Error Detection & Correction Methods

Errors

- When bits are transmitted over the computer network, they are subject to get corrupted due to interference and network problems.
- The corrupted bits leads to spurious data being received by the destination and are called errors.

false or fake

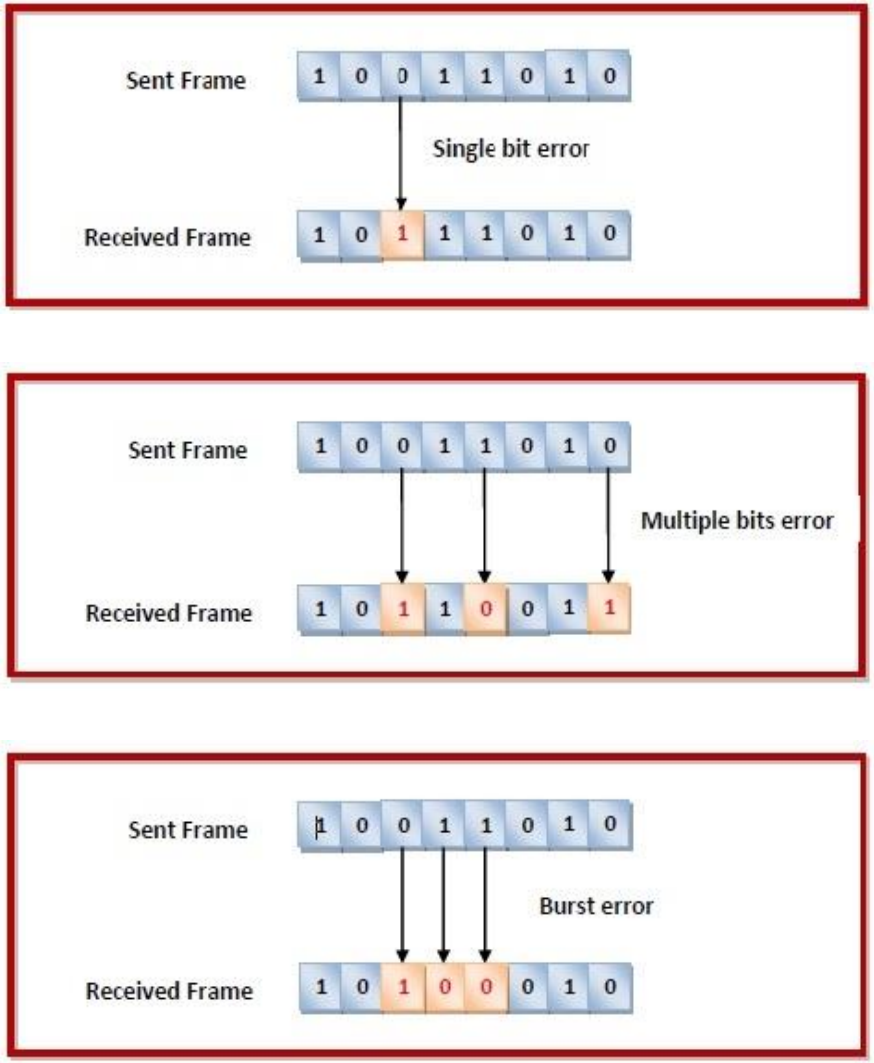
Types of Errors

Errors can be of three types, namely single bit errors, multiple bit errors, and burst errors.

- **Single bit error** – In the received frame, only one bit has been corrupted, i.e. either changed from 0 to 1 or from 1 to 0.

- **Multiple bits error** – In the received frame, more than one bits are corrupted.

- **Burst error** – In the received frame, more than one consecutive bits are corrupted.



Error Control

Error control can be done in two ways

- **Error detection** − Error detection involves checking whether any error has occurred or not. The number of error bits and the type of error does not matter.
- **Error correction** − Error correction involves ascertaining the exact number of bits that has been corrupted and the location of the corrupted bits.

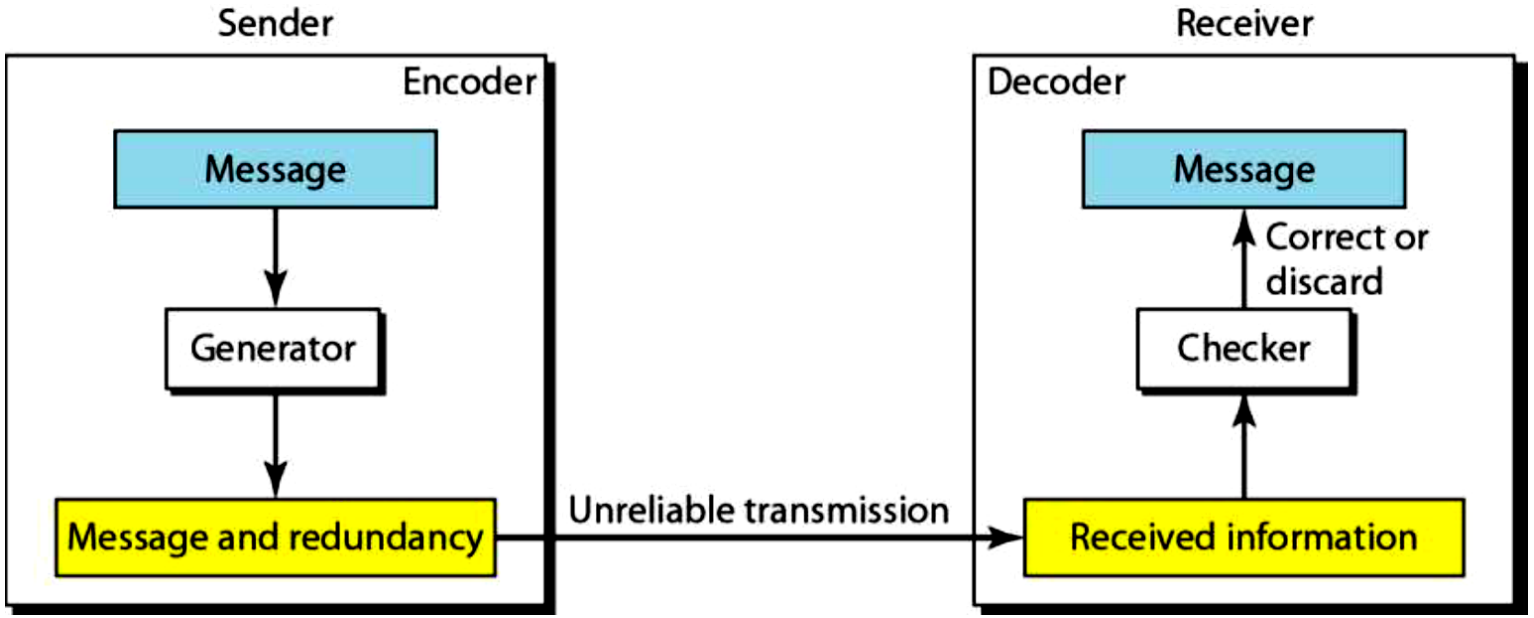
For both error detection and error correction, the sender needs to send some additional bits along with the data bits. The receiver performs necessary checks based upon the additional redundant bits. If it finds that the data is free from errors, it removes the redundant bits before passing the message to the upper layers.

Redundancy

the state of being not or no longer needed or useful

oproyojoniota

- To detect or correct errors, redundant bits of data must be added



Error Detection Techniques

There are three main techniques for detecting errors in frames:

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

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.

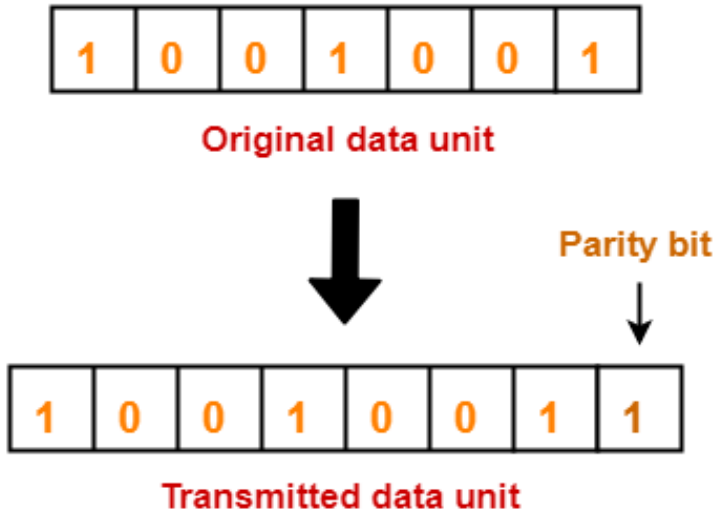
The parity check is suitable for single bit error detection only.

Parity Check Example

- Consider the data unit to be transmitted is 1001001

At **Sender** Side-

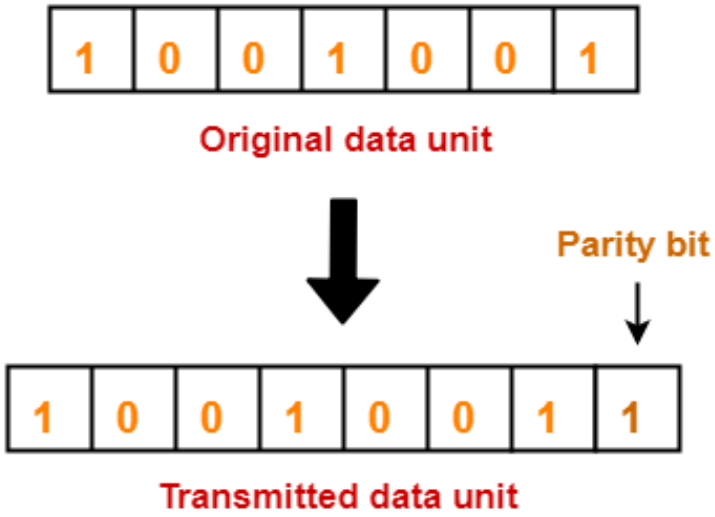
- Total number of 1's in the data unit is counted.
- Total number of 1's in the data unit = 3.
- Clearly, even parity is used and total number of 1's is odd.
- So, parity bit = 1 is added to the data unit to make total number of 1's even.
- Then, the code word 10010011 is transmitted to the receiver.



Parity Check Example

At **Receiver** Side-

- After receiving the code word, total number of 1's in the code word is counted.
- Consider receiver receives the correct code word = 10010011.
- Even parity is used and total number of 1's is even.
- So, receiver assumes that no error occurred in the data during the transmission.



How many bit errors can PB detect ?

10001110 ---→ 10101110 => error !

10001110 ---→ 10100110 => **No error detected** !!!

Conclusion – 1 PB can only detect an odd number of errors !

Limitation-

- This technique can not detect an even number of bit errors (two, four, six and so on).
- If even number of bits flip during transmission, then receiver can not catch the error.

Checksum

In this error detection scheme, the following procedure is applied

- For error detection by checksums, data is divided into fixed sized frames or segments.
- **Sender's End** – The sender adds the segments using 1's complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.
- **Receiver's End** – The receiver adds the incoming segments along with the checksum using 1's complement arithmetic to get the sum and then complements it.
- If the result is zero, the received frames are accepted; otherwise they are discarded.

Checksum **Example-1**

Consider the data unit to be transmitted is-

10011001111000100010010010000100

Consider 8 bit checksum is used.

Solution

Step-01:

At sender side, The given data unit is divided into segments of 8 bits as-

10011001	11100010	00100100	10000100

Solution

10011001	11100010	00100100	10000100

Now, all the segments are added and the result is obtained as-

- 10011001 + 11100010 + 00100100 + 10000100

  = 1000100011

- Since the result consists of 10 bits, so extra 2 bits are wrapped around.
- 00100011 + 10 = 00100101 (8 bits)
- Now, 1’s complement is taken which is 11011010.
- Thus, checksum value = 11011010

Solution

Step-02:

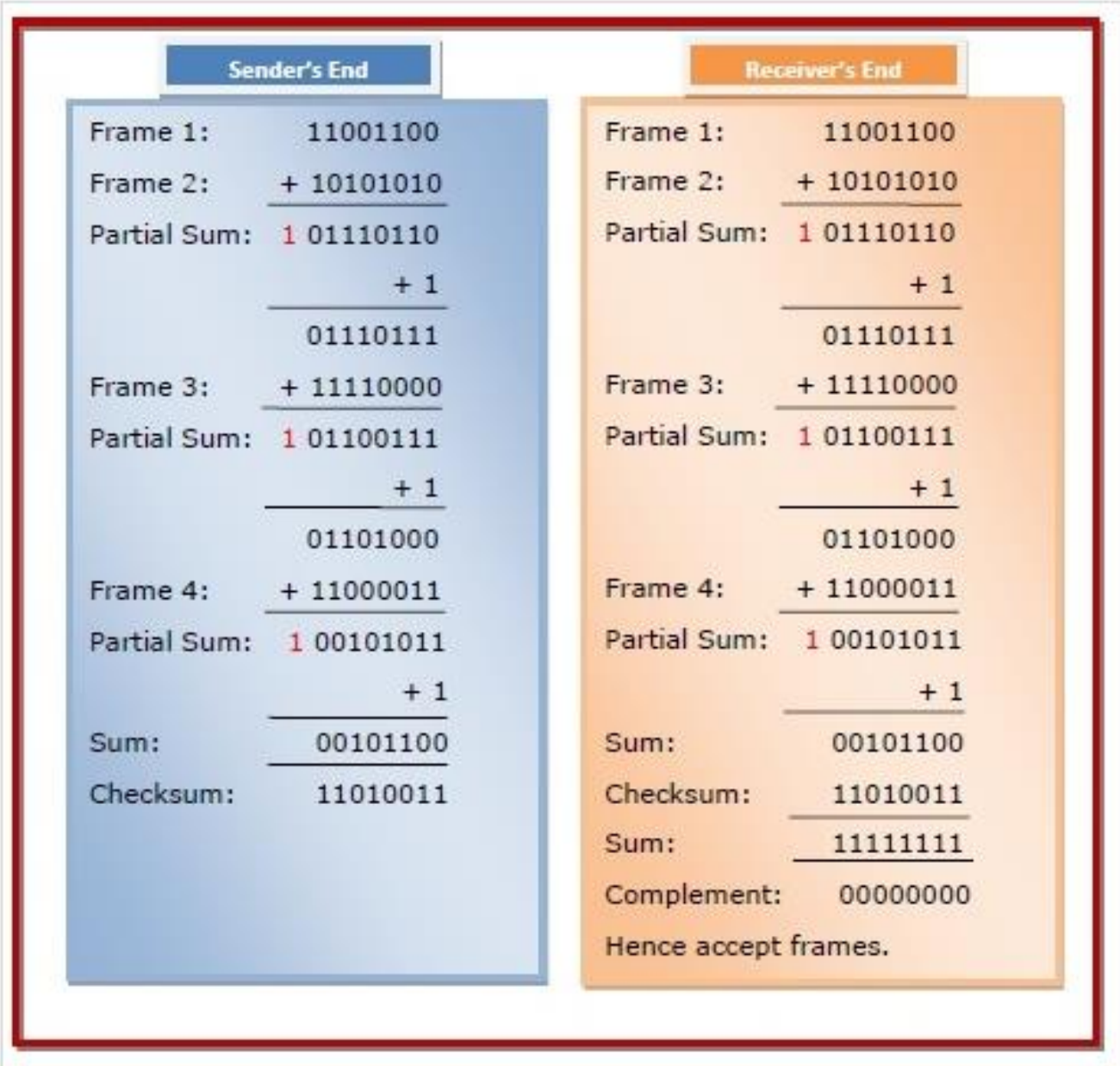
- The data along with the checksum value is transmitted to the receiver.

Step-03:

- At receiver side, The received data unit is divided into segments of 8 bits.
- All the segments along with the checksum value are added.
- Sum of all segments + Checksum value = 00100101 + 11011010 = 11111111
- Complemented value = 00000000
- Since the result is 0, receiver assumes no error occurred in the data and therefore accepts it.

Checksum
Example-2

- Suppose that the sender wants to send 4 frames each of 8 bits, where the frames are 11001100, 10101010, 11110000 and 11000011.



Sender's End	
Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011

Receiver's End	
Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011
Sum:	11111111
Complement:	00000000
Hence accept frames.	