Name of Teacher: Miss Radhika M. Patil

Class: B.Sc. Computer Science (Entire)- III **Semester : 5**

Course Title: Data Communication

UNIT 4
Data Link Layer

- In the OSI model, the data link layer is a 4th layer from the top and 2nd layer from the bottom.
- It provides a well defined service interface to the network layer.
- It deals with the transmission errors.
- It regulates the flow of data so that slow receivers are not swamped by fast senders.

To accomplish these goals, the DLL takes the packets from network layer and encapsulates them into frames for transmission.

- **Design Issues of Data Link Layer:**

The main functions and the design issues of this layer are

1. Providing services to the network layer
2. Framing
3. Error Control
4. Flow Control

1. Providing services to the network layer

- In the OSI Model, each layer uses the services of the layer below it and provides services to the layer above it.
- The data link layer uses the services offered by the physical layer.
- The primary function of this layer is to provide a well defined service interface to network layer above it.

The types of services provided can be of three types −

1. Unacknowledged connectionless service
2. Acknowledged connectionless service
3. Acknowledged connection - oriented service

1. Unacknowledged connectionless service :

- Here, the data link layer of the sending machine sends independent frames to the data link layer of the receiving machine.
- The receiving machine does not acknowledge receiving the frame.
- No logical connection is set up between the host machines.
- Error and data loss is not handled in this service.
- This is applicable in Ethernet services and voice communications.

2. Acknowledged connectionless service :

- Here, no logical connection is set up between the host machines, but each frame sent by the source machine is acknowledged by the destination machine on receiving.
- If the source does not receive the acknowledgment within a specified time, then it resends the frame.

3. Acknowledged connection-oriented service

- This is the best service that the data link layer can offer to the network layer.
- A logical connection is set up between the two machines and the data is transmitted along this logical path.
- The frames are numbered, that keeps track of loss of frames and also ensures that frames are received in correct order.

The service has three distinct phases −

1. **Set up of connection –** A logical path is set up between the source and the destination machines. Buffers and counters are initialized to keep track of frames.
2. **Sending frames –** The frames are transmitted.
3. **Release connection –** The connection is released, buffers and other resources are released.

It is appropriate for satellite communications and long-distance telephone circuits.

2. Framing:

- Data-link layer takes the packets from the Network Layer and encapsulates them into frames.
- If the frame size becomes too large, then the packet may be divided into small sized frames.
- Smaller sized frames makes flow control and error control more efficient.
- Frame Management forms the heart of DLL.

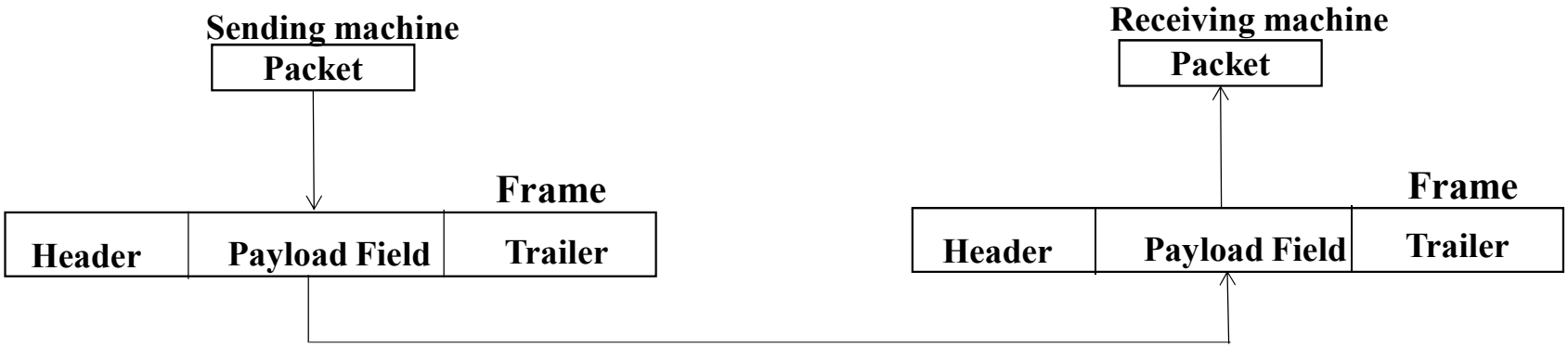


Fig. Relationship between packet and frame

- **Parts of a Frame**

A frame has the following parts −

1. **Frame Header** − It contains the source and the destination addresses of the frame.
2. **Payload field** − It contains the message to be delivered.
3. **Trailer** − It contains the error detection and error correction bits.

3. Error Control

- The data link layer ensures error free link for data transmission. The issues it caters to with respect to error control are −
- Dealing with transmission errors
- Sending acknowledgement frames in reliable connections
- Retransmitting lost frames
- Identifying duplicate frames and deleting them
- Controlling access to shared channels in case of broadcasting

4. Flow Control

- The data link layer regulates flow control so that a fast sender does not drown a slow receiver. When the sender sends frames at very high speeds, a slow receiver may not be able to handle it.
- There will be frame losses even if the transmission is error-free.
- The two common approaches for flow control are −

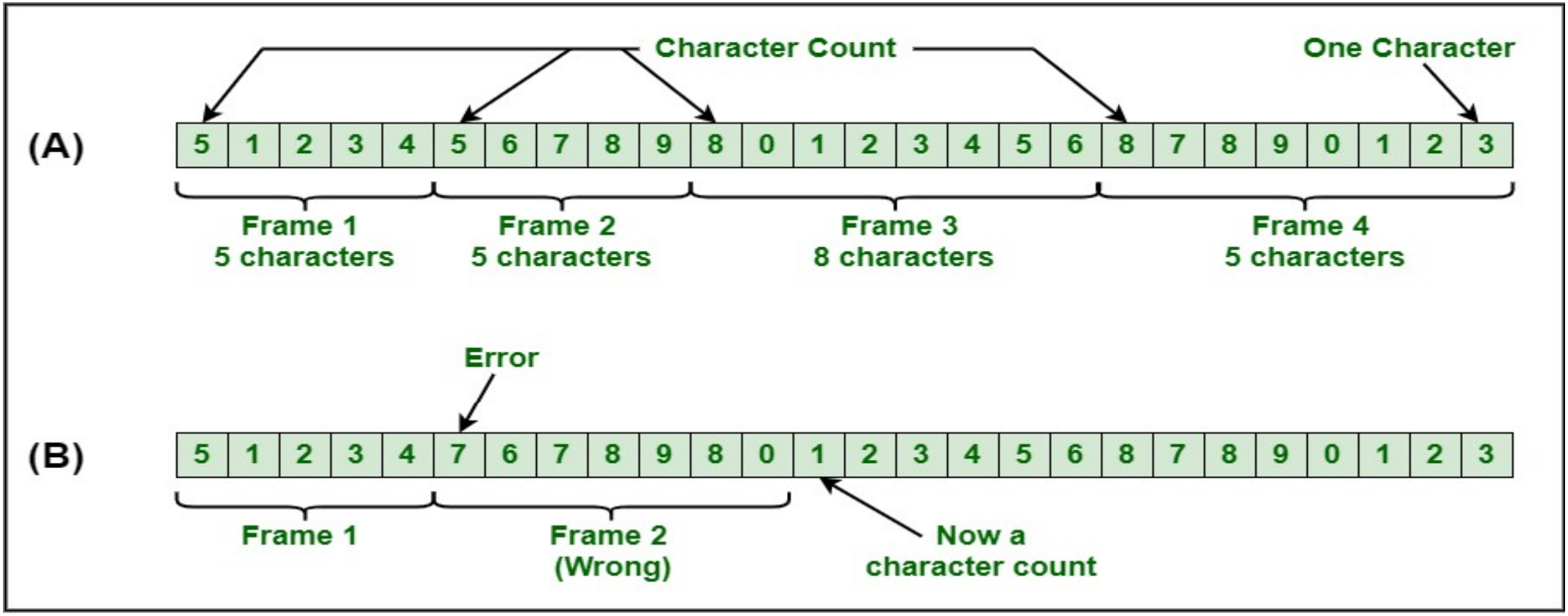
1. **Feedback based Flow Control**
   In these protocols, the sender sends frames after it has received acknowledgments from the user.
   This is used in the data link layer.
2. **Rate based Flow Control**
   These protocols have built in mechanisms to restrict the rate of transmission of data without requiring acknowledgment from the receiver. This is used in the network layer and the transport layer

- **Framing Methods:**

1. Character count/Byte Count
2. Flag Byte with byte Stuffing
3. Starting and Ending flags with bit stuffing
4. Physical Layer Coding Violation.

1. Character count/Byte Count:

- This method uses a field (**character count**) in the header to specify the **number of characters** in the frame.
- This character count or byte count at destination indicates that how many characters that the frame follow and where the end of the frame is.
- This technique is shown in fig. A for four frames of sizes 5, 5, 8 and 8 characters respectively.
- The disadvantage is that if the count is garbled by a transmission error, the destination will lose synchronization and will be unable to locate the start of the next frame. So, this method is rarely used.
- Checksum is also incorrect so the destination knows that the frame is bad.
- Asking for retransmission is also not possible because destination does not know how many characters to skip over to get to the start of retransmission.
- For this reason this method is rarely used.



A Character Stream

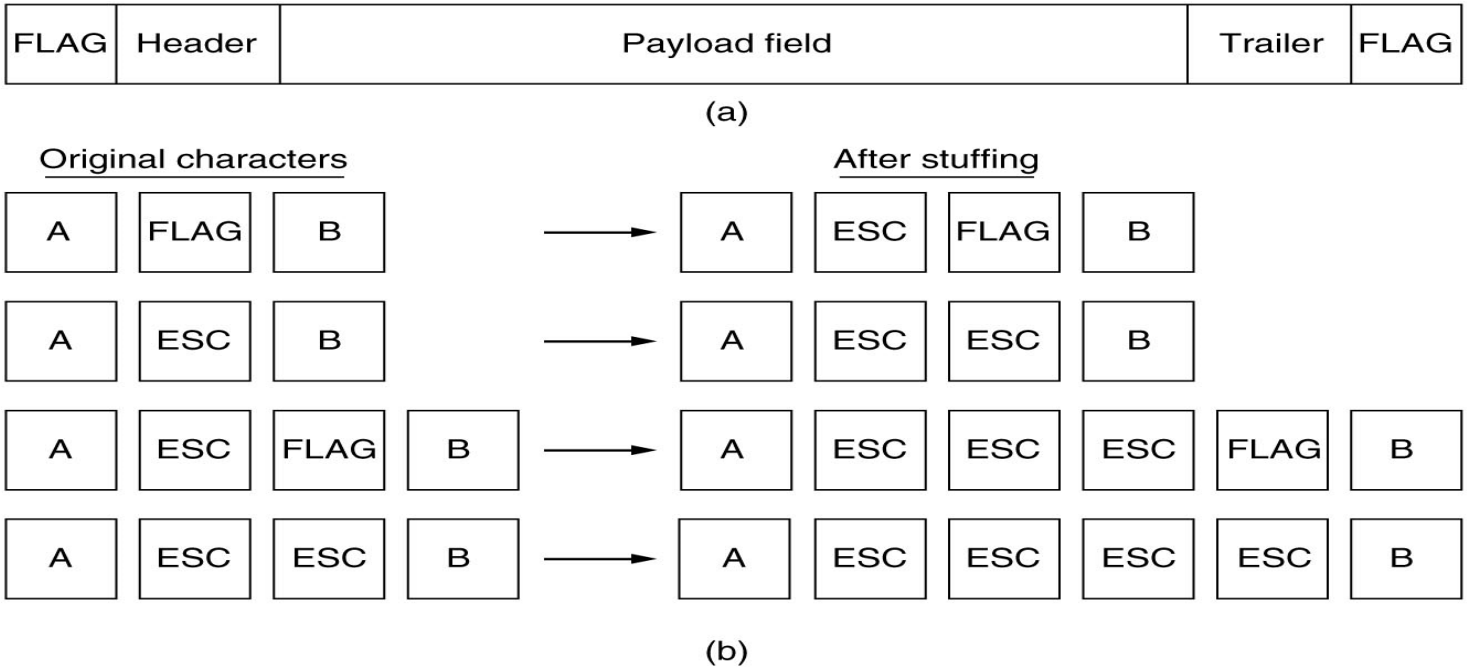
(A) Without Errors
(B) With one Error

2. Flag Byte with Byte Stuffing:

- In this method each frame starts and ends with special byte called Flag byte or FLAG as shown in fig a.
- In this way if the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame.
- Two consecutive flag bytes indicate the end of one frame and start of next frame.

Fig a. A Frame starts and ends with FLAG byte.

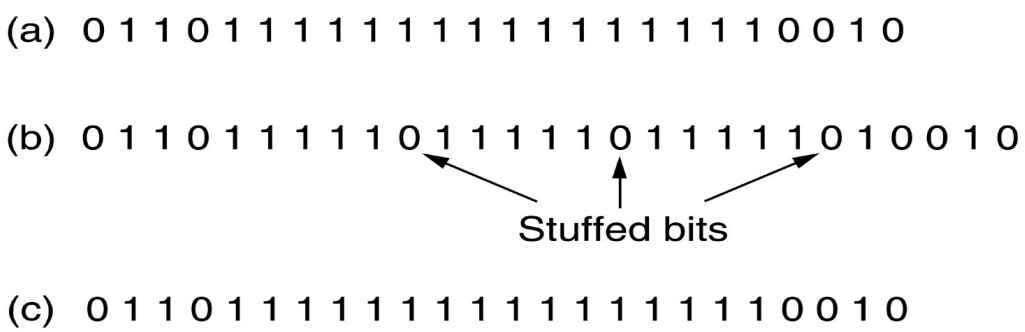
- But the problem occurs with this method when binary data such as object programs / floating point numbers are being transmitted.
- It may easily happen that the flag byte's bit pattern occurs in the data.
- To solve this problem senders DLL insert a special escape (ESC) byte just before each accidental FLAG byte in the data.
- The DLL on receiving end removes the ESC byte before the data given to the network layer.
- This technique is called **Byte Stuffing or Character Stuffing.**
- Thus a framing flag byte can be distinguished from one byte in the data by presence or absence of an ESC byte before it.



- Problem occurs when ESC byte occurs in the middle of the data .
- To solve this problem , an ESC byte is again stuffed with ESC byte.
- Thus, any single ESC byte is part of ESC sequence, whereas a double one indicates that a single ESC occurred naturally in the data as shown in fig b.

3. Starting and Ending flags with bit stuffing:

- This technique allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary no.of bits per character.
- Each frame begins and ends with a special bit pattern , 0111110 (in fact , a flag byte).
- Whenever senders DLL encounters 5 consecutive 1's in data it automatically stuffs a 0 bit into outgoing bit stream.
- This bit stuffing is analogous to byte stuffing in which an ESC byte is stuffed into outgoing character stream before a flag byte in the data.
- When receiver sees 5 consecutive incoming 1 bit followed by a 0 bit it automatically destuffs (i.e. deletes) the 0 bit.
- If the user data contains the flag pattern i.e. 01111110 then this flag is transmitted as 011111010 but stored in receivers memory as 01111110.



(a) Original data.

(b) Stuffed data transmitted.

(c) De-stuffed data received.

4. Physical layer coding violations

1. The final framing method is physical layer coding violations and is applicable to networks in which the encoding on the physical medium contains some redundancy.
2. In such cases normally, a 1 bit is a high-low pair and a 0 bit is a low-high pair.
3. This means that every data bit has a transition in middle, making it easy for receiver to locate the bit boundaries.

- **Error Detection and Correction:**

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

- **Types of Errors:**

1. Single bit error

In a frame, there is only one bit, anywhere though, which is corrupted

2. Multiple bits error

Frame is received with more than one bits in corrupted state.

3. Burst error

Frame contains more than1 consecutive bits corrupted.

- **Error Detection :**

➢ Error detection, as name suggests, simply means detection or identification of errors.

➢ These errors may cause due to noise or any other impairments during transmission from transmitter to the receiver, in

➢ communication system.

➢ It is class of technique for detecting garbled i.e. unclear and distorted data or message.

- **Error Detection Methods :**

1. Parity Check
2. Checksum
3. Cyclic Redundancy Check (CRC)

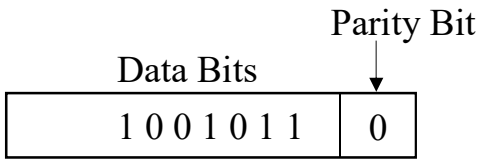
1. Parity Check:

- The parity check is done by adding an **extra bit, called parity bit** to the data to make a number of 1s either even in case of even parity or odd in case of odd parity.
- While creating a frame, the sender counts the number of 1s in it and adds the parity bit in the following way
- **In case of even parity:** If a number of 1s is even then parity bit value is 0. If the number of 1s is odd then parity bit value is 1.
- **In case of odd parity:** If a number of 1s is odd then parity bit value is 0. If a number of 1s is even then parity bit value is 1.
- On receiving a frame, the receiver counts the number of 1s in it. In case of even parity check, if the count of 1s is even, the frame is accepted, otherwise, it is rejected. A similar rule is adopted for odd parity check.
- The parity check is suitable for **single bit error detection** only

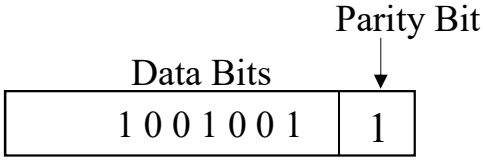
- **Even parity:**

- Even parity means the **number of 1's** in the given data **including parity** bit should be **even.**
- For even parity the parity bit is set to 1 or 0 such that the no.of 1's in the entire data is even.

Example:



OR

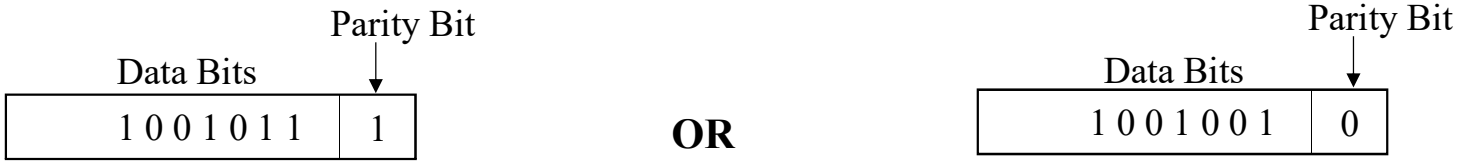


- **Odd parity:**

➢ Odd parity means the **number of 1's** in the given data **including parity** bit should be odd**.**

➢ For odd parity the parity bit is set to 1 or 0 such that the no.of 1's in the entire data is odd.

Example:



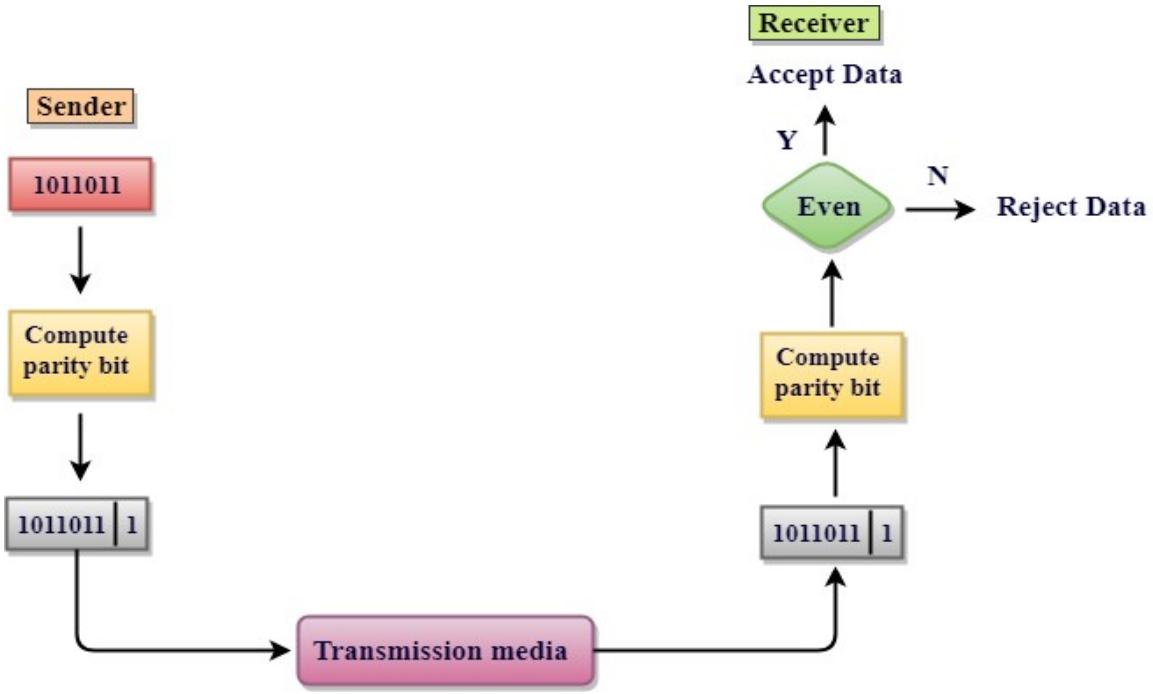
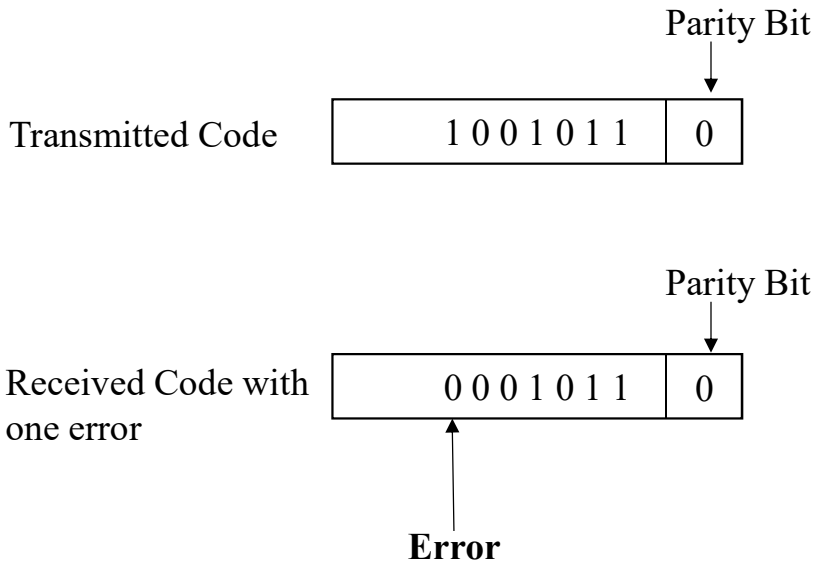


Fig. Parity Checking (Even parity is considered here)

- Parity checking at receiver can detect the presence of an error if the parity of the receiver signal is different from expected parity.
- This means that if it is known that the parity of the transmitted signal is always going to be even and if the received signal has an odd then the receiver can conclude that the received signal is not correct.
- If an error is detected then the receiver will ignore the received byte and request for retransmission of same byte to the sender.

Example:



2. Checksum:

In this method :

- **Sender follows the following steps:**

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

- **Receiver follows the following steps:**

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

Example:

Consider the following block of 16 bits is to be sent using a checksum of 8 bits.

Data: 1 0 1 0 1 0 0 1 0 0 1 1 1 0 0 1

the numbers are added using one's complement as

```
  1 0 1 0 1 0 0 1
  0 0 1 1 1 0 0 1
-----------------
  1 1 1 0 0 0 1 0
```

Sum is 1 1 1 0 0 0 1 0

Complement this sum to get checksum

So

Checksum= 0 0 0 1 1 1 0 1

The pattern sent is

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

Now suppose , the receiver receives the pattern sent in above example and there is no error

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

When receiver adds the three sections, it will get all 1's which after complementing is all zeros (0's) which shows that there is no error.

1 0 1 0 1 0 0 1
0 0 1 1 1 0 0 1
0 0 0 1 1 1 0 1
Sum 1 1 1 1 1 1 1 1
Complement= 0 0 0 0 0 0 0 0

This means pattern is error free.

3. Cyclic Redundancy Check (CRC):

1. It is most commonly used method.
2. CRC is a different approach to detect if the received frame contain valid data.
3. This technique involves binary division of data bits being sent.
4. The divisor is generated using polynomials
5. The sender performs a division operation on bits being sent and calculates the **remainder.**
6. Before sending the actual data or bits, the sender adds the remainder at the end of actual bits.
7. In division process **XOR operation** is used **for subtraction.**
8. The actual **data bits plus remainder** is called a **code word.** Sender transmits data bits as code word.
9. Let's assume k message bits and n bits of redundancy.

i.e. XXXXXX / yyyy

k bits n bits

Block of length (k+n)

Let M(X) be the message polynomial.
Let P (X) be the generator polynomial.
Note that,
P (X) is fixed for given CRC scheme and is known by both sender receiver.

10. Create a block polynomial F (X) based on M (X) and P (X) such that F (X) is divisible by P (X).

Where,

F (X) = M (X) + C (X) (remainder after division)

Example:

Send $x^4 + x^3 + 1$ by using the polynomial $x^2 + 1$.

Solution:

Here,

M (X)= $x^4 + x^3 + 1$ i.e. M (X)= 1 1 0 0 1

P (X) = $x^2 + 1$ i.e. P (X) = 1 0 1

Divide M (X) by P (X) to find remainder C (X).

- **At Sender:**

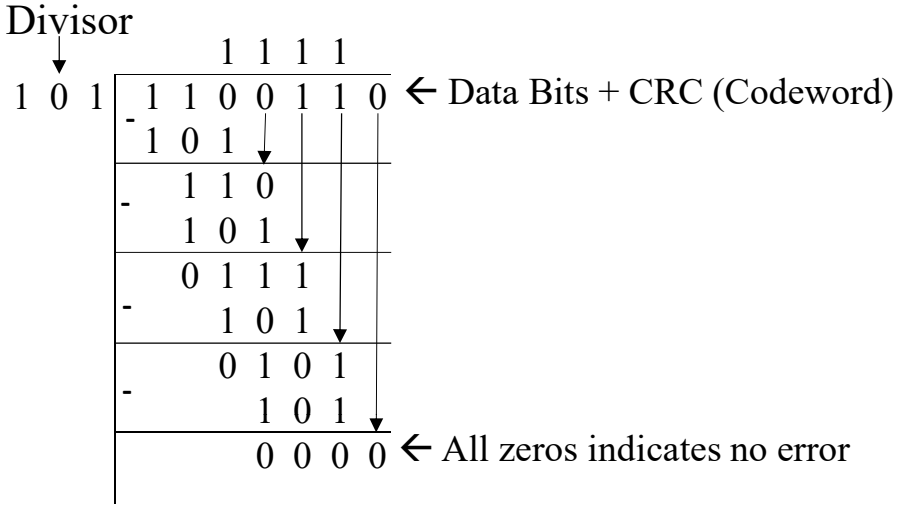
```
Divisor
   ↓          1 1 1
 1 0 1 | 1 1 0 0 1
       - 1 0 1
       ---------
         0 1 1 0
       -   1 0 1
       ---------
         0 1 1 1
       -   1 0 1
       ---------
           0 1 0   ←CRC
```

C(X) =10

F (X)= M (X)+C(X)

F (X)= 1 1 0 0 1 1 0 ← Code word

Now sender sends 1 1 0 0 1 1 0 to the receiver.
At Receiver:
Again divide F(X) by P(X) as:



11. At the other end, the receiver performs division operation on codeword using the same CRC divisor.
12. After division, if the remainder contains all zeros then the data bits are accepted otherwise it is considered as some data corruption occurred in the transmission.

- **Error Correction:**

Error correction can be done in two ways:

1. Backward Error Correction:

When the receiver detects an error in the data received, it requests back the sender to retransmit the data unit.

2. Forward Error Correction:

When the receiver detects some error in the data received, it executes error-correcting code, which helps it to auto-recover and to correct some kinds of errors.

The first one, Backward Error Correction, is simple and can only be efficiently used where retransmitting is not expensive. But in case of wireless transmission retransmitting may cost too much. In the latter case, Forward Error Correction is used.

Error Correction Method:

- **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 occurrences 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 bit is 1. If the total number of 1s is odd, then the value of parity bit is 0

- **Algorithm of Hamming code:**

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

Relationship between Error position & binary number.

Error Position	Binary Number
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Let's understand the concept of Hamming code through an example:

Suppose the original data is 1010 which is to be sent.

Total number of data bits 'd' = 4
Number of redundant bits r :

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

$2^r >= 4+r+1$

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

Total number of bits = d+r = 4+3 = 7;

- **Determining the position of the redundant bits**

➢ The number of redundant bits is 3.

➢ The three bits are represented by r1, r2, r4.

➢ The position of the redundant bits is calculated with corresponds to the raised power of 2.

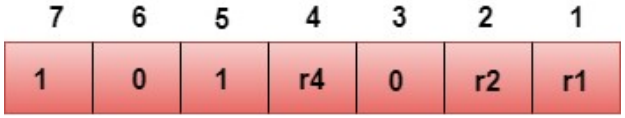
➢ Therefore, their corresponding positions are **1, 2^1, 2^2**.

The position of r1 = 1

The position of r2 = 2

The position of r4 = 4

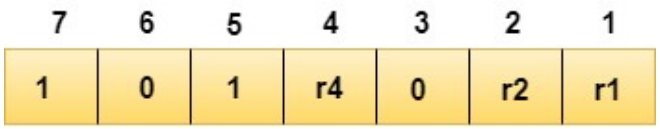
- **Representation of Data on the addition of parity bits:**



- **Determining the Parity bits**
- **At Sender:**

1. Determining the r1 bit

The r1 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the first position.

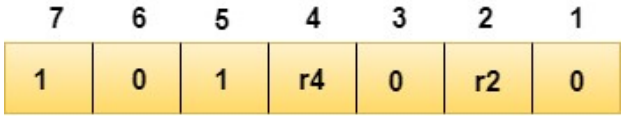


1. We observe from the above figure that the bit positions that includes 1 in the first position are 1, 3, 5, 7.
2. Now, we perform the even-parity check at these bit positions.
3. The total number of 1 at these bit positions corresponding to r1 is **even, therefore, the value of the r1 bit is 0**.

r1 = 0

2. Determining r2 bit:

The r2 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the second position.

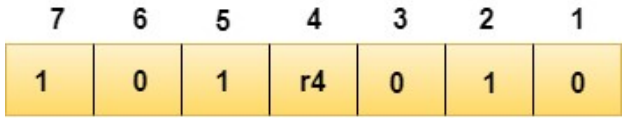


1. We observe from the above figure that the bit positions that includes 1 in the second position are **2, 3, 6, 7**. Now, we perform the even-parity check at these bit positions.
2. The total number of 1 at these bit positions corresponding to r2 is **odd, therefore, the value of the r2 bit is 1**.

r2 = 1

3. Determining r4 bit:

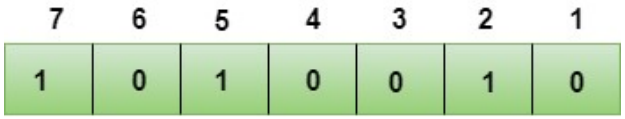
The r4 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the third position.



1. We observe from the above figure that the bit positions that includes 1 in the third position are **4, 5, 6, 7**.
2. Now, we perform the even-parity check at these bit positions.
3. The total number of 1 at these bit positions corresponding to r4 is **even, therefore, the value of the r4 bit is 0**.

r4 = 0

Data transferred is given below:

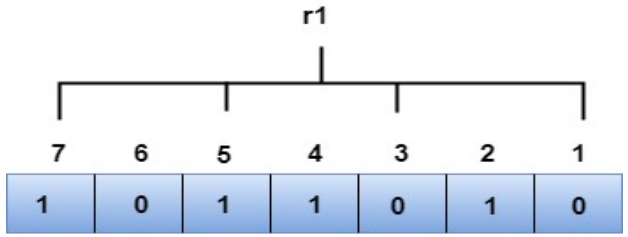


Suppose the 4th bit is changed from 0 to 1 at the receiving end, then parity bits are recalculated.

- **At Receiver:**

R1 bit

The bit positions of the r1 bit are 1,3,5,7

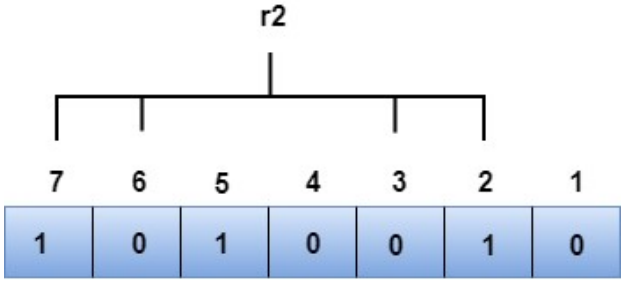


1. We observe from the above figure that the binary representation of r1 is 1100.
2. Now, we perform the even-parity check, the total number of 1s appearing in the r1 bit is an even number.
3. Therefore, the value of r1 is 0.

r1 = 0

R2 bit

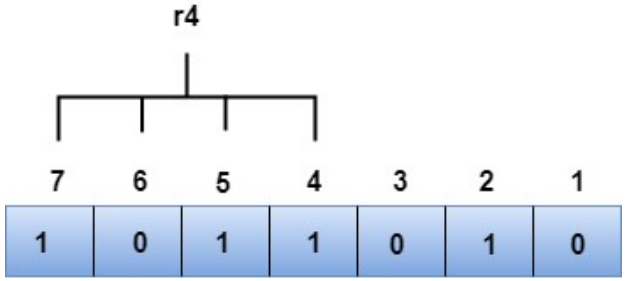
The bit positions of r2 bit are 2,3,6,7.



1. We observe from the above figure that the binary representation of r2 is 1001.
2. Now, we perform the even-parity check, the total number of 1s appearing in the r2 bit is an even number.
3. Therefore, the value of r2 is 0.

R4 bit

The bit positions of r4 bit are 4,5,6,7.



1. We observe from the above figure that the binary representation of r4 is 1011.
2. Now, we perform the even-parity check, the total number of 1s appearing in the r4 bit is an odd number.
3. Therefore, the value of r4 is 1.

r4 = 1

- The binary representation of redundant bits,

  i.e., r4 r2 r1 is 100, and its corresponding decimal value is 4.

  Therefore, the error occurs in a 4^{th} bit position.

  The bit value must be changed from 1 to 0 to correct the error.

THANK YOU…