UNIT-II

Data Link Layer

LALIT KUMAR

Introduction :- The Physical layer deals with the transmission of signals over different transmission medias. For achieving reliable, efficient communication between two adjacent (near) machines the data link layer plays an important role. This layer (DLL) basically deals with frame formation (framing), flow control, error control, addressing and link management. While sending data from source to destination the communication circuits make errors. The data link protocols used for communication take care of all these problems.

Position of Data link layer :- fig shows the position of data link layer in the five layer Internet model. It is the second layer.

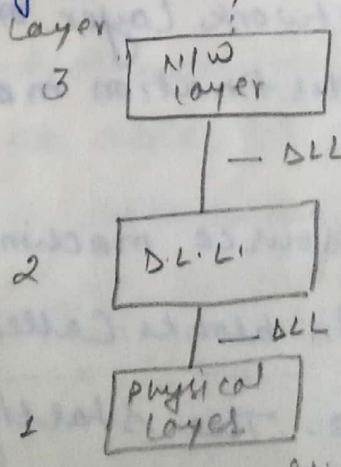


fig: Position of Data link layer

It receives services from the Physical layer and provides services to the network layer.

Data Link Layer Design Issues:

For effective data communication between two directly connected transmitting and receiving stations the data link layer has to carry out a number of specific functions as follows:

(i) Data transfer (services provided to the N/w layer)
(ii) Frame Synchronisation
(iii) Flow control
(iv) Error Control
(v) Addressing
(vi) [Control and data on the same link.
(vii) Link Management.]

1) Services provided to the Network Layer :- A well define service interface to the network layer. The principle service is transferring data from the network layer on source machine to the network on destination machine.

2) Frame Synchronization :- The source machine sends data in blocks called frames to the destination machine. The starting and ending of each frame should be recognised by the destination machine.

② Flow Control :- The source machine must not send data frames at a rate faster than the destination machine can accept them.

④ Error Control :- The error made in bits during transmission from source to destination machine must be detected and corrected.

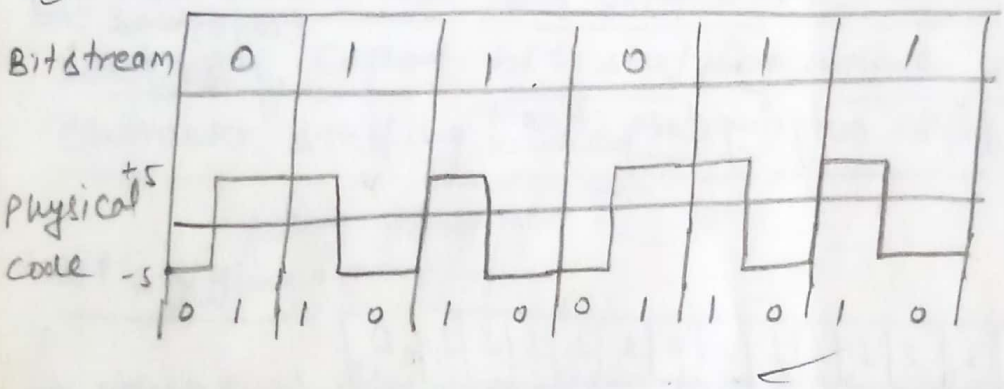
⑤ Addressing :- On a multipoint line, such as many machines connected together (LAN), the identity of the individual machines must be specified while transmitting the data frames.

⑥ Control and data on same link :- The data and control information is combined in a frame and transmitted from the source to destination machine. The destination machine must be able to recognise control information from the data being transmitted.

⑦ Link management :- The initiation, maintenance and termination of the link between the source and destination is required for effective exchange of data. It ~~required~~ requires co-ordination and co-operation among

→ Normally a 1 bit is encoded into a 10 pairs and a 0 bit is encoded into a 01 pair.

Note: " Many data link protocols use the combination of the character count technique with one of the other techniques so as to have an extra safety.



Encoding Technique:

1. Digital to Digital
2. Digital to Analog
3. Analog to digital
4. Analog to analog

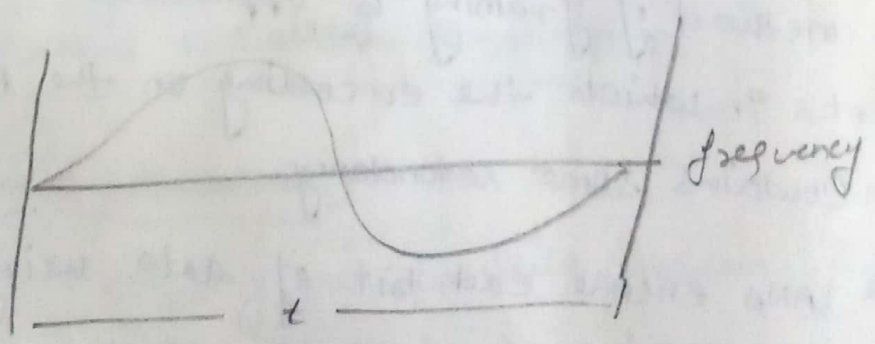


fig analog form of data (Continuous way)

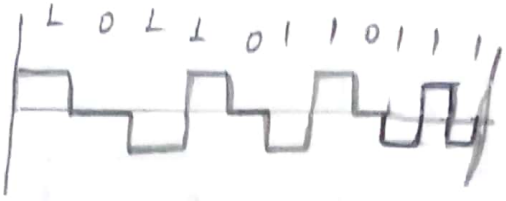
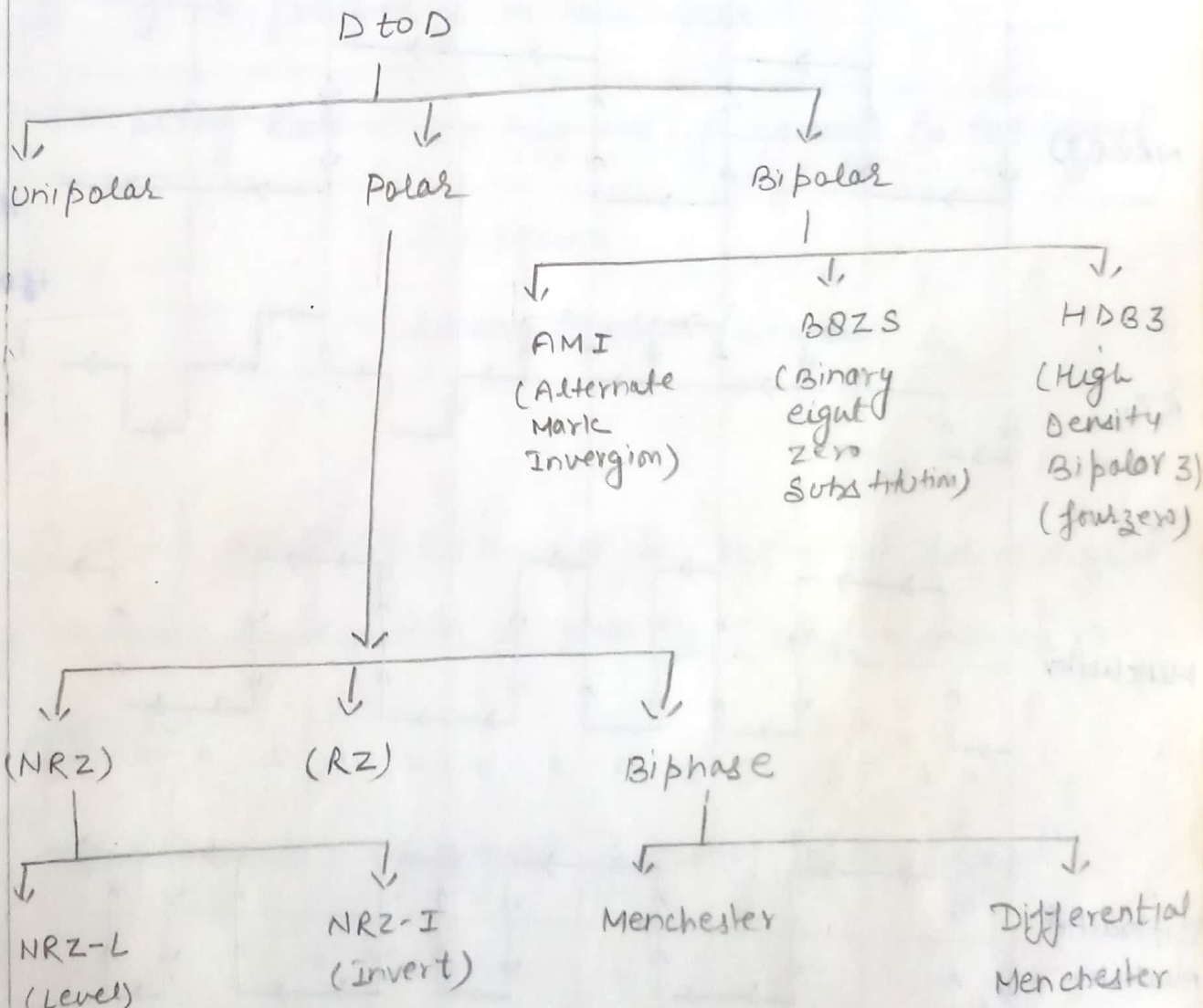


fig: digital form of data (Non continuous)

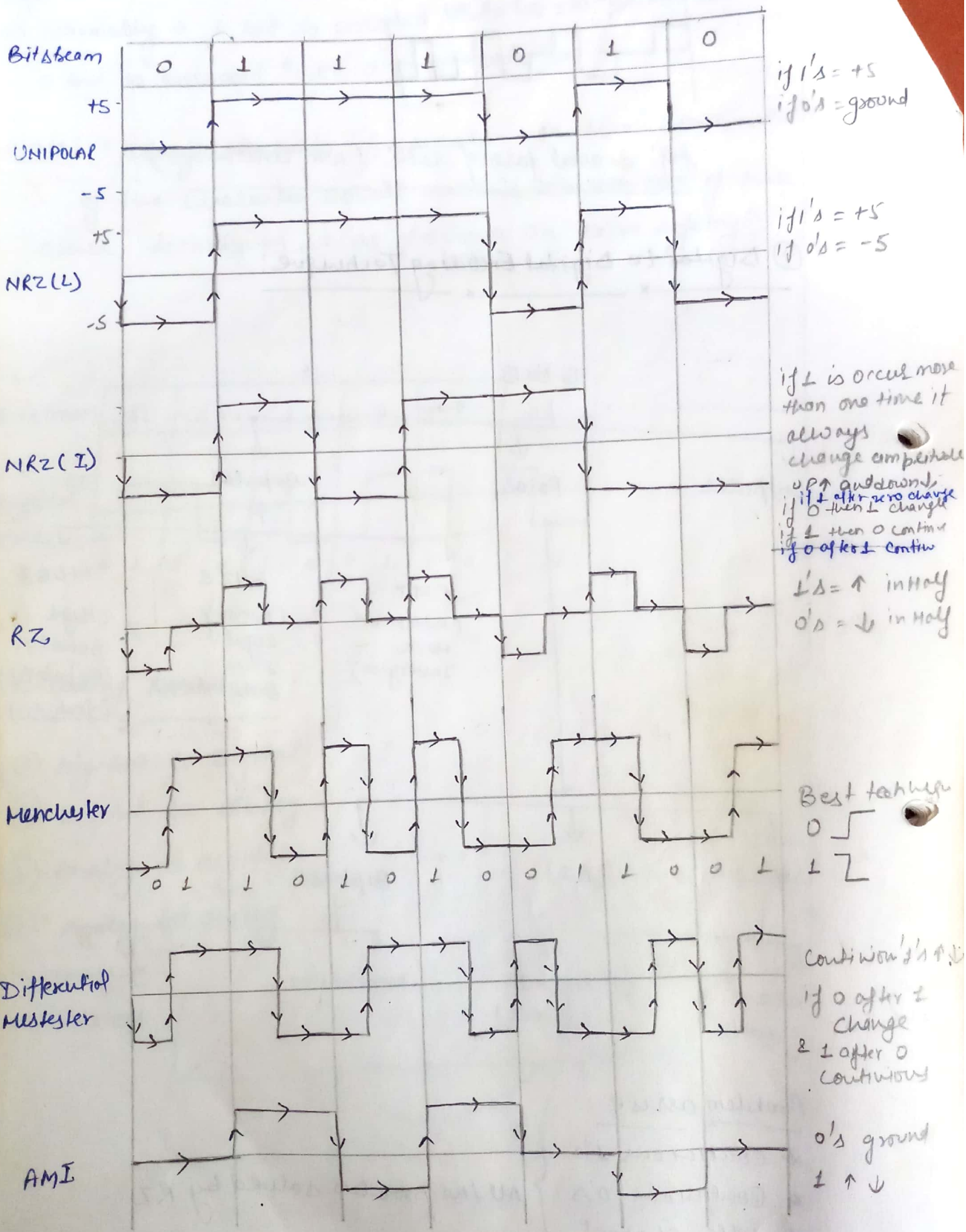
① Digital to Digital Encoding Technique:



Problem arrise:

* Continuous 1's
* Continuous 0's
* idle channel

} All the problem solved by RZ



Numericals:

LAUT KUMAR

(1.) What sampling rate is needed for a signal with a bandwidth of 10,000 Hz (1000 Hz to 11000 Hz)?

Solution:-

We know that the sampling rate must be twice of higher frequency in the signal.

Sampling Rate = 2 x Highest frequency in the signal

= 2 x 11000

= 22000 samples/second Ans

(2.) What sampling Rate will be required for a signal with a Bandwidth of 5000 Hz (1000 Hz to 6000 Hz)?

Solution:

Sampling Rate = 2 x Highest frequency in the signal

= 2 x 6000

= 12000 sample/second Ans

Bit Rate: After determining the number of bits per sample we can evaluate the bit rate by using the following expression (formula).

Bit Rate = Sampling Rate x No. of bits per Sample

Numerical 3:- It is required to digitize the human voice. What will be the bit rate? Assume 8 bits per sample are assigned.

Sol: We know that the human voice generally consists of frequency from 0 to 4000 Hz. therefore the sampling Rate will be:

Sampling Rate = 2 x higher frequency

= 2 x 4000

= 8000 sample / second

Bit Rate = Sampling Rate x No. of bits per Sample

= 8000 x 8

= 64000 bps

Bit Rate = 64 Kbps } Ans

Numerical 4: Given that a composite periodic signal is decomposed into five sine waves with frequencies of 100, 300, 500, 700 & 900 Hz. Calculate the bandwidth of this composite analog signal? Draw the spectrum assuming all components have a maximum amplitude of 10 V.

Sol^{n}: Let

f_H be the highest frequency

f_L be the lowest frequency

B_T is the bandwidth, then

$$B_T = f_H - f_L$$

$$B_T = 900 - 100 = 800 \text{ Hz}$$

Bandwidth = 800 Hz Ans.

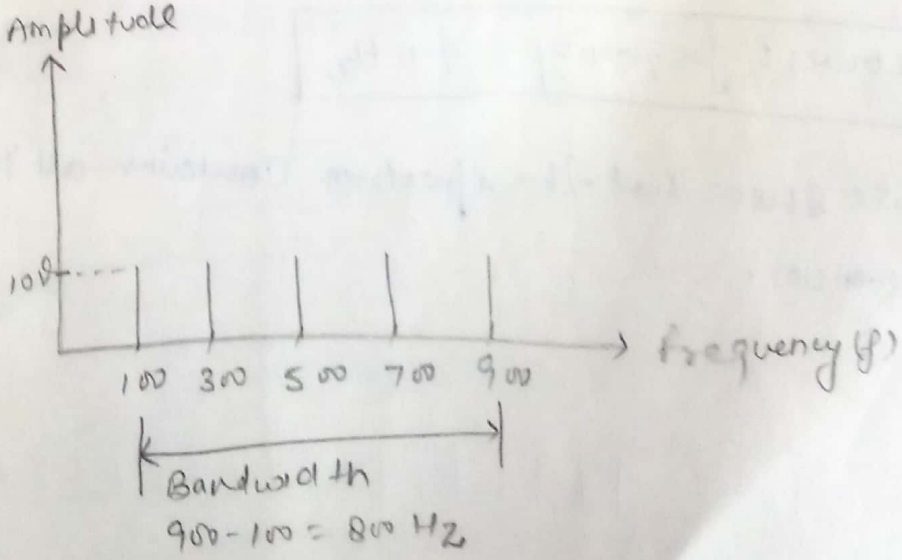


fig spectrum with freq. & amplitude

Numerical 5: A signal has a Bandwidth of 20 Hz. The highest frequency is 60 Hz. What is the lowest frequency? Draw the spectrum if the signal contains all integral frequency of the same amplitude.

Sol^n:

Let

f_H be the highest frequency

f_L be the lowest frequency

B_T is the Bandwidth.

then we know that

$$B_T = f_H - f_L$$

$$20 = 60 - f_L$$

$$f_L = 60 - 20$$

$$f_L = 40 \text{ Hz}$$

Lowest frequency = 40 Hz

figure show that the spectrum contains all integral frequencies.

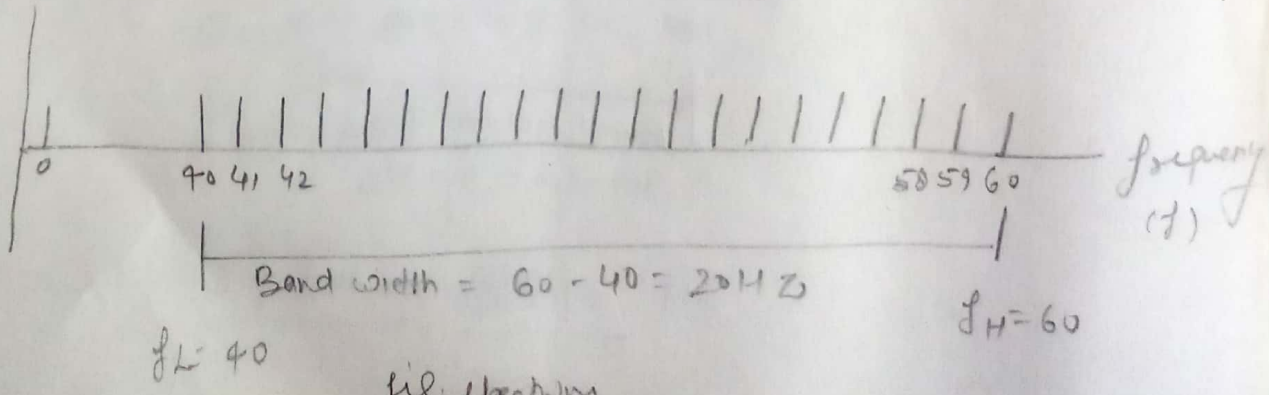


fig: spectrum

Error control: The next problem to be dealt with is to make sure that all frames are eventually delivered to the network layer at the destination, in proper order.

→ Generally the receiver sends back some feedback (Passitive or negative) to convey the information about whether it has received a frame or not.

→ A positive acknowledgement (feed back) indicates a successful and error free delivery of a frame, where as a negative acknowledgement means that something has gone wrong and that the frame needs to be retransmitted.

→ Due to the presence of noise burst a frame may lost (disappear, vanish) completely. So the Receiver does not Receive anything and it does not react at all (No acknowledgement).

→ This problem is solve (overcome) by introducing a timer in the D.L.L.

function of a Timer:

→ As soon as a sender transmits a frame, it also starts the data link timer.

→ The timer timing is set by taking into account the factors such as the time required for the frame to reach the destination, processing time at the destination and the time required for the acknowledgement to return back.

→ Normally the frame is received correctly and the acknowledgement will return back to the sender before the timer run out.

→ This shows that a frame has been received and the timer cancelled.

→ But if a frame is lost or acknowledgement is lost, then the timer will go off. This will alert the sender that their is some problem.

→ The solution to this problem is that the sender retransmits the same frame.

→ But when a frame is transmitted multiple times, there is a possibility that the receiver will accept the same frame two or more times and pass it to the network layer more than once.

→ To avoid this each outgoing frame is assigned a distinct sequence number. This will help the receiver to distinguish retransmission.

Introduction to error Detection and Correction:

→ Networks must be able to transfer data from one device to another with complete accuracy.

→ Yet anytime, data are transmitted from source to destination, they can become corrupted in passage.

→ Many factors, including line noise, can alter or wipe out one or more bits of a given data unit.

→ Reliable systems must have a mechanism for detecting and correcting such errors.

→ "Data can be correct corrupted during transmission for reliable communication, error must be detected and corrected."

→ Error detection and correction are implemented either at the DLL or the transport layer of the OSI model.

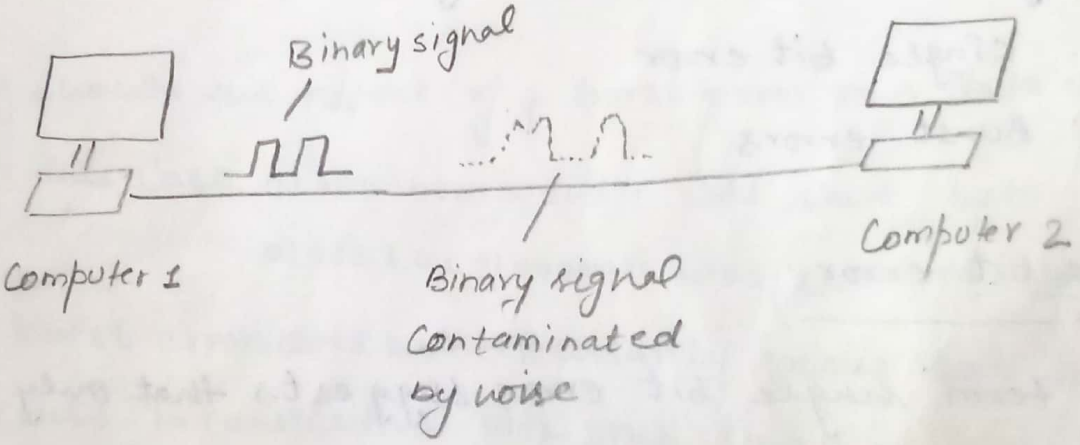


fig: Noise contaminates the Binary signal

Types of Errors!

→ The error introduced in the data bits during their transmission can be categorised as:

* Content errors
* Flow integrity errors

→ The content errors are noting but errors in the contents of a message for example: zero (0)

may be received as '1' (one) or vice-versa. Such errors are introduced due to noise (तारमा आउने electrical disturbance) added into the data signal during its transmission.

→ Flow integrity errors means missing blocks of data. It is possible that a data block may be lost the network as it has been delivered to a wrong destination

→ (Depending on the number of bits in error we can classify the errors into two types as,

* 1. Single bit error
* 2. Burst errors)

1. (Single bit error:)

→ The term single bit error suggests that only one bit in the given data unit such as byte is in error, character, or packet.

→ (That means only one bit will change from 1 to 0, or 0 to 1.

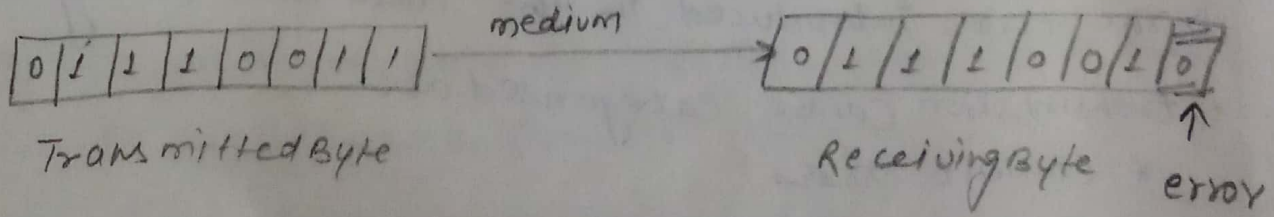


fig Single bit error)

(2) Burst errors:

A Burst error means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

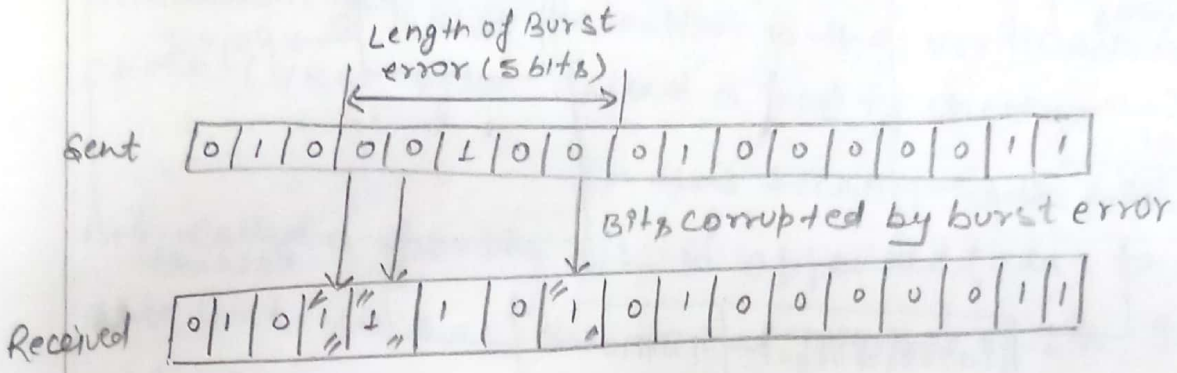


fig Burst error of length five

fig shows the effect of a burst error on a data unit. In this case 0100010000100000 11 was sent, but 0101110101000011 was received. Note that a burst error does not necessarily means that the errors occurs in consecutive bits. The length of the burst is measured from the first corrupted bit to the last corrupted bit. some bits in between may not have been corrupted.

Error Detection:- Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination.

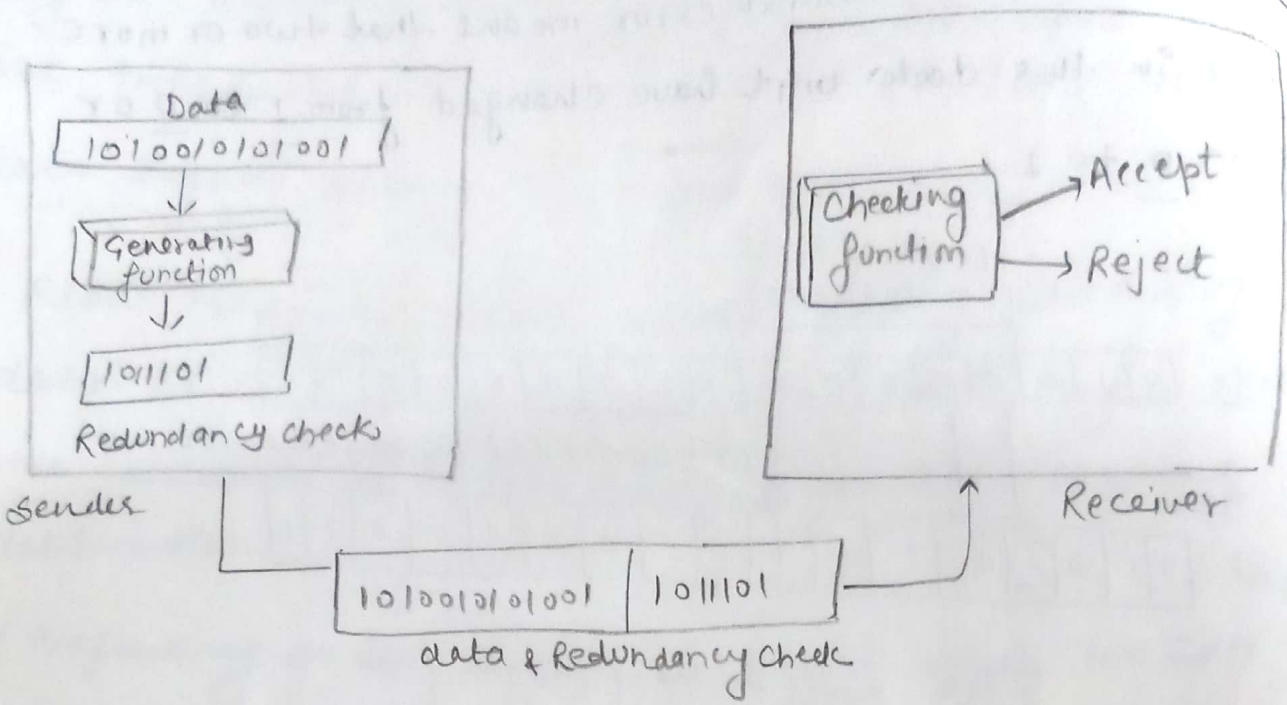


fig: Redundancy

Detection Method :- four types of Redundancy checks are used in data communications :

(i) Vertical Redundancy check (VRC) also called parity check.
(ii) Longitudinal Redundancy check (LRC)
(iii) Cycle Redundancy check (CRC).
(iv) Check Sum.

The first three, VRC, LRC, CRC normally implemented in the physical layer for use in the data link layer. The fourth, check sum, is used primarily by upper layers.

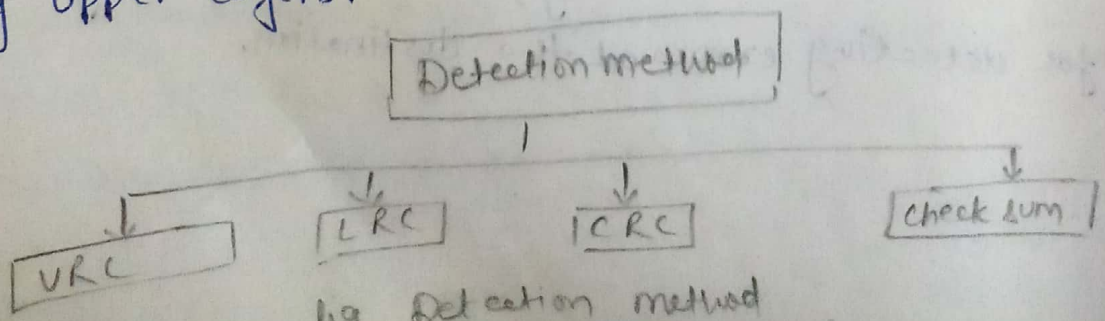


fig Detection method

(1) Vertical Redundancy Check (VRC):- The most common and least expensive mechanism for error detection is the vertical redundancy check (VRC), often called a parity check.

In this technique, a redundant bit, called a parity bit, is appended (add) to every data unit so that the total number of 1's in the unit (including the parity bit) becomes even.

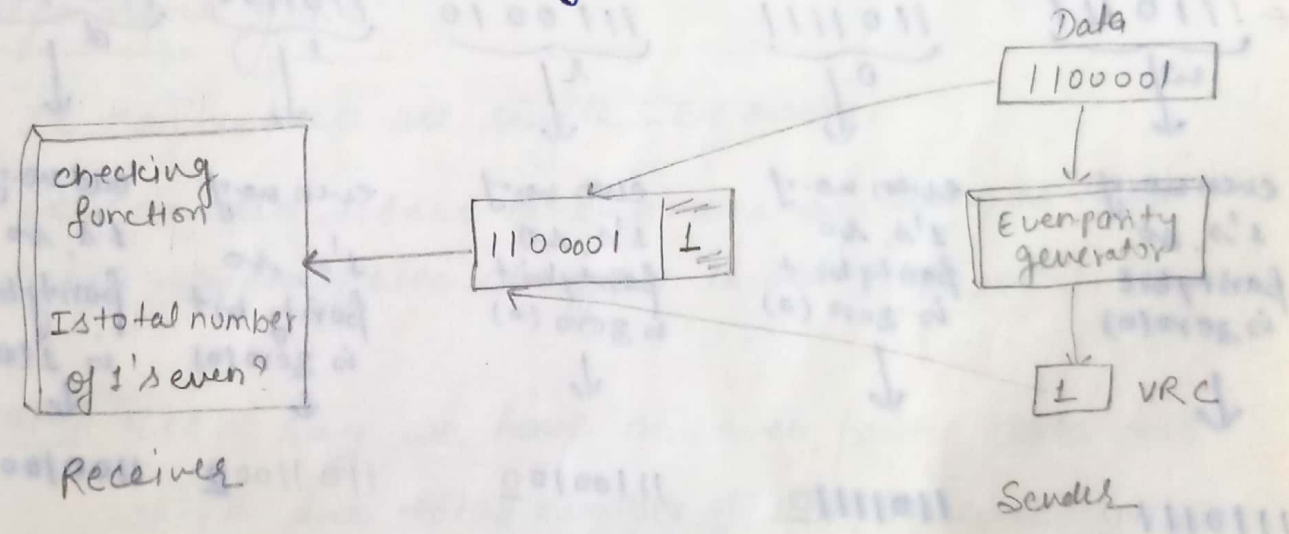


fig: Even Parity VRC concept

NOTE: We are discussing here even-parity checking, where the number of 1's should be an even number. Some system may use odd-parity checking, where the number of 1's should be odd. The principle is the same; the calculation is different.

example: Imagine the sender wants to send the word 'world'. In ASCII table (See Appendix A) (Page-809), the five character are coded as:

1110111 (w) 1101111 (o) 1110010 (r) 1101100 (l) 1100100 (d)

Solution:

1110111 (w)	1101111 (o)	1110010 (r)	1101100 (l)	1100100 (d)
even no. of 1's, so parity bit is zero (0)	even no. of 1's, so parity bit is zero (0)	even no. of 1's, so parity bit is zero (0)	even no. of 1's, so parity bit is zero (0)	odd no. of 1's, so parity bit is 1 (one)
← 11101110	11011110	11100100	11011000	11001001 (Sender)

(the following shows the actual bits sent (the parity bits are underlined)

Now the word "world" is received by the receiver without being corrupted in transmission.

← 11101110 11011110 11100100 11011000 11001001 (Receiver)

Now the receiver counts the 1's in each character and comes up with even numbers (6, 6, 4, 4, 4). The data would be accepted.

example: Now suppose the word 'world', is received by the receiver but corrupted during transmission.

← 1111 1110 1101 1110 1110 1100 1101 1000 1100 1001

Sol The receiver counts the 1's in each character and comes up with even and odd numbers (7, 6, 5, 4, 4), the receiver knows that the data are corrupted, discards them, and asks for retransmission.

Performance of VRC:

① It can detect all single-bit errors.

② It can also detect burst errors as long as the total number of bits changed is odd (3, 5, etc).

Note: Let's say we have an even parity data unit where the total number of 1's, including the parity bit, is 6: 1000 1110 11. If any three bits change value, the resulting parity will be odd and the error will be detected.

1 1 1 1 1 1 0 1 1 : 9, 0 1 1 0 1 1 1 0 1 1 : 7, 1 1 0 0 0 1 0 0 1 1 : 5

all odd. The VRC checker would return a result of 1 and the data unit would be rejected. The same holds true for any odd number of errors.

Suppose, however that two bits of the data unit are changed: 1000111011 ⇒ 1 1 1 0111011 : 8, 1 1 00 0 11011, 1 0000 1101 0 : 4. In each case the number of 1's in the data unit is still even. The VRC checker will add them and return an even number although the data unit contains 2 (two) errors.

VRC cannot detect errors where the total number of bits changed is even. If any two bits change in transmission, the changes cancel each other and the data unit will pass a parity check even though the data unit is damaged. The same holds true for any even number of errors.)

(2) Longitudinal Redundancy check (LRC):- (In longitudinal redundancy check (LRC), a block of bits is organized in a table (rows & columns). For example, instead of sending a block of 32 bits, we organize them in a table made of four row and eight columns. After we can calculate the parity bit for each column and create a new row of eight bits, which are the parity bits for the whole block.)

Note that the first parity bit in the fifth row is calculated based on all first bits. The second parity bit is calculated based on all second bits, and so on. We then attach the eight parity bits to the original data and send them to the receiver.

"In longitudinal redundancy check (LRC), a block of bits is divided into rows and a redundant row of bits is added to the whole block".

for example:

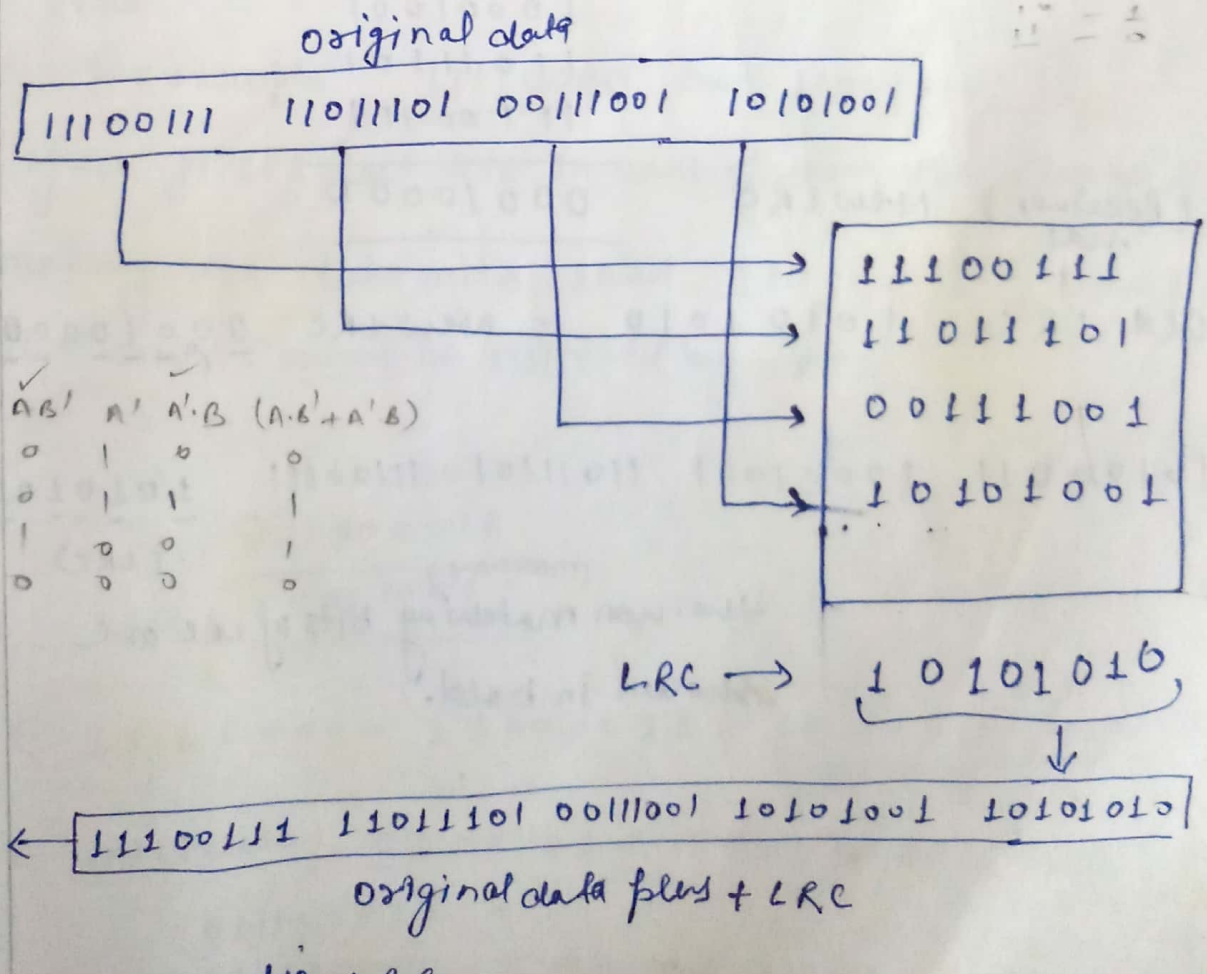
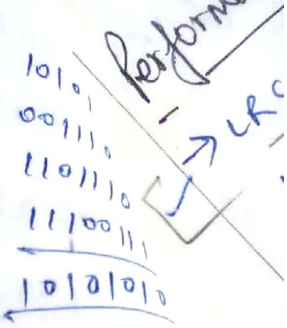


fig LRC

for example:

Suppose the following block is sent:

← 10101001 00111001 11011101 11100111 10101010 (LRC)



However, it is hit by a burst noise of length eight and some bits are corrupted.

← 10100011 10001001 11011101 11100111 10101010 (LRC)

old LRC

When the Receiver check the LRC, some of the bits do not follow the even-parity rule and the whole block is discarded. (the and ask for retransmission

10100011
10001001
11011101
11100111

(Receiver side) New LRC 00010000

Old LRC 10101010 - New LRC 00010000

← 10100011 10001001 11011101 11100111 10101010 (LRC)

(matching) the non matching bits of LRC are shown in bold.

ask for retransmission.

10101001 00111001 11011101 11100111 10101010

10101001
00111001
11011101
11100111
10101010

old LRC = 10101010
New = 1010 1010 } Accepted.

Performance:

→ LRC increases the likelihood (probability) of detecting burst errors.

As we showed in the previous example, an LRC of n bits can easily detect a burst error of n bits.

→ A burst error of more than n bits is also detected by LRC with a very high probability.

→ There is, however one pattern of errors that remains exclusive (Elusion).

drawback

→ If two bits in one data unit are damaged and bits in exactly the same positions in another data unit are also damaged, the LRC checker will not detect an error.

for example: 11110000 and 11000011

if the first & last bits in each of them are changed, making the data units read 01110001 & 01000010, the error cannot be detected by LRC.

⇒

```
  11110000
  11000011
  --------
  00110011   LRC
  --------
```

Sender ← 11110000 11000011 00110011 actual data + LRC

LRC (old)

Receiver ← 01110001 01000010 00110011 (New LRC)

```
  01110001
  01000010
  --------
  00110011   New CRC
  --------
```

both same

Note!
When the Receiver check the LRC, is same as sending LRC but this is two bits error in each frame.

(iii) CRC (Cyclic Redundancy Check):-

Numerical:- The following bit stream is encoded using VRC, LRC and even parity. Locate and correct the error if it is present.

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

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

1 1 1 0 0 0 0 1

Solution:

									VRC bit (even parity)
Byte 1 →	1	1	0	0	0	0	1	1	
Byte 2 →	1	1	1	1	0	0	1	1	
" 3 →	1	0	1	1	0	0	1	0	
" 4 →	0	0	0	0	1	0	1	0	
Bit in error " 5 →	(0)	0	1	0	1	0	1	0	← error (wrong parity)
" 6 →	0	0	1	0	1	0	1	1	
" 7 →	1	0	1	0	0	0	1	1	
" 8 →	0	1	0	0	1	0	1	1	
" 9 →	(1)	1	1	0	0	0	0	1	LRC bit
	5	4	6	2	4	0	8	6(1's)	

LRC → LRC bit (even parity)

error wrong parity

Note: the Parity bit corresponding to row five (5th) and column 1 indicate wrong parity. therefore the first bit in row 5th (encircled bit) is incorrect.

Cyclic Redundancy Check (CRC)

→ The third and most powerful of the redundancy checking techniques is the cyclic redundancy check (CRC).

→ Unlike VRC and LRC, which are based on addition, CRC is based on binary division.

→ In CRC, instead of adding bits together to achieve a desired parity, a sequence of redundant bits, called the CRC or the CRC remainder, is appended to the end of a data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.

→ At its destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be intact and is therefore accepted. A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

Procedure to obtain CRC: The redundancy bit used by CRC are derived by following the procedure given below:

→ (i) Divide the data unit by a predetermined divisor.

→ (ii) Obtain the remainder. It is the CRC.

Requirement of CRC: A CRC will be valid if and only if it satisfies the following requirements:

(i) It should have exactly one less bit than divisor.

(ii) Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.)

CRC generator and checker :-

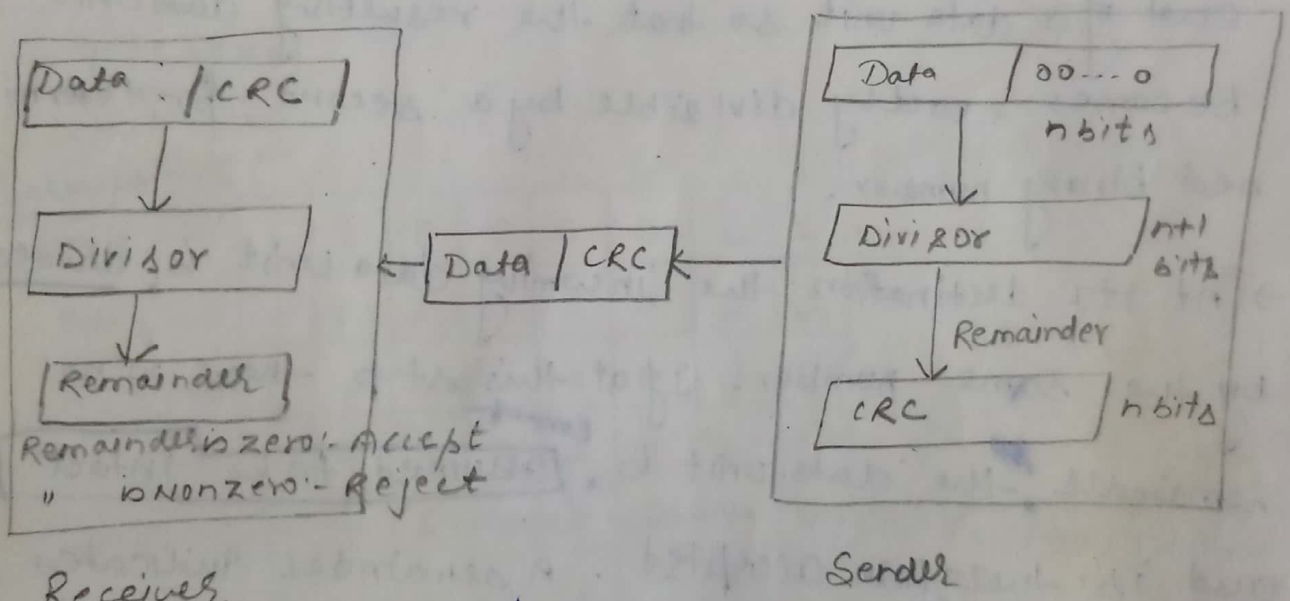


fig CRC generator & checker.

Step 1: A string of n zero's (0) is appended to the data unit. The number n is one less than the number of bits in the predetermined divisor, which is n+1 bits.

Step 2: The newly elongated data unit is divided by the divisor using a process called binary division. The remainder resulting from the division is the CRC.

Step 3: The CRC of n bits derived in step 2 replace the appended 0's at the end of the data unit. Note that the CRC may consist of all zero's (0's).

Step 4: The data unit arrives at the receiver data first, followed by the CRC. The receiver treats the whole string as a unit and divides it by the same divisor that was used to find the CRC remainder.

Step 5: If the string arrives without error, the CRC checker yields a remainder of zero and the data unit passes. If the string has been changed in transit, the division yields a non-zero remainder and the data unit does not pass.

Procedure to subtract :-

XOR fun.

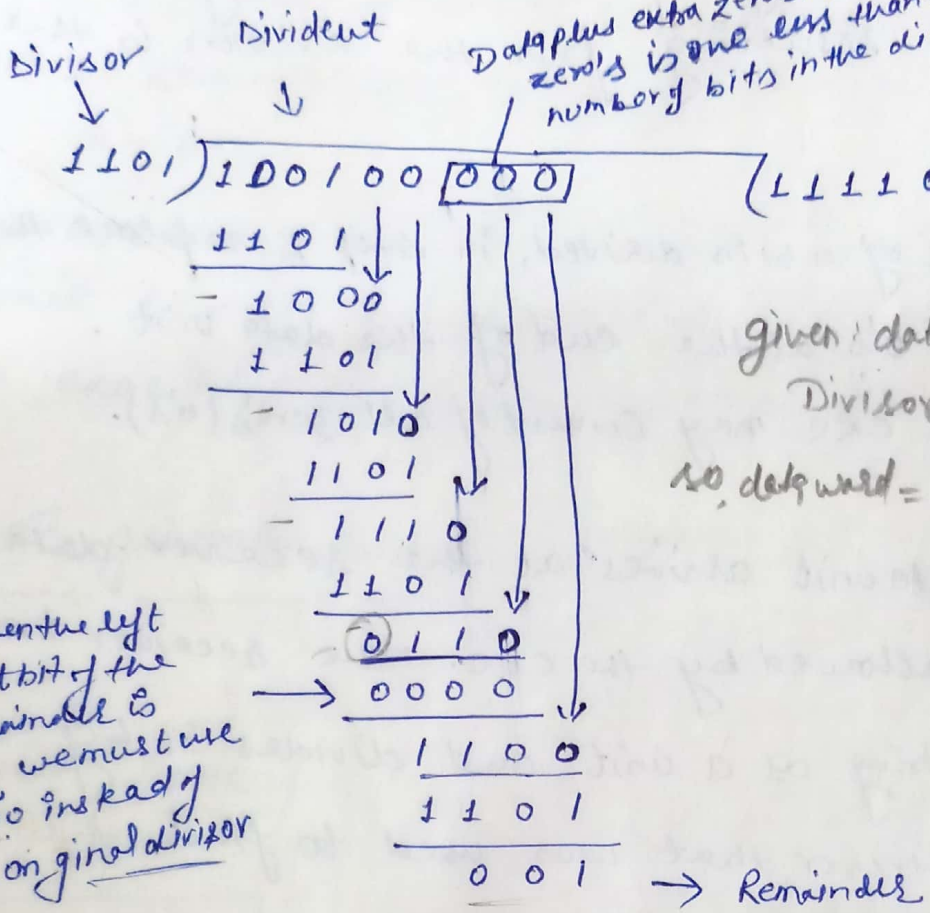
mod-2 addition

Carry is detect.

```
  1101          1011
  1011          1101
  ----          ----
   110           110
```

The CRC Generator :-

A CRC Generator uses modulo-2 division.



given : dataword = 100100
Divisor = 1101
so, dataword = 100100000

The CRC Checker :-

A CRC checker functions exactly like the CRC generator. After receiving the data appended with the CRC, it does the same process of module-2 division. If the Remainder is all zero's (0's), the CRC is dropped and the data is accepted; otherwise, the received stream of bits is discarded and data are resent if fig

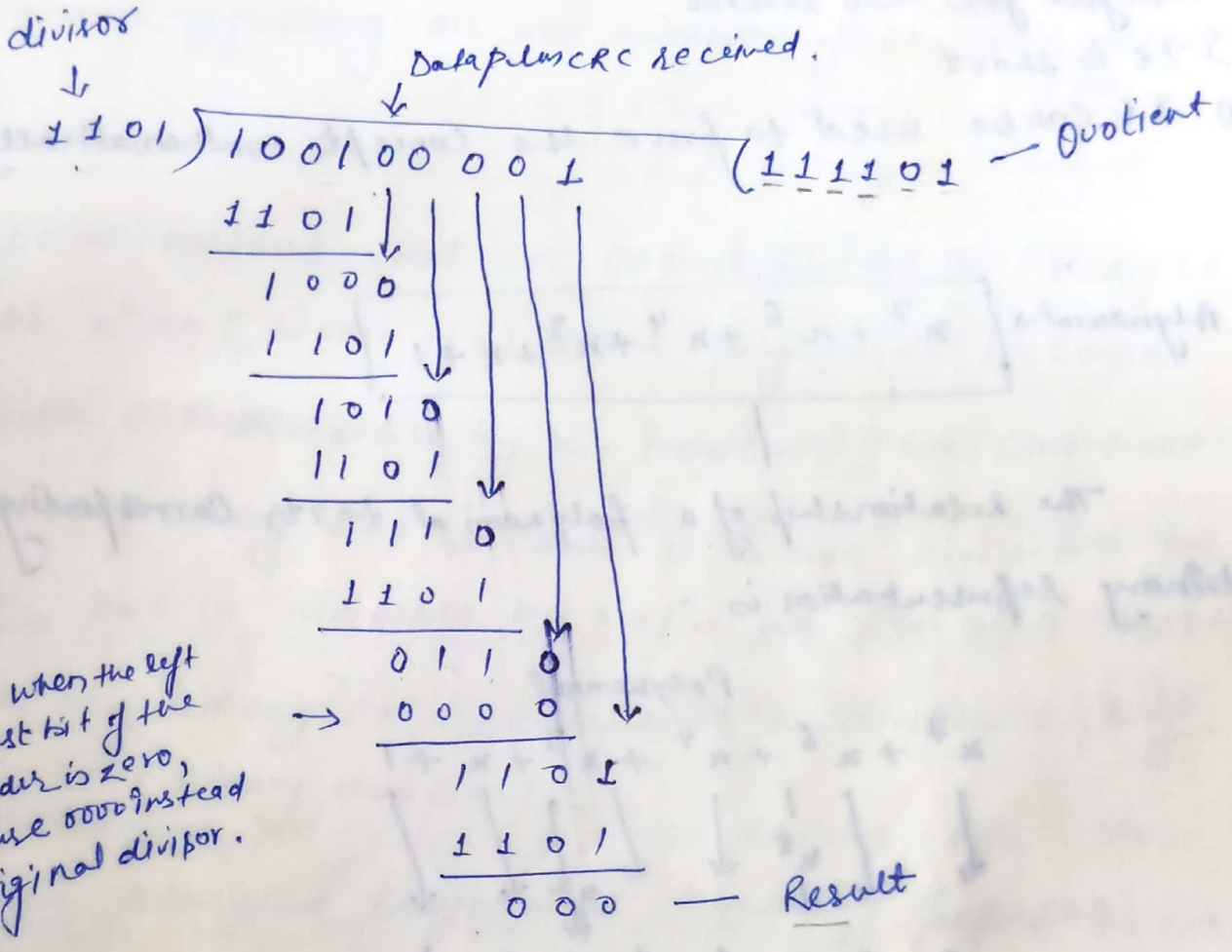
[In fig. shows the same process of division in the receiver. we assume that there is no error. The remainder is therefore all zero's (0's) and the data are accepted.]

data: 10010000

CRC + 001

10010001 → Data plus CRC Received

1101 → Divisor



When the left most bit of the remainder is zero, we must use 0000 instead of the original divisor.

So: The remainder is all zero's, then the data are accepted.

[**Polynomials:** The CRC generator (the divisor) is most often represented not as a string of 1's and 0's, but as an algebric polynomial. The polynomial format is useful for two reasons:

i) It is short

ii) It can be used to prove the concept mathematically.

A Polynomial → $x^7 + x^6 + x^4 + x^3 + x + 1$

The relationship of a polynomial to its corresponding binary representation is:

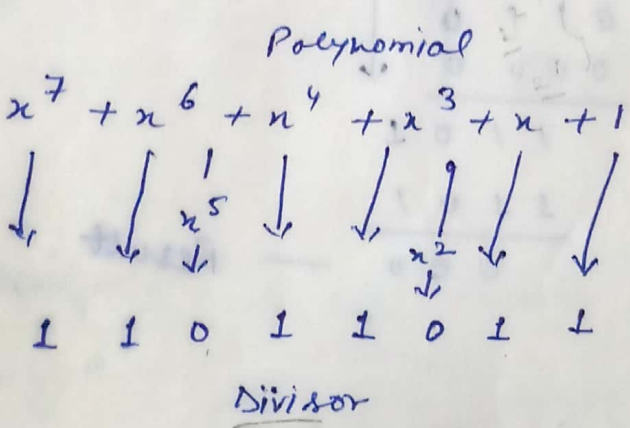


fig: A polynomial representing a divisor

A polynomial should be selected to have at least the following properties:

→ *: It should not be divisible by x

→ *: It should be divisible by x+1]

→ The first condition guarantees that all burst errors of a length equal to the degree of the polynomial are detected.

→ The second condition guarantees that all burst errors affecting an odd number of bits are detected.

Note:

$1 = 1$ $\quad x = 10$

$x+1 = 1+1$

100
10
110

It is obvious that we cannot choose x (binary 10) as x^2+x (binary 110) as the polynomial because both are divisible by x. However, we can choose $x+1$ (binary 11) because it is not divisible by x, but is divisible by $x+1$. We can also choose x^2+1 (binary 101) because it is divisible by $x+1$ (binary division).

The standard polynomials used by popular protocols for CRC generation are as follows:

The numbers 12, 16, & 32 refer to the size of the CRC remainder.

The CRC divisors are 13, 17 & 33 bits respectively..

Standard polynomials:

CRC-12 $\quad x^{12}+x^{11}+x^3+x^2+x+1$

2 (00) 01 10 11 100 101 6 (110)

(2)

Numericals:

(1) The codeword is received as 1100 1001 0101 1. Check whether their are errors in the received code word, is the divisor is 10101. (The divisor corresponds to the generator polynomial.)

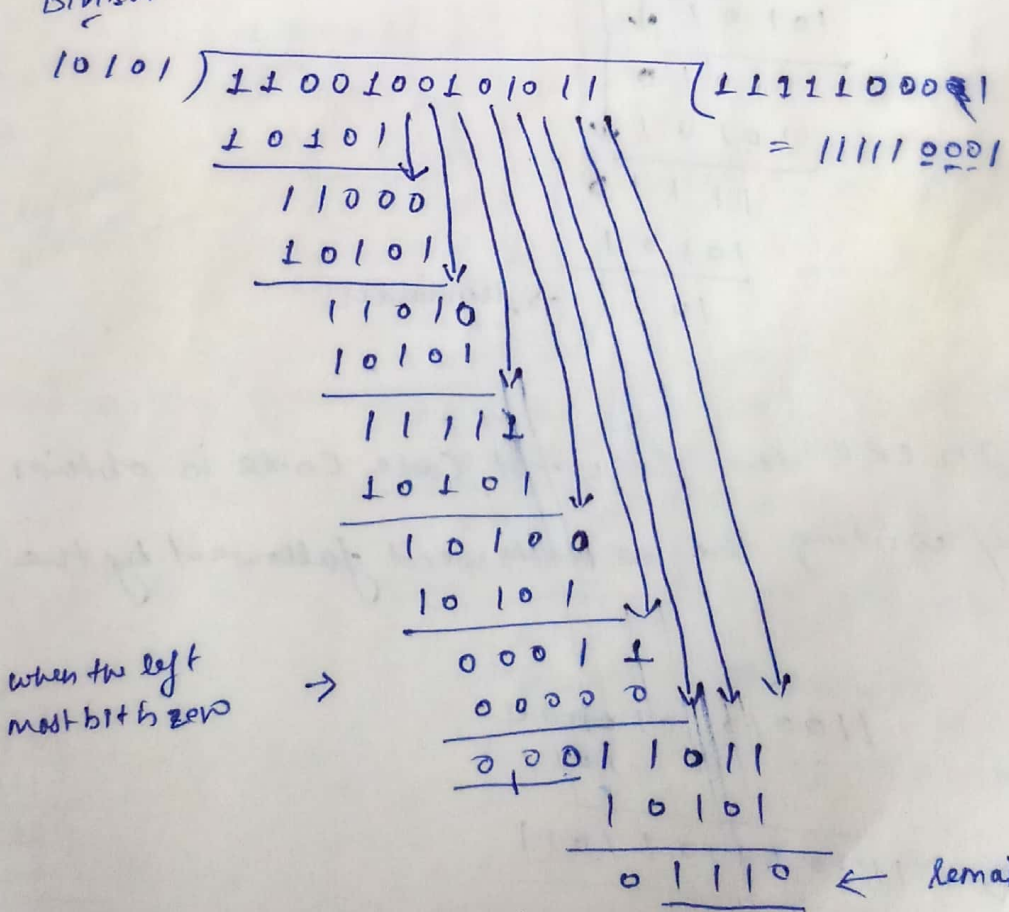
Solution:

(1) The codeword is formatted by adding the divident and the remainder

(2) If there is no remainder then there are no errors. But if there is remainder after division, then there are errors in the received codeword.

Dataword = 1100100101011

Divisor = 10101



Note: The Non zero remainder show that there are errors in the received code word.

H.W

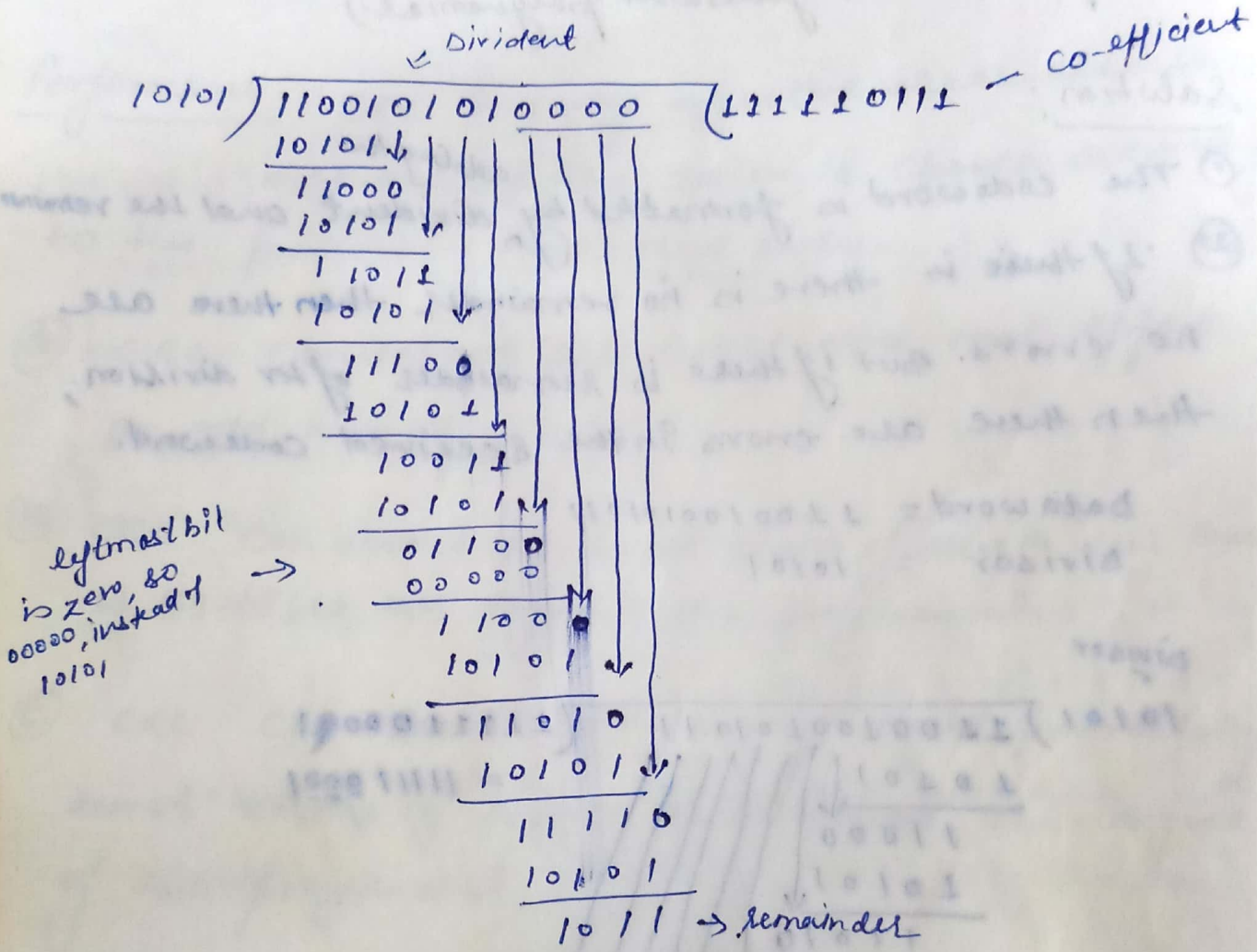
(2) Generate the CRC code for the data word of 11001010.

→ The divisor is 10101. A+B

Solution:

Dataword = 11001010

divisor = 10101 so Dataword = 11001010 10000



Codeword: In CRC the required code code is obtain by writing the data word followed by the remainder

∴ 11001010 10000
1011

Codeword → 11001010 1011

(3) Write the steps to compute the checksum in CRC code.

Sol^{n}: Calculate CRC for the frame 1101011011 and the Generator polynomial $= x^4+x+1$, & also check whether there an error in the received code word.

Sol^{n}: The Generator polynomial actually acts as the divisor in the process of CRC Generation.

20/2 A+B

Data word = 1101011011

Divisor = x^4+x+1 or = x^4+x+1

$\Rightarrow$ | x^3 x^2 |

1 0 0 1 1

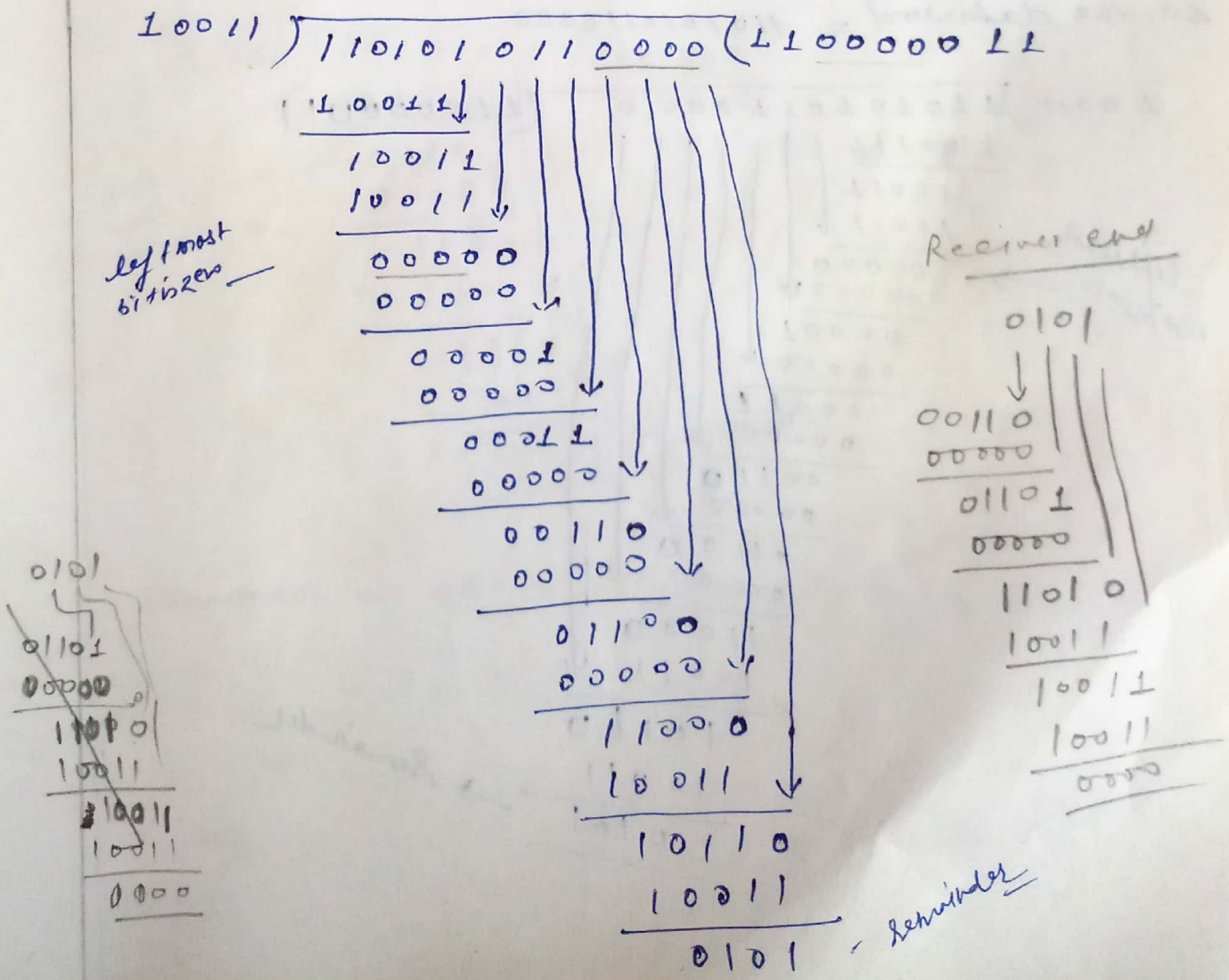
$= 1\times2^4 + 1\times2^1 + 1\times2^0$

$= 16+2+1\times1 = (19)_{20}$

Divisor $\Rightarrow$ 10011

$= (19)_{10} = (10011)_2$

So Data word = 1101011011 0000



4) Checksum :-

→ The error detection method used by the higher layer protocols is called checksum. Like VRC, LRC and CRC; checksum is based on the concept of redundancy.

Checksum Generator: The sender follows these steps:

1. The data unit is divided into k stations, each of n bits.
2. All stations are added together using one's complement to get the sum.
3. The sum is complemented and becomes the checksum.
4. The checksum is sent with the data.

Checksum Checker: The Receiver follows these steps.

1. The unit is divided into k stations, each of n bits.
2. All stations are added together using one's complement to get the sum.
3. The sum is complemented.
4. If the result is zero, the data are accepted otherwise, they are rejected.

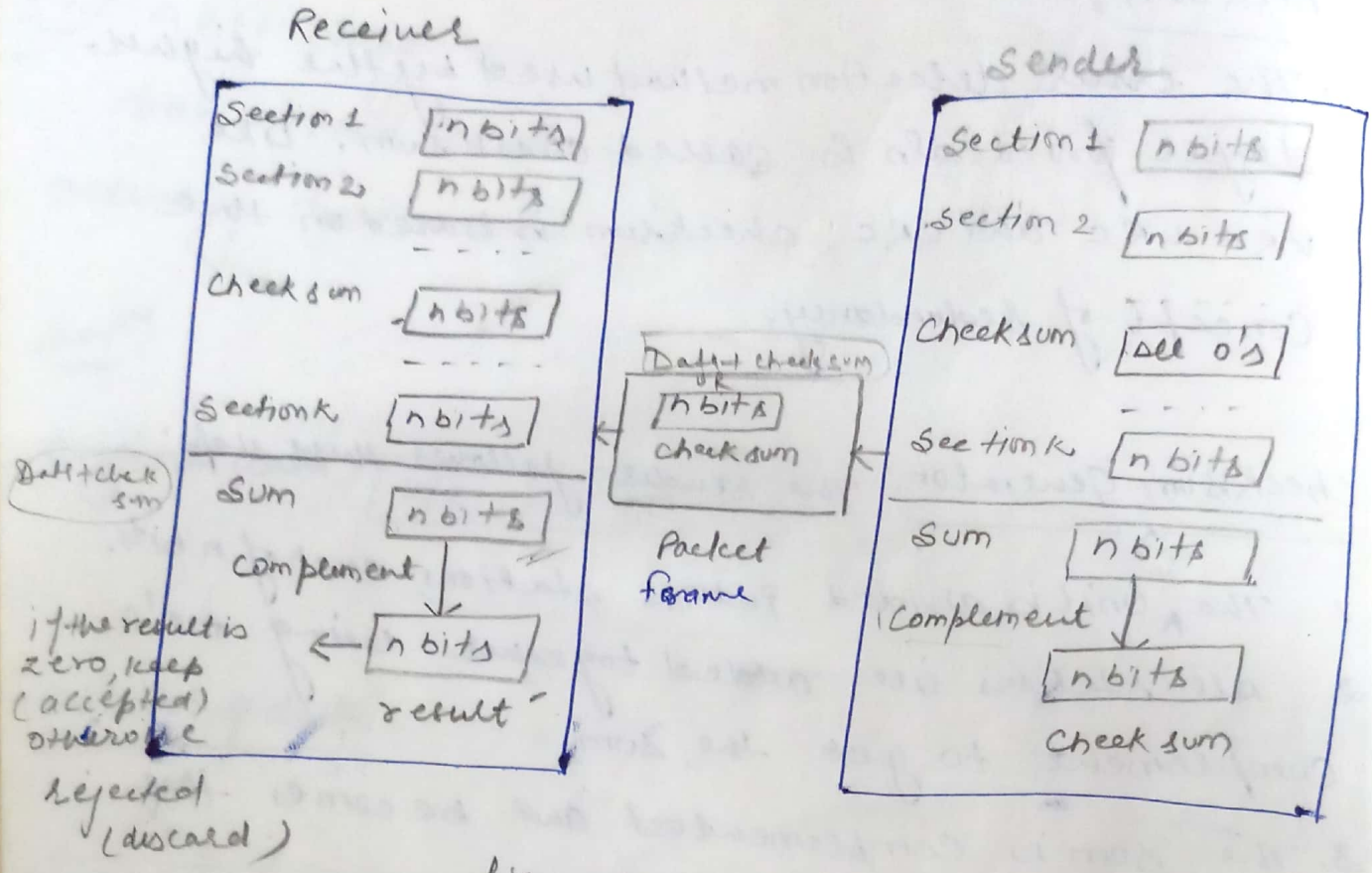


fig: checksum

for example: Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

← 10101001 00111001

the numbers are added using one's complement arithmetic

```
  10101001
  00111001
  --------
  11100010
```

SUM ⇒ 11100010

⇓

1's complement

checksum ⇒ 00011101

```
1- 10101001
2- 00111001
  --------
  01100010
+ 1
  --------
  01100011
```

```
  00111100
- 10101001
- 10111001
  --------
① 11111110
+ 1
```

The pattern sent is:

← 10101001 00111001 00011101 (Sending side)
Checksum

Now suppose the Receiver receives the pattern sent & there is no error.

← 10101001 00111001 00011101 (Receiving side)

```
  10101001
  00111001
  00011101
 ----------
  11111111
```
SUM

⇓

Complement 00000000 (null) means that the pattern is ok.

Note:
When the receiver adds the three sections together, it will get all 1's, which after complementing, is all 0's and shows that there is no error.
So, the data is accepted.

⇒ Now suppose there is a burst error of length five that affects four bits.

10101111 11111001 00011101

When the receiver adds the three sections together, it gets

```
  1 10 10 10 1  1
  10101111
  11111001
  00011101
 ----------
  11000101   Result
 ----------
         1   Carry
 ----------
  11000110
```

Sum 11000110

complement 00111001 (there is not all zero, so the discarded. (Corrupted bits (data).

Adding in one's Complement:

Carry

First no. 1 1 1 1 1 1 1 1
10000101010000 11

Second " 00101011110101 01

result : 10110001000 11000

Adding in one's Complement with Carry from the last Column

Carry

First number 1 1 1 1 1 1 1
10000101010000 11

Second " 10101011110101 01

Result 00110001010110 00

1

final Result 00110001010110 01

Adding a number to its Complement :-

+A = 1010101010

-A = 0101010101

-0 = 1111111111 ⇒ complement

0 = 0000000000 ⇒ All zero then data is correct & accepted.

Error Correction:

Hamming code: We have examined the number of bits required to cover all of the possible single bit error states in a transmission. But how do we manipulate those bits to discover which state has occured? A technique developed by R.W. Hamming provides a ~~pair~~ practical solution.

Positioning the Redundancy Bits: The Hamming code can be applied to data units of any length and uses the relationship between data and redundancy bits.

for example: a seven-bit ASCII code requires four(4) redundancy bits that can be added to the end of the data unit or interspersed with the original data bits.

The bits are placed in positions 1, 2, 4 & 8 (the positions in an 11 bit sequence that are powers of 2) ($2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$). we refer to these bits as r_1, r_2, r_3, r_4 & r_8.

n-2017-12 numericals:

What is Hamming code? Calculate the hamming code for following message string: 1100101 with each and every step explained clearly

Sol: Bit stream (string) = 00110100011

Hamming Code 11000101100

redundant bit:

$r_1 = 0$, $r_4 = 1$

$r_2 = 0$ $r_8 = 0$

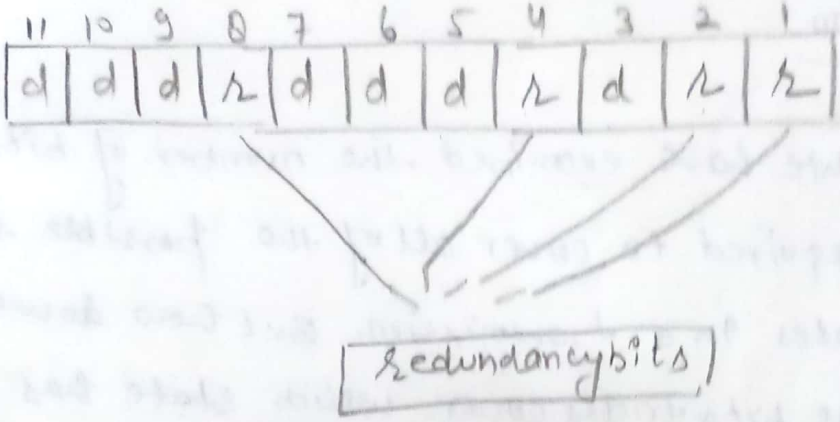


fig: Positions of redundancy bits in Hamming code

In the Hamming code, each r bit is the VRC bit for one combination of data bits. The combinations used to calculate each of the four r values for a seven-bit data sequence are as follows:

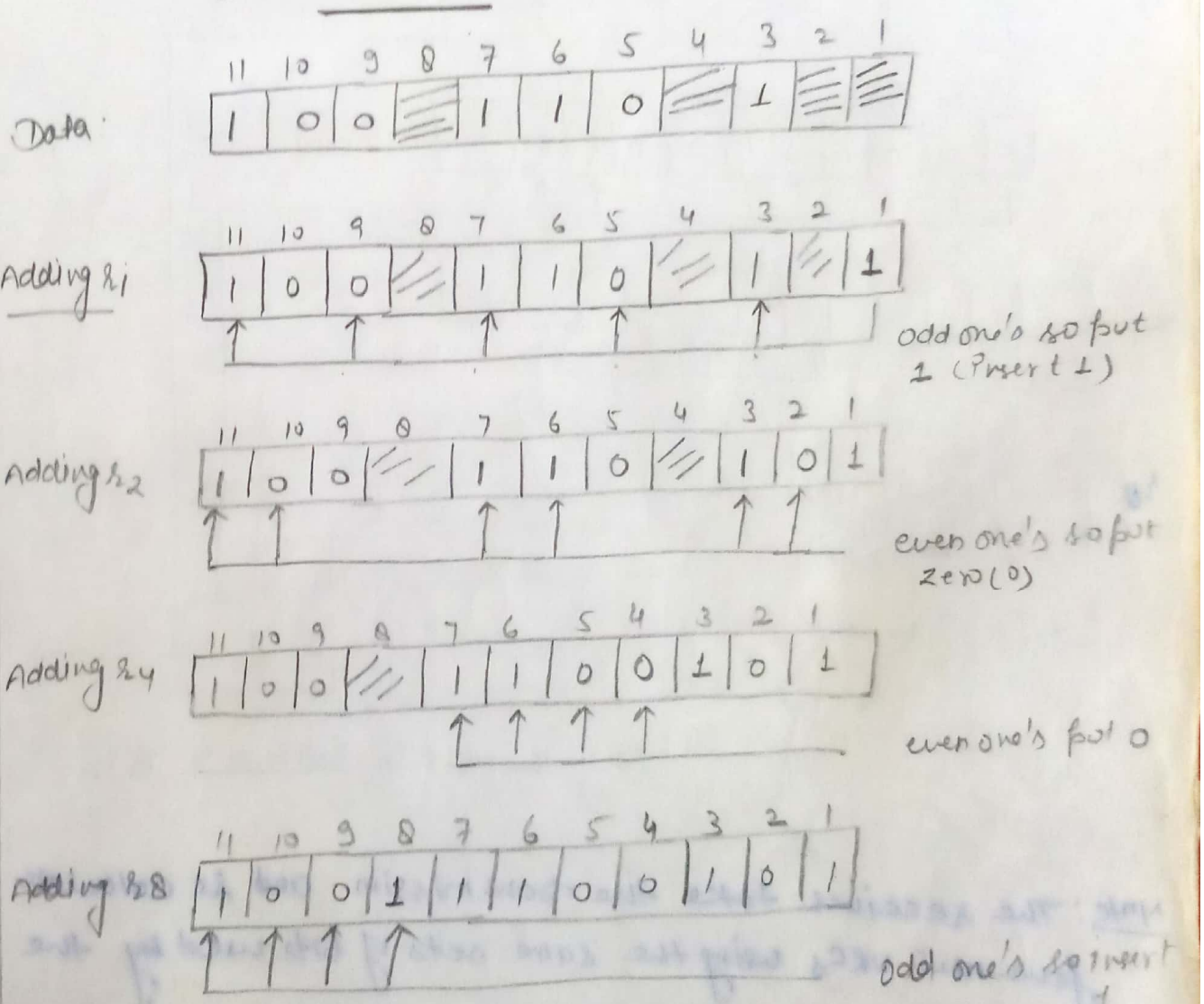
- r_1 : bits 1, 3, 5, 7, 9, 11
- r_2 : bits 2, 3, 6, 7, 10, 11
- r_4 : bits 4, 5, 6, 7,
- r_8 : bits 8, 9, 10, 11, 12 13 14 15

Calculating the r values:

In the first step, we place each bit of the original character in its appropriate position in the 11-bit unit. In the subsequent (Next) steps, we calculate the even parities for the various bit combinations. The parity value for each combination is the value of the corresponding r bit.

for example:, the value of r_1 is calculated to provide even parity for a combination of bits 3, 5, 7, 9 & 11. The value of r_2 is calculated to provide even parity with bits 3, 6, 7, 10, & 11, and so on. The final 11-bit code is sent through the transmission line.

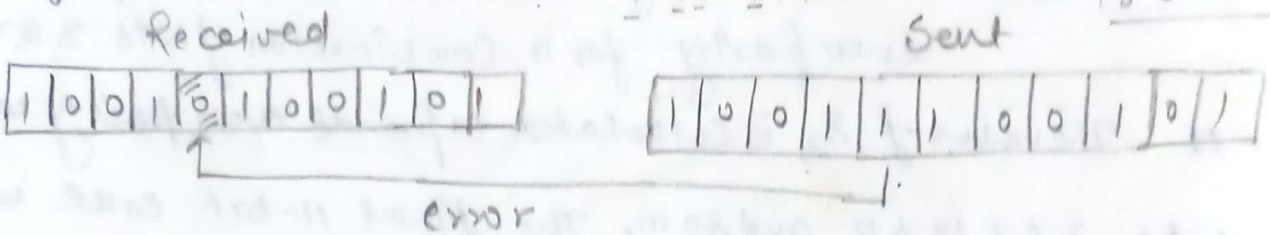
Data: 1001101

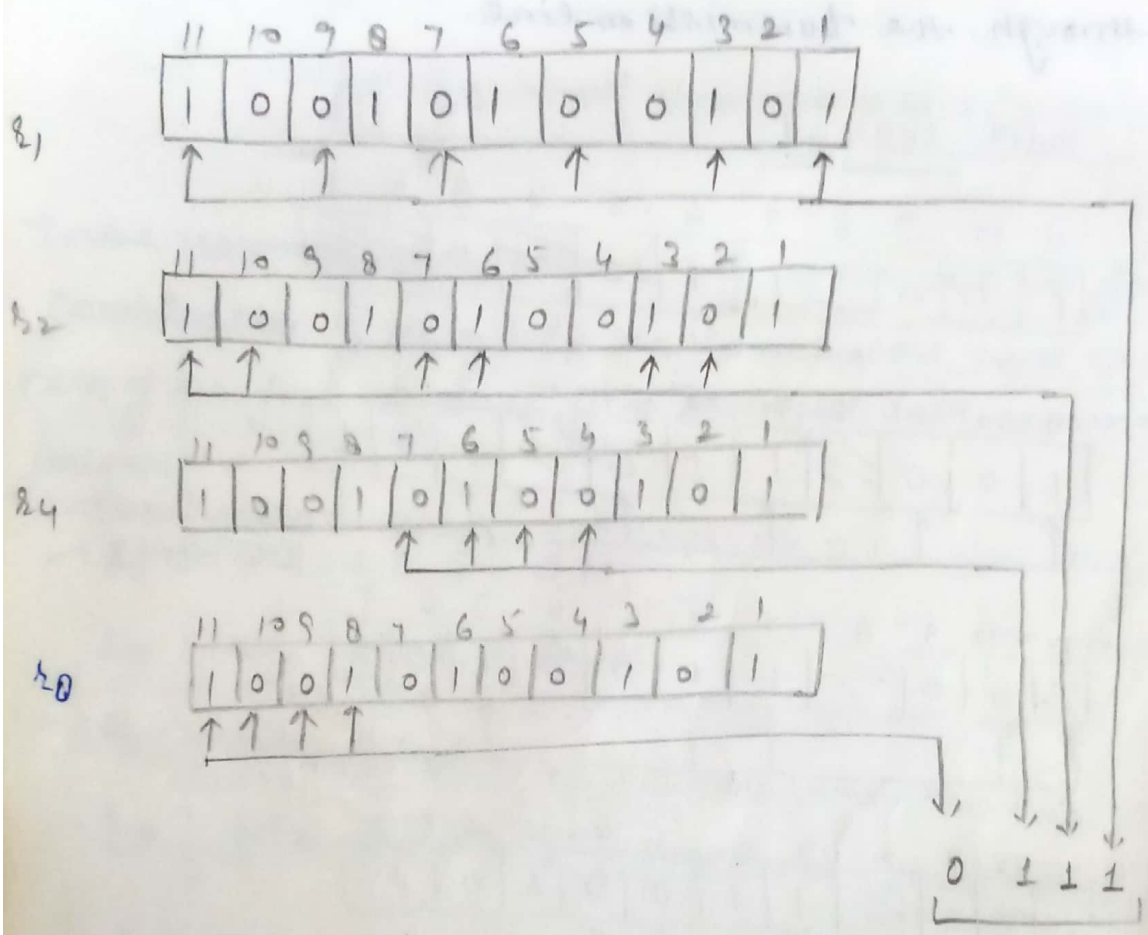


so the code is sent: 10011100101

If Receiver side Receive 10011100101 same & r_1, r_2, r_4, r_8. Contains all 1's is even then accepted, otherwise error correction.

Single bit error:





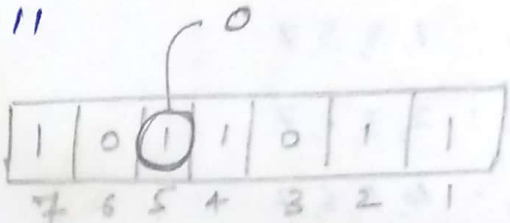
Now $(0111)_2 = (7)_{10}$, the bit in position 7 is in error.

Note: The receiver take the transmission and he calculate, four new CRCs using the same sets of bits used by the sender plus the relevant parity (r) bit for each set. Then the assembles the new parity values into a binary number in order of r position (r_8, r_4, r_2, r_1). The binary number $(0111)_2$ $(7)_{10}$ in decimal, which is precise location of the bit in error. Once the bit is identified, the receiver can reverse its value and correct the error.

Numerical (1) If the seven bit Hamming codeword received by a receiver is 1011011. Assuming the even parity state whether the received codeword is correct or wrong. If wrong locate the bit in error.

Sol:

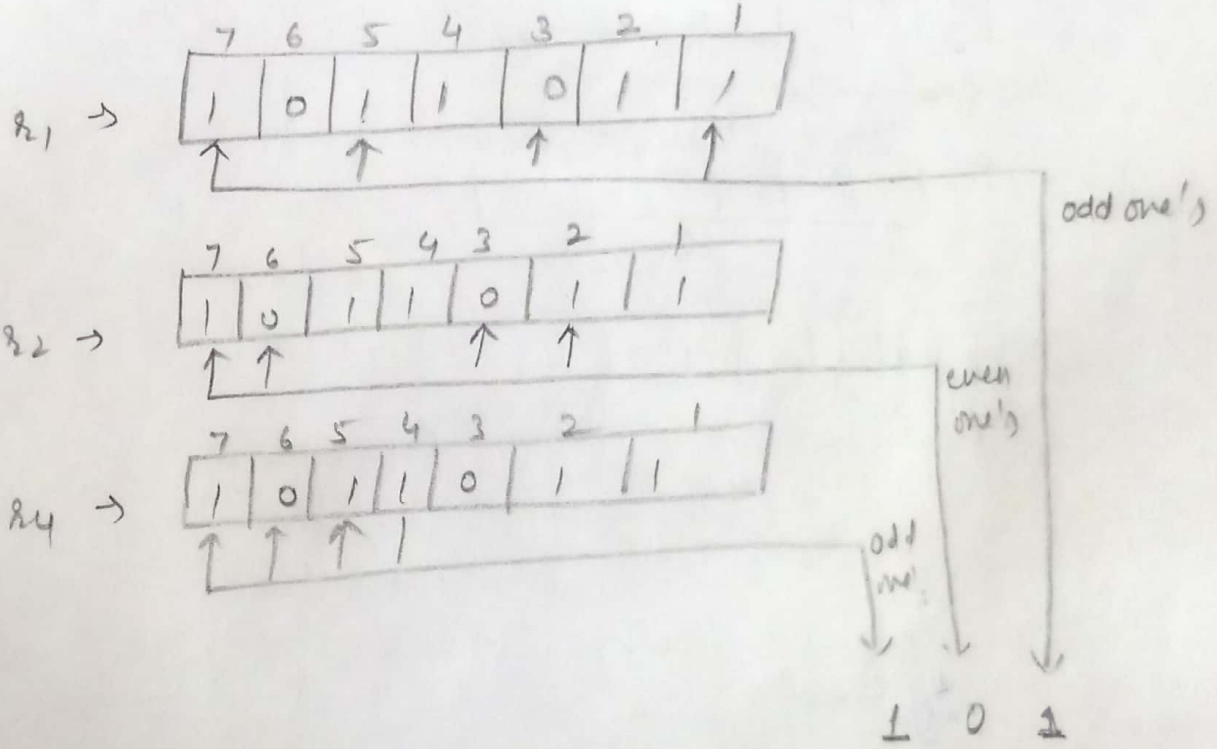
Bit received 1 0 1 1 0 1 1



R1 : 1, 3, 5, 7

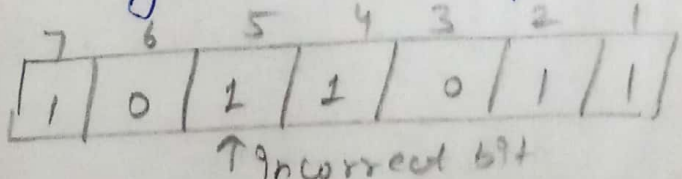
R2 : 2, 3, 6, 7

R4 : 4, 5, 6, 7



Now $(101)_2 = (5)_{10}$

Hence bit 5 of the transmitted codeword is in error



Now correct the error: Invert the Incorrect bit to obtain the correct codeword as follows: Correct Codeword = 1 0 0 1 0 1 1

Numerical 2: A seven bit Hamming code received as 1110101, what is the correct code?

Sol:

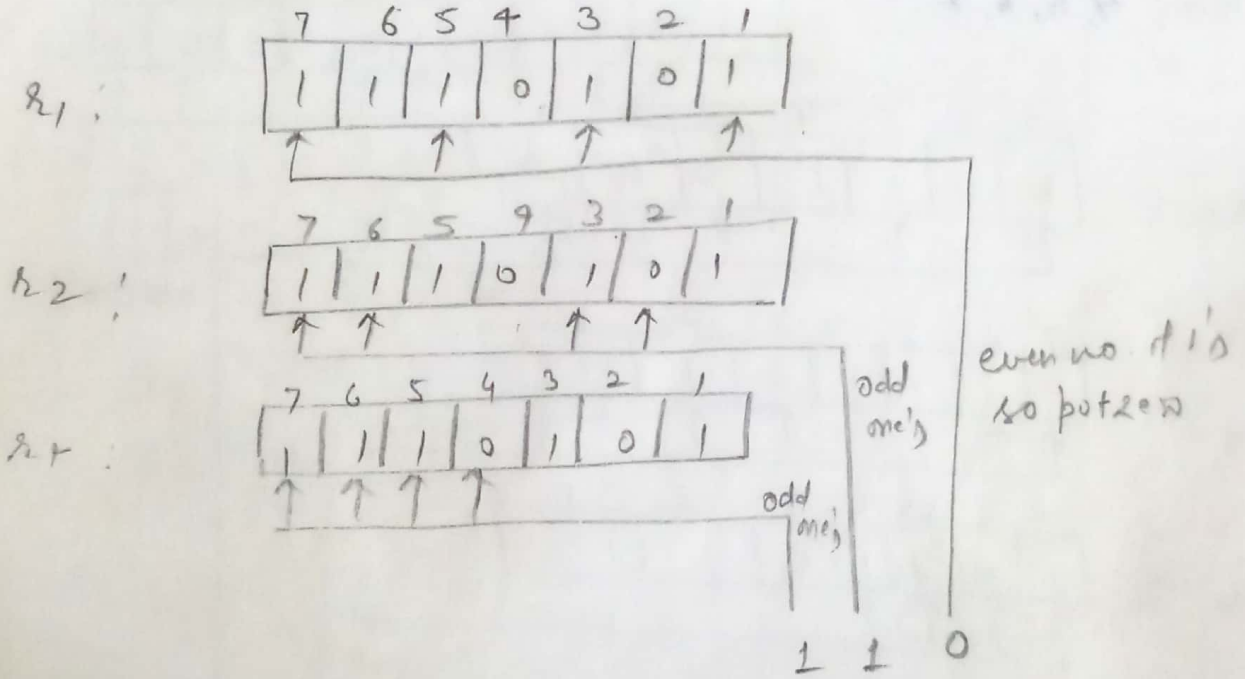
Received Codeword

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

R_1 : 1, 3, 5, 7

R_2 : 2, 3, 6, 7

R_4 : 4, 5, 6, 7



Now $(110)_2 = (6)_{10}$

So 6 bit in the received codeword is incorrect. So invert it.

1	1	1	0	1	0	1

↑ incorrect bit

Correct Codeword :

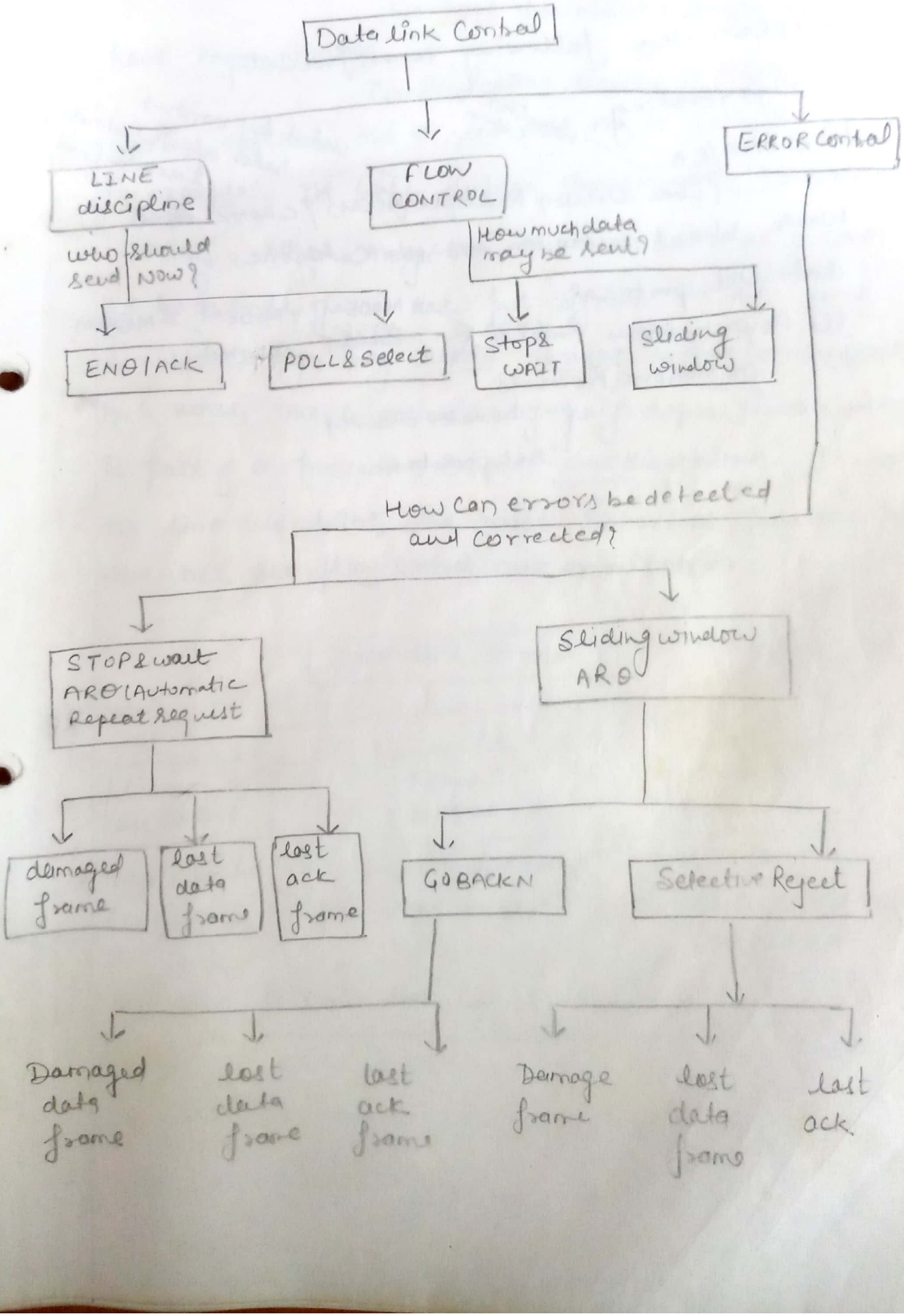
7	6	5	4	3	2	1
1	0	1	0	1	0	1

↑ invert bit

Correct Codeword = 1010101

UNIT-2(C) - 61-80

Data link control :-



Data link control: In the Physical layer of the OSI model, we have transmission but we do not yet have communication.

"Communication requires at least two devices working together, one to send and one to receive." Figure

for example: In half-duplex transmission, it is essential that only one device transmit at a time.

If both ends of the link put signals on the line simultaneously, they collide, leaving nothing on the line but noise. (to come into collision - to clash) The co-ordination of half-duplex transmission is part of a procedure called "line discipline". In addition to line discipline, the most important functions in the DLL are flow control and error control.

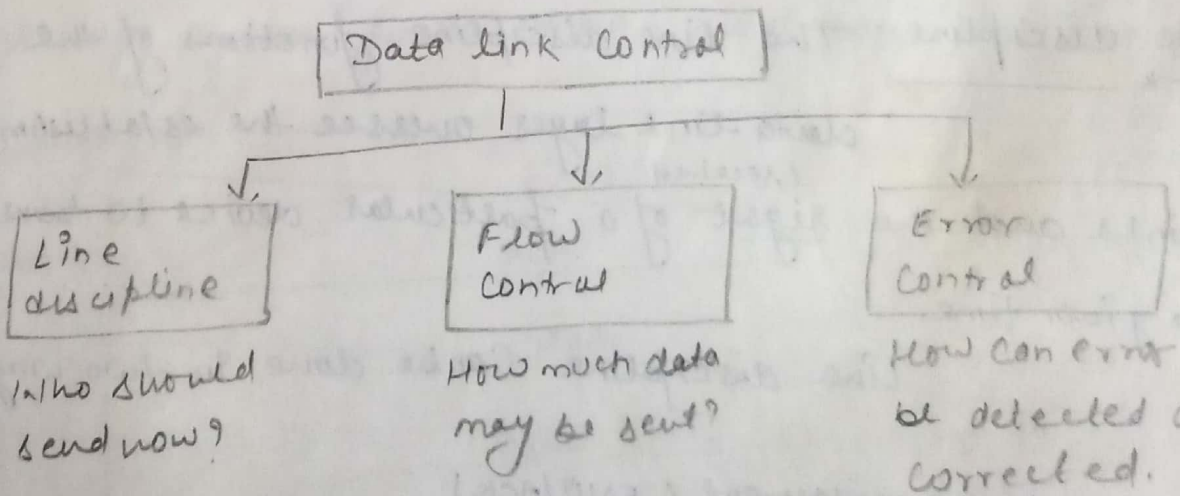


fig Data link layer function

→ Line discipline coordinates the link systems. It determines which devices can send and when it can send.

→ Flow control coordinates the amount of data that can be sent before receiving acknowledgement. It also provides the receiver's acknowledgement of frame. received intacts (entire, untouched, complete), and so is linked to error control.

→ Error control: means error detection and correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and co-ordinates the retransmission of those frames by the sender.

(1) Line discipline: The line discipline functions of the data link layer oversee the establishment of links and the right (receiver) of a particular device to transmit at a given time.

(Line discipline can be done in two ways:

(i) Enquiry/acknowledgement (ENQ/ACK)

(ii) Poll/select

The first method is used in peer to peer communication, the second method is used in primary-secondary communication.)

(i) ENQ/ACK — In both half-duplex and full duplex transmission the initiating device establishes the session. In half-duplex, the initiator then sends its data while the responder waits. The responder may take over the link when the initiator is finished or has requested a response. In full duplex, both devices can transmit simultaneously once the session has been established.

How it works: The initiator first transmits a frame called an enquiry (ENQ) asking if the receiver is available to receive data. The receiver must answer either with an (ACK) frame if it is ready to receive or with a negative acknowledgement (NAK) frame if it is not. (If the response is positive, then initiator is free to send its data) once all of its data have been transmitted, the sending system finishes with an end of transmission (EOT) frame.

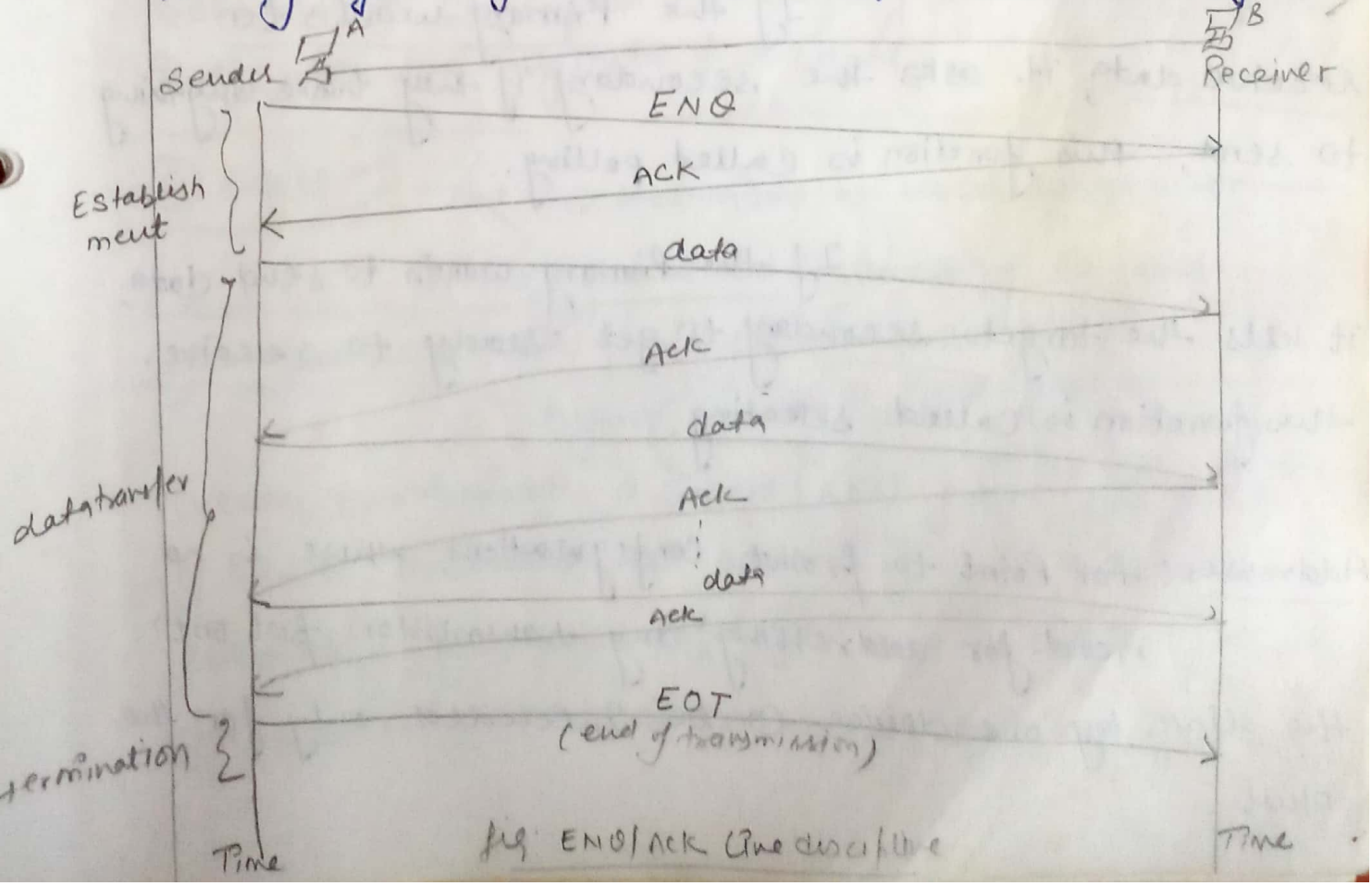


fig: ENQ/ACK (one direction)

(ii) Poll/select :- The Poll/select method of line discipline (डिसिप्लीन) works with topologies where one device is designated as a primary station and other devices are secondary stations.

How it works : Whenever a multipoint link consists of a primary device and multiple secondary devices using a single transmission line, all exchanges must be made through the primary device even when the ultimate destination is a secondary device.

The primary device controls the link; the secondary devices follows its instructions. It is up to the primary to determine which device is allowed to use the channel at a given time. The primary, therefore, is always the initiator of a session.

If the Primary wants to receive data, it asks the secondary if they have anything to send, this function is called polling.

If the Primary wants to send data, it tells the targets secondary to get ready to receive. This function is called selecting.

Addresses : For Point to point configurations, there is no need for addressing, any transmission put on to the link by one device can be intended only for the other.

for the primary device in a multipoint topology to be able to identify and communicate with a specific secondary device, however there must be an addressing convention. for this reason, every device on a link has an address that can be used for identification.

Poll/select protocals identify each frame as being either to or from a specific device on the link. Each secondary device has an address that differentiates it from the others.

In any transmission, that address will appear in a specified portion of each frame called an address field or header depending on the protocol. If the transmission comes from the primary device, the address indicates the ~~receiv~~ recipient of the data. If the transmission comes from a secondary device, the address indicates the originator of the data.

(a) Select :- The select mode is used whenever the primary device has something to send. (Remember that the primary controls the link).

Before sending data, the primary creates and transmits a Select (SEL) frame, one field of which includes the address of the intended secondary. Multipoint topologies use a single link for several devices, which means that any frame

on the link is available to every device. As a frame makes its way down the link, each of the secondary devices checks the address field. only when a device recognizes its own address does it open the frame and read the data.

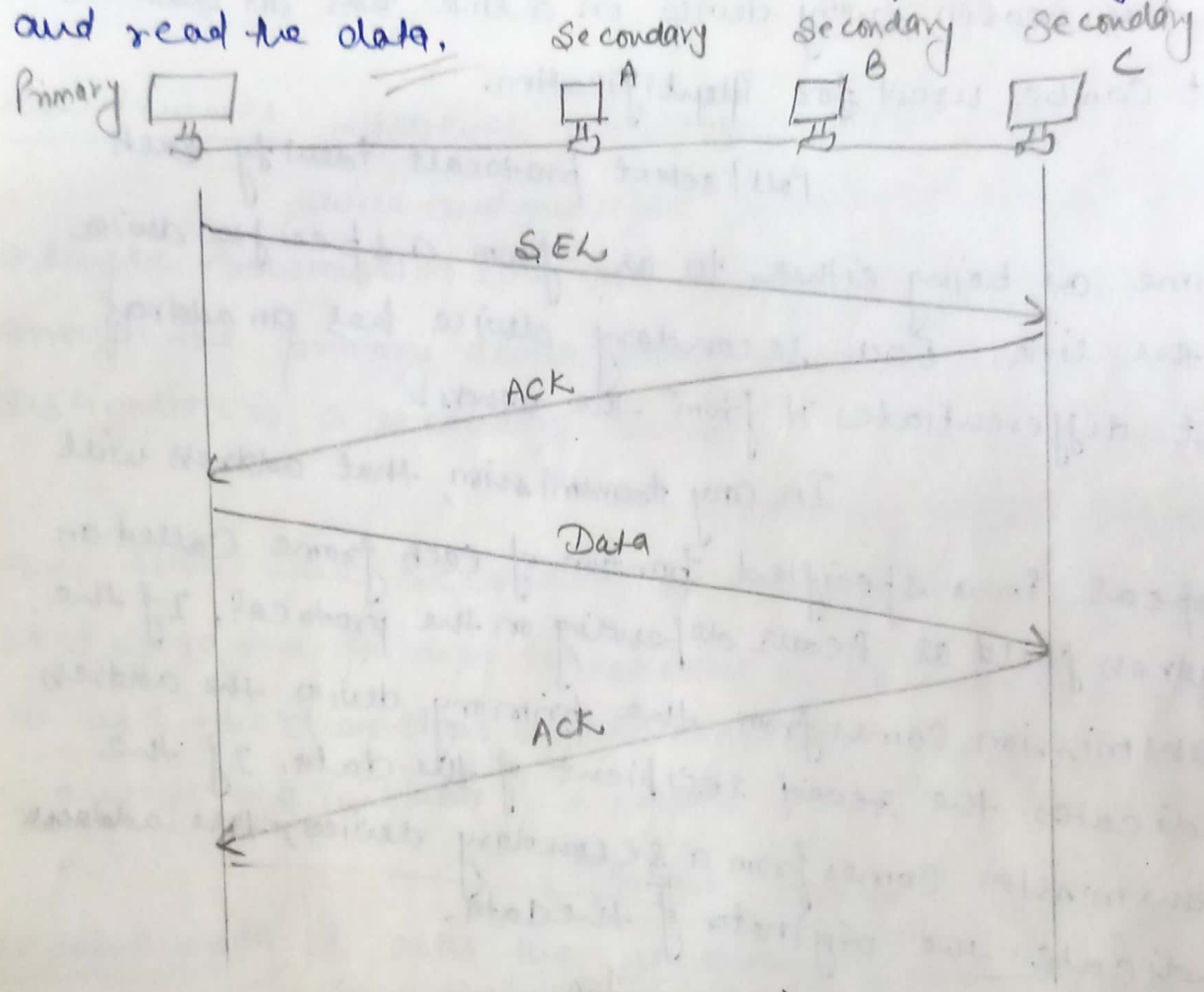


fig: select

(b) **Poll:** When the primary is ready to receive data, it must must ask (Poll) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has noting to send. If the response is negative (a NAK frame), the primary then polls the next

secondary in the same way until it finds one with data to send. When the response is positive (a data frame) the primary reads the frame and returns an acknowledgement (ACK frame) verifying its receipt.

The secondary may send several data frames one after the other, or it may be required to wait for an ACK before sending each one, depending on the protocol being used.

There are two possibilities for terminating the exchange: either the secondary sends all its data, finishing with an end of transmission (EOT) frame, or the primary says, "Time's up". Which of these occurs depends on the protocol and the length of the message. Once a secondary has finished transmitting, the primary can poll the remaining devices.

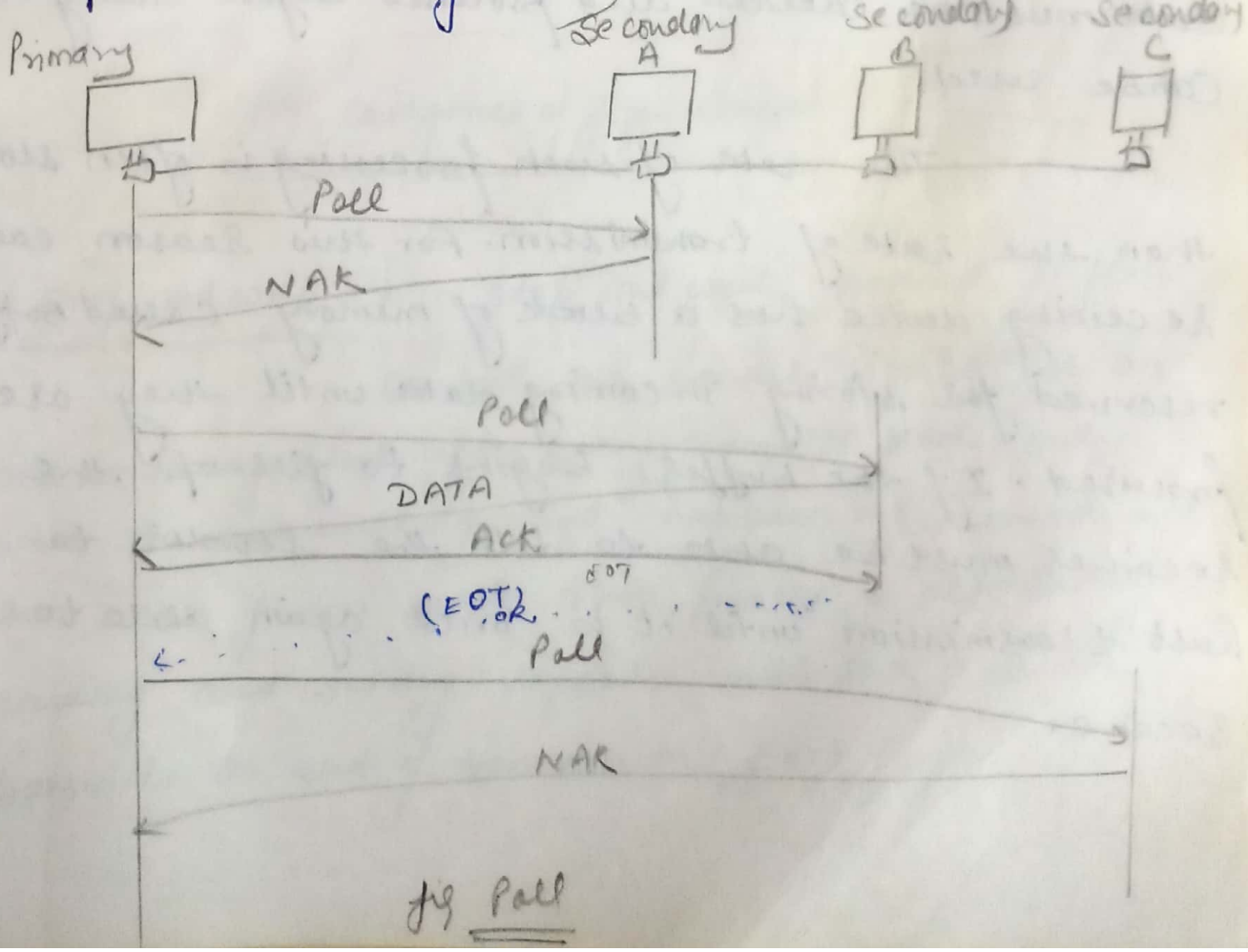


fig Poll

(2) Flow control: The second aspects of data link control is flow control. In most protocols, flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the Receiver. The flow of data must not be allowed to overwhelm the (Over power) Receiver.

Any receiving device has a limited speed at which it can process incoming data. The receiving and a limited amount of memory in which to store incoming data. The receiving device must be able to inform the sending device before these limits are reached and to request that the transmitting device send fewer frames or stop temporarily. Incomming data must be checked and processed before they can be used.

The rate of such processing is often slower than the rate of transmission. For this reason, each receiving device has a block of memory, called buffer, reserved for storing incoming data until they are processed. If the buffer begins to fill-up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

"Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgement."

Two methods have been developed to control the flow of data across communications links:

(i) Stop and wait
(ii) Sliding window.

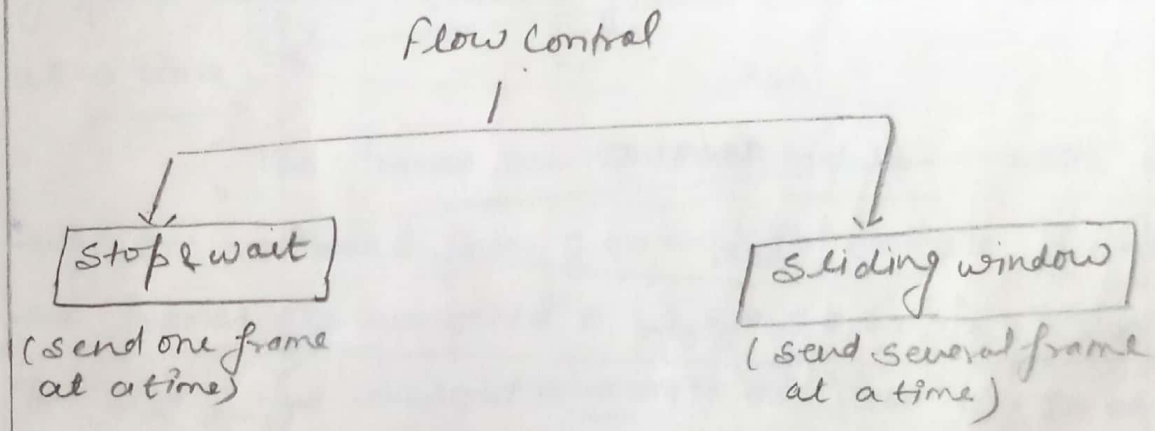


fig: categories of flow control

(i) Stop and wait: In stop and wait method of flow control, the sender waits for an acknowledgement after every frame it sends. only when an acknowledgement has been received is the next frame sent. This process is alternately sending and waiting repeats until the sender transmits an end of transmission (EOT) frame.

"In the stop and wait method of flow control, the sender sends one frame and waits for an acknowledgement before sending the next frame".

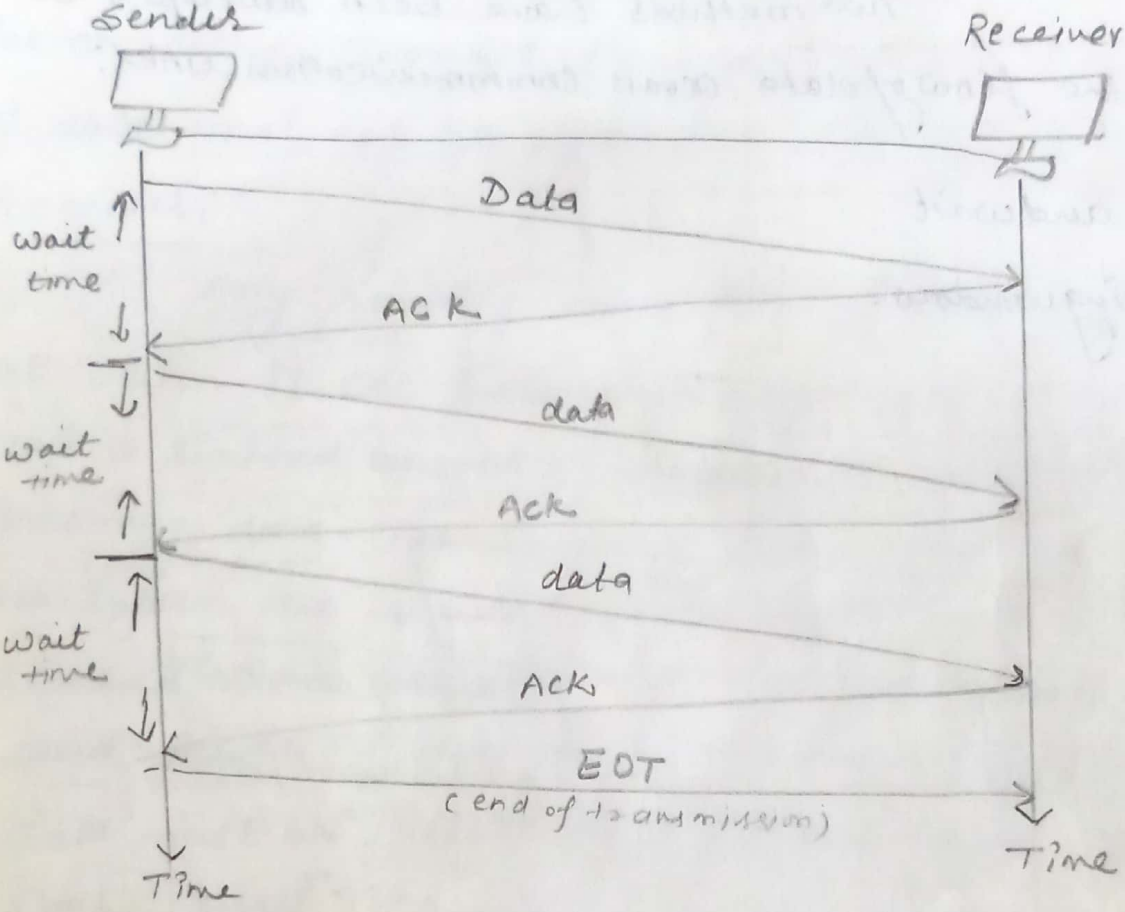


fig: stop and wait

The advantage of stop and wait is simplicity: each frame is checked and acknowledged before the next frame is sent.

The disadvantage is inefficiency: stop and wait is slow. Each frame must travel all the way to the receiver and an acknowledgement must travel all the way back before the next frame can be sent. If the distance between devices is long, the time spent waiting for ACK's b/w each frame can add significantly to the total transmission time.

(ii) Sliding window: In the sliding window method of flow control, the sender can transmit several frames before needing an acknowledgement. Frames can be sent one right after another. Meaning that the link can carry several frames at once and its capacity can be used efficiently. The Receiver acknowledge only some of the frames, using a single Ack to confirm the receipt of multiple data frames.

("In the sliding window method of flow control, several frames can be in ~~transm~~ transit at a time.")

(The frames are numbered modulo-n, which means they are numbered from 0 to n-1. for example, if n=8, the frames are numbered 0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1... The size of the window is n-1 (in this case, 7). In other words the window cannot cover the whole module (0-6) (8 frames); it covers one frame less.

The window can hold n-1 frames at either end; therefore, a maximum of n-1 frames may be sent before an acknowledgement is required.)

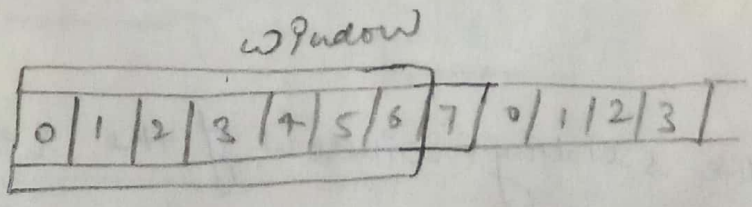


fig: Sliding window

An example: A sample transmission that uses sliding window flow control with a window of seven frames.

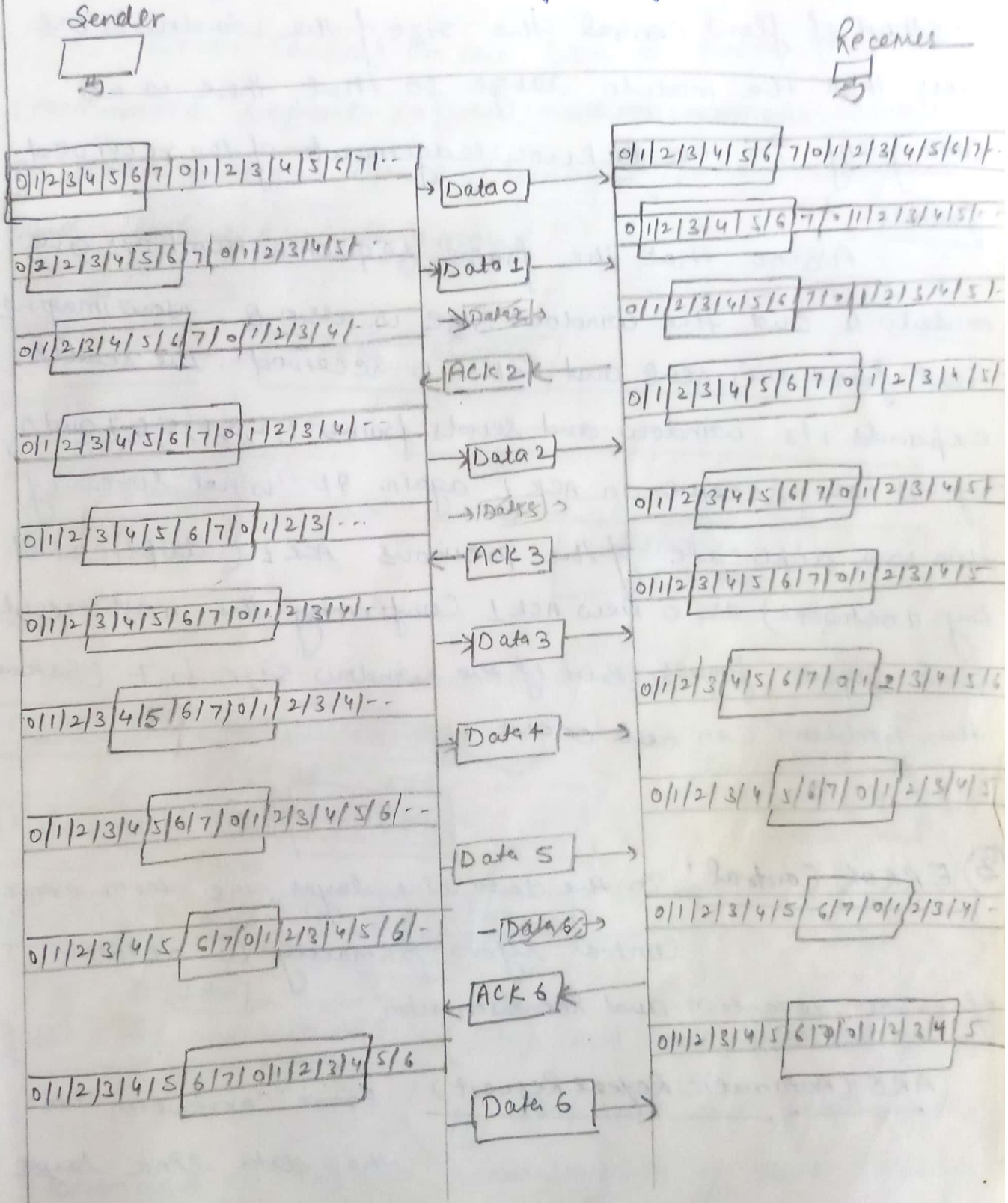
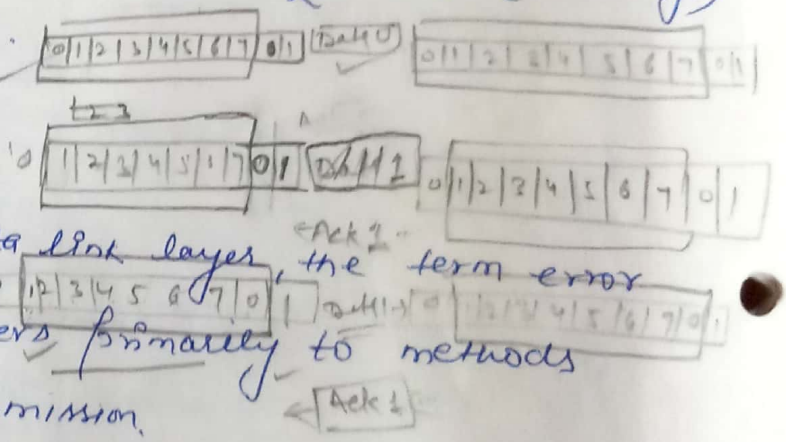


Fig. Example of sliding window

(Note: (More about window size): In the sliding window method of flow control, the size of the window is one less than the modulo range so that there is no ambiguity (doubtfulness, uncertain) in the acknowledgement of the received frames.)

Assume that the frame sequence numbers are modulo-8 and the window size is also 8. Now imagine that frame 0 is sent and ACK 1 is received. The sender expands its window and sends frames 1, 2, 3, 4, 5, 6, 7 and 0, if it now receive an ACK 1 again, it is not sure if this is a duplicate of the previous ACK 1 (duplicated by a network) or a new ACK 1 confirming the most recently sent eight frames. But if the window size is 7 (instead of 8) this problem can not occur.



(2) (ERROR Control): In the data link layer, the term error control refers primarily to methods of error detection and retransmission.

(ARQ (Automatic Repeat Request)): Error correction in the data link layer is implemented simply: any time an error is detected in an exchange, a negative acknowledgment (NAK) is returned and the specified frames are retransmitted.

This process is called ARQ (Automatic Repeat request.

→ ["Error Control in the DLL is based on ARQ (Automatic Repeat request) which means retransmission of data in three cases: damaged frame, lost frame and lost acknowledgment."]

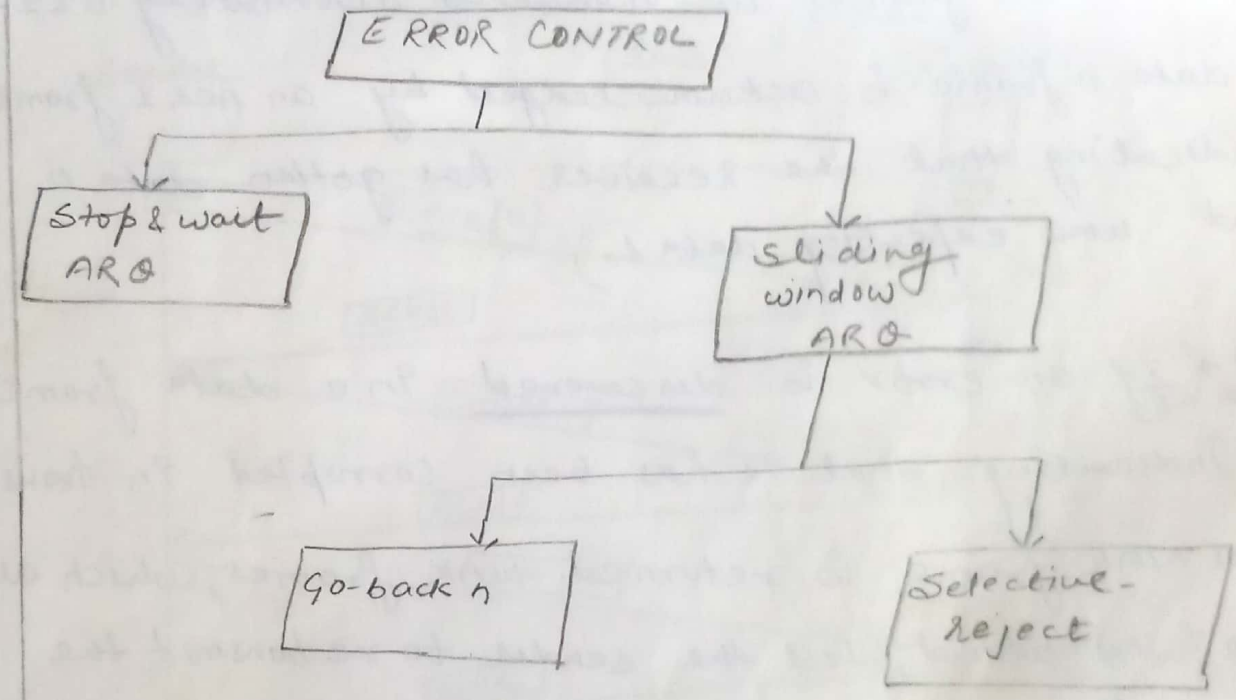


fig: categories of error control

(i) Stop-and-wait ARQ: (Stop and wait ARQ is a form of stop and wait flow control extended to include retransmission of data in case of lost or damaged frames.) For retransmission to work, four features are added to the basic flow control mechanism.

(i) → (The sending device keeps a copy of the last frame transmitted until it receives an acknowledgement for that frame. Keeping a copy allows the sender to retransmit lost or damaged frames until they are received correctly.)

(ii) → (For identification purpose, both data frames and ACK frames are numbered alternately 0 & 1. A data 0 frame is acknowledged by an ACK 1 frame, indicating that the receiver has gotten data 0 and now expecting data 1.)

(iii) → (If an error is discovered in a data frame, indicating that it has been corrupted in transit, a NAK frame is returned. NAK frames, which are not numbered, tell the sender to retransmit the last frame sent.)

IV → (The sending device is equipped with a timer. If an expected acknowledgement is not received with in an allotted time period, the sender assumes that the last data frame was lost in transit and sends it again.)

(a) damaged frame (stop & wait ARQ)
(b) Last data frame (")
(c) Last Acknowledgement frame (stop & wait ARQ)

(a) Damaged frames :- When a frame is discovered by the receiver to contain an error, it returns a NAK frame and the sender retransmits the last frame. →

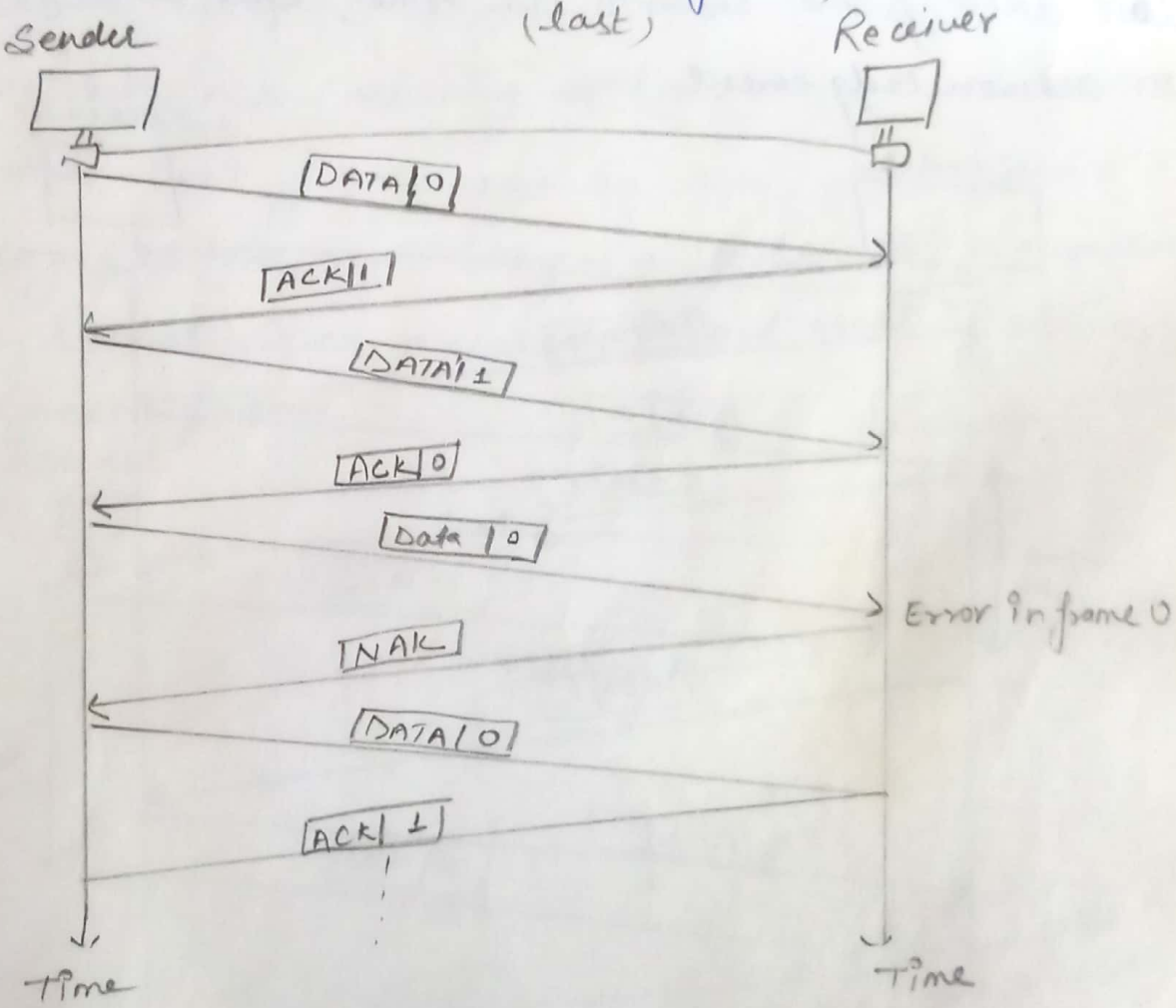


fig: stop & wait ARQ, damaged frame

(b) Lost data frame: (The Sender is equipped with a timer that starts every time a data frame is transmitted, If the frame never makes it to the receiver, the receiver can never acknowledge it, positively or negatively. The sending device waits for an ACK or NAK frame until its timer goes off (time out), at which point it tries again. It retransmits the lost data frame, restarts its timer, and waits for an acknowledgement.)

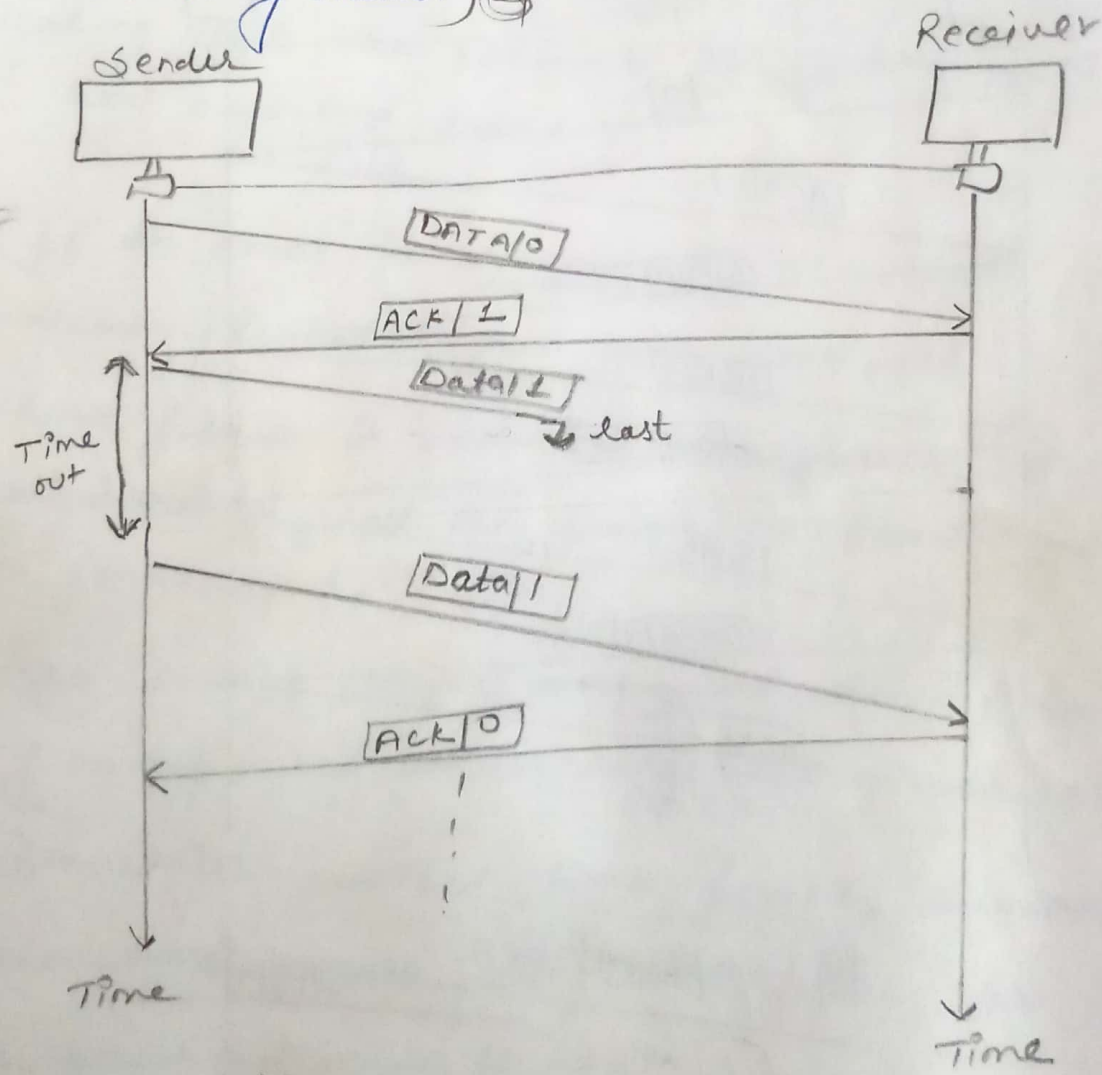


fig Stop & wait ARQ, lost data frame

(c) Lost Acknowledgement: (The data frame has made it to the receiver and has been found to be either acceptable or not acceptable. But the ACK or NAK frame return by the receiver is lost (LOST) in transit. The sending device waits until its timer goes off, then retransmits the data frame. The receiver checks the number of the new data frame. If the lost (LOST) frame was a NAK, the receiver accepts the new copy and returns the appropriate ACK (assuming the copy arrives undamaged) If the lost (LOST) frame was an ACK, the receiver recognizes the new copy as a duplicate, acknowledges its receipt, then discards it and waits for the next frame.)

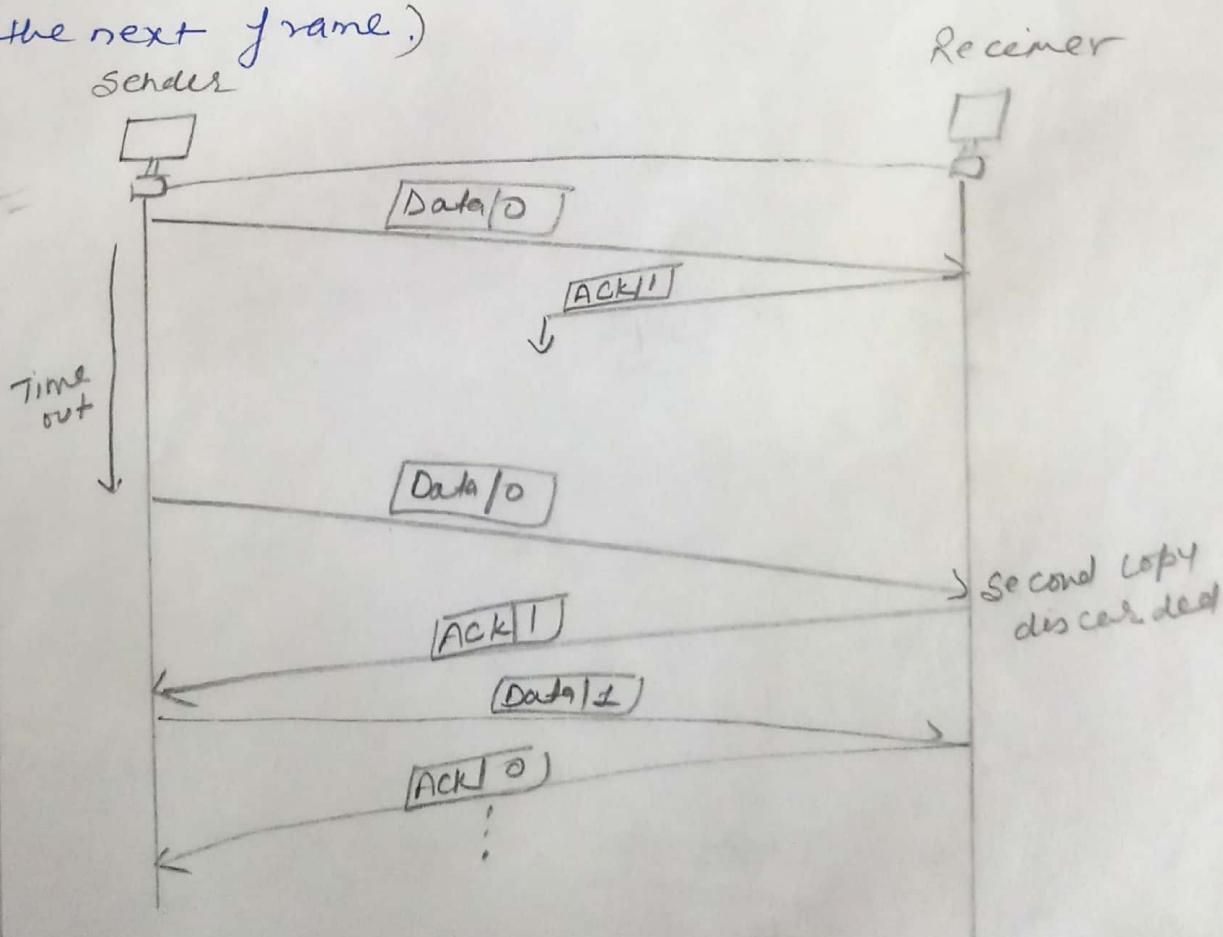


fig: Stop & wait ARQ, lost ACK frame

LALITKUMAR

(81) Sliding window ARQ: Among the several popular mechanisms for continuous transmission error control, two protocols are the most popular: Go back N ARQ and Select Reject ARQ, both Based on Sliding window flow control. To extend sliding window to cover retransmission of lost or damaged frames, three features are added to the basic flow control mechanism:

(i) → The sending device keeps copies of all transmitted frames until they have been acknowledged.

(ii) → In addition to ACK frames, the receiver has the option of returning a NAK frame if the data have been received damaged. The NAK frames tells the sender to retransmit a damaged frame. Because sliding window is a continious transmission mechanism (as opposed to stop & wait), both ACK and NAK frames must be numbered for identification.

ACK frames, carry the number of the next frame expected. NAK frames, on the other hand, carry the number of the damaged frame itself.

(iii) → Like stop and wait ARQ, the sending device in sliding window ARQ is equipped with a timer to enable it to handle lost acknowledgement.

In sliding window ARQ, n-1 frames (the size of the window may be sent before an acknowledgement must be received. If n-1 frames are awaiting acknowledgement, the sender starts a timer and wait's before sending any more. If the alloted time has timout (it finished) with no acknowledgement, the sender assumes that the frames were not received and retransmits one or all of the frames depending on the protocal.

(a) Go Back-n ARQ :- In this sliding window go back-n ARQ method, if one frame is lost or damaged, all frames sent since the last frame acknowledged are retransmitted.

(i) Damaged frame:

What if frame 0, 1, 2 and 3 have been transmitted, but the first acknowledgement received is a NAK 3, ? Remember that a NAK means two things:

(a) A positive ACK of all frames received prior to the damaged frame &
(b) A negative ACK of the frame id indicated. If the first ACK is a NAK 3, it means that data frames 0, 1, 2 were all received in good shape. only frame 3 must be resend.

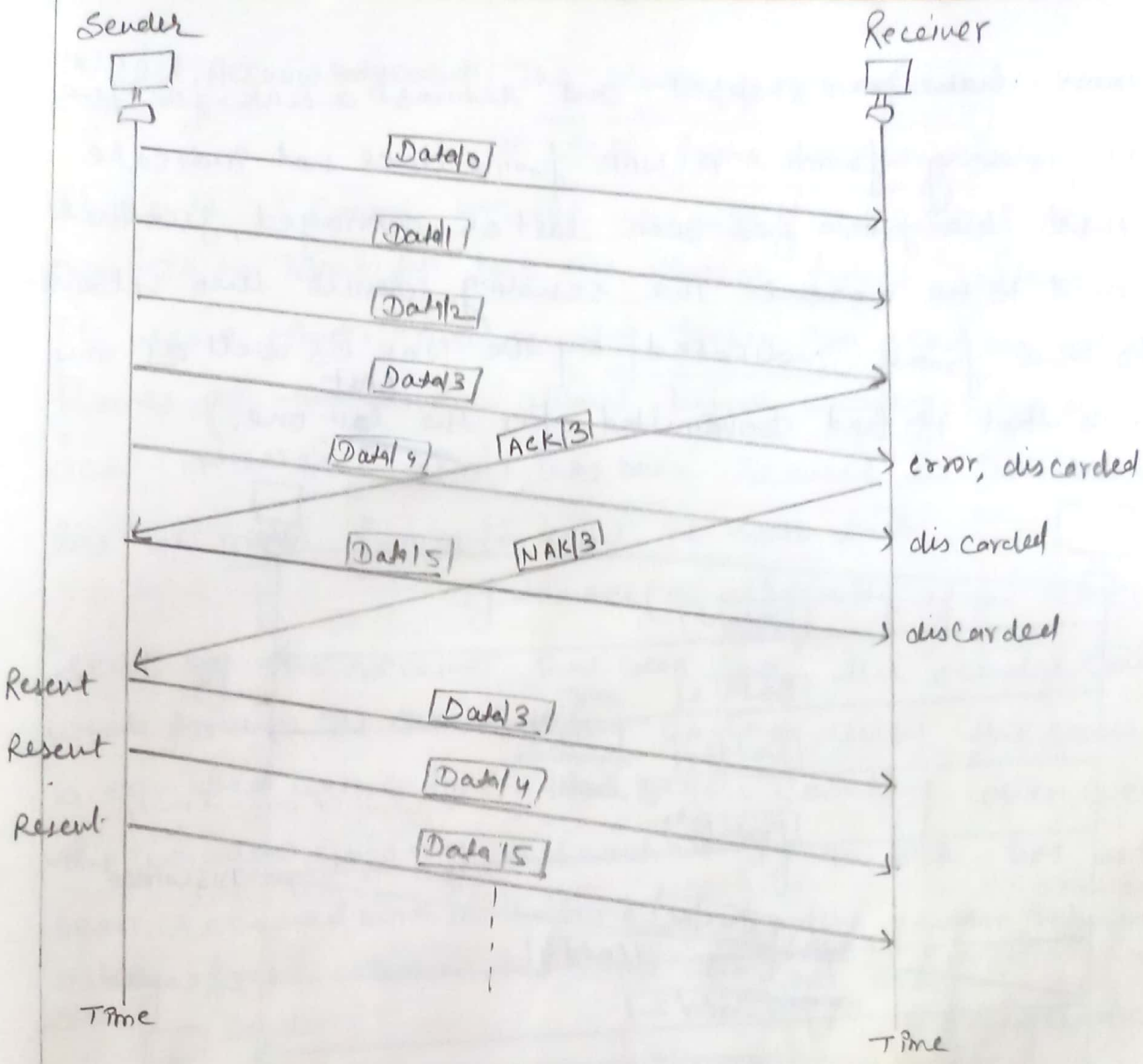


fig: Go Back N, Damage data frame

(ii) Last Data Frame: Sliding window protocals require that data frames be transmitted sequentially. If one or more frames are so noise corrupted that they become last in transit, the next frame to arrive at the receiver will be out of sequence. The receiver checks the (identification) (Identifying number on each frame, discovers that one

or more have been skipped, and returns a NAK for the first missing frame. A NAK frame does not indicate whether the frame has been lost or damaged, just that it need to be resent. The sending device then retransmits the frame indicated by the NAK as well as any frames that it had transmitted after the lost one.)

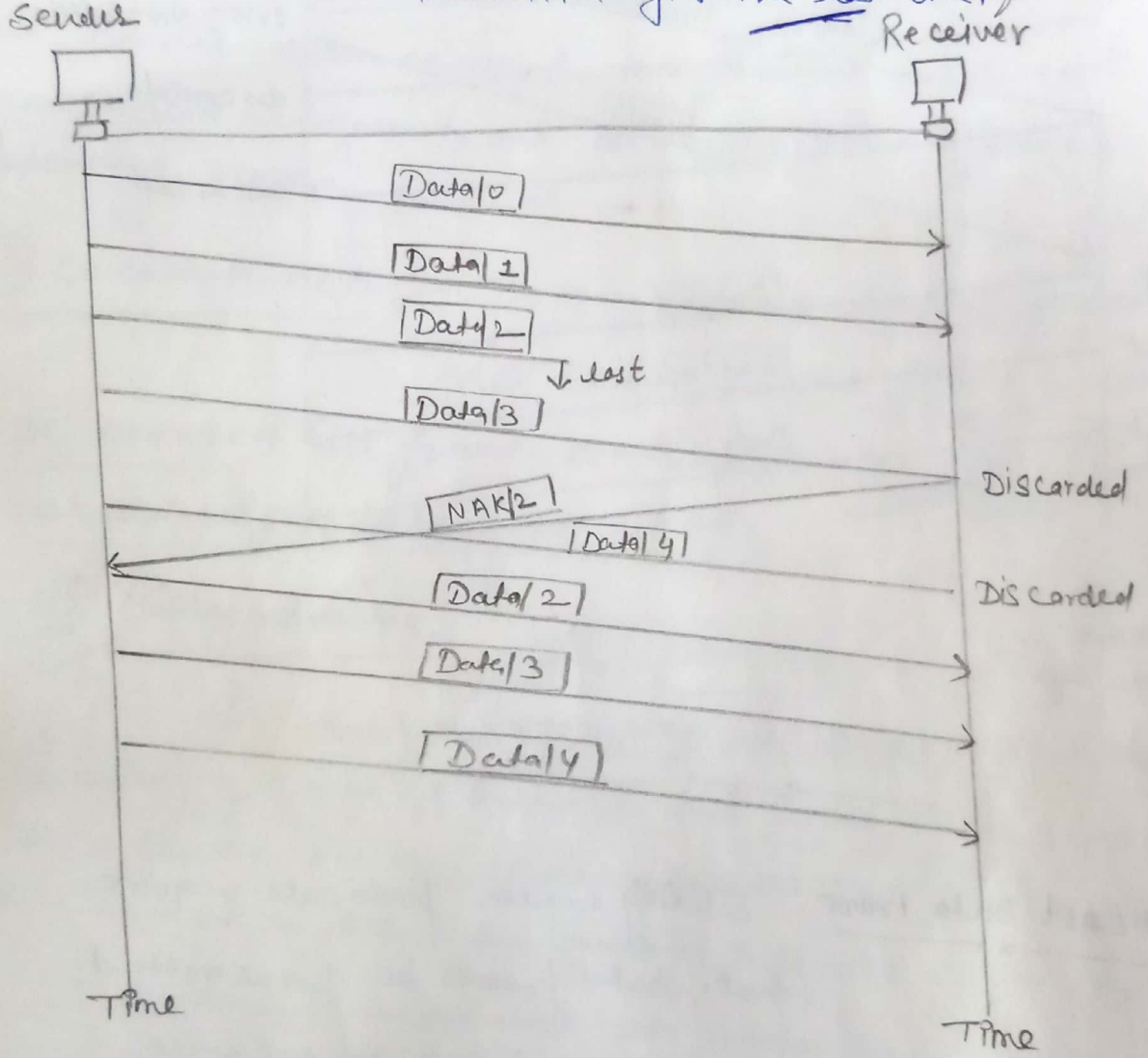


fig: Go Back n, last data frame

(iii) Last Acknowledgement: The sender is not expecting to receive an Ack frame for every data frame it sends. It cannot use the absence of sequential Ack numbers to identify lost ACK or NAK frames, Instead, it uses a timer. The sending device can send as many frames as the window allows before waiting for an ack; once that limit has been reached or the sender has no more frames to send, it must wait.

If the ACK (or, especially, if the NAK) sent by the receiver has been lost, the sender could wait forever. TO avoid tying up both devices, the sender is equipped with a timer that begins counting whenever the window capacity is reached. If an ack. has not been received with in the time limit, the sender retransmits every frame transmitted since the last ACK.

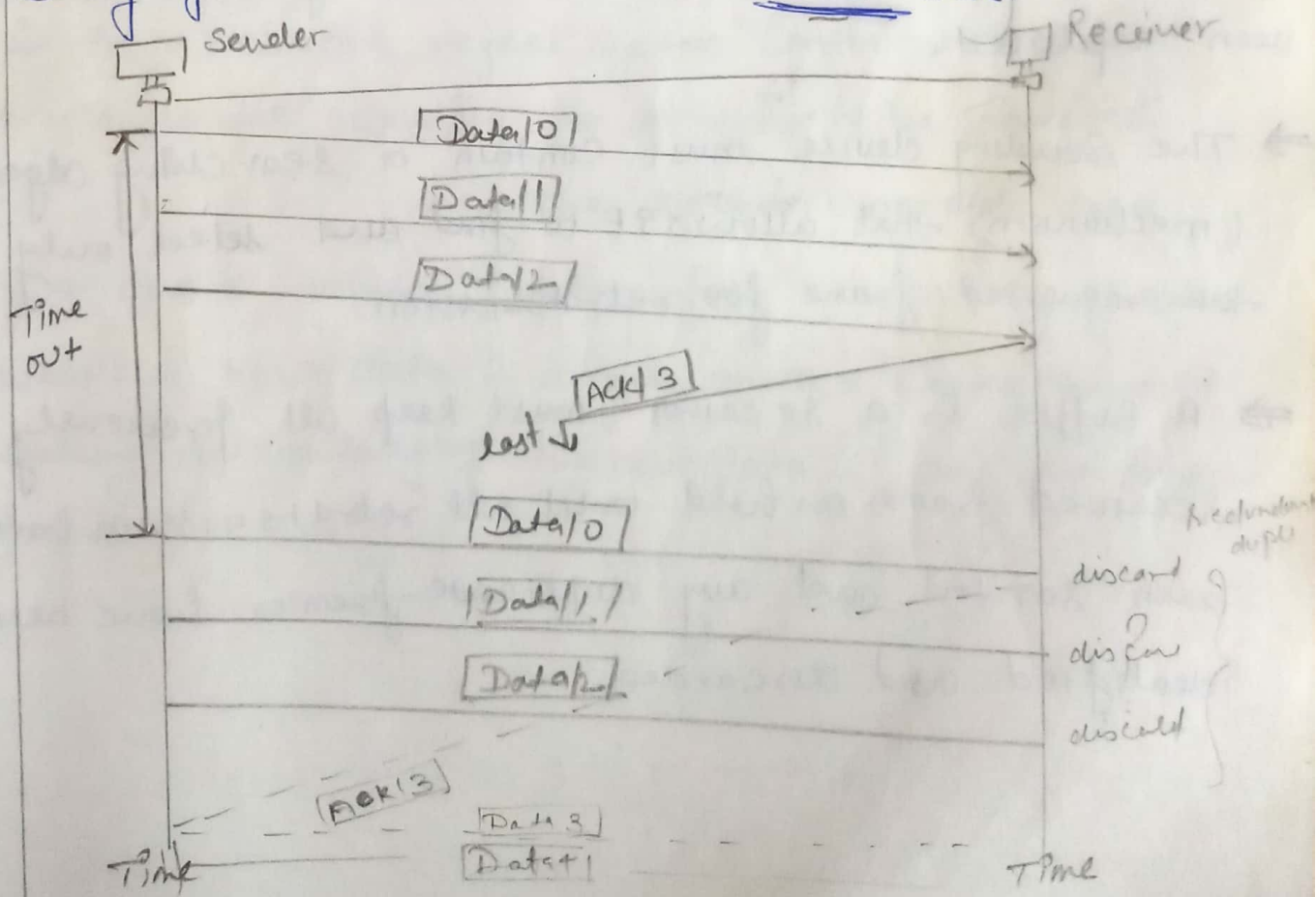


fig: GoBack n, last Ack

(b) Selective-Reject ARQ: In selective-reject ARQ, only the specific damaged or lost frame is retransmitted. If a frame is corrupted in transit, a NAK is return and the frame is resent out of sequence. The receiving device must be able to sort the frames it has and insert the retransmitted frame into its proper place in the sequence. To make such selectivity possible, a selective Reject ARQ system differ from a go-back-n ARQ system in the following way.

⇒ The Receiving device must contain sorting logic to enable it to reorder frames received out of sequence. It must also be able to store frames received after a NAK has been sent until the damaged frame has been replaced.

⇒ The Sending device must contain a searching algorithm (mechanism) that allows it to find and select only the requested frame for retransmission.

⇒ A Buffer in a receiver must keep all previously received frames on hold until all retransmission have been sorted and any duplicate frames have been identified and discarded.

⇒ To aid (help) selectivity, Ack numbers, like NAK numbers must refer to the frame received (or last) instead of the next frame expected.

⇒ This complexity requires a smaller window size than is needed by the go-back-n method if it is to work efficiently. It is recommended that the window size be less than or equal to $(n+1)/2$, where $n-1$ is the go back n window size.

(d.) Damaged frames: Frame 0 and 1 are received but not acknowledged. Data 2 arrives and is found to contain an error, so a NAK 2 is returned. NAK 2 tells the sender that data 0 and data 1 have been accepted, but that data 2 must be resent. The Receiver in a selective reject system continues to accept news while waiting for an error to be corrected.

The receiver accepts data 3, 4, and 5 while waiting for a new copy of data 2. When the new data 2 arrives, an RR 5 can be returned ~~Acking~~ acknowledging the new data 2 and the original frames 3, 4 and 5.

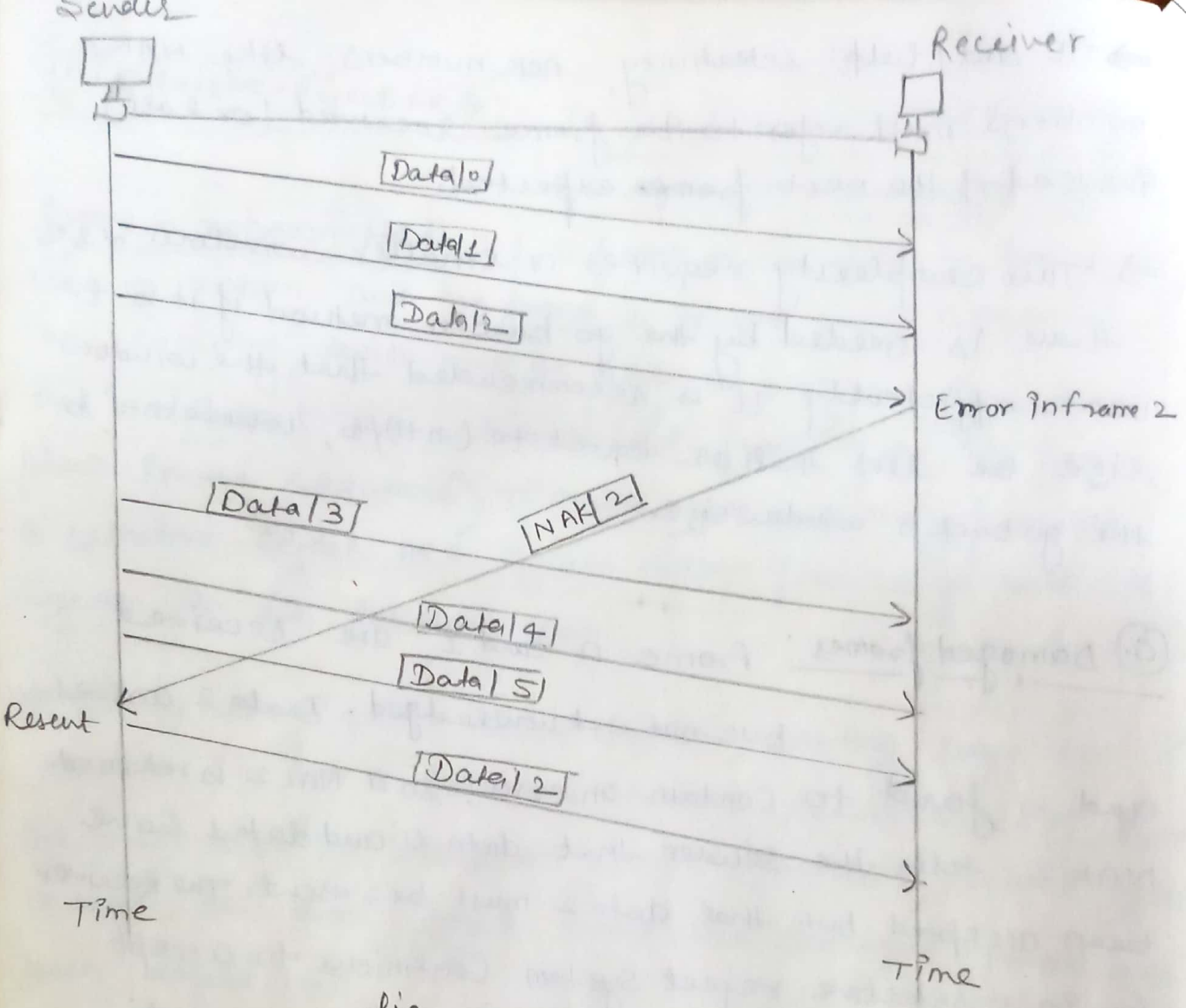


fig Selective Reject, damaged data frame

? → figure.

(b) Last frames: If a frame is lost, the next frame will arrive out of sequence. When the receiver will discover the discrepancy and return a NAK. of course, the receiver will recognize the ommision only if other frame follow. If the last frame was the last of the transmission, the receiver does nothing and the sender treats the silence like a lost acknowledgement.

(c) Last Acknowledgement: LOST

Last Ack and NAK frames are treated by Selective-reject ARQ just as they are by go-back-n ARQ. When the sending device reaches either the capacity of its window or the end of its transmission, it sets a timer. If no acknowledgement arrives in the time allotted, the sender retransmits all of the frames that remain unacknowledged. In most cases, the receiver will recognize any duplications and discard them.

Only theory

Note: In damage & Last frame only Retrans app. frame which is not Acop. Rec
but in Lost ack, all frame send ack

Comparision between Go-Back-n and Selective Reject:

Although retransmission (retransmitting) only specific damaged or lost frames may seem more efficient than resending undamaged frame as well, it is in fact less so. Because the complexity of the sorting and storage required by the receiver, and the extra logic needed by the sender to select specific frames for retransmission, selective-reject ARQ is expensive and not often used. In other words, Selective-reject gives better performance, but in part practice it is usually discarded in favour of go-back-n for simplicity of implementation.

Last Frame