

UNIT-2

Data Link Layer

(1) Need:-

- [4th layer from top & 2nd from down]
- ① It hides the details of underlying hardware & represents itself to upper layer as the medium to communicate.
 - ② It is responsible for converting data stream / packets of network layer into frames.
 - ③ It transfers data b/w adjacent network nodes in a WAN or between nodes on the same LAN.
 - ④ It is concerned with local delivery of frames b/w nodes on the same level of the network.
 - ⑤ Inter network routing & global address (IP) are higher level functions (Network layer), allowing data link protocols to focus on local delivery, addressing (MAC) & media arbitration.
 - ⑥ When a devices attempt to use a single medium simultaneously, frame collisions occur. Data link protocols specify how devices detect & recover from such collisions & also provide mechanism to reduce or prevent them.
• DDL Protocol are Ethernet, PPP, token ring.
 - ⑦ Data link layer has 2 layers -
 - (1) Logical Link control (LLC) - deals with flow & error
 - (2) Media Access Control (MAC) - deals with actual com

Date / /

(2) Services Provided:-

(1) Framing:- It encapsulate each network layer datagram into frame before transmission onto the link

- Framing
- Link Access
- Reliable delivery
- Flow control
- Error detection
- Error correction
- Half Duplex & Full Duplex

- A frame consists of a data field, in which the network layer datagram is inserted & add header & trailer

↓
header

↓
Error detection (CRC)

- Physical addresses used in frame headers to identify source & destination.

(2) Link Access:- When multiple nodes share a simple broadcast link, the channel access protocol serves to coordinate the frame transmission of the many nodes.

(3) Reliable Delivery:- DLT provides a reliable delivery service that means it transmits the network's layer datagram without any error.

- A reliable delivery service is achieved by transmission & acknowledgements.

Date / /

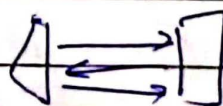
(4) Flow Control:- If the rate at which the data is absorbed by the receiver is less than the rate produced in the sender, then the receiver's buffer can overflow & frames can get lost. A receiving node can receive the frames at a faster rate than it can process the frame. To overcome this problem DLL, it uses the flow control so that constant data rate is maintained on both sides & no data get corrupted/lost.

(5) Error detection:- Error may be introduced into a transmitted frame due to signal attenuation & electromagnetic noise.

- DLL protocol provides a mechanism to detect one or more errors
- This is done by having the transmitting node set error detection bits in the frame & having the receiving node perform an error check.

(6) Error Correction:- It is similar to error detection except that a receiver can not only detect whether errors have been introduced in the frame but can also determine exactly where in the frames the errors have occurred & hence correct the errors.

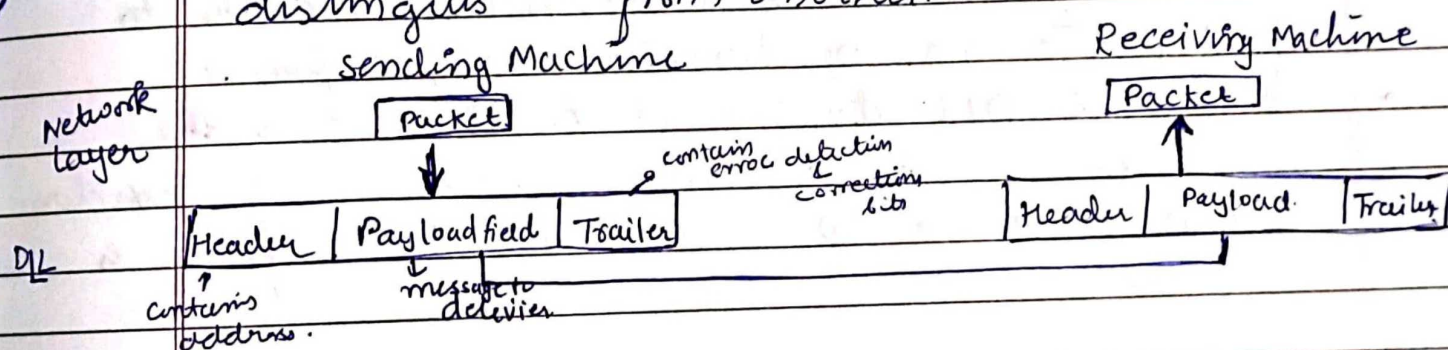
(7) Half Duplex & Full Duplex:-



Date / /

- ⑦ Half Duplex & full Duplex:- In a Half-Duplex mode, only one node can transmit the data at a time & in a full duplex modes, both nodes at the end of a link may transmit packet at the same time.

① Framing:- The data link layer needs to pack data packets delivered from the network layer into separate manageable frames so that each frame is distinguish from another.



Since the physical layer merely accept & transmits a stream of bits without any regard to meaning or structure, it is upto DL to create & recognize frame boundaries.

- This can be accomplished by attaching special bit patterns to the beginning & end of the frames
- If these bit patterns can accidentally occur in data, special care must be taken to make sure these patterns are not incorrectly interpreted as frame delimiters

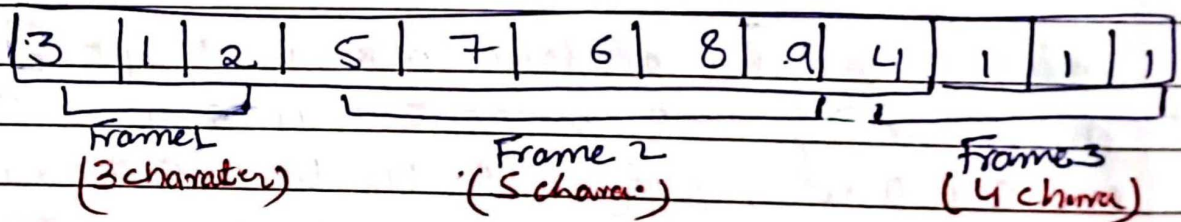
Date / /

The following are the four framing methods that are widely used.

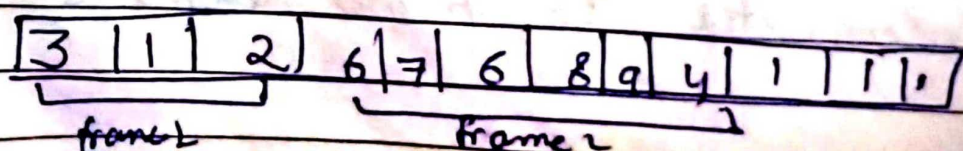
- (1) character count
- (2) starting & ending characters with character stuffing
- (3) starting & ending flag with bit stuffing
- (4) Physical layer coding violations

(1) Character Count:-

- This method uses a field in the header to specify the no. of characters in the frame.
- When the DLE at the destination sees the character count, it knows how many characters follow & hence where the end of the frame is



Disadvantage:- If the character count is garbled by a transmission error, the destination will lose synchronization & will be unable to locate the start of the next frame. So this method is rarely used.



Date / /

(2) Character stuffing

In this method, each frame starts with the ASCII character sequence DLE STX, ~~ETX~~ & ends with the sequence DLE ETX

DLE - Data Link Escape

STX - Start of Text

ETX - End of Text

- It overcomes the drawbacks of the character count method.

Example Data send - ABC
by N/w Layer

DLE STX A B C DLE ETX

- If the destination ever loses synchronization it only has to look for DLE STX & DLE ETX character.
- If data being transmitted consists of characters like DLE STX & DLE ETX, a technique called character stuffing is used.
- The sender's DLL inserts an ASCII DLE character just before the DLE character in the data & the receiver's DLL removes this DLE before the data is given to the network layer.

Example:- AB DLE C

character
stuffing.

DLE STX AB DLE DLE C DLE ETX

stuffing DLE

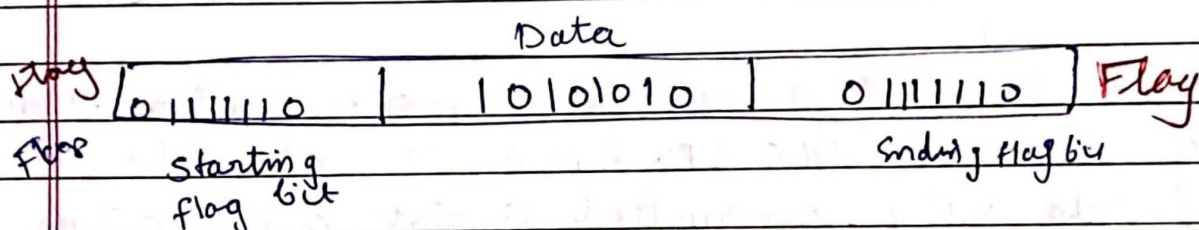
remove

Date / /

Disadvantage:- Character stuffing is closely associated with 8 bit characters & this is a major problem in transmitting arbitrary size characters.

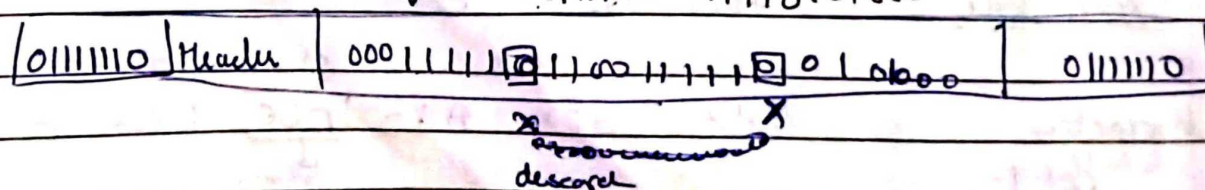
(3) Bit stuffing:- This method allows data frames to contain an arbitrary no. of bits & allows character codes with an arbitrary no. of bits per character.

Example:- Most protocols use a special bit pattern flag 01111110 as the delimiter to define the beginning & the end of the frames.



- If the flag pattern appears in the data, whenever the sender's bit encounters five consecutive 1's in the data, it automatically stuffs a zero bit into the outgoing bit stream. This technique is called bit stuffing.

Example:- Data from upper layer
00011111101111010100



Example:-
2110111111111111010
01101111 011111 011111 010010

stuffing bit

Date / /

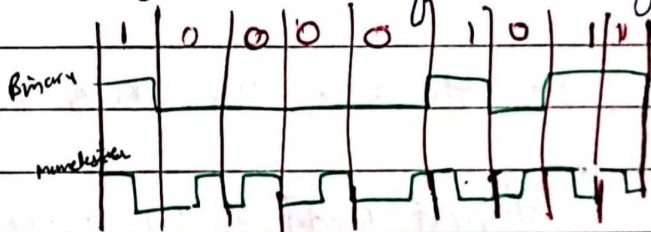
When receiver sees 5 consecutive 1's in the incoming data stream, followed by a zero bit, it automatically destuffs the 0 bit.

- The boundary between two frames can be determined by locating the flag pattern.
- In this we stuff single bit (instead of 1 byte) to prevent the pattern from looking a flag

(4) Physical Layer coding violations :

It is application to networks in which the encoding on the physical medium contain some redundancy.

- In such cases normally, a 1 bit is a high-low pair (10) & a 0 bit is a low-high pair (01) (Manchester coding is used)
- The combination of low-low & high-high which are not used for data may be used for making frame boundaries



- ① odd parity bit stuffing
- ② even parity bit stuffing

① Odd

0	0	1	1	1	0
---	---	---	---	---	---

odd

receive

odd (odd+1) X
even

~~if even no diff it is not receive.~~

0	0	1	1	1	0
---	---	---	---	---	---

 → error mch
(adding +1)

odd

Date / /

Even parity

0010 1 → Parity bit
original

01100 → Parity bit

odd parity

0010 0 → Parity
original

0110 1 → Parity bit
error

② Flow Control:-

Types of framing - There are two types of framing.

① Fixed size:- The frame is of fixed size & there is no need to provide boundaries to the frame, length of the frame itself acts as delimiter.

• Drawback:- It suffers from internal fragmentation of data size is less than frame size.

• Solution:- Padding.

② Variable size:- In this there is need to define end of frame as well as beginning of next frame to distinguish. This can be done in two ways:-

(1) Length Field:- We can introduce a length field in the frame to indicate the length of frame. Used in Ethernet (802.3)

This problem with this is that sometimes the length field might get corrupted.

(2) End Delimiter (ED):- We can introduce an ED (pattern)

Date / /

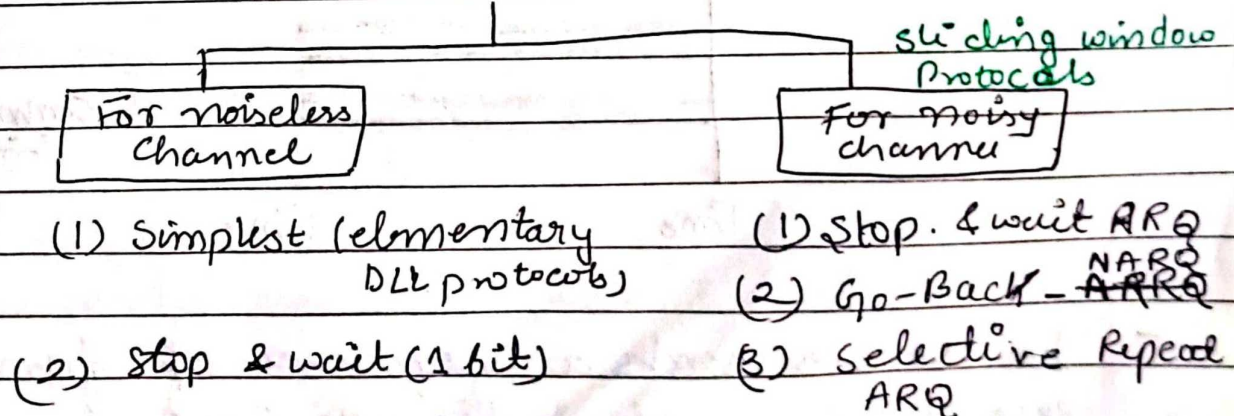
to indicate the end of the frame. Used in Token Ring. The problem with this is that ED can occur in the data.

This can be solved by - Byte stuffing, Bit stuffing

(2) Flow Control :-

- Flow control is a technique that allows two devices working at different speed to communicate with each other.
- Consider a situation in which the sender transmits frames faster than the receiver can accept them. If the sender keeps sending frames at high rate, at some point the receiver buffer will be completely full & will start losing some frames.
- This problem may be solved by introducing flow control.
- Most flow control protocols contain a feedback (Ack) mechanism to inform the sender when it should transmit the next frame.

Flow control Protocols in DLL



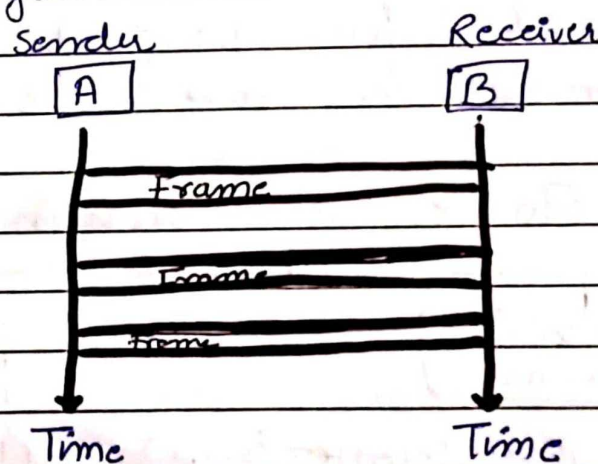
DLL PROTOCOL:-

(A) For noiseless channel:-

(1) Elementary Data Link Protocol (Simplex)

The following assumptions has been made for developing the simplex protocol.

- (i) The channel is a perfect noiseless channel.
- (ii) An ideal channel in which no frames are lost, duplicated or corrupted.
- (iii) No flow & error control used.
- (iv) It is a unidirectional protocol in which data frames are travelling in only one direction.
- (v) Both transmitting & receiving hosts are always ready.
- (vi) Infinite buffer space is available.
- (viii) Processing time that is small enough to be negligible.



Communication Using
Simplex protocol

- The sender sends a sequence of frames without even thinking about the receiver.
- No acknowledgements or retransmission used.

Date / /

- If frame has 'd' data bit & 'h' overhead bit channel bandwidth. b - bits / second.

$$\text{Maximum channel utilization} = \frac{\text{data size}}{\text{Frame size}}$$

$$= \frac{d}{d+h}$$

$$\text{Maximum data throughput} = \frac{d}{d+h} * \text{channel bandwidth}$$

$$= \frac{d}{d+h} * b.$$

(2) Stop & Wait Protocol:-

[Error free]

The following assumption has been made for developing the stop & wait protocol -

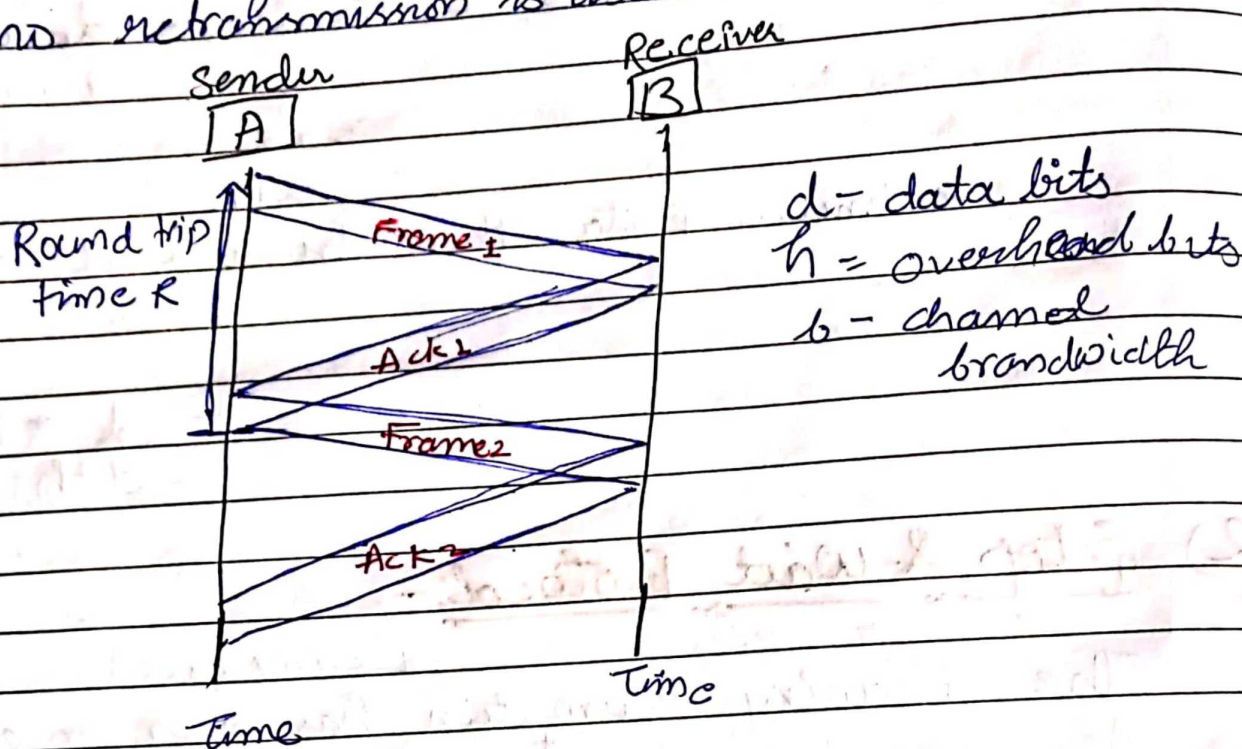
- (i) The channel is a perfect noiseless channel.
- (ii) Flow control used
- (iii) It is a bidirectional protocol in which frames are travelling in both direction.
- (iv) Both transmitting & receiving hosts are always not ready.
- (v) Processing time considerable
- (vi) Finite buffer space is available ~~rec~~ the receiver may not be always ready to receive next frame.

- The sender sends one frame & wait for feedback from the receiver (Ack).

In stop & wait protocol, receiver sends a positive acknowledgement frame to sender to

Date / /

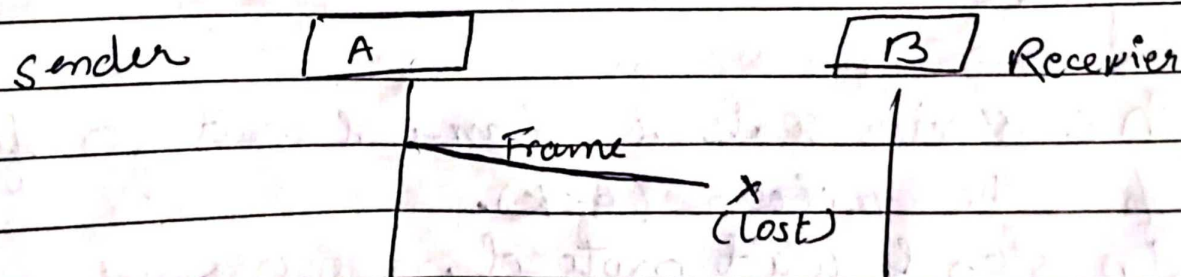
- transmit the next data frame.
- As error-free communication channel assumed, no retransmission is used.



Stop & wait protocols only gives us an idea of how to control flow control, but as noiseless channels are nonexistent, hence error occurs.

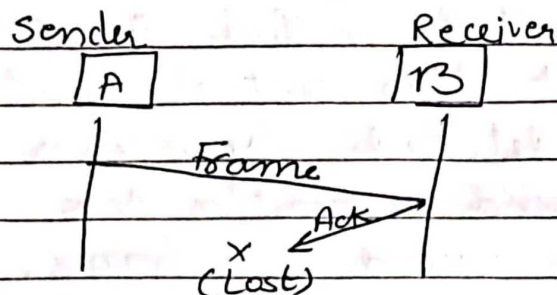
Problems of Stop & wait Protocol -

(1) Last Data:- Sender wait for Ack & Receiver wait for data for infinite amount of time.



Date / /

(2) Lost Acknowledgement :- Sender wait for acknowledgement & receiver wait for data for infinite amount of time.



(3) Delay Acknowledgement :- After timeout on sender side, a long delayed acknowledgement might be considered as ack of some other recent packet.

(B) Sliding Window Protocol / Noisy channel :-

- It removed the drawbacks of stop & wait protocol.
- It uses error control (work on noisy channel)
- It is used where reliable in order delivery of packets is required, such as in the DLL of OSI model as well as in the TCP.

The sliding window ARQ has 3 techniques.

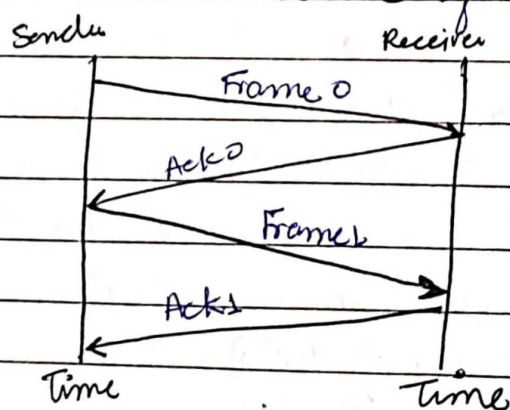
- (1) 1 bit or stop & wait ARQ
- (2) Go Back N ARQ
- (3) Selective Repeat ARQ

ARQ - [Automatic Repeat Request].

It ensure a sequence of information packets is delivered in order & without errors or duplication despite transmission errors & losses.

(1) 1 bit Or Stop & Wait ARQ Protocol:-

- Above three problems are resolved by stop & wait ARQ that does both error controls & flow control.
- An error control method is incorporated with stop & wait flow control protocol.
 - If error is detected by receiver, it discards the frame & send a negative ACK (NAK) causing sender to resend the frame.
 - stop & wait ARQ implements sliding window protocol concept with window size 1.
 - This is the simplest flow control protocol in which the sender transmits a frame & then waits for an acknowledgement, either (+)ve or (-)ve, from the receiver before proceeding.
 - If a positive acknowledgement is received, the sender transmit the next frame else it retransmits the same frame. ~~Sender~~ Receiver.



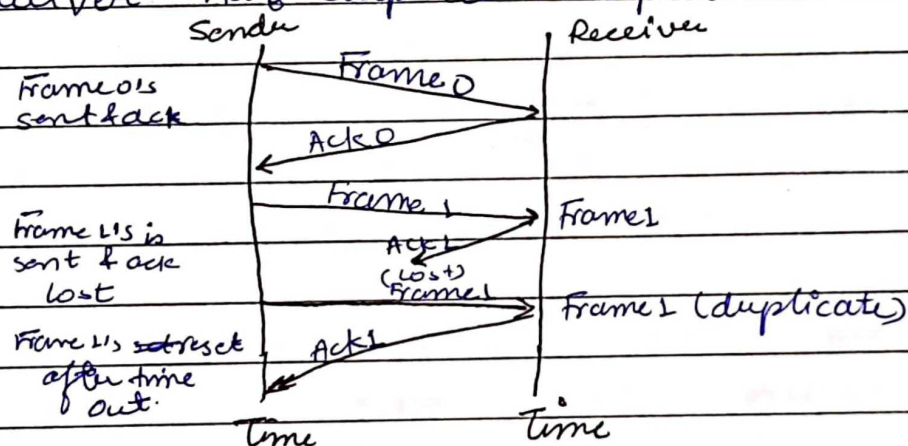
Drawback - If a packet or an acknowledgement is completely destroyed in transit due to a noise burst, a deadlock will occur because the sender can not proceed until it receives an acknowledgement.

Date / /

Solution:- Use timer on the sender's side
when the frame is transmitted, the timer is set. if there is no response from the receiver within a certain time interval the timer goes off & the frame may be retransmitted

Timer introduces a problem:-

suppose timeout & sender retransmits a frame but receiver actually received the previous transmission but acknowledgement was lost. so receiver has duplicated copies.



- It is very inefficient protocol.
- At any one moment, only one frame is in transition.
- The sender will have to wait at least one round trip time before sending next.

For noisy channel, pure stop & wait protocol will break down & solution is to incorporate some error control mechanism

Date / /

(2) GO BACK - N FRQ Protocol -

In this the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement packet from the receiver.

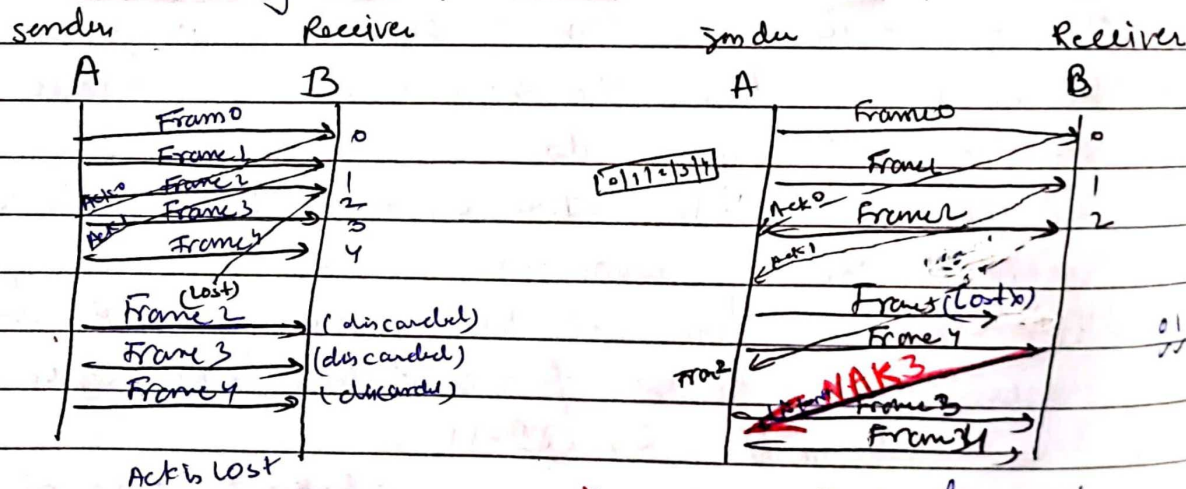
- To improve the efficiency of transmission multiple frames must be in transition while waiting acknowledgement.
- It improves stop & wait by not waiting.
- It uses n-bit sequence numbering.
- sequence numbers are modulo 2^m , where m is the size of the sequence number field in bits
eg:- $m=4$ sequence no - 0 to 15 (0- 2^m-1)
- We can repeat the sequence
 $0, 1, 2, \dots, 15, 0, 1, 2, \dots, 15, 0, 1, 2, \dots, 15$
- It introduces the window of size n ($n > 1$)
- A window is a collection of adjacent sequence numbers.
- The range which is the concern of the sender is called Sender sliding window.
- The range that is the concern of the receiver is called receiver sliding window.
- The send window is an imaginary box ^{covering} ~~covering~~ the sequence numbers of the data frames which can be in transit ($2^m - 1$)
- The receiver window is an abstract concept defining an imaginary ~~box~~ ^{box} of size 1 with one single variable R_n . The window slides when a correct frame has arrived, sliding occurs one slot at a time.

Date / /

- There is one times.
- The receiver sends a positive acknowledgement if a frame was arrival safe & in order. If a frame is damaged or is received out of order, the receiver is sent & will discard all subsequent frames until it receives the one it is expecting.
- The silence of the receiver caused the time of the unacknowledged frame at the sender side to expire. It cause the sender to go back & resend all frames, beginning with the with expired times.

For example:- Suppose the sender has already sent frame 6 but the timer for frame 3 expires.

This means that frame 3 has not been acknowledged the sender goes back & sends frame 3, 4, 5 & 6 again. That is why the protocol is called Go-Back-N-ARQ.



Date / /

Advantages:- ① The sender can send many frames at a time.

- ② Timer can be set for a group of frames.
- ③ efficiency is more.
- ④ waiting time is low.
- ⑤ we can alter the size of sender window.

Disadvantages:-

- ① It is very inefficient for a noisy link.
- ② Buffer requirement sender needs to store the last N packets.
- ③ scheme is inefficient when round trip delay large & data transmission rate is high.
- ④ if NACK is lost, a long time is wasted until re-transmission of all packets until another NACK is sent.

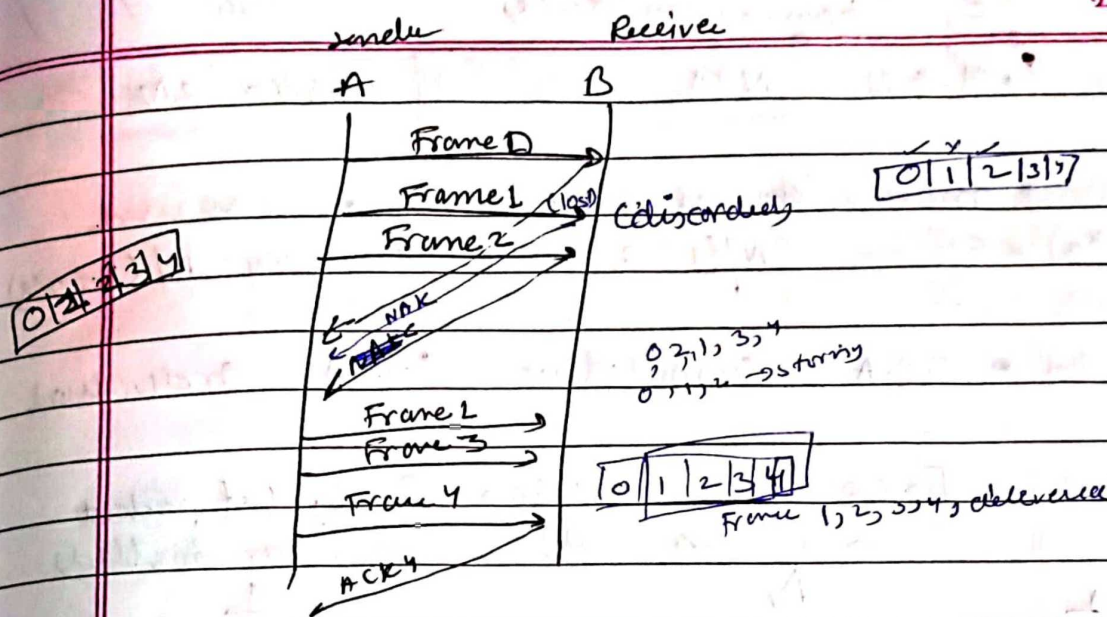
(3) Selective Repeat Protocol:-

- Go-Back-N ARQ inefficient because multiple frames are resent when errors or losses occur.
- There is another mechanism used for noisy links that does not resend N frames when just one frame is damaged, it only resends the damaged frame. This mechanism is called as "Selective Repeat ARQ".
- Timeout causes individual corresponding frame to be resent.
- NAK causes retransmission of oldest unacknowledged frame before the corresponding timer expires & then thus improve performance.

Date / /

- This protocol also uses two windows, but size of sender & receiver windows must be almost one half of 2^m .
eg:- $m=2$ window size = $\frac{2^m}{2} = \frac{2^2}{2} = 2$
(sender & receiver)
- In this protocol rather than discard all the subsequent frames following a damaged or lost frame, the receiver's DLL simply stores them in buffers.
- When the sender does not receive an acknowledgment for the first frame, its time goes off after a certain time interval & it retransmits only the last frame.
- In this protocol receiver stores correct frame, the bad one. Sender retransmits the bad one after noticing.
- Receiver passes data to network layer & acknowledges with the highest number.
- In this protocol receiving window > 1, so need large memory.
- Ack for frame n implicitly acknowledges all frames $\leq n$.
- In SRP, the retransmitted frame is received out of sequence, so the receiver is capable of sorting the frame in a proper sequence.
- The sender must be capable of searching the for which the NAK has been received.

Date / /



* Comparison b/w Stop wait, Go-Back & Selective Repeat :-

Stop & Wait ARQ

- Send window size = 1
- Receiver " " = 1
- The sender sends the frame & wait for acknowledgement once the Ack reaches the sender, it transmits the next frame.

[No buffer at R. & S]

$$(1+1) = 2$$

Total Buffer = 2

Go-Back -N ARQ

- Send window size = N (or $2^m - 1$)
- Receiver " " = 1
- The sender N packets/frame which is equal to the window size once the entire window is sent, the sender waits for a cumulative Ack to send more frame.

[No buffer]

$$(N+1)$$

T.B

Selective Repeat ARQ

- Send window size = N ($2^m - 1$)
- Receiver " " = N
- The sender sends frame of window size N & the receiver ack all frames ^{whether} there were received in order or not

[Buffer]

$$(N+N)$$

T.B

$a =$ Ratio of Propagation delay & Transmission delay.

$$a = \frac{TP}{TE} \quad \begin{matrix} \text{(Propagation of time)} \\ \text{(Transmission time)} \end{matrix}$$

Date / /

• Minimum sequence

$$M.SN = N+1$$

$$M.SN = 2N$$

Seq. No. - 2

• Rarely used

• more often used due to less complexity

$$\text{Efficiency} = N/(1+2*a)$$

$$\text{Efficiency} = 1/(1+2*a)$$

$$\text{Efficiency} = N/(1+2*a)$$

• Type of Ack = Individual

• T.O.A = Cumulative

• T.O.A = Individual

Number of retransmission in case of Packet drop - 1
(only lost packet)

[The entire window is retransmitted]
N

• only lost packet is retransmitted
1

• Half duplex

• Full duplex

• Full duplex

Cpu usage - low

• Moderate

• High due to searching & sorting required at sender & receiver side

• Receive in order delivery
(Not accept)

• Receive in-order delivery only

• In order delivery only -
out of order

• Simple

• Less complicated

• more complicated

• Bandwidth requirement is low

• Bandwidth requirement is high because even if a single packet is lost, entire window has to be retransmitted, more
Thus error rate is high, it wastes a lot of bandwidth

• Bandwidth requirement is moderate

Date / /

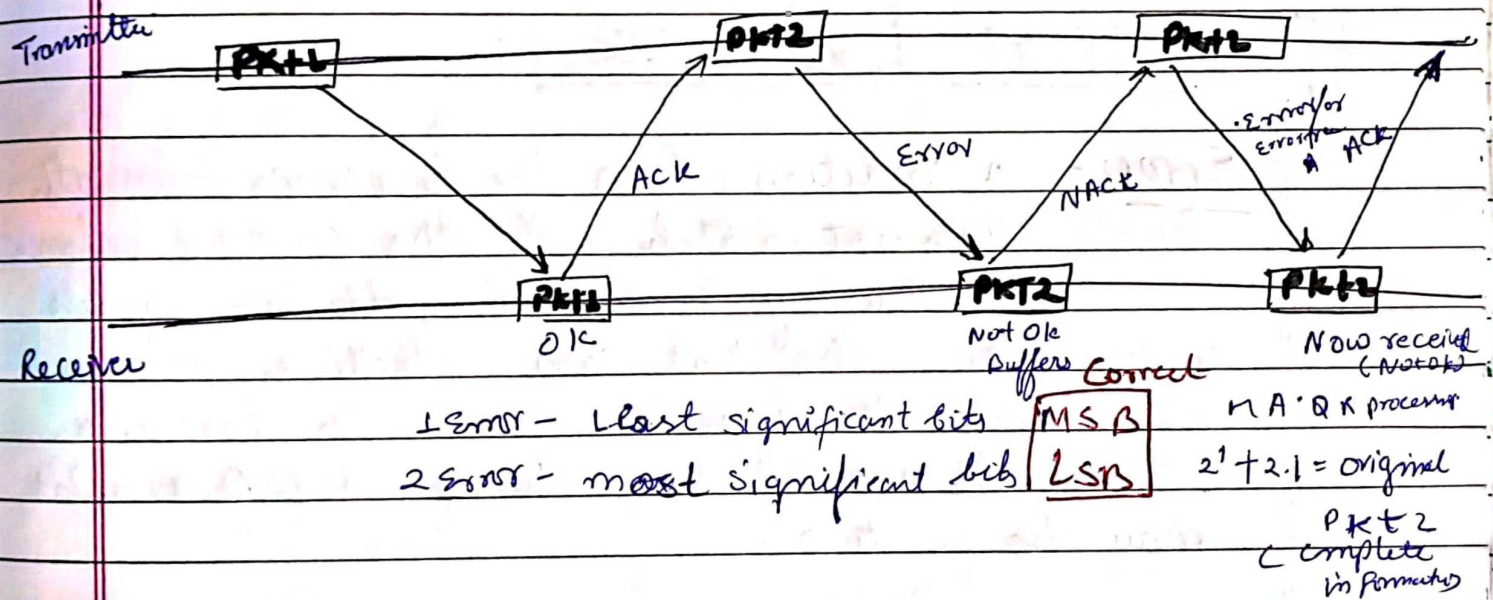
* Hybrid ARQ (HARQ) :-

HARQ stands for Hybrid Automatic Repeat Request.

HARQ = ARQ + FEC (Forward Error Correction)
/ soft combining

Soft combining is an error correction technique in which bad packets are not discarded but stored in a buffer.

The basic idea is that two or more packets received with insufficient information can be combined together in such a way that total signal can be decoded.



The transmitter sends packet 1, if it is received at receiver without error, then receiver sends an ACK signal to sender/transmitter.

Again transmitter sends a packet 2, it arrives but contains error at receiver. so receiver sends a NAK signal to transmitter.

Packet 2 (error) is not thrown away, as it is done

in conventional ARQ. but it is stored in a buffer at receiver.

- The transmitter sends another packet 2. It also arrives with error, it is also buffered. (later)

Now, at receiver. If combining these two bad packets (2 + 2.1) provides the complete information then receives send an ACK signal to transmitter otherwise the process must continue & another NACK is sent.

Error Detection:-

Error:- A condition when the receiver's information does not match with the sender's information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from sender to receiver. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Error Detecting codes (Implemented either at Data link layer or Transport Layer of OSI Model) whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if any error

Date / /

has occurred during transmission of the msg.

Basic approach used for error detection is the use of redundancy bits, where additional bits are added to facilitate detection of errors.

Some popular techniques for error detection are:

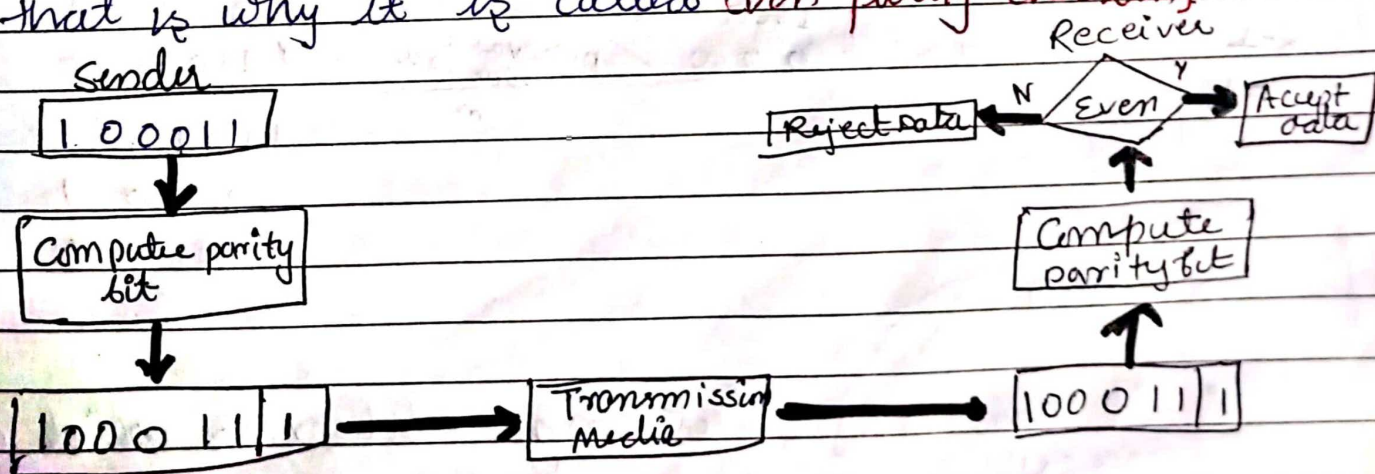
- ① Simple Parity check
- ② Two-dimensional Parity check
- ③ checksum
- ④ Cyclic redundancy check

4) Simple parity check :-

Blocks of data from the source are subjected to a check bit or parity bit generator form, where a parity of:

- 1 is added to the block if it contains odd number of 1's, and
- 0 is added if it contains even number of 1's

This scheme makes that total number of 1's even, that is why it is called even parity checking.



→ message bits
→ $m+1$ bits

→ Even Parity → no of '1' even

→ odd parity → no of '0' odd

eg: $1010 \underline{0}$
 $1110 \underline{1}$

error → $1110 \underline{1}$

(i) $0110 \underline{1}$ → detect error.

(ii) $0010 \underline{1}$ → error b by detect no brc but it detect it correct.

(iii) $0000 \underline{1}$ → error

→ Can detect all single bit error in code word

→ Can detect all odd no of errors also

Hamming distance:-

XOR
min = 2

0000

0001

1111 H.D = 4

0101

1000

1101 → H.D = 3

valid code word

00000

00011

00101

00110

01001

01010

01100

01111

10001

10010

10100

10111

11000

11011

11101

11110

suppose

$\Rightarrow d$

$[d-1]$

2-1

= 1

detect

2 bit change

$d = 11000$

✓ H in valid code word

1 bit change

$d = 10000$

✓ min valid code word

Drawbacks:-

- It can only detect single-bit error which are rare or (i)
- If two bit are interchanged, then it cannot detect errors (eg) (ii)

② Two-D parity check:-

Parity check bits are calculated for each row, which is equivalent to a simple parity check bit.

Parity check bits are also calculated for all columns, then both are sent along with the data. At the receiving end these are compared with the parity bits calculated on the received data.

original data

10011001	11100010	00100100	10000100
----------	----------	----------	----------

Drawbacks:-

If two bits in one data unit are corrupted & two bits exactly the same position in another data unit are also corrupted, then 2D parity checked will not be able to detect the error.

Column →
parities

10011001	10
11100010	0
00100100	0
10000100	0
11011011	0

Row parities

This technique cannot be used to detect the 4-bit errors or more in some cases.

100110010	111000100	001001000	100001000	110110110
-----------	-----------	-----------	-----------	-----------

Data to be sent

③ Checksum:- In checksum error detection scheme, the data is divided into K segments each of m bits.

- In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.
- At the receiver's ~~send~~ end, all received segments are

$$\begin{aligned} 0+0 &= 0 \quad 0 \rightarrow \text{carry} \\ 0+1 &= 1 \quad 0 \\ 1+0 &= 1 \quad 0 \\ 1+1 &= 0 \quad 1 \\ 1+1+1 &= 1 \end{aligned}$$

$$\begin{aligned} 0-0 &= 0 \quad \text{Sub} \quad 0 \rightarrow \text{borrow} \\ 0-1 &= 1 \quad 0 \\ 1-0 &= 1 \quad 0 \\ 1-1 &= 0 \quad 0 \end{aligned}$$

Date / /

added using 1's complement arithmetic to get the sum. The sum is complemented.

- If the result is zero, the received data is accepted otherwise discarded.

10011001 11100010 00100100 10000100

Original Data

10011001 11100010 00100100 10000100

I Method

k=4, m=8

Sender

Receiver

1 10011001

1 10011001

2 11100010

2 11100010

0 01111011

0 01111011

0 01111000

0 01111000

3 00100100

3 00100100

1 01000000

1 01000000

4 100000100

4 100000100

0 00100100

0 00100100

sum : 00100101

checksum : 11011010

checksum : 11011010

sum : 11111111

complement : 00000000

Conclusion: Accept data

- Check+Sum
 - Sender side → checksum Creation / checksum Generator
 - Receiver side → checksum validation / checksum checker
1. Break the original message into 'k' number of blocks with 'n' bits in each block.
 2. sum all the 'k' data blocks
 3. Add the carry to the sum, if any.
 4. Do 1's complement to the sum = checksum.

Date / /

IInd Method :-

Sender

1 0 0 1 1 0 0 1

1 1 1 0 0 0 1 0

0 0 1 0 0 1 0 0

1 0 0 0 0 1 0 0

1 0 0 0 1 0 0 0 1 1

1 0

Sum: 0 0 1 0 0 1 0 1

checksum: 1 1 0 1 1 0 1 0

- collect all the data blocks including the checksum
- Sum all the data blocks & checksum.
- If the result is all 1's Accept; Else, Reject.

1 0 0 1 1 0 0 1

1 1 1 0 0 0 1 0

0 0 1 0 0 1 0 0

1 0 0 0 0 1 0 0

1 0 0 0 1 0 0 0 1 1

1 0

Sum: 0 0 1 0 0 1 0 1

checksum: 1 1 0 1 1 0 1 0

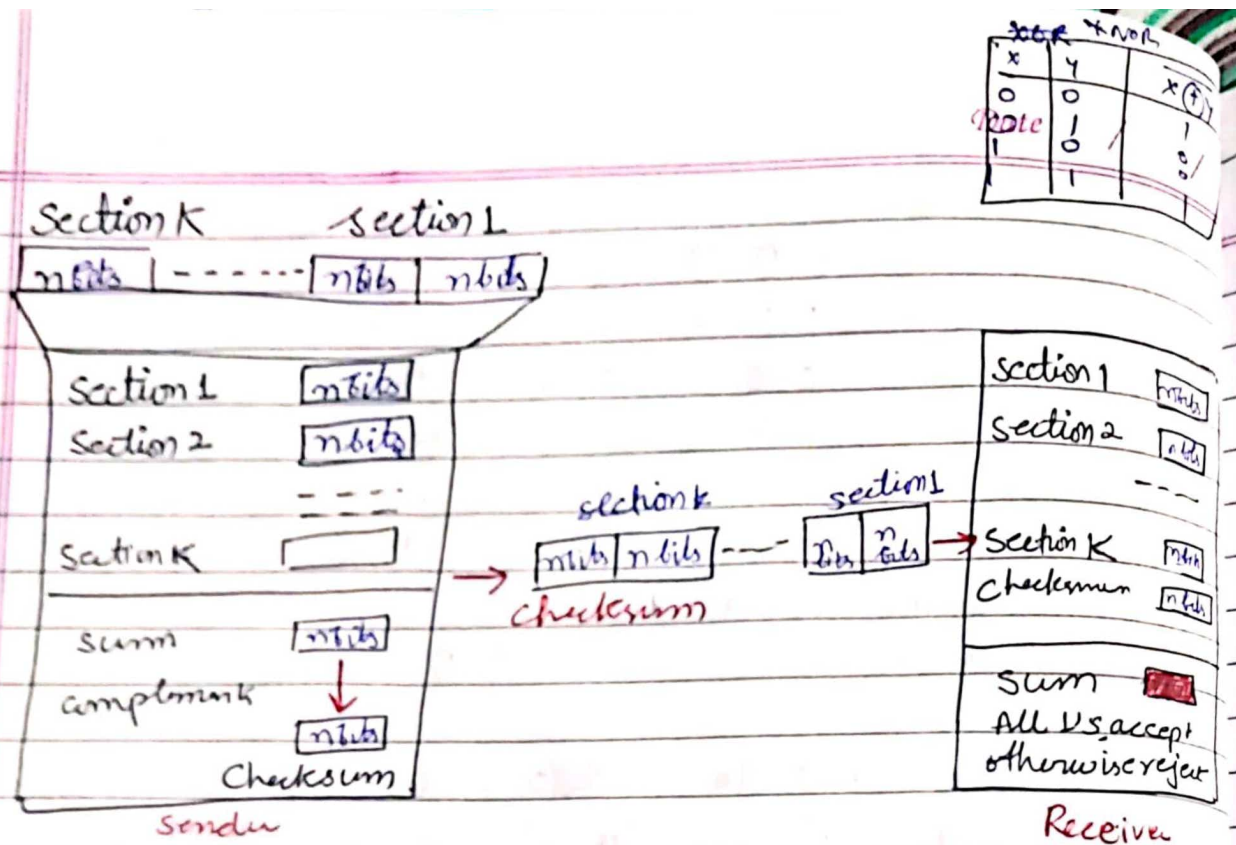


1 1 1 1 1 1 1 1

Performance of checksum:-

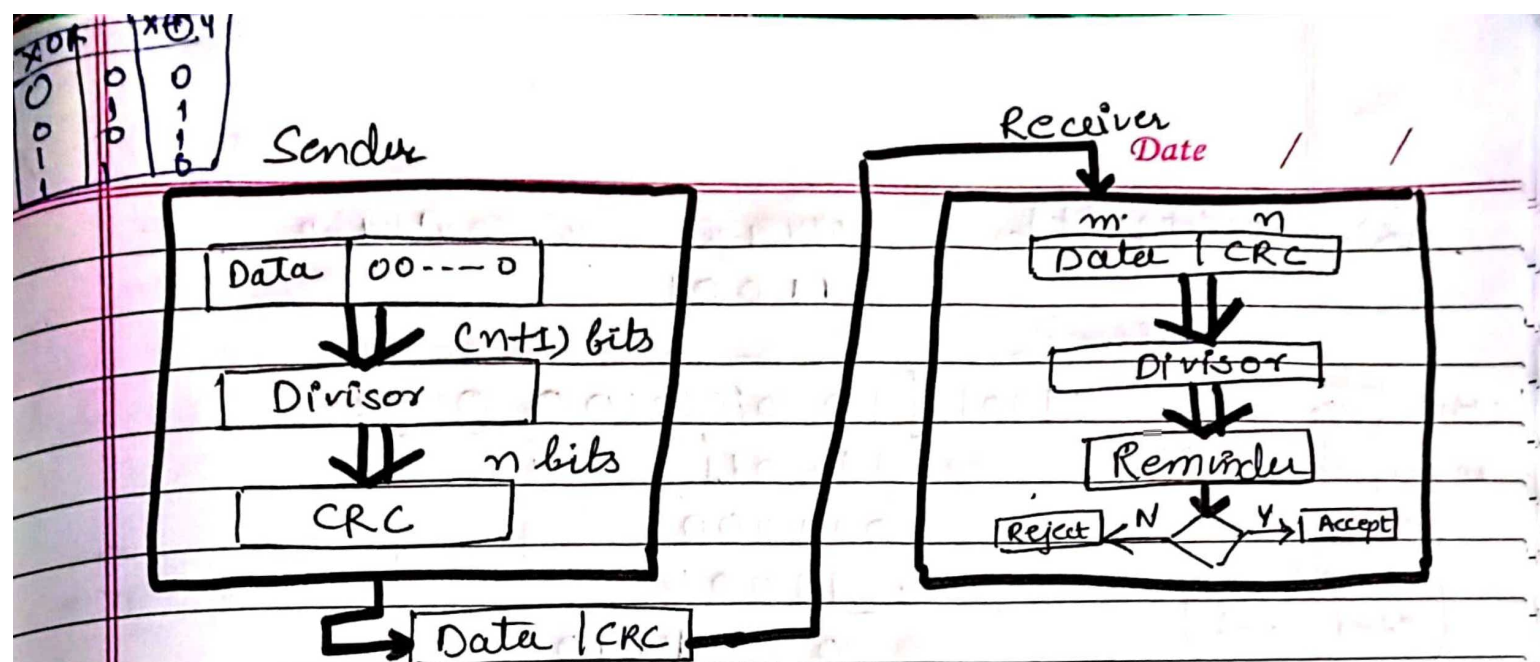
no error.

- The checksum detects all errors involving an odd no. of bit
- It detects most errors involving an even number of bits.
- If one or more bits of a segment are damaged and the corresponding bit or bits of opposite value in a second segment are also damaged, the sum of those columns will not change and the receiver will not detect the error(s).



④ Cyclic Redundancy Check (CRC)

- Unlike checksum scheme, which is based on addition, CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the 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 correct & is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit & therefore must be rejected.



Example:-

Original message

1010000

Generator polynomial
 $x^3 + 1$

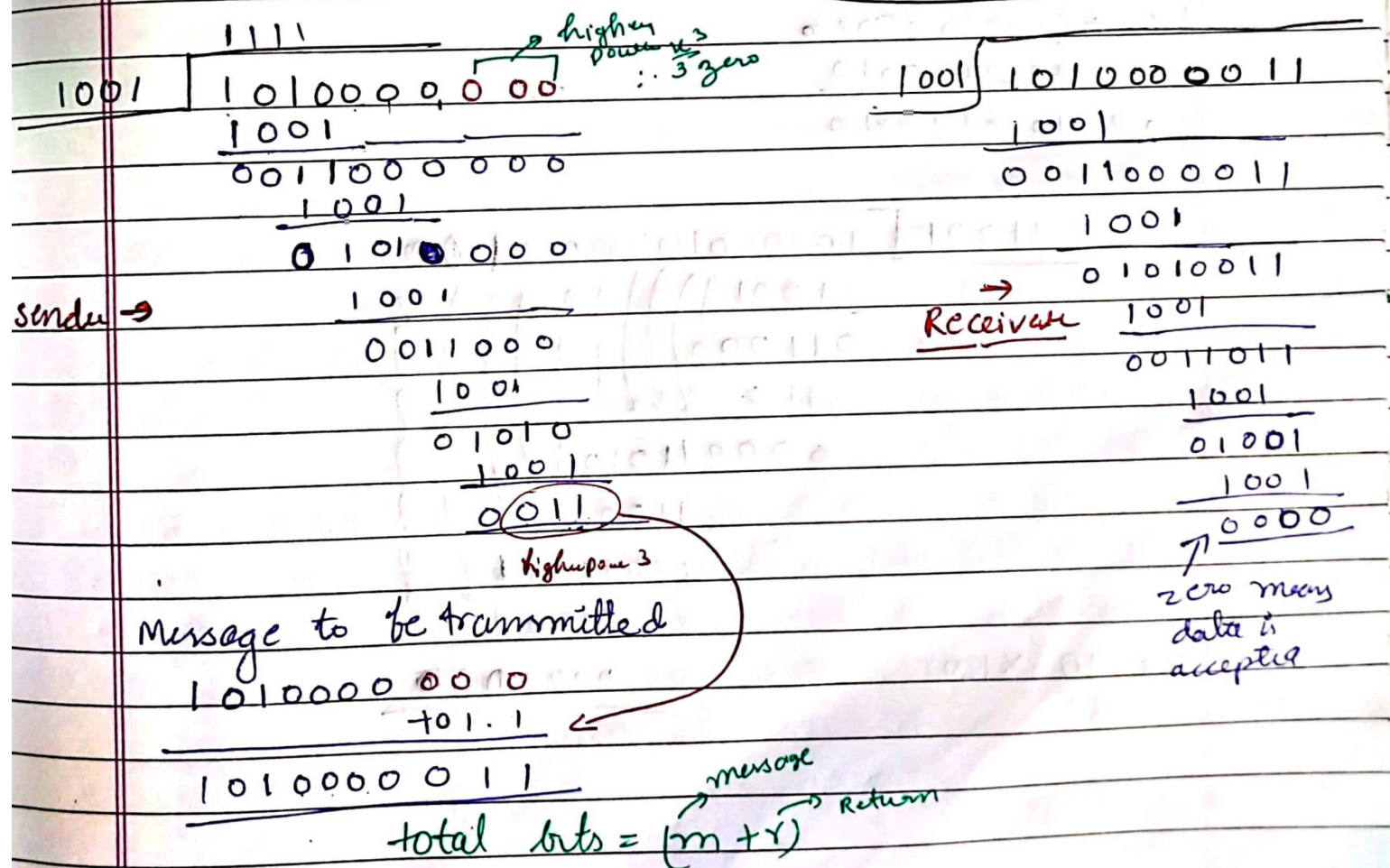
$1 \cdot x^3 + 0x^2 + 0x^1 + 1 \cdot x^0$

CRC generator

1001 4-bits

If CRC generator is of n bit then append $(n-1)$ zeros in the end of original message

XOR



(2). $x^4 + x^3 + 1$ $1x^4 + 1x^3 + 0x^2 + 0x + 1x^0$
11001

if no. is given
then (total no. - 1)
in polynomial higher
power
(Divisor/
total no. - 1)

$$\begin{array}{r} 11001 \overline{) 101010100000} \\ \underline{11001} \\ 011000 \\ \underline{11001} \\ 00001010 \\ \underline{11001} \\ 00011000 \\ \underline{11001} \\ 000010 \end{array}$$

$$\begin{array}{r} 101010100000 \\ \underline{1010000000} \\ 101010100000 \end{array}$$

$$\begin{array}{r} 11001 \overline{) 101010100000} \\ \underline{11001} \\ 011000 \\ \underline{11001} \\ 00001010 \\ \underline{11001} \\ 00011000 \\ \underline{11001} \\ 000010 \end{array}$$

Efficiency = $\frac{10}{14} \times 100$

Date / /

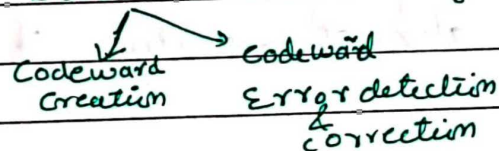
Error Correction:-

Error correction are used to detect & correct the errors when data is transmitted from the sender to the receiver.

Error correction can be handled in two ways:-

- Backward error correction:- Once the error is discovered, the receiver request the sender to retransmit the entire data unit.
- Forward error correction:- In this case, the receiver uses the error - correcting code which automatically corrects the errors.

Code = Data bits + Parity (Redundant)



For correcting the errors, one has to know the exact position of the error.

eg:- If we want to calculate a single-bit error, the error correction code will determine which one of seven bit is in error. To achieve this, we have to add some additional redundant bits.

Suppose r is the number of redundant bits and d is the total number of the data bits. The number of redundant bits r can be calculated by using the formula:

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

Hamming Code :-

Parity bits :- The bit which is appended to the original data of binary bits so that the total number of 1s is even or odd.

Even parity :- To check for even parity, if the total number of 1s is even, then the value of the parity bit is 0, if the total number of 1s occurrence is odd, then the value of the parity bit is 1.

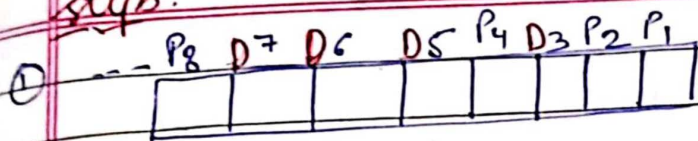
Odd parity :- To check for odd parity, if the total number of 1s is even, then the value of parity bits is 1. If the total number of 1s is odd, then the value of parity bit is 0.

Algorithm :-

- An information of 'd' bits are added to the redundant bits 'r' to form $d+r$.
- The location of each of the $(d+r)$ digits is assigned a decimal value.
- The 'r' bits are placed in the positions $1, 2, \dots, 2^{r-1}$.
- At the receiving end, the parity bits are recalculated. The decimal value of the parity bits determines the positions of an error.

Date / /

steps:



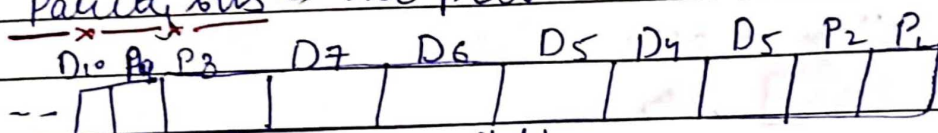
② Mark all bit positions that are power of two as parity bits

$$2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, \dots$$

$$(1, 2, 4, 8, \dots)$$

③ All other bit positions are used for data (original bits).

Parity bits $\Rightarrow$ use positions



P₁ $\Rightarrow$ check 1, skip 1, ^{check 1} skip 1.
(1, 3, 5, 7, 9, \dots)

P₂ $\Rightarrow$ check 2, skip 2, check 2, skip 2,
= (2, 3, 6, 7, 10, 11, \dots)

P₄ $\Rightarrow$ check 4, skip 4, check 4, skip 4,
= (4, 5, 6, 7, 12, 13, 14, 15, \dots)

⑤ Set a parity bit 1 $\Rightarrow$ if total no. of 1's in positions it checks is odd.
0 $\Rightarrow$ if total no. of 1's in positions it checks is even.

Calculate the no. of Parity (Redundant) bits

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

$\downarrow$ data bits $\rightarrow$ redundant bit

eg. If $m=4$

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

$$2^r \geq 5 + r$$

$$2^3 \geq 5 + 3$$

$$2^3 \geq 8$$

$$\boxed{r=3} \text{ bits}$$

eg. (2)

$m=7$

$$2^r \geq 7 + r + 1$$

$$2^r \geq 8 + r$$

$$2^4 \geq 8 + 4$$

$$\boxed{r=4} \text{ bits}$$

Que)

A bit word 1011 is to be transmitted.
Construct the error parity seven bit hamming
Code for this data.

Ans)

$m=4$ bit (1011)

$r=?$

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

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

$$2^r \geq 5 + r$$

$$\boxed{r=3} \text{ bits}$$

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

$$P_1 \rightarrow P_1, D_3, D_5, D_7$$

$$? \ 1 \ 1 \ 1$$

$$\boxed{1} \ 1 \ 1 \ 1 \text{ (even parity)}$$

$$P_1 \rightarrow D_3 \oplus D_5 \oplus D_7$$

$$= 1 \oplus 1 \oplus 1$$

$$= 0 + 1 = 1$$

Date / /

$$P_2 \Rightarrow P_2 D_3 D_6 D_7$$

$$P_2 \quad 1 \quad 0 \quad 1$$

$$0 \quad 1 \quad 0 \quad 1 \quad (\text{even parity})$$

$$P_2 \Rightarrow D_3 \oplus P_6 \oplus D_7$$

$$= 1 \oplus 0 \oplus 1$$

$$= 1 \oplus 1 = 0$$

$$P_4 \Rightarrow P_4 D_5 D_6 D_7$$

$$P_4 \quad 1 \quad 0 \quad 1$$

$$0 \quad 1 \quad 0 \quad 1$$

$$\text{Hamming code} = 1010101$$

Ques) If 7 bit hamming code is received as 1011011. Assuming even parity state whether it is correct or not? if not locate the bit in error.

Ans

$$D_7 \quad D_6 \quad D_5 \quad P_4 \quad D_3 \quad P_2 \quad P_1$$

$$1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1$$

Error word

$$P_4 \quad P_2 \quad P_1$$

$$1 \quad 0 \quad 1$$

$$P_4 = P_4 D_5 D_6 D_7$$

$$1 \quad 1 \quad 0 \quad 1 \quad (\text{not even parity}) \quad \text{error}$$

$$P_2 = P_2 D_3 D_6 D_7$$

$$1 \quad 0 \quad 0 \quad 1 \quad (\text{even parity}) \quad \text{no error}$$

$$P_2 = 0$$

$$P_1 = P_1 D_3 D_5 D_7$$

$$1 \quad 0 \quad 1 \quad 1 \quad (\text{not even parity}) \quad \text{error}$$

$$P_1 = 1$$

$$(101)_2$$

$$\Rightarrow 5^{\text{th}} \text{ bit}$$

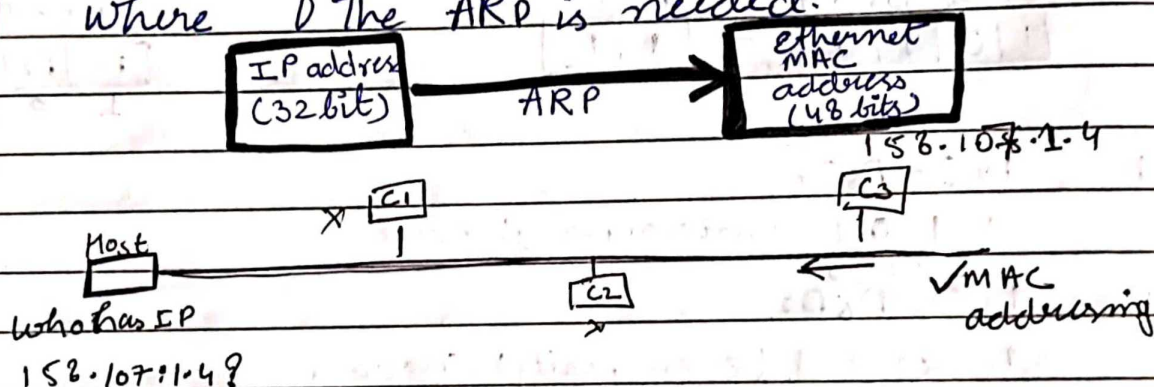
$$\text{corrected code word} = 1001011$$

Protocol ARP/RARP/GARP

* ARP (Address Resolution protocol)

IP & MAC
logical physical
address address
universal

- It is a simple communications protocol used in IP/Ethernet networks
- It's main purpose is to discover & associate IP addresses to physical MAC hardware addresses.
 - When a computer needs to send data across a network to another physical device, it must find the physical address (MAC) address of the destination device. This is where the ARP is needed.



- ARP is used to find the MAC address of device on a network using only the IP address.
- The ARP protocol will make a broadcast out to the network asking for the MAC address of the destination IP address.
- ARP requests are broadcasted & broadcasting is for more expensive, computer maintains a cache of recently acquired IP to physical address mapping. Thus before sending an ARP request, a machine always looks its ARP cache to

Date / /

- find the desired mapping.
- ARP request only broadcasted if no mapping found in the cache.

MAC address
Ethernet - 6 bytes or 48 bits } DLL protocols may have different MAC addresses
Token ring - 2 or 6 bytes
FDDI - 2 or 6 bytes

- The machine with the IP address will respond with its MAC address.
- It is used by the Internet Protocol (IP) specifically IPv4 to map IP network address to the hardware addresses.
- It is used by a data link protocol.
- when ARP returns a MAC address, it is placed in a cache table. When the next request comes in for the same IP address look first in the cache table.

* RARP (Reverse ARP) :-

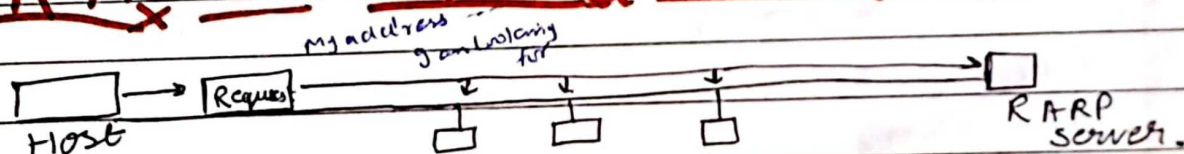


Fig: RARP request in broadcast

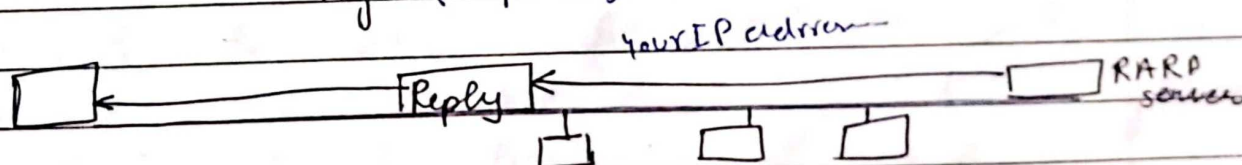
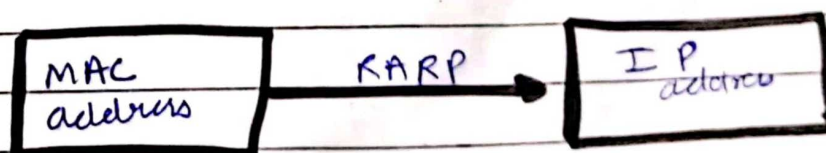


Fig RARP reply is unicast



(host)

- It is used when device does not know its own IP address.
- RARP originally developed for diskless work station.
- Reverse RARP is a networking protocol used by client machine in a LAN to request its IP address from the gate way - router's ARP table.

- The network administrator creates a table inside RARP server in the router to map the MAC address to corresponding IP address.

- When a new machine is setup or any machine which don't have memory to store IP address, it can dynamically find IP address when address is known.

Sender broadcast a RARP request, supplying its physical network address in the target header address field.

- Only to machines authorized to supply the RARP service (RARP services) process the request & send a reply filling in the target IP address.

Date / /

ARP

- ① ARP stands for Address Resolution Protocol.
- ② Through ARP, (32-bit) IP address mapped into (48-bit) MAC address.
- ③ In ARP, broadcast MAC address is used.
- ④ ARP table is managed or maintained by local host.
- ⑤ ~~Protocol~~, Receiver's MAC address is fetched.
- ⑥ ARP table uses ARP reply for its update.
- ⑦ Hosts & routers use ARP for knowing the MAC addresses of other hosts & routers in the networks.

RARP

- ① RARP stands for Reverse Address Resolution Protocol.
- ② Through RARP, (48-bit) MAC address of 48 bits mapped into (32-bit) IP address.
- ③ While in RARP broadcast IP address is used.
- ④ RARP table is managed or maintained by RARP server.
- ⑤ IP address is fetched.
- ⑥ RARP table uses RARP reply for configuration of IP addresses.
- ⑦ RARP is used by small users having less facilities.

* GARP (Gratuitous ARP)

- A gratuitous ARP is an ARP response that was not prompted by an ARP request.
 - The GARP is sent as a broadcast, as a way for a node to announce or update its IP to MAC mapping to the entire network.
 - No reply is expected for a GARP.
 - Let one computer in network changes its IP address for some technical requirement.
 - The change in IPv4 address must be informed to all other devices in the LAN because there may be MAC address to incorrect previous IPv4 address mapping inside the ARP table of other devices.
 - Gratuitous ARP is mainly used by a TCP/IP device to inform other device in the LAN, any change in its MAC address or IPv4 address.
- Since the destination MAC address is the broadcast MAC address, the switch will flood the GARP packet to all its connected ports.
- GARP is often performed by a computer when it is first booted up. When a NIC's is first powered on, it will automatically out its MAC address to the entire network.

Date / /

- It allows my switches to know the location of the physical devices & DHCP server to know where to send an IP address if needed & requested.