0 1 ) 1 1 1 0 0 1 1 1
          1 0 0 1
          ---------
            1 1 1 0
            1 0 0 1
            -------
              1 1 1 1
              1 0 0 1
              -------
                1 1 0 1
                1 0 0 1
                -------
                  1 0 0 1
                  1 0 0 1
                  -------
                  0 0 0 0   ---> Remainder is 0, data accepted.
```

Error Correction Techniques

→ used to detect & correct the errors when data is transmitted from S to R.

i) Backward error correction: Once the error is discovered, the receiver requests the sender to retransmit the data.

ii) Forward error correction: R uses error-correcting code which automatically corrects the errors.

→ A single additional bit can detect the error, but can not correct it.

→ For correcting the errors, we have to know exact positn of error.

→ If we want to calculate single bit error, the error correctn code will determine which one of 7 bits is in error. To achieve this, we have to add some additional redundant bits.

→ Suppose r is the no. of redundant bit & d is the total no. of data units. The ~~see~~ no. of redundant bits are calculated by

$$2^r >= d+r+1$$

eg. if value of d is 4, then possible smallest value that satisfy the above relation would be 3.

To determine the position of bit which is in error, a tech. developed by R. W Hamming Code, which can be applied to any length of data unit & uses the relationship b/w data units & redundant units.

Hamming Code:

Parity bits: The bit which is appended to ori. data of binary bits so that total no. of 1s is even or odd.

Even parity: if the total no. of 1s is even, then the value of parity bit is 0. If the total no. of 1s occurrence is odd, then the value of parity bit is 1.

Odd parity: if the total no. of 1s is Even, then value of parity bit is 1. If total no. of 1s is odd, then value of parity bit is 0.

Algorithm:

- An info. of 'd' bits are added to the redundant bit 'r' to form d+r.
- Location of each (d+r) digit is assigned a decimal value.
- 'r' bits are placed in the position $1, 2, \ldots 2^{k-1}$
- At the receiving end, parity bits are recalculated. The decimal value of parity bits determines the position of an error.

Relationship b/w error position & binary no.

Error position	Binary no.
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Eg: Suppose ori' data 1010 to be sent.

Total no. of data bits $d = 4$

no. of redundant bits r : $2^r \geq d + r + 1$

$2^r \geq 4 + r + 1$

Therefore, the value of r is 3 that satisfy the above relation.

Total no. of bits = $d + r = 4 + 3 = 7$

Determining the position of redundant bits:

The no. of redundant bit is 3. the three bits are represented by r_1, r_2, r_4. The position of the redundant bits is calculated with corresponds to the raised powers of 2. therfore, their corresponding positions are $1, 2^1, 2^2$.

the position of $r_1 = 1$

$r_2 = 2$

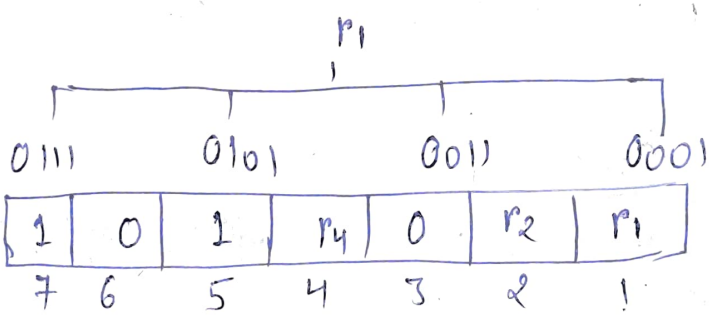
$r_4 = 3$

Representation of data on the addition of parity bits:

7	6	5	4	3	2	1
1	0	1	r4	0	r2	r1

Determining the parity bits:

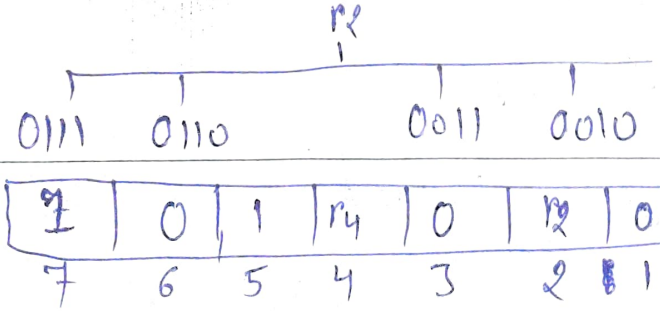
r1 bit: binary representation includes 1 in the first position.



~~7, 5, 3, 1~~
1 1 0 r1 even
7, 5, 3, 1

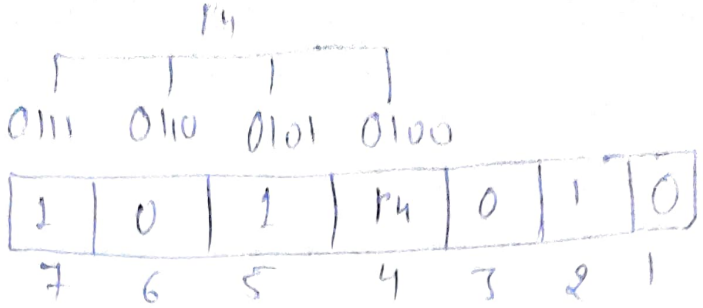
1 in the first position are 1, 3, 5, 7. perform even parity check at these bit positions. Total no. of 1 at these bit positions corresponding to r1 is even, therefore the value of r1 bit is 0.

r2 bit: 1 in the second position. i.e. 2, 3, 6, 7



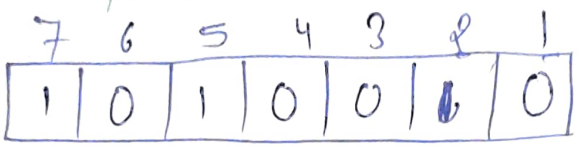
~~2, 3, 6~~
7, 6, 3, 2
1 0 0 r2 odd

1 in r2 is odd, value of r2 bit is 1.



total no. of 1 in r4 is even, therefore, the value of r4 bit is 0.

Data transferred:



Suppose 4th bit is changed from 0 to 1 at the receiving end, then parity ~~check~~ bits are recalculated.

* Detection vs Correction:

The correct of error is more ~~differe~~ difficult than the detection.

→ In Error detection, we see only any error has occured.

→ In Error correction, we need to know that exact no. of bits that are corrupted and their location in the msg. The member of the error and size of the msg are imp. factor.

* Error Correction Methods:

1) Automatic Request for Retransmission (ARQ)/Automatic Repeat ReQuest

2) Forward error correction (FEC)

1) ARQ:

→ It is a group of error-control protocols for transmission of data over noisy or unreliable Comm^n n/w.

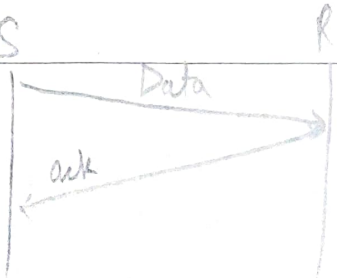
→ These protocols resides in DDL & TL of OSI model.

→ They provide automatic retransmission of frames that are corrupted or lost during transmission.

→ It is also called Positive Ack. with Retransmission (PAR).

Working principle:

The receiver sends an ack. msg back to the sender, if it receives a frame correctly. If the S does not receive the ack of a transmitted frame before a specified time i.e timeout occurs. The S understand that frame has been corrupted or lost during transit. So, S retransmits the frame. This process is repeated until the correct frame is transmitted.



Types of ARQ:

ARQ Protocols

- Stop & Wait ARQ
- Go-Back-N ARQ
- Selective Repeat ARQ

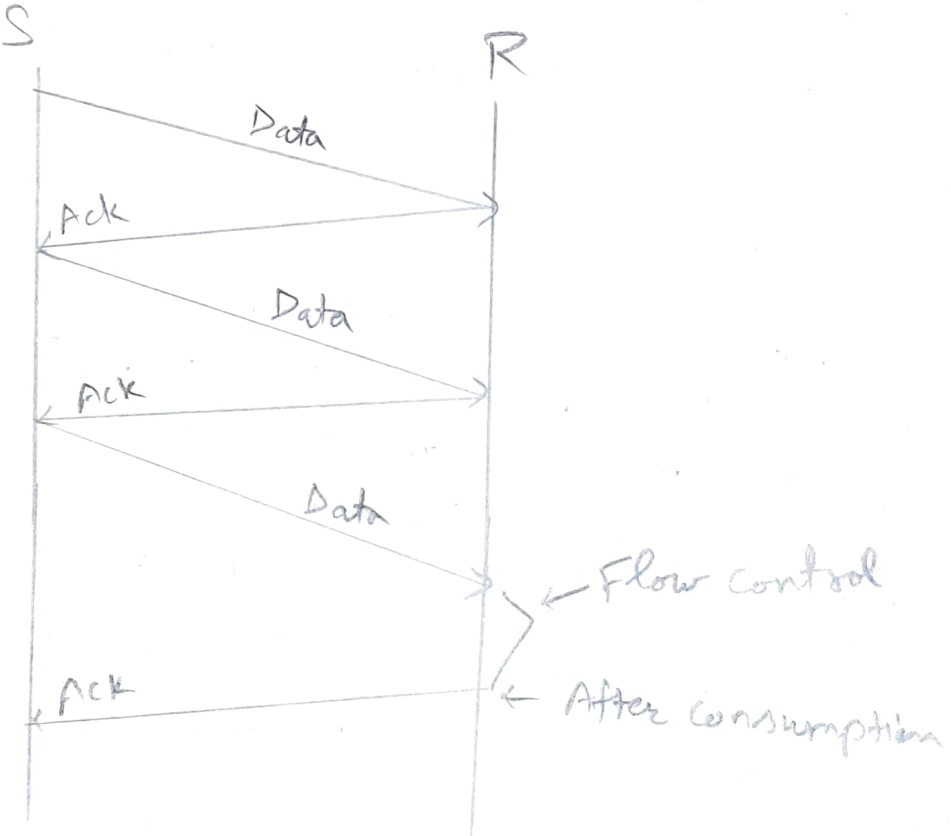
i) Stop and Wait ARQ :

Sender:

1) Send one packet (Data) at a time.
2) Send next pkt only after receiving ack for previous.

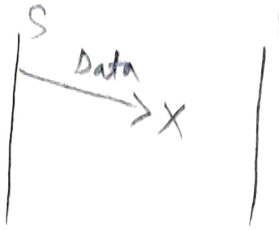
Receiver:

1) Send ack after receiving the pkt.
2) After consuming pkt ack. need to be sent (flow control)



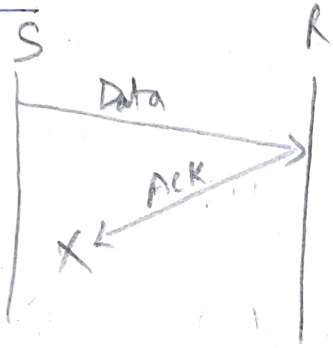
Problems:

a) Lost Data:



Sender waits for Ack
R waits for data
For infinite amount of time

b) Lost Ack:



S waits for Ack for infinite amount of time.

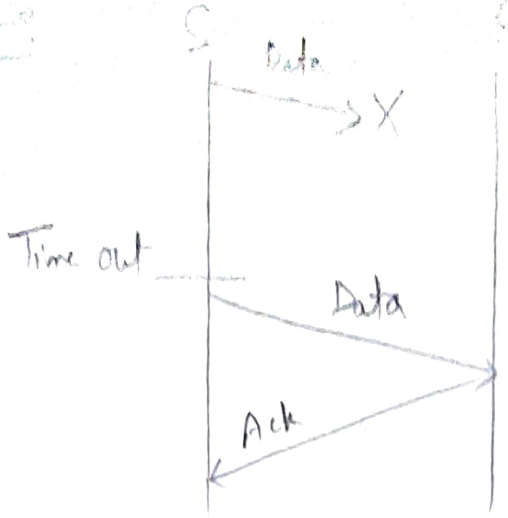
c) Delayed Ack / Data: after time out on sender side, a long delayed ack might be wrongly considered as ack of some other recent pkt.

Stop and wait + Time out + Seq. No (data) + Seq. no. (ACK)

↑ Lost data ↑ Lost Ack ↑ Delayed Ack

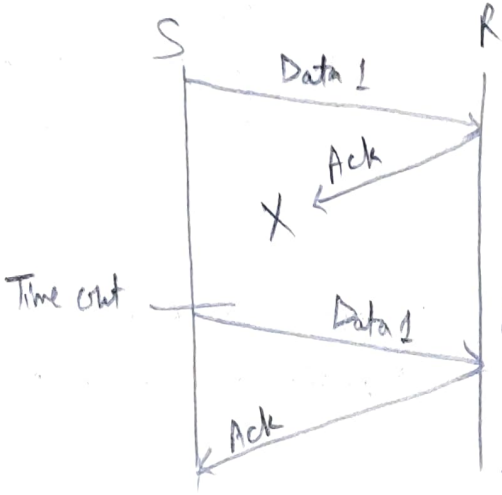
All 3 prob. are resolved by Stop and Wait (ARQ) that does both error control & flow control.

a) Time Out:



Retransmission of data after Time out resolves Lost Data Problems

b) Sequence No. (Data):

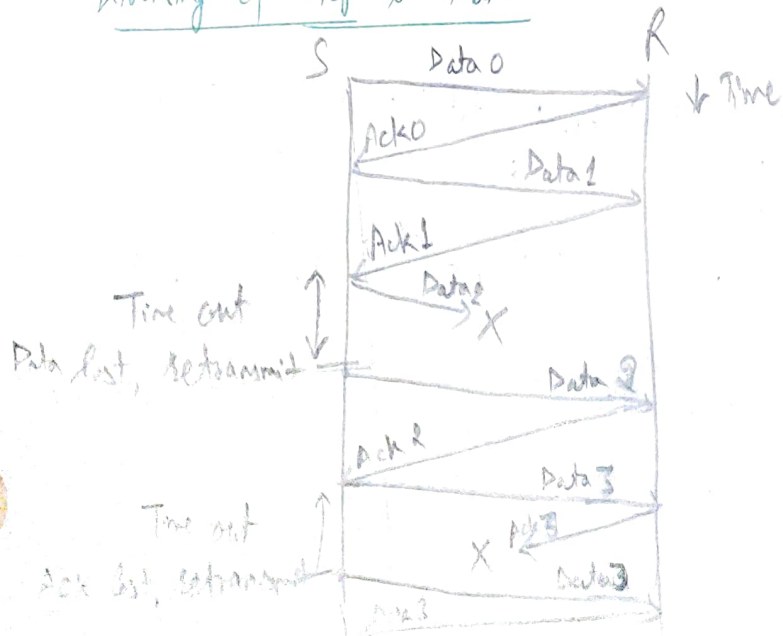


Data 1 Consumed and ack sent

Receiver sends ack instead of -ve ack bcoz of seq. no. attached to S pkt.

c) Delayed Ack: This is resolved by introducing seq. no. for ack. also.

Working of Stop & Wait:



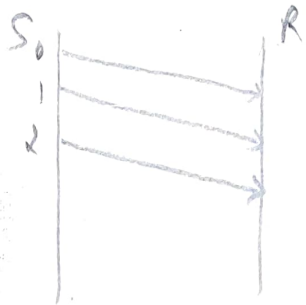
ii) Go-Back-N ARQ:

→ Sending multiple frames before receiving the ack for the first frame. It uses the concept of sliding window, So it is called sliding window protocol.

→ The frames are sequentially numbered and a finite no. of frames are sent.

→ If the ack of a frame is not received within the time period, all frames starting from that frame are retransmitted.

Eg: N is the sender's window size. Suppose it is GO-Back-3, means three frames can be sent at a time before expecting the ack from the receiver.



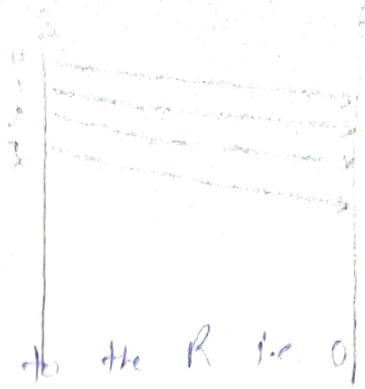
Working:

Suppose 11 frames to be sent, these frames are represented as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and these are seq. no. of frames. Seq. no. is decided by S's window size.

Let's consider the window size 4, which means four frames can be sent at a time before expecting the ack of the first frame.

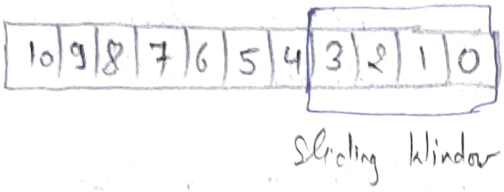
1) 

Window size: 4

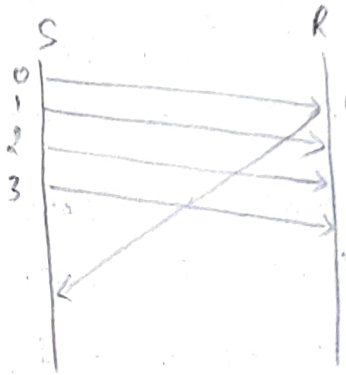


Sender send the first 4 frames to the R i.e 0, 1, 2, 3 and wait for ack of 0th frame.

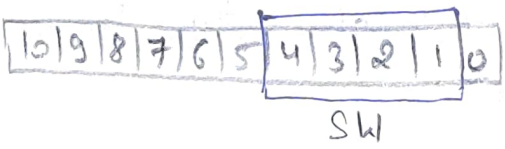
2) R sent the ack for frame 0 & S. has successfully received it.



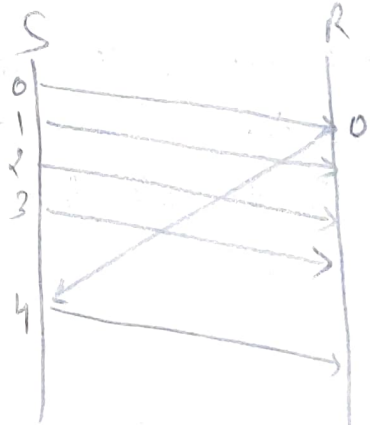
WS : 4



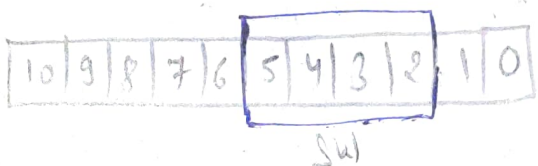
3) Sender will send next frame i.e 4 the window slides containing 4 frames i.e. 1, 2, 3, 4



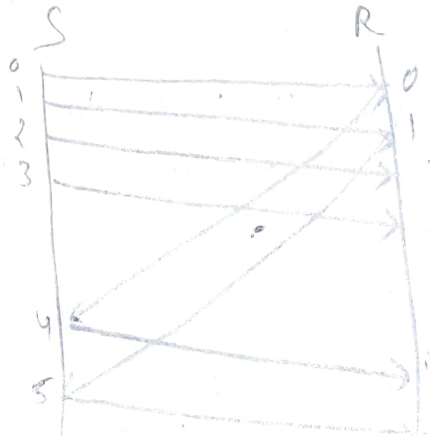
RS : 4



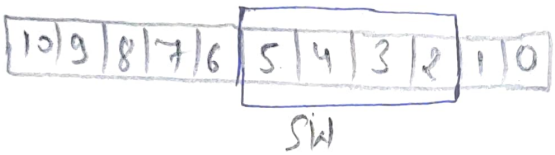
4) R will send the ack. for frame no. 1. After receiving ack, S will send next frame i.e. frame no. 5, and the window will slide having four frames (2,3,4,5).



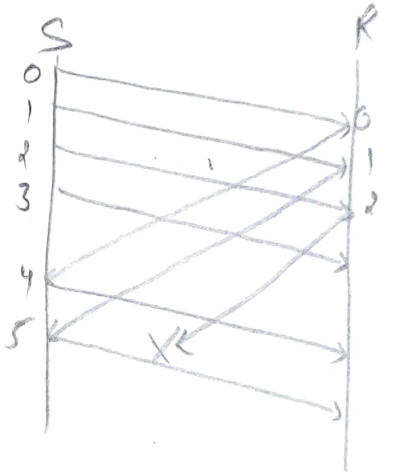
RS : 4

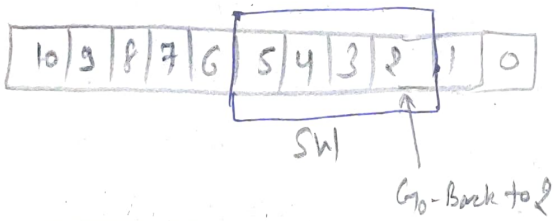


5) Now let's assume that R is ~~not~~ send the ack of frame 2 but this ack is lost. Instead of sending frame 6 the sender Go Back to 2, which is the first frame of current window, Retransmit all frames in the current window ie. 2, 3, 4, 5

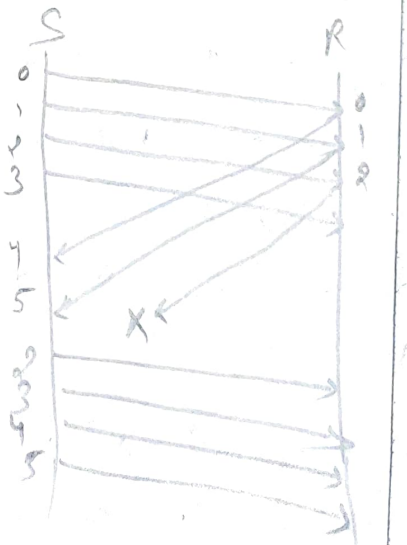


WS : 4





WS : 4



Imp. Points:

→ N determines the S's WS, the size of R's window is always 1.

→ Does not consider the corrupted frames & simply discard them.

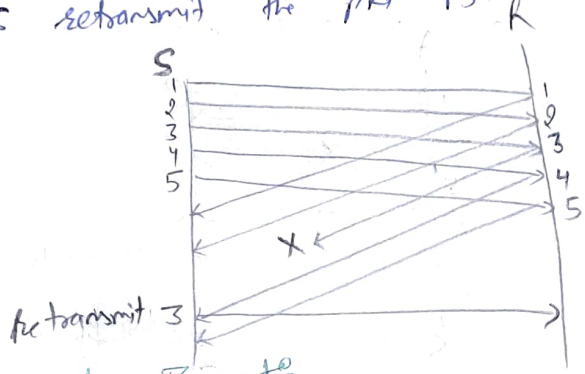
→ Does not accept the frames which are out of order & discard them.

→ If the S does not Receive ack, then it Retransmit the all current window frames.

iii) Selective Repeat ARQ

→ Provides for sending multiple frames before receiving the ack. for the first frame. However, here only incorrect or lost frames are retransmitted, while the good frames are received and buffered.

Eg: S sends the data pkt from P1 to P5. Now P1 & P2 received by R successfully but P3 get lost. So Unlike GO-Back-N ARQ, in selective Repeat. S send the pkt P4 & P5, which the R will receive and after that, S retransmit the pkt P3.



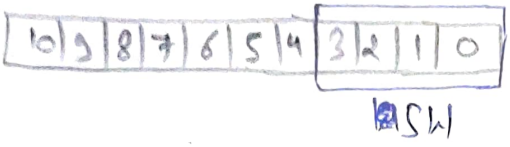
Retransmission Request:

- Implicit Retransmission Request: It's the R's duty to ack or give +ve feedback of received data pkt and -ve feedback for those which were lost or damaged during transmission and that too before the expiry of time out timer.
- Explicit Retransmission Request: the R can request retransmission of just one pkt.

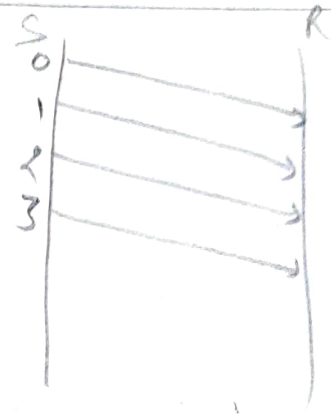
Note: The S's window must be equal to R's window and WS should be less than or equal to half of seq. no.

Working :

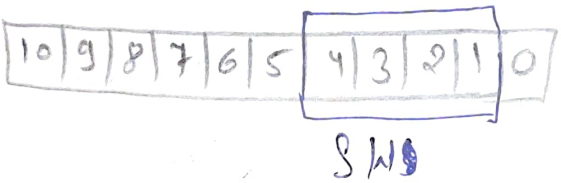
1)



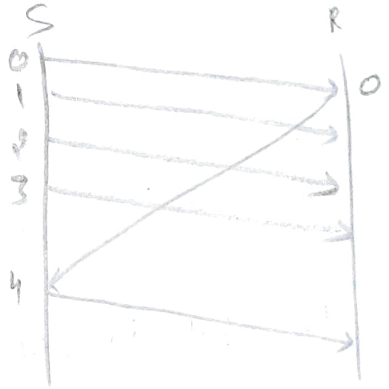
WS : 4



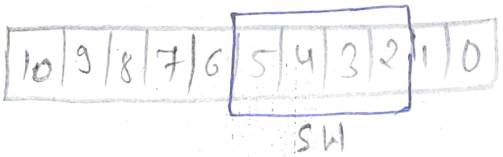
2)



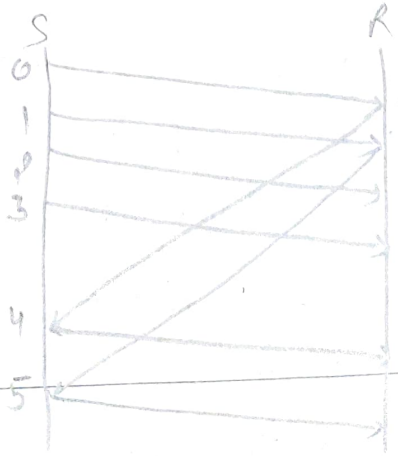
WS : 4



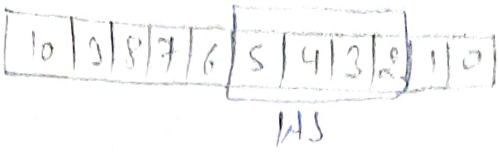
3)



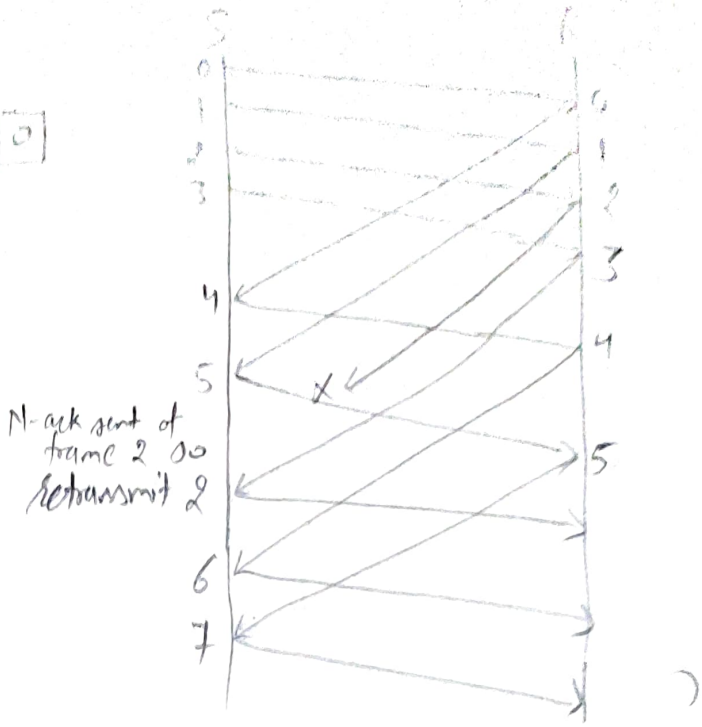
WS : 4



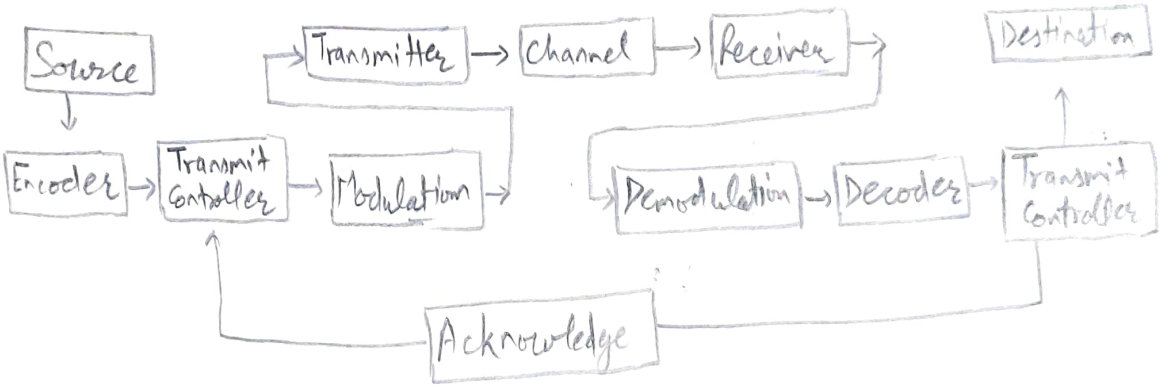
4) if frame no 2 is lost the R does not send the ack of frame 2.



WS : 4



Block diagram of ARQ:



2) Forward Error Correction (FEC):

→ It is an error correction technique to detect & correct a limited no of errors in transmitted data without the need for retransmission.

→ S sends a redundant error-correcting code along with the data frames.

→ The R performs necessary checks based upon the additional redundant bits.

If it finds that the data is free from errors, it executes

Error-correcting codes that generates the actual frame.

→ If it finds the error then removes the redundant bits before passing the msg. to upper layers.

Advantage:

→ FEC does not require handshaking b/w S & D. It can be used for broadcasting of data to many 'D' simultaneously from a single S.

→ FEC saves BW required for retransmission. So it is used in Real time System.

Disadvantage:

→ if there are too many errors, then frames need to be retransmitted.

Error Correction Codes for FEC:

1) Block Codes: The msg is divided into fixed-sized blocks of bits to which redundant bits are added for error correction.

2) Convolutional Codes: The msg. consist of data streams of arbitrary length and parity symbols are generated by the sliding app. of a Boolean fun^n to the data stream.

3) Hamming Codes: It is a block code is capable of detecting up to three simultaneous bit errors and correcting single bit errors.

4) Binary Convolution Code: an encoder processes an I/P seq. of bits of arbitrary length and generates a seq. of o/p bits.

5) Reed - Solomon Code: They are block codes that are capable of correcting burst errors in the received data block.

6) Low - Density Parity Check code: It is a block code specified by a parity check matrix containing a low density of 1s. They are suitable for large block sizes in very noisy channels.

* Error - Detecting Codes - Checksums:

→ When bits are transmitted over the comp. n/w, they are corrupted due to interference and n/w prob.

→ Error detection techniques are responsible for checking whether any error has occured or not in the frame.

→ For error detection, the sender needs to send some additional bits along with data bits.

→ The receiver performs necessary checks based upon the additional redundant bits.

→ If it finds that the data is free from errors, it removes the redundant bits before passing the msg to upper layer.

Techniques: Parity check, Checksum & CRC.

PODDRNIMA
COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

Checksum:

→ This is a block code method where a checksum is created based on the data values in the data blocks to be transmitted using some algo and appended to data.

→ When the R gets this data, a new checksum is calculated & compared with the existing checksum.

→ A non-match indicates an error.

Error detection by checksum: data is divided into fixed sized frames or segments.

- Sender's End: The sender adds the segments using 1's complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.

- Receiver's End: The R adds the incoming segment along with the checksum using 1's complement arithmetic to get the sum and then complements it.

If the result is Zero, the received frames are accepted, otherwise discarded.

Sender End

Frame 1: 11001100

Frame 2: + 10101010

Partial Sum: 101110110

+1

01110111

Frame 3: + 11110000

Partial Sum: 101100111

+1

01101000

Frame 4: + 11000011

Partial Sum: 100101011

+1

Sum: 00101100

Checksum 11010011

Receiver End

Frame 1: 11001100

Frame 2: + 10101010

Partial Sum: 101110110

+1

01110111

Frame 3: + 11110000

Partial Sum: 101100111

+1

01101000

Frame 4: + 11000011

Partial Sum: 100101011

+1

Sum: 00101100

Checksum: + 11010011

Sum: 11111111

Complement: 00000000

Hence frame accepted.

* Block Coding:

divide the msg into blocks, each of k bits, called data words. We add r redundant bits to each block to make the length $n = k + r$. The resulting n-bit blocks are called code words.

We have a set of data words, each of size k, and a set of codewords each of size n. With k bits, we can create a combination of 2^k data words and with n bits, create a combination of 2^n code words. Since $n > k$, the no. of possible codewords is greater than the no. of possible data words.

The block coding process is one-to-one, the same data word is always encoded as the same code word. This means that we have $2^n - 2^k$ code words that are not used. We call these code words invalid or illegal.

* Linear Block Coding:

The parity bits & msg bits have a linear combination, which means that the resultant code word is the linear combination of any two code words.

Let's consider some block of data, which contains k bits in each block. These bits are mapped with the blocks which has n bits in each block. Here $n > k$.

The transmitter adds redundant bits which are $n-k$ bits. The ratio k/n is the code rate. It is denoted by r and value of r is $r < 1$.

The n-k bits are called parity bits. Parity bits help in error detection and error correction and also in locating the data.

In data being transmitted, the left most bit of code word corresponds to the msg bit and right most bits of code word to the parity bit.

* Cyclic Codes:

The cyclic property of a code word is that any cyclic shift of a code word is also a code word. Cyclic codes follow this cyclic property.

For a linear code C, if every code word i.e. $C = C_1, C_2, \ldots C_n$ from C has a cyclic right shift of components, it becomes a code word. This shift (right) is equal to n-1 cyclic left shift. Hence, it is invariant under any shift. So, the linear code C, invariant under any shift called as cyclic code.

Cyclic codes are used for error correction. They are mainly used to correct double errors and burst errors.

Eg. If 1011000 is a code word and we cyclically left-shift, then 0110001 is also a code word.

In this case, bits in the first word a_0 to a_6 and bits in the second word b_0 to b_6, we can shift the bits

$b_1 = a_0$, $b_2 = a_1$, $b_3 = a_2$, $b_4 = a_3$, $b_5 = a_4$, $b_6 = a_5$, $b_0 = a_6$

Polynomial Codes:

→ A polynomial code is a linear code having a set of valid code words that comprises of polynomials divisible by a shorter fixed polynomial is known as generator polynomial.

→ They are used for error detection & correction during the transmission of data as well as storage of data.

Types:

- Cyclic Redundancy Code
- Bose - Chaudhuri - Hocquenghem (BCH) Codes
- Reed Solomon Codes

Representation:

Bit strings are represented by polynomials whose coefficients are either 0 or 1. A 'k' bit word is represented by a polynomial ranging from x^0 to x^{k-1}. The order of this polynomial is the power of the highest coff. i.e. (k-1).

Eg. a 8-bit word 11001101 is represented by following poly. of order 7.

$1x^7 + 1x^6 + 0x^5 + 0x^4 + 1x^3 + 1x^2 + 0x^1 + 1x^0$

$= x^7 + x^6 + x^3 + x^2 + 1$

Modulo 2 Arithmetic

Polynomial code operations are done by modulo 2 arithmetic.

- **Addition & Subtraction:** addⁿ & sub. has no carries or borrows. Thus, both operat has same as XOR (exclusive OR) operation.

Operand 1	Operand 2	Modulo 2 Add.	Modulo 2 sub.
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Eg.

```
  11001011          11001011
+ 10101111        - 10101111
----------        ----------
  01100100          01100100
```

- **Multiplication:** same as AND operation.

Operand 1	Operand 2	Modulo 2 Multiplication
0	0	0
0	1	0
1	0	0
1	1	1

Generator Polynomial:

When msg. are encoded using polynomial code, a fixed polynomial called generator polynomial $G(x)$ is used. The length of $G(x)$ should be less than the length of msg. it encodes.

✱ Sliding Window Protocols

→ It is a DDL protocol for reliable & sequential delivery of data frames. The SW is also used in TCP.

→ In this, multiple frames can be sent by a sender at a time & before receiving an ack. from the R.

→ SW refers to imaginary boxes to hold frames.

→ SW is also known as windowing.

Working Principle:

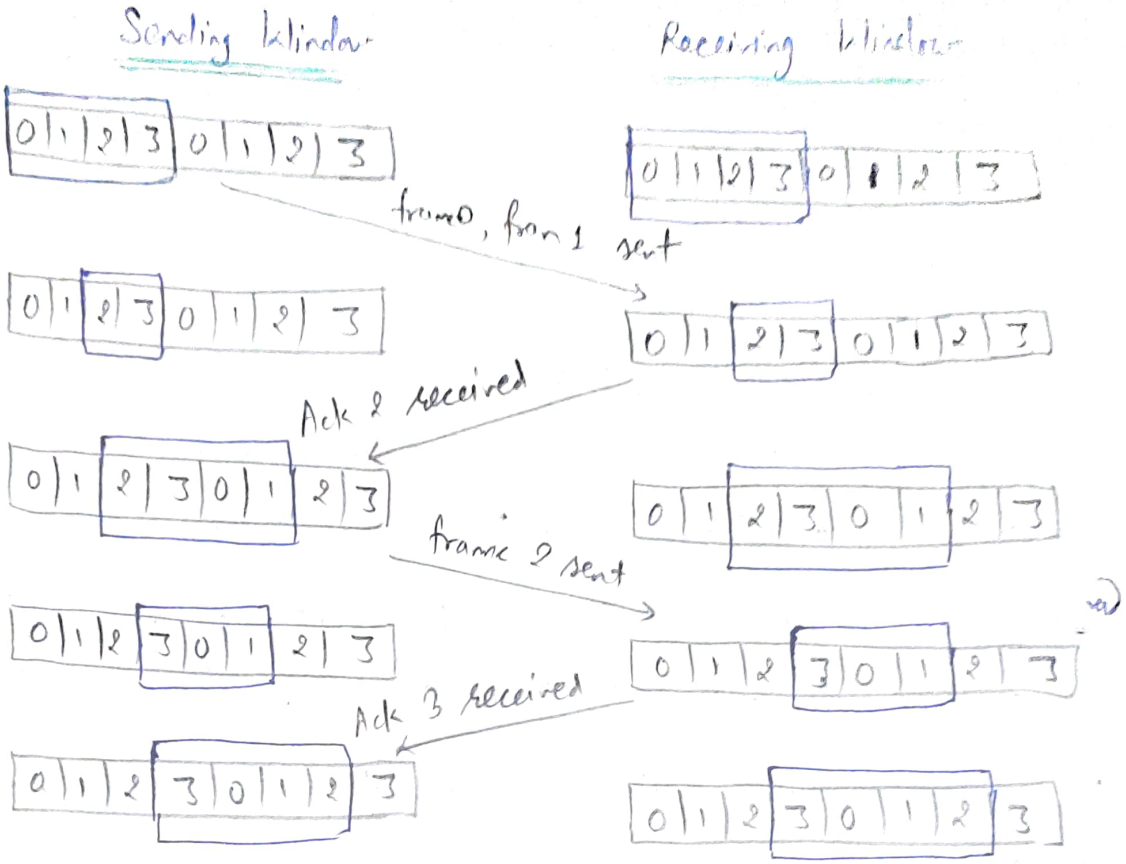
The sender has a buffer called sending window and receiver has a buffer called receiving window.

Eg. If sending window size is 4, then seq. no. will be 0, 1, 2, 3, 0, 1, 2, ... so on.

The size of Receiving window is the max. no. of frames that the R can accept at a time. It determines the max. no. of frames that the S can send before receiving ack.

Eg. Suppose that we have S window and R window each of size 4. So the seq. no. of both the window will be 0, 1, 2, 3, 0, 1, 2 And so on.

following diagram shows position of window after sending the frames and receiving ack.



Types:

1) Go-Back-N ARQ

2) Selective Repeat ARQ

Difference b/w Go-Back-N ARQ & Selective Repeat:

Go-Back-N ARQ	Selective Repeat ARQ
1) If a frame is corrupted or lost, all subsequent frames have to be sent again.	Only frame is sent again, which is corrupted or lost.
2) If it has a high error rate, it waste a lot of BW.	There is a loss of low BW.

3)	It is less complex.	More Complex bcoz it has to do sorting and searching as well. and it requires more storage.
4)	It does not require sorting.	Sorting is done to get frames in correct order.
5)	Does not require searching	required searching
6)	It is used more.	It is used less bcoz of more complexity.

* Piggy Backing:

→ Transmission of data needs to be bi-directional. This is called as full-duplex transmission.

→ Acheive full duplex transmission by having two separate channels ~~→~~ one for forward data transfer and other for separate transfer i.e. for ack.

→ A better solutn would be use each channel (forward & reverse) to transmit frames both ways, with both channels having same capacity. If A and B are two users, then the data frames from A to B are intermixed with the ack from A to B.

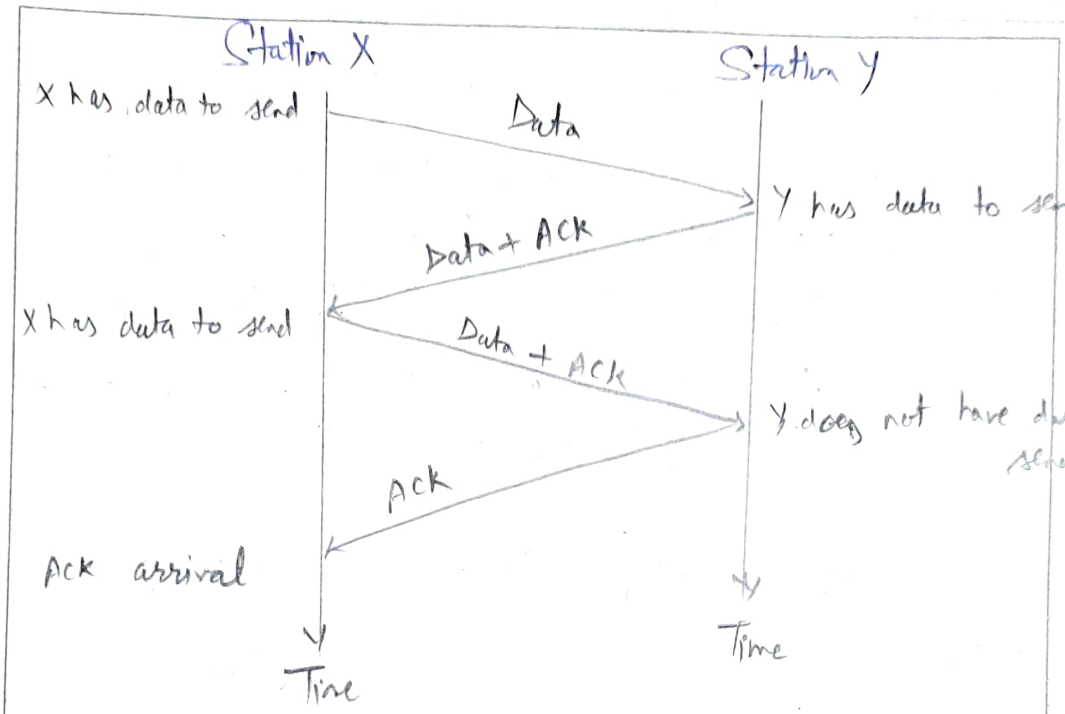
→ In two way comm^n, whenever a data frame is received, the receiver waits & does not send a control frame (ack) back to sender immediately.

→ The R waits until its NL (nlw layer) passes in the next data pkt. The delayed ack is then attached to this outgoing data frame.

→ This tech. of temporarily delaying the ack so that it can be hooked with next outgoing data frame is know as piggybacking.

Eg^n: When station X wants to comm^n with station Y.

- If station X has both data & ack to send, it sends a data frame with ack field containing the seq. no. of the frame to be acknowledged.
- If station X has only an ack to send, it waits for a finite period of time to see whether a data frame is available to be sent. If a data frame becomes available, then it piggybacks the ack with it, otherwise, it sends an ACK frame.
- If station X has only a data frame to send, it adds the last ack with it. The station Y discards all duplicate ack., Alternatively, Station X may send the data frame with the ack field containing a bit combination denoting no ack.



* Multiple Access Protocols

The DLL is responsible for transmission of data b/w two nodes. its main fun^n are:

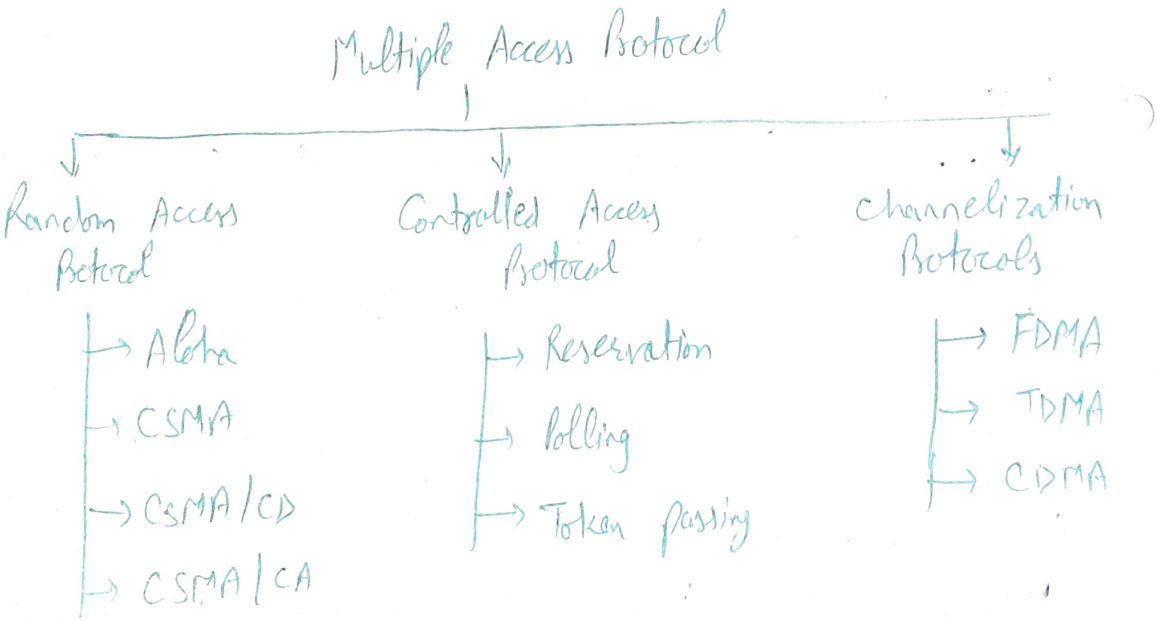
- Data Link Control & . Multiple access control

Data Link Control: it is responsible for reliable transmission of msg over transmission channel by using techniques like framing, error control, flow control.
eg. Stop & Wait ARQ.

Multiple Access Protocol: If there is no dedicated link between the sender & receiver then the data link control layer is sufficient, however if there is no dedicated link present then multiple stations can access the channel simultaneously. Hence, multiple access protocol are required to ↓↓ Collision and avoid crosstalk.

Eg. In a classroom full of students, when a teacher ask a question and all the students (or station) start answering simultaneously (send data at same time) then a lot of chaos is created (data overlap or data lost) then it is the job of a teacher (multiple access protocol) to manage the students and make them ans. one at a time.

Thus, protocols are required for sharing data on non dedicated channels.



Multiple Access Protocol's: It is a set of protocols operating in the Medium Access Control Sublayer (MAC sublayer) of the open system Interconnection (OSI) model.

These protocols allow a no. of nodes or users to access a shared n/w channel.

Several data streams originating from several nodes are transferred through the multi-point transmission channel.

The objectives of multiple access protocols are optimization of transmission time, min. of collisions and avoidance of crosstalks.

Random Access Protocol: All stations have same superiority that is no station has more priority than another station. Any station can send data depending on medium's state (idle or busy).

It has two features:

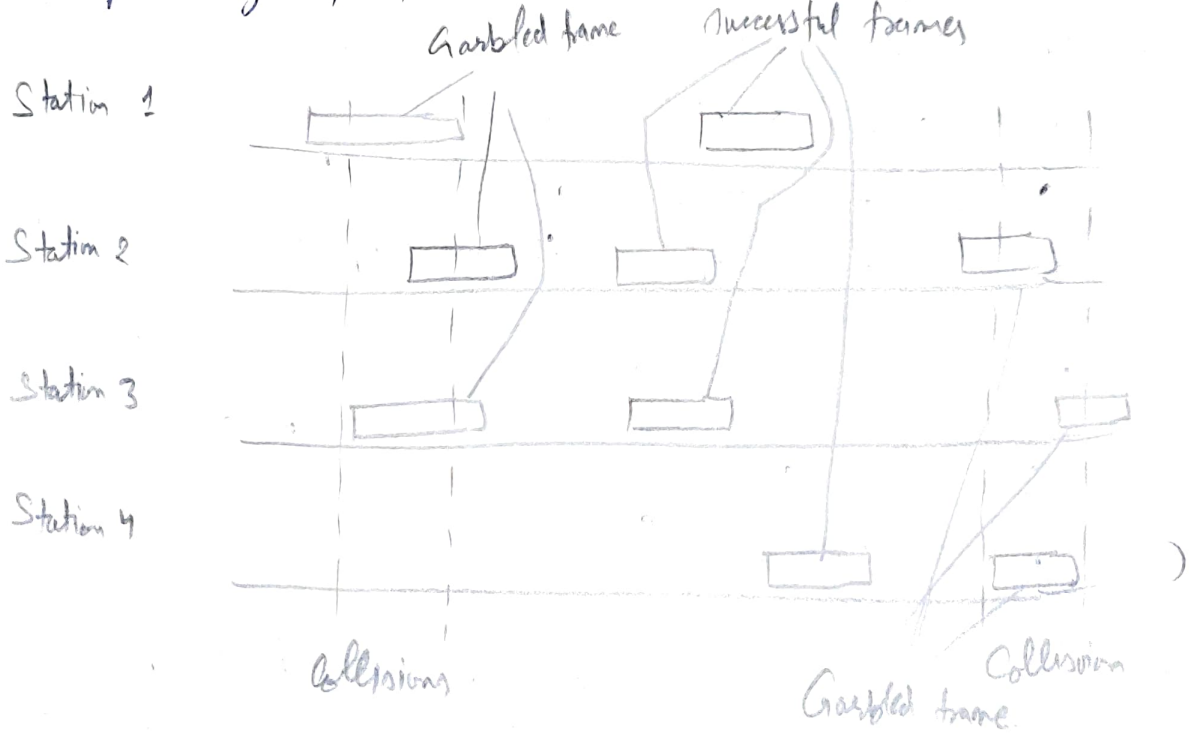
↳ There is no fixed time for sending data

↳ There is no fixed sequence of stations for sending data.

Aloha: It was designed for wireless LAN but it is also applicable for shared medium. Multiple station can transmit data at same time and can hence lead to collision and data being garbled.

↳ Pure Aloha ↳ Slotted Aloha

Pure Aloha: When a station sends data it waits for an ack. If the ack doesn't come within the alloted time then the station waits for a random amount of time called back off time (T_b) and re-sends the data. Since diff. station wait for diff. amount of time, The probability of further collision decreases.



~~In this two cases are possible.~~

[A collision occur if more than one frame tries to occupy the channel at the same time.]

2 Cases are possible:

1) Case 1:

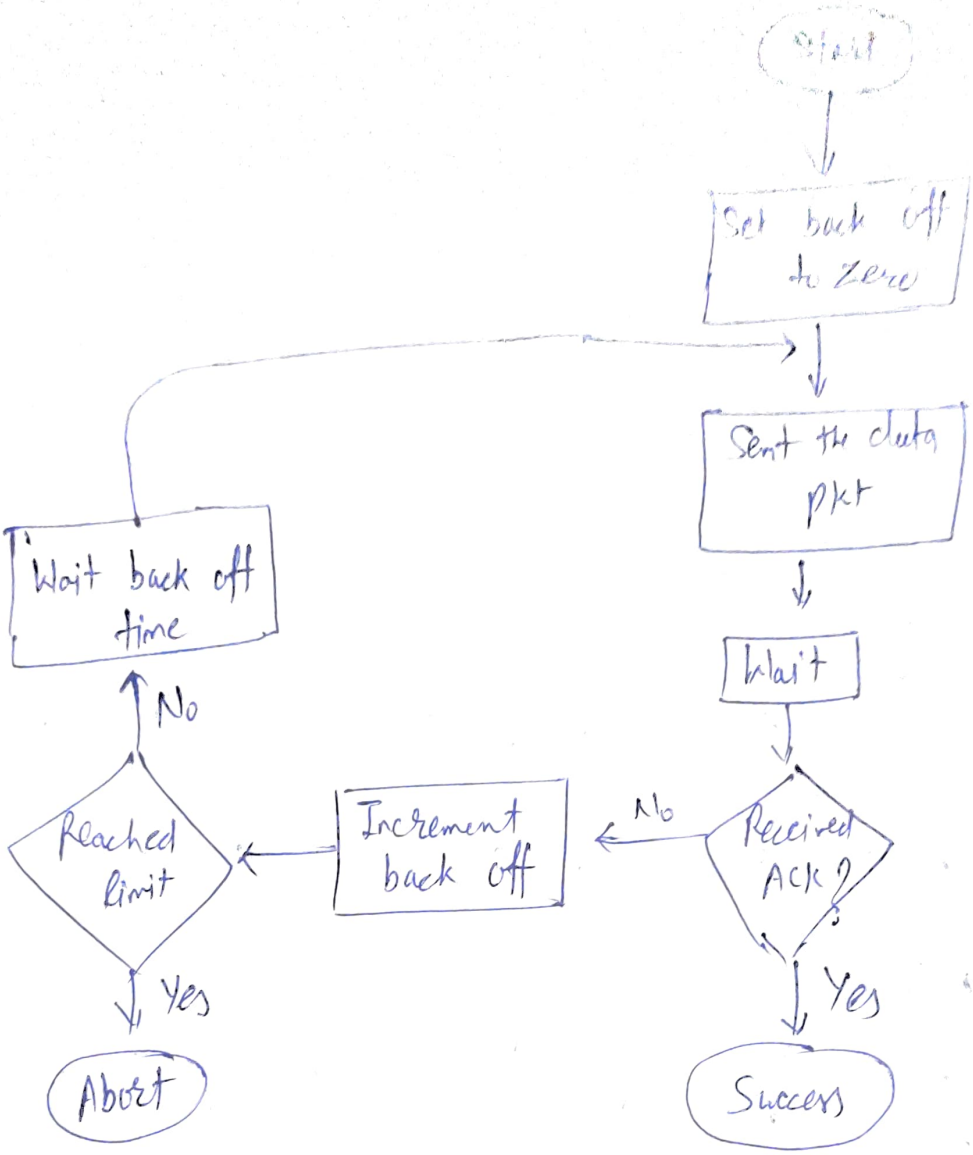
- Transmitting station receives an ack. from the receiving station.
- transmitting station assumes that transmission is successful.

2) Case 2:

- Transmitting station does not receive any ack within specified time from the receiving station.
- transmitting station assumes that transmission is unsuccessful.

Then,

- Transmitting station uses a Back off Strategy and wait for some random amount of time.
- After back off time, it transmit the data pkt again.
- It keeps trying until the back off limit is reached after which it aborts the transmission.



(Flow chart of Pure Aloha)

Throughput of Pure Aloha (efficiency):

Let T be the frame time, i.e. the time required for 1 frame to be transmitted.

Let G be the no of transmission attempts per frame time.

The probability that k frames are generated during the frame time is given by Poisson distribution

$$P(k) = \frac{G^k e^{-G}}{k!}$$

We can say that probability ~~that~~ of 0 frames are generated ~~(k=0)~~ ($k=0$) during the frame time is e^{-G}

In Case of pure aloha, the vulnerable time period so that collision does not occur b/w two frames is equal to two frames times, i.e. $2T$. In $2T$ time, avg. no. of transmission attempts is $2G$.

The probability that 0 frames are initiated in the vulnerable time period will be:

$$P(0) = e^{-2G}$$

The throughput S, is calculated as the no. of transmission attempts per frame time, G, multiplied by the probability of success $P(0)$.

$$S = G\,P(0)$$

$$S = G\,e^{-2G}$$

Max. Throughput of Pure Aloha?

max. throughput occurs when $G = 0.5$

$$S_{max} = 0.5\,e^{-2\times 0.5} = 0.5\,e^{-1} = \frac{1}{2e} = 0.184$$

(Length of frame)

Vulnerable Time = 2 × frame transmission time

Throughput = Ge^{-2G}

Max. Throughput = 0.184 for $G = 0.5$

vulnerable time = $2 \times T_{fr}$

Slotted Aloha: It was invented for improve the efficiency of pure aloha.

→ The time of shared channel is divided into discrete intervals called slots.

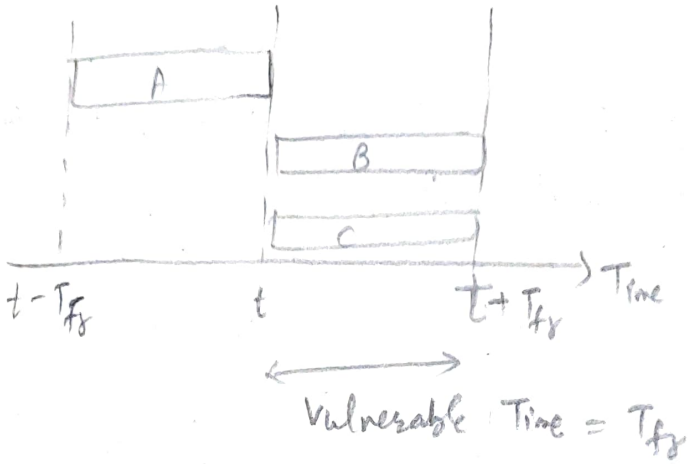
→ The station can send a frame only at the beginning of the slot and only one frame is sent in each slot.

→ If any station is not able to place the frame onto the channel at the beginning of the slot i.e. it misses the time slot then the station has to wait until the beginning of the next time slot

→ There is still a possibility of collision: ~~the~~ two stations try to send at the beginning of the same time slot.

→ In this colligion is reduced to one half.

The Vulnerable time is now reduced to one-half equal to T_{fr}.



Slotted Aloha vulnerable time = T_{fr}

Throughput (efficiency):

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

Where G = No. of stations willing to transmit data at the beginning of the same time slot

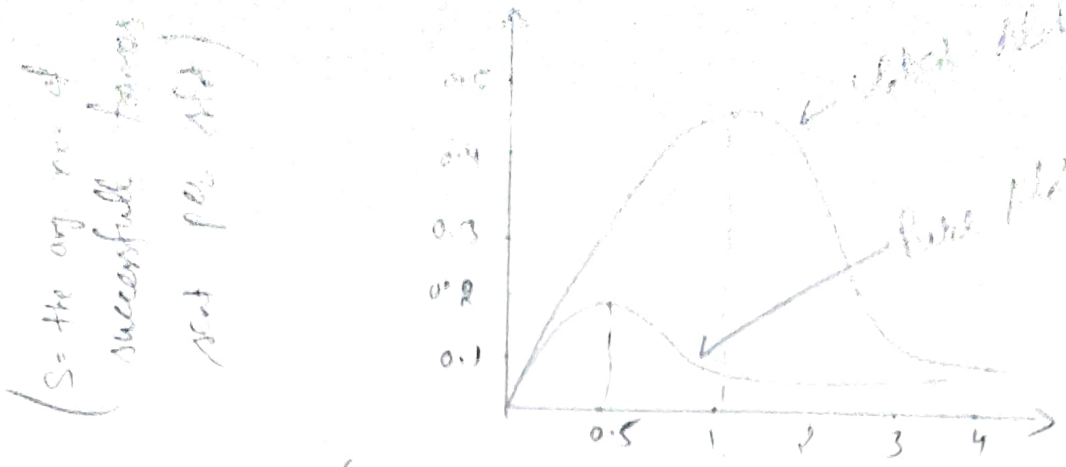
Max. Throughput:

$\eta = 36.8\%$ when $G = 1$

$\eta = .368$

max. efficiency is high due to less no. of collision.

Comparison of the Pure & Slotted Aloha



(G = traffic rate measured as the avg. no. of frames generated / slot)

Slotted Aloha has a double throughput efficiency than the pure Aloha.

Difference b/w Pure Aloha and Slotted Aloha:

Pure Aloha	Slotted Aloha
1) any station can transmit the data at any time.	transmit data at the beginning of any time slot.
2) time is continuous and not globally synchronized.	time is discrete & globally synchronized.
3) Vulnerable time in which collision may occur = $2T_t$	Vulnerable time in which collision may occur = T_t
4) Probability of successful transmission of data pkt = $G \times e^{-2G}$	$G \times e^{-G}$

5)	max. efficiency = 18.4% (occurs at $G = \frac{1}{2}$)	36.8% (occurs at $G = 1$)
6)	simple in implementation.	Reduces the no. of collision to half and doubles the efficiency of pure aloha.

CSMA:

- Stands for Carrier Sense Multiple Access.
- it ensures fewer collisions. as the station is required to first sense the medium (for idle or busy) before transmitting data.
- If it is idle then it sends data, otherwise it waits till the channel becomes idle.
- However there is still chance of collision in CSMA due to propagation delay.

Eg: if station A wants to send data, it will first sense the medium. If it finds the channel idle, it will start sending data. However, by the time the first bit of data is transmitted (delay due to propagation delay) from station A,

If station B senses the channel before station A wakes up, it will also find it idle and will also send data. This will result in collision of data from station A and B.

CSMA is based on the principle "sense before transmit" or "listen before talk."

CSMA can reduce the possibility of collision, but it cannot eliminate it.

CSMA access Modes

- **1-Persistent:** The node senses the channel, if idle it sends the data, otherwise it continuously keeps on checking the medium for being idle and transmits unconditionally (with 1 probability) as soon as the channel gets idle.

- **Non-persistent:** The node senses the channel, if idle it sends the data, otherwise it checks the medium after a random amount of time (not continuously) and transmits when found idle.

- **P-Persistent:** The node senses the medium, if idle it sends the data with p probability. If the data is not transmitted ((1-p) probability) then it waits for some time and checks the medium again, now if it is found idle then it sends with p probability. This repeat continues until the frame is sent. It is used in Wifi and pkt radio sys.

- **O-persistent:** Superiority of nodes is decided beforehand and transmission occur in that order. If the medium is idle, node waits for its time slot to send data.

CSMA/CD:

→ Stands for Carrier Sense Multiple Access with Collision Detection.

→ It is a n/w protocol for carrier transmission that operates in the Medium Access Control (MAC) layer.

→ It senses or listen whether the shared channel for transmission is busy or not and defers transmission until the channel is free.

→ The collision detection technology detects collision by sensing transmission from other station.

→ On a detection of a collision, the station stops transmitting, sends a jam signal, then waits for random time interval before retransmission.

Frame Format of CSMA/CD:

PR	SFD	DA	SA	L	D	FCS
7 byte	1 byte	6 byte	6 byte	2 byte	46 to 1500 byte	4 byte

Preamble: (PR) It is 7 bytes (56 bits) that provides bit synchronization. It consits of alternating 0s and 1s. The purpose is to provide alert and timing pulse.

Start frame delimiter (SFD): It is one byte field with unique pattern: b_0 to b_7 is 10101011. It marks the beginning of frame.

Destination Address (DA): It is a 6 byte field contains physical address of pkt's destination.

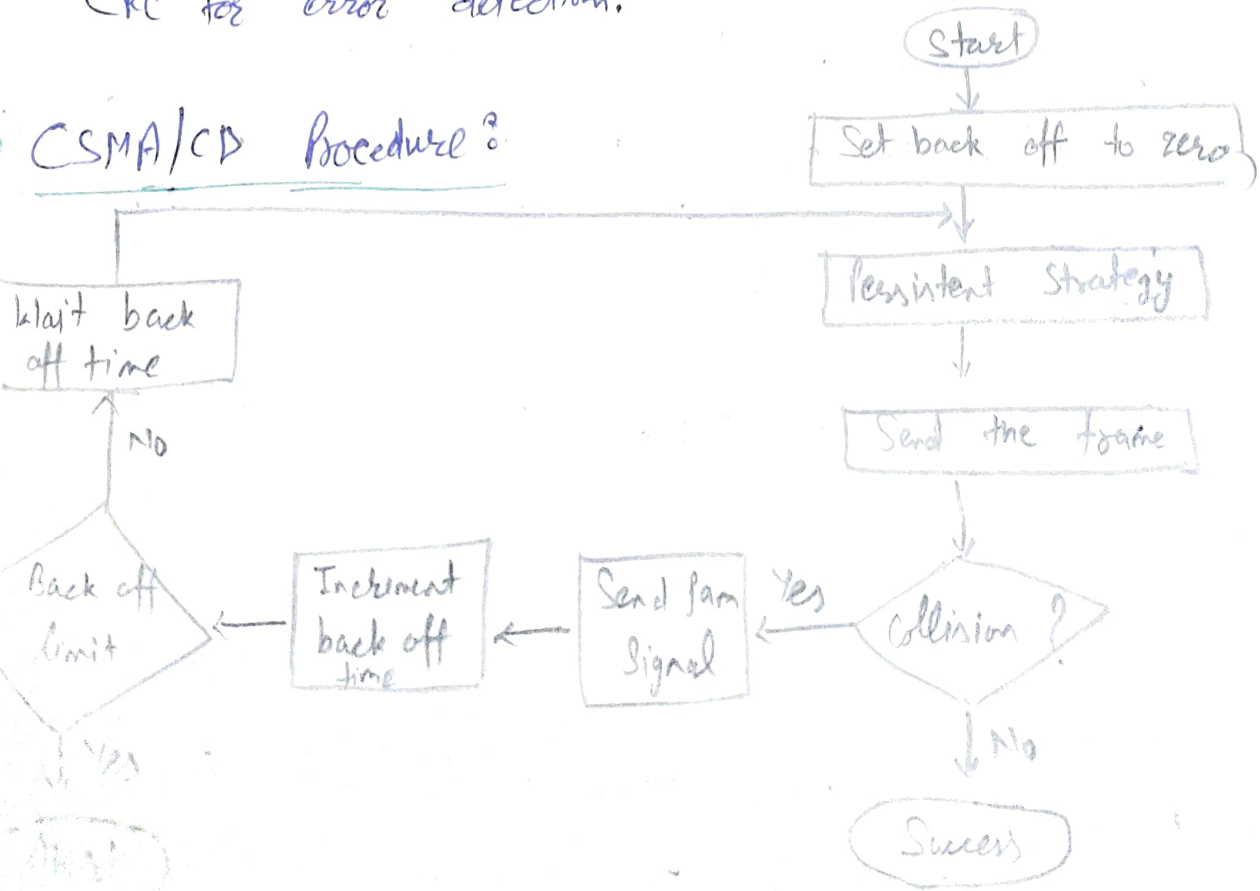
Source Address (SA): It is also a 6 byte field and contains physical address of source or last device to forward the pkt.

Length (L): This two byte field specifies the length or no. of bytes in data field.

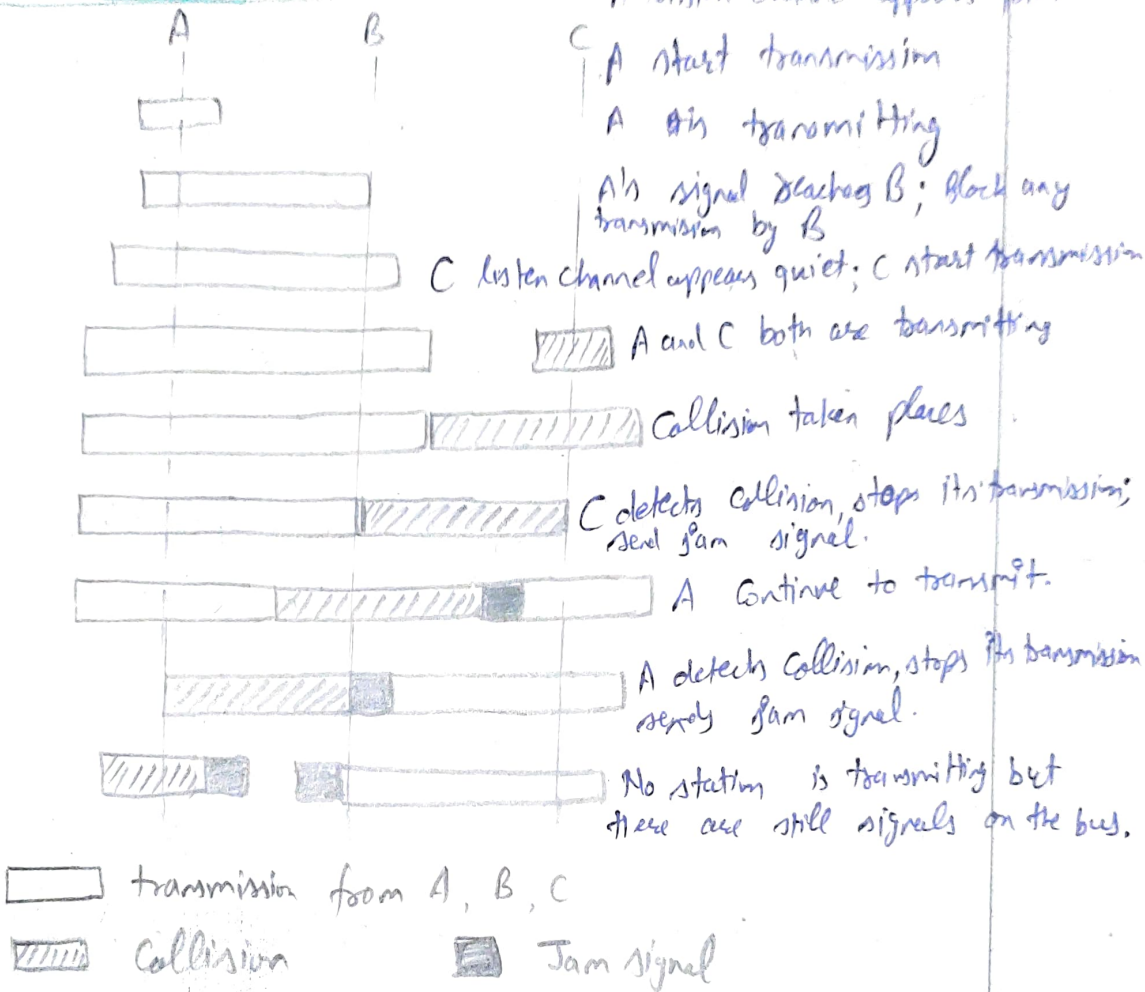
Data (D): It can be 46 to 1500 bytes, depending upon the type of frame and the length of information field.

Frame check Sequence (FCS): This for byte field containing CRC for error detection.

CSMA/CD Procedure:



CSMA/CD Scheme:



How CSMA/CD works?

Step1. Check if the sender is ready for transmitting data pkts

Step2. Check if transmission link is idle?

Sender checks transmission link is idle or not. For this it continuously senses transmission from other nodes. Sender sends dummy data on the link. If it does not

[illegible] signal, then it senses the link is busy and there are no collisions, it sends the data. Otherwise it halt from sending data.

Step 3: Transmit the data & check for collisions.

S transmits its data on the link. CSMA/CD does not use ack system. It checks for the succesful and unsuccessful transmission through collision signals.

During transmission, if collision signal is received by the node, transmission is stopped. The station then transmits a jam signal onto the link and waits for (random) time interval before it resends the frame.

After some random time, it again attempts to transfer data and repeats above process

Step 4. If no collision was detected in propagation, the S completes its frame transmission and resets the counters.

CSMA/CA:

→ Stands for Carrier Sense Multiple Access with Collision Avoidance.

→ The process of collision detection involves Sender receiving ack signals.

→ If there is just one signal (its own) then the data is successfully sent but if there were two signals (its own and the one with which it has collided) then it means a collission has occured.

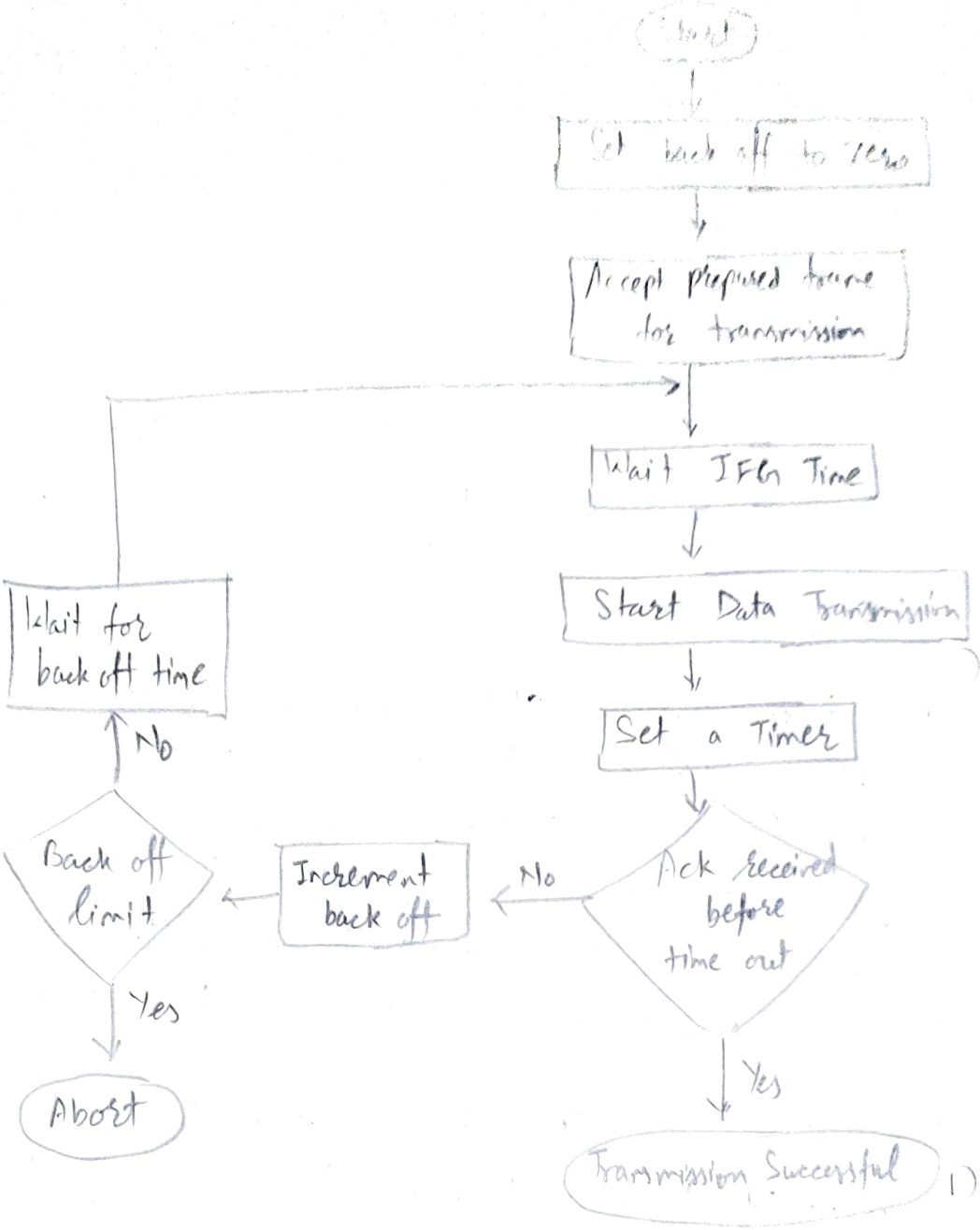
→ To distinguish b/w these two cases, collision must have a lot of impact on received signal.

CSMA/CA avoids collision by:

1) Interframe space: Station waits for medium to become idle and if found idle it does not immediately send data (to avoid collision due to propagation delay) rather it waits for a period of time called IFS. after this time it again checks the medium for being idle. The IFS duration depends on the priority of station.

2) Contention Window: It is the amount of time divided into slots. If the sender is ready to send data, it chooses a random no. of slots as wait time which doubles every time medium is not found idle. If the medium is found busy it does not restart the entire process, rather it restarts the timer when the channel is found idle again.

3. Acknowledgement: The sender re-transmits the data if ack is not received before time out.



* Difference b/w CSMA/CD and CSMA/CA:

CSMA/CD	CSMA/CA
1) Effective after a collision	Effective before a collision.
2) It is used in wired network	used in wireless network.
3) Only reduces the recovery time.	minimizes the possibility of collision.

4)	Resends the data frame whenever a conflict occurs.	Will first transmit the intent to send for data transmision.
5)	Used in 802.3 standard.	Used in 802.11 standard.
6)	More efficient than simple CSMA.	Similar to simple CSMA.