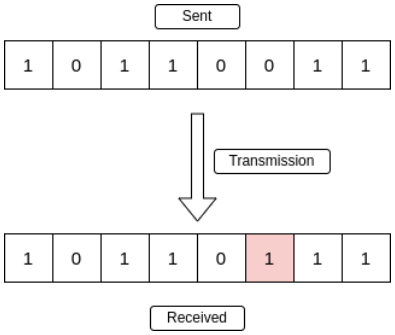
Error detection & correction

NETWORK 2

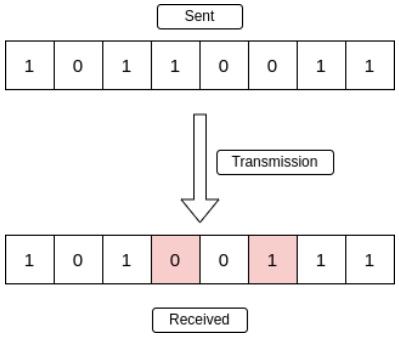
Razie poorgholamrezaee | 40003793 | 4022

Types of error

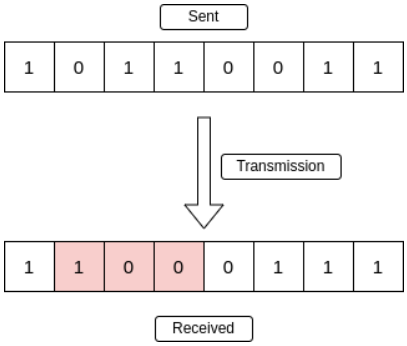
- Single-Bit Error: one bit is altered.



- Multiple-Bit Error: more than one bit in is affected.



- Burst Error: several consecutive bits are flipped.



Error detection methods

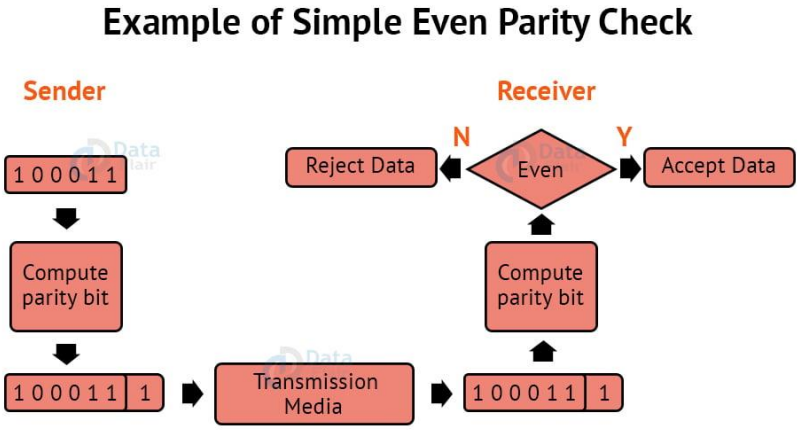
Error is a condition when the receiver's information does not match the sender's information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits. That means a 0 may change to 1 or a 1 may change to 0. To prevent such errors, error-detection codes are added as extra data to digital messages. This helps in detecting errors that may have occurred during transmission. Some of them are as follows:

SIMPLE PARITY CHECK

The simplest method is to add a redundant bit which indicates the parity of 1's. There are two types:

1. Even parity: total number of 1's is even
2. Odd parity: total number of 1's is odd

I implemented even parity. There is an example:



In order to compute parity bit, we can add all bits and find the reminder by 2.

$$\text{Parity} = (1+0+0+0+1+1) \% 2 = 1$$

Disadvantages

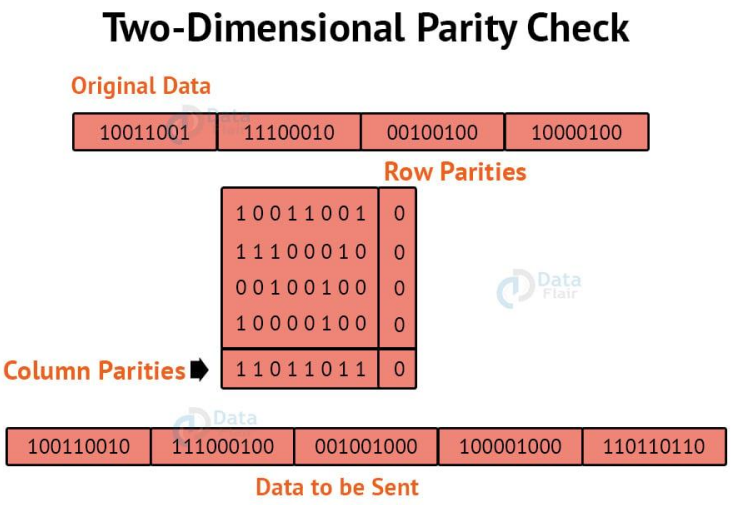
- Even number of bit error can't be detected.
- The location of error can't be found.
- The error can't be corrected.
- In order to detect more errors, more check bits should be added.

Advantages

- All single bit errors and multiple bit errors with odd errors can be detected.
- Simple to implement.
- Efficient in terms of computational complexity and memory requirements.

TWO-DIMENSIONAL PARITY CHECK

Parity check bits are calculated for each row, same as simple parity check bit, and each column. At the receiving end, these are compared with the parity bits calculated on the received data.



Disadvantages

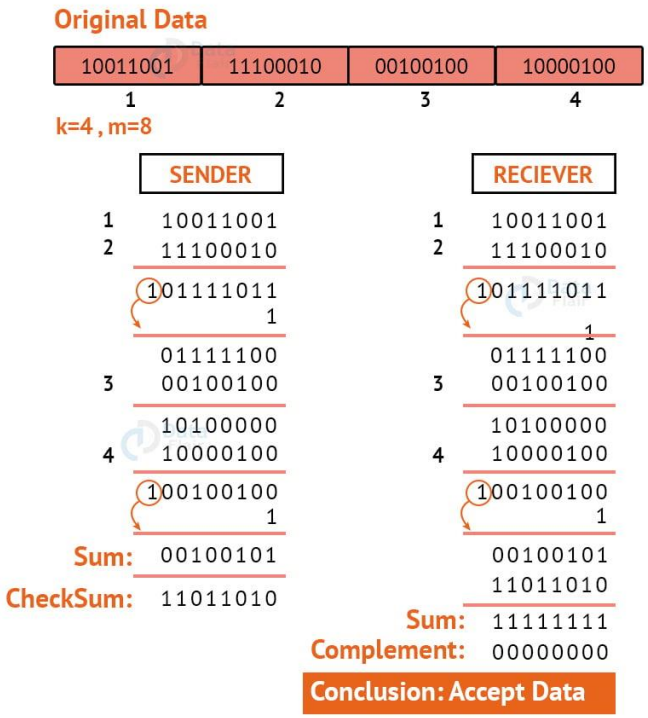
- If two bits in one data unit are corrupted and two bits exactly the same position in another data unit are also corrupted, then 2D Parity checker will not be able to detect the error.
- This technique cannot be used to detect the 4-bit errors or more in some cases.

Advantages

- All single bit errors and multiple bit errors with odd errors in each row or column can be detected.
- Detect more errors compared to simple parity.

CHECKSUM

The process involves dividing the data into equally sized segments and using a 1's complement to calculate the sum of these segments. The calculated sum is then sent along with the data to the receiver. At the receiver's end, the same process is repeated and if all zeroes are obtained in the sum, it means that the data is correct.



Disadvantages

- Error is not detected, if one sub-unit of the data has one or more corrupted bits and corresponding bits of the opposite value are also corrupted in another sub-unit. Error is not detected in this situation because in this case the sum of columns is not affected by corrupted bits.
- If chunks of data are flipped, the error can't be detected. Because the sum will remain the same (order is not important).
- Adding or removing o bytes wouldn't affect the sum.
- Sensitivity to the Data Packet size => small : errors not detected, large : overhead of computation

Advantages

- simplicity and efficiency
- suitable for real-time data packet transmissions because it is fast

CRC (CYCLIC REDUNDANCY CHECK)

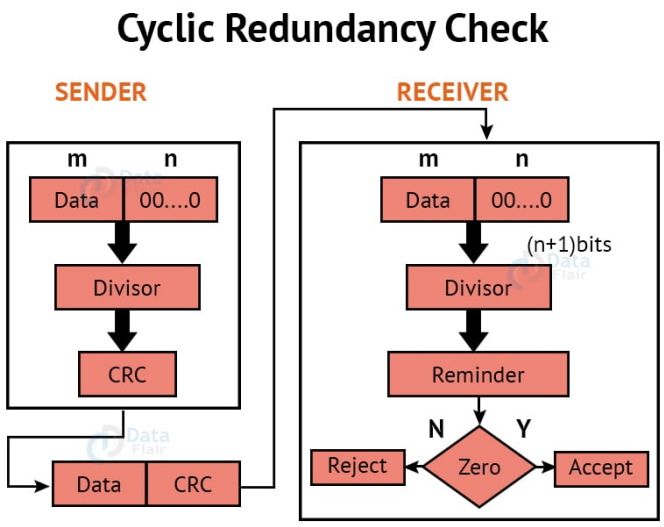
CRC is based on binary division.

A sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of the data unit so that the resulting data unit becomes exactly divisible by a predetermined binary number.

At the destination, the data unit is divided by the same number. If there is no remainder, the data unit is assumed to be correct and is therefore accepted.

A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

Check bits = n bit => p = n+1 bit



Disadvantages

- Can't detect errors with error pattern dividable by p.
- Can't correct errors.

Advantages

- Effective in detecting single-bit errors, multiple-bit errors and burst errors.
- Simple code suitable for various applications.

Error Correction Methods

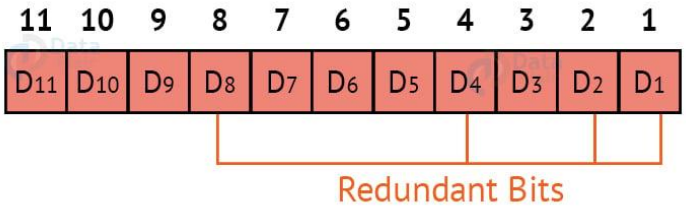
Error Correction codes are used to detect and repair mistakes that occur during data transmission from the transmitter to the receiver.

We will need to add some more redundant bits to find the location and then correct the error(s).

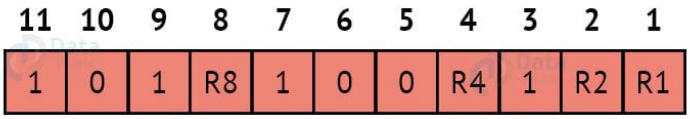
HAMMING CODE

Example: Data to be transmitted is 1011001

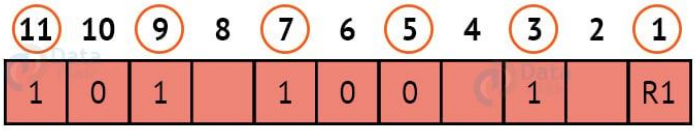
Redundant bits are always placed at positions that correspond to the power of 2, so the redundant bits will be placed at positions: 1, 2, 4 and 8.



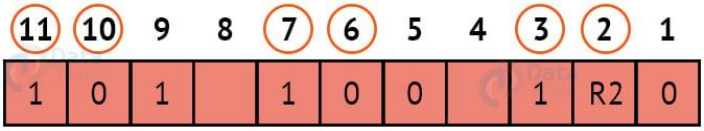
If we place the data bits:



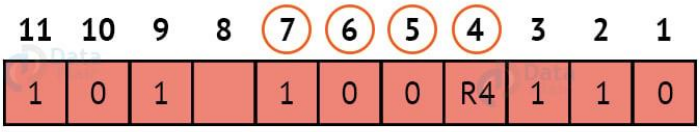
Computing R1 which is even parity of places where the first bit (bit $\log^1_2 = 0$) in binary index is 1 => 001, 011, 101, 111, 1001, 1011



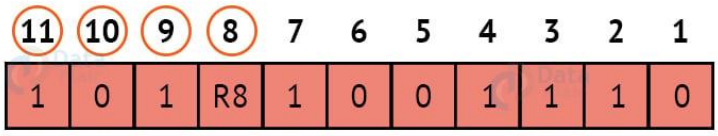
R2: even parity where (bit $\log^2_2 = 1$) is 1 => 010, 011, 110, 111, 1010, 1011



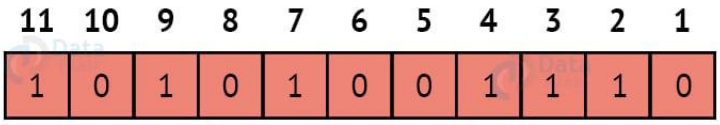
R4: even parity where (bit $\log^4_2 = 2$) is 1 => 100, 101, 110, 111



R8: even parity where (bit $\log^8_2 = 3$) is 1 => 1000, 1001, 1010, 1011



The final result is:



At the receiver the index of all 1s are added and the result is the place of error. Knowing the place of error, we can correct it by flipping the bit.

Disadvantages

- Limited Multiple Error Correction
- Hard to correct in large segments (probability of error is higher)

Advantages

- simple and efficient
- not much redundancy for larger segments of data (256 bit -> 8 bit redundant)