U → 03

Data link layer

Date
Page

Errors

* Error control, Error detection and Error correction :-

① Error detection (E·D) :-

Types of Errors

① single bit errors → Errors where only 1 bit is changed either 1 → 0 or 0 to 1.

eg :- 1100010 ⟶ 1110010
(S) (R)

② Burst error → errors where two or more bits in the data are changed. The first bit that is corrupted to last bit that is corrupted will be considered as the burst error.

eg 1100010 ⟶ 1000111
(S) ↓ (R)
Burst error bits

Error detection (E·D)

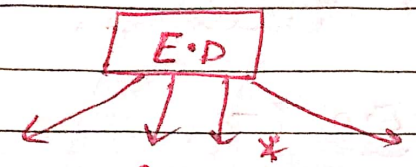
* E·D uses the concept of Redundancy, which means adding extra bits to the data, to detect errors.

(It only tells yes / NO)

(parity).

Date. / Page. /

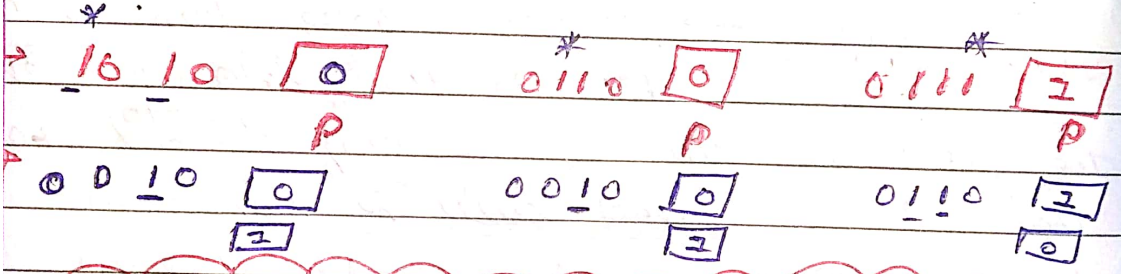
Redundancy is process of adding extra bits to existing bits.



(V·R·C OR V·P·C) → [Vertical Redundancy check].

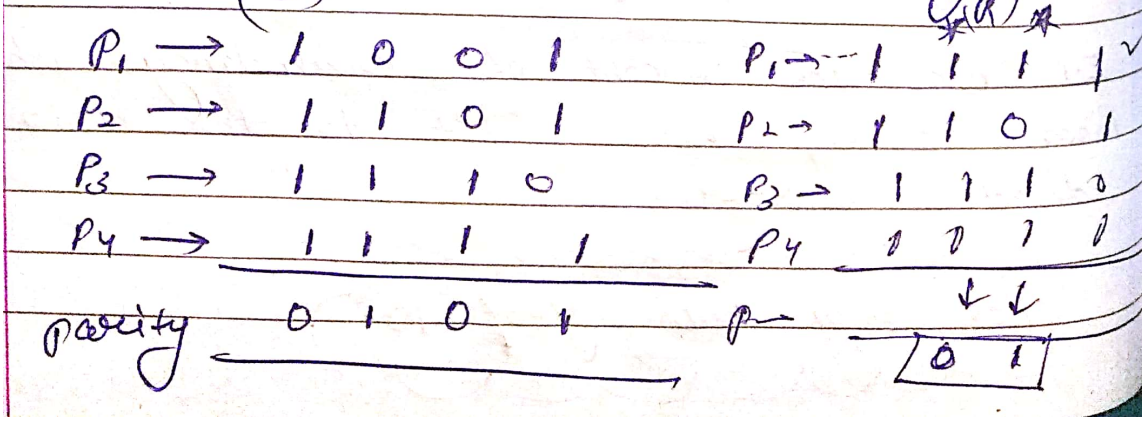
Rule to calculate parity (P)

No. of 1's are even = 0
" " 1's are odd = 1



drawback → if even no. of bits changed then it can't detect errors.

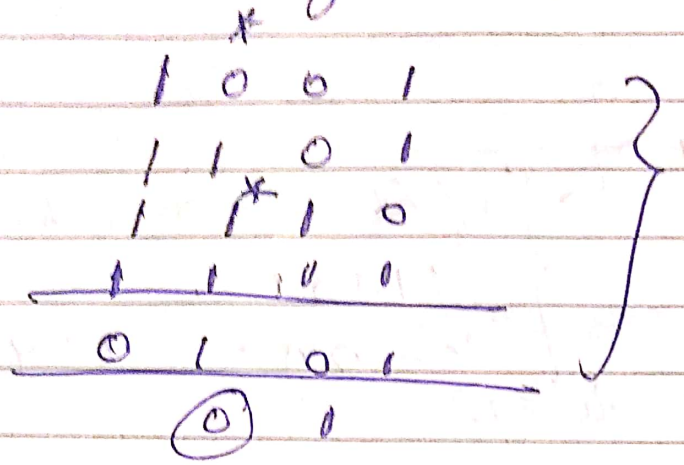
L·R·C → [Longitudinal Redundancy check]



Date. / Page. /

* if error is in some position in some queue of two different packets.

eg



* drawback → if two bit of two diff packet at the same position are changed Reciever can't detect error.

③ * C.R.C → cyclic Redundancy check

* The most powerful and widely used E.D method.

Total bits = m + r

(m = original menage) (r = parity)

(q.) $x^4 + x^3 + 1$ ← divisor = 1010101010 ← dividend = ?

find CRC parity bits?

Date. / Page. /

④ checksum left for next class.

(8) D = 1 0 1 0 1 0 1 0

Div = $x^4 + x^3 + 1$ → if it is given in binary 1 1 0 0 1

$r = 4$ → 4 bit mein dummy. $r = (n-1) = 5-1 = 4$.

⇒ $x^4 + x^3 + 1$

$1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x + x^0 \cdot 1$

1 1 0 0 1 — 5 bit

perfor (XOR) on every bit

same = 0

diff = 1. } XOR

11001) 1 0 1 0 1 0 1 0 1 0 0 0 0 (— 0 0 0 0 dummy

1 1 0 0 1 ...

~~1~~ 1 1 0 0 0

1 1 0 0 1

~~0 0 0 0~~ 1 1 0 1 0

1 1 0 0 1

~~0 0 0~~ 1 1 0 0 0

1 1 0 0 1

0 0 | 0 0 1 0

parity

R

Date. / Page. /

```
11001 ) 10101010100   0010
        11001 ↓
        -----
        01100
         11001
        -----
        00001101 0
            11001
           -------
           00011001
              11001
             -------
Remainder →  00000
             -------
```

(No Error.)

∴ =

(4) **Check sum :-** In check sum each word is added to the previous word and total sum (check sum) is calculated.

There are two ways :-

(a) **simple check sum comparison →**

Eg. 7, 11, 12, 0, 6 $\quad 7+11+12+0+6 = 36$

data	cs	→	7, 11, 12, 0, 6	36
S			R.	

($C_S == C_{S.I}$ No Error)

R - calculate CS

(b) **using 1's complement →**

(1) check sum = 36 → convert in Binary.

(2) Binary :

wrap → 10 0100
→ 10
0110 (6 in decimal)
inverse → 1001 (9 in decimal)
↓
Now this is need Checksum main.
Real check sum = 9.

7, 11, 12, 0, 6 | 9 → Recieve

In Reciever :

① 7, 11, 12, 0, 6 | 9

Chek sum = 45.

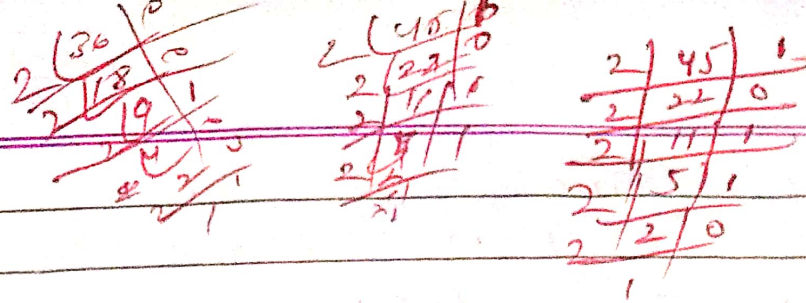
10 1101 (binary)
10
1111
Inverse 0000 → No error.

Error correction

* Hamming code :-

⇒ It is used to detect and correct errors as well.

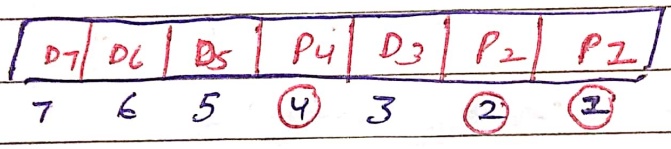
Rules :-

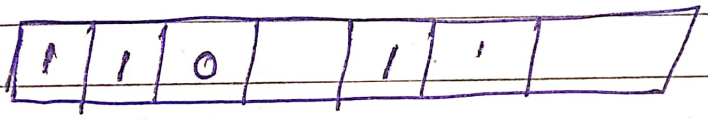


Date. |
Page. |

(1.) All bit positions that are power of 2 will be marked as parity. (ie; $2^0, 2^1, 2^2$ ——).

⇒ (1, 2, 4, 8, 16, ——) and rest bits are considered as data. eg. (7 bit info)





(2) To calculate the structure Rule says to "check one skip one". and so on.

Rule 1 :- P(1) :- check 1, skip 1
start with parity :- 1, 3, 5, 7, —— @ bits a

Rule 2 :- P(2) :- check 2, skip 2.
start with parity :- 2, 3, 6, 7, ——

Rule 3 :- P(4) :- check 4, skip 4.
start with parity. 4, 5, 6, 7, 12, 13, 14, 15

P_1	P_2	P_4
P_1 B_3 D_5 D_7	P_2 P_3 P_6 P_7	P_4 P_5 P_6 D_7
P_1 1 0 1	P_2 1 1 1	P_4 0 1 1
$P_1 = 0$ No Error	$P_2 = 1$ Error	$P_4 = 0$
odd 1 = 0	Even 1 = 0	

Date. / Page. /

A 7 bit hamming code is Recieved 1011011. Assume Even parity and detect if there is error in it or not. if yes! correct it

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	1	0	1	1

lets Analyse code first:-

1 3 5 7

P_1 D_3 P_5 D_7

1 0 1 1

(odd 1 = 1)

detection $P_1 = 1$

error

2 3 6 7

1 0 0 1

even 1 = 0

$P_2 = 0$ NO error

4, 5, 6, 7

1 1 0 1

odd 1 = 1

$P_4 = 1$

P_3 P_2 P_1

Error code = 101

Date.
Page.

Q.) Assume an 11 bit hamming code data 10011101

D_{11} D_{10} D_9 P_8 D_7 D_6 D_5 P_4 D_3 P_2 P_1

1	0	0	P_8	1	1	0	P_4	1	P_2	P_1

P_8 P_4 P_2 P_1

(1) P_1 :- C1, S1

1, 3, 5, 7, 9, 11,

P_1 1 0 1 0 1

Even 1 = odd = 0

$P_1 = 0$ No Error

(2) P_2 :- C2, S2

2, 3, 6, 7, 10, 11

P_2 1 1 1 0 1

P_2 Even 1 = 0

$P_2 = 0$ No Error

(3) P_4 :- C4, SK4

4, 5, 6, 7,

P_4 0 1 1

Even 1 = 0

$P_4 = 0$

P_8 :- C8, SK8

8, 9, 10, 11,

P_8 0 0 1

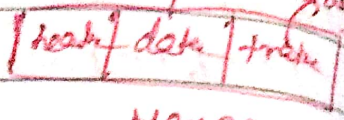
address

Date. / Page.

#o Ch-2

→ Ethernet

#o MAC Frame



Preamble	SFD	DA	SA	Length	data	C
7 bytes	1 byte	6 byte	6b	2b.		4b

Sync.

Preamble :- Alternating 0's and 1's that alerts the reciever of incoming frames and the need for the clocks synchronisation.

SFD :- Start frame delimiter. it is the begning of (10101011) this is the final chance of synchron

Dest Add :- Reciever's MAC

Source Add :- sender's MAC

length :- Size of data or No of bytes of the data.

means → size of data in bytes.

data & padding :- { Minm → 46 bytes. Maxm → 1500 bytes.

CRC :- used for Error detection.

frame length

Date
Page

All frame :- Minm = 64 bytes (512 bits)
Maxm = 1518 bytes (12,144 bits)

Ch → 03 → Data control

protocols

① Noiseless channel :-	② Noisy channel
ⓐ simplest ⓑ stop-and-wait	left

Types of framing

① fixed size :- Here there is no need to define the boundary of the frame, size of the frame acts as the delimiter. (start or end of frame acts as limit)

② variable size framing :- Here there is a need to define the start and end of the frame (e.g. FLAG). In order to distinguish one frame from another.

(Read from PPT)

Bit stuffing & byte stuffing

Date. / Page. /

Protocols

for Noiseless channel :-

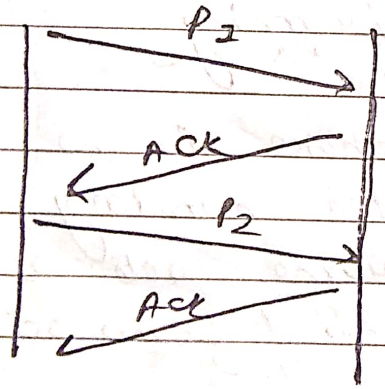
Simplest :-

No concept of Acknowlegement.

does not have flow & error control.

As No Ack., No flow control. in the sense that there is no concept of Ack. here. flow control here means the duration a sender is willing to wait for Ack before sending the next frame.

Stop and wait :-



- It has concept of Ack.
- Next packet will only be sent when Ack for prev one is Recieved
- if Ack is not Recieved it waits infinite amount of time, next packet can't be sent.

drawback :-

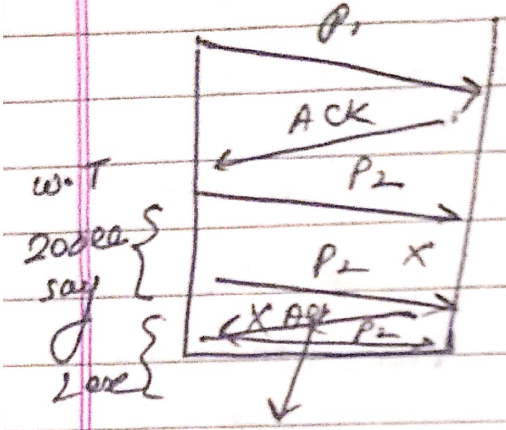
Lost Ack., lost frame, delayed frame Ack it wait infinite No. of time

(a) Ack is piggybacked

② it has flow control but NOT Error control

Noisy channel

① **Stop and wait ARQ :-** (Automatic Repeat Request)



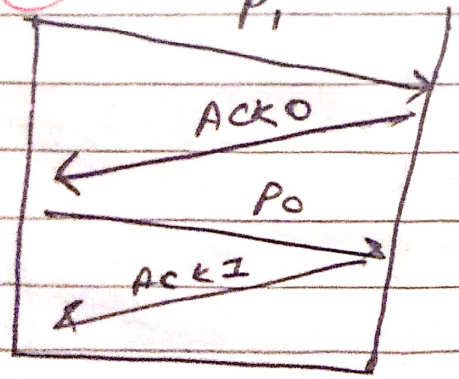
(Retransmission)

◉ It has both flow & error control.

* **Round trip time :-** RTT

② R.T.T = Amount of time taken by a packet to reach the Reciever + Amount of time taken by the Acknowledgement to be Recieved by the sender

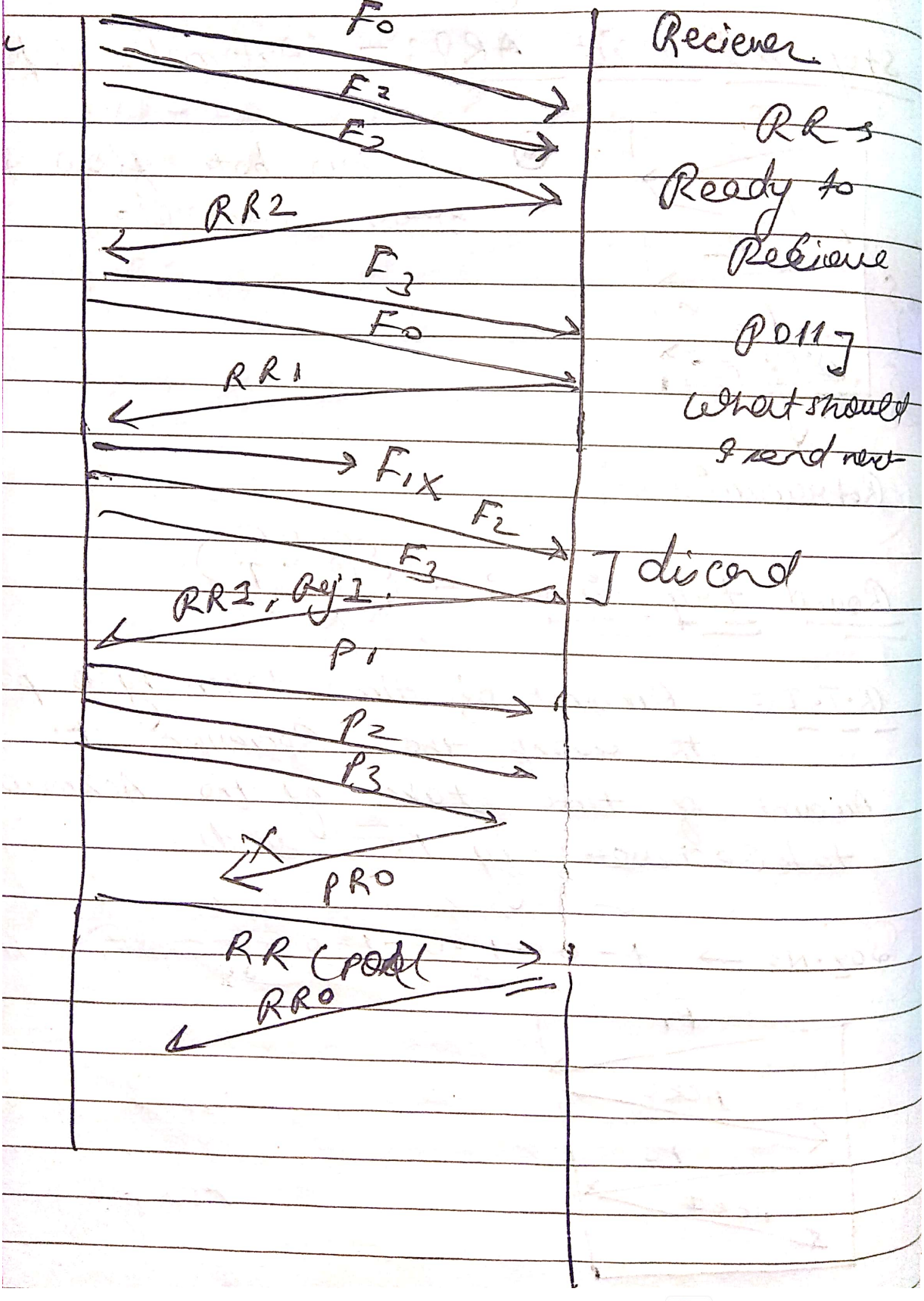
Seq.No → 1-0-1-0-1-0 ----



Date.
Page.

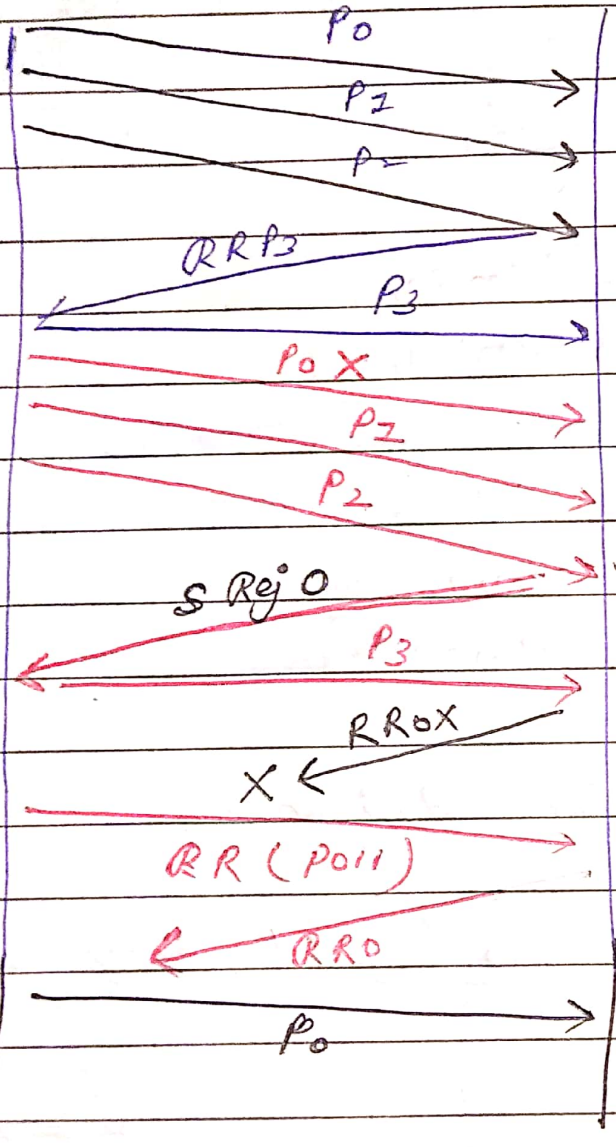
sliding window protocols :-

Multiple packets one Ack concept.



Date. |
Page. |

Selective Reject technique :-



$\{k > k_{max}\}$

- It will attempt maxm 16 times only. (0-15)
- After 16 it will abort the whole process.

} buffer store.

o ═══ o

Ch-04 Multiple Access

Date.
Page.

CSMA :- carrier sense multiple Access.

"Sense channel before you transmit"

CSMA - CD (collision detection)
CSMA - CA (collision Avoidance)

CSMA - CD :- ① It detects collision after it occurence.
Happens in wired Networks.

CSMA - CA :- (collision Avoidance)

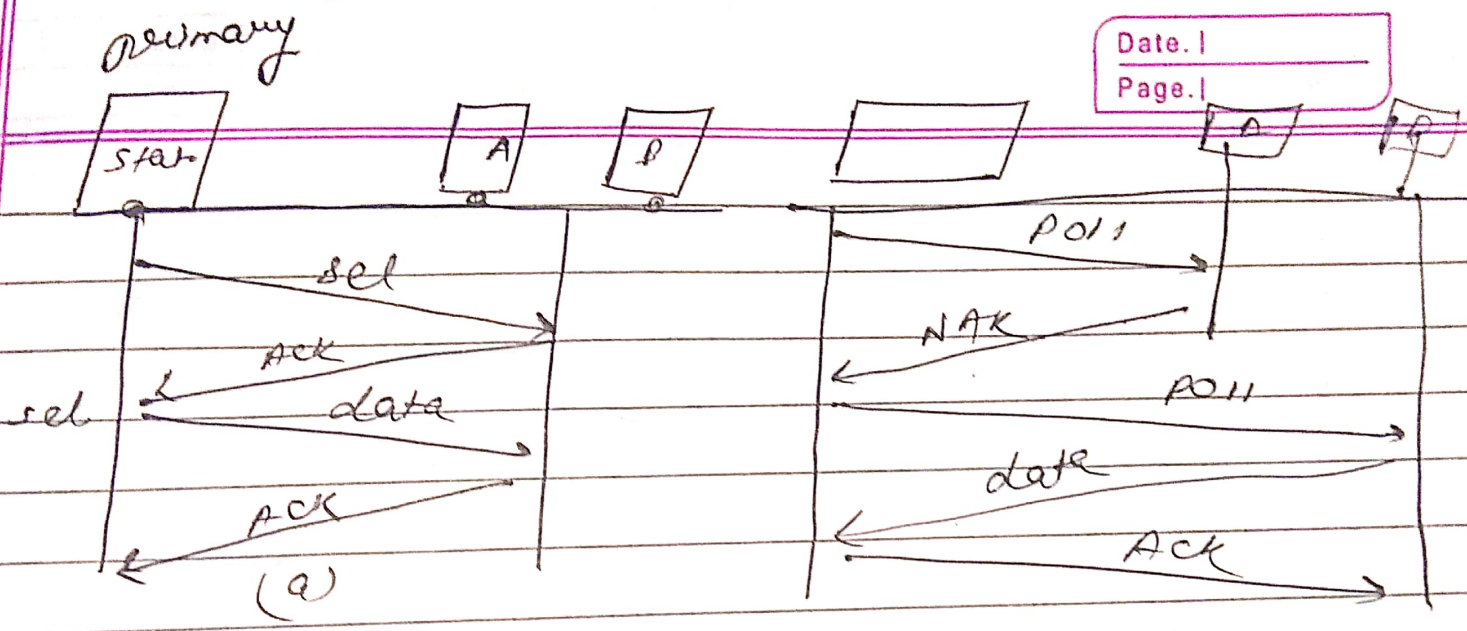
Here steps are taken to avoid collision before it happens.

It happens mostly in wireless Networks.

o=o

Controlled Access

In sel mesage primary station asks the sec. stations if they are free to Recieve the data; ie. primary station has some data to send to sec. stations and sel is sent to Just confirem if the sec stations are free to recieve it or not.



(2) poll :- using poll primary station, acknowledges ~~reciev~~ sec. station (Reciever) that it is free right now and is ready to recieve data from sec. station if they have any.