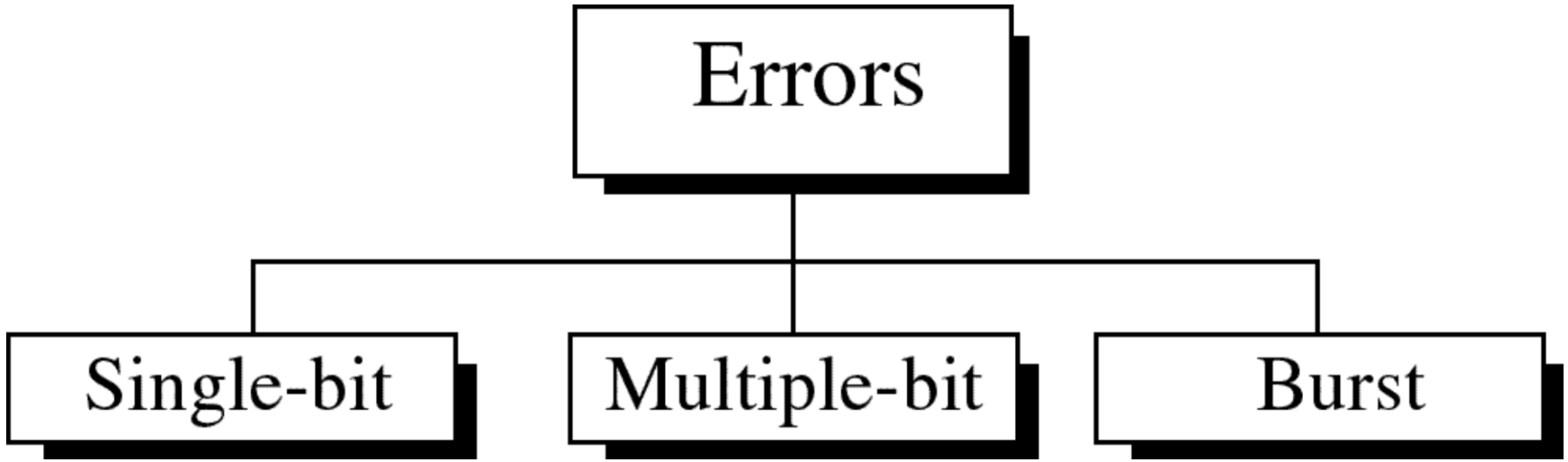
Error Detection and Correction

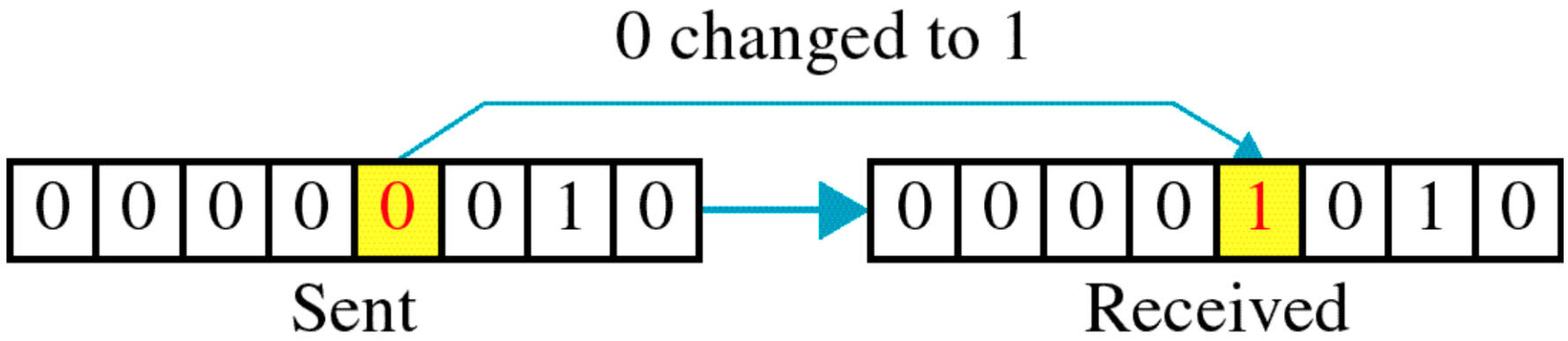
Why

- Networks must be able to transfer data from one device to another with complete accuracy.
- Data can be corrupted during transmission.
- For reliable communication, errors must be detected and corrected.
- Error detection and correction are implemented either at the data link layer or the transport layer of the OSI model.

Types of Errors



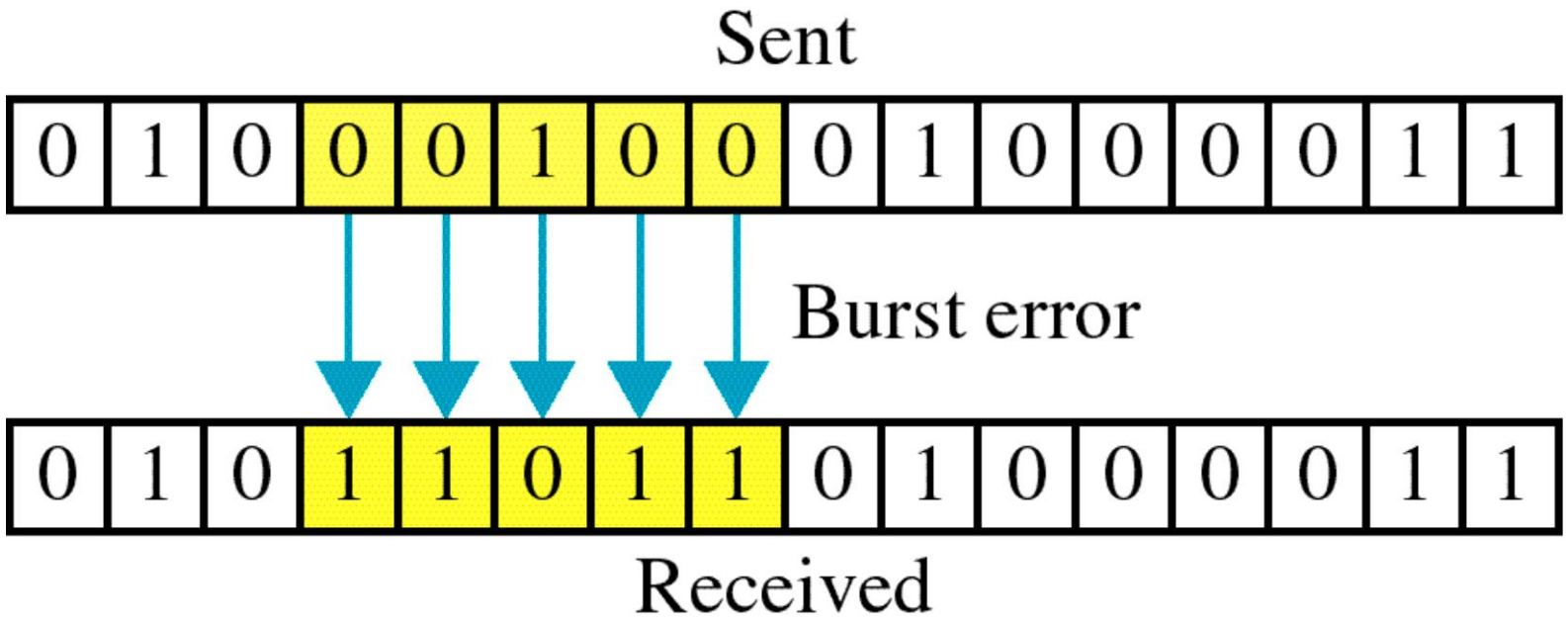
Single-bit error

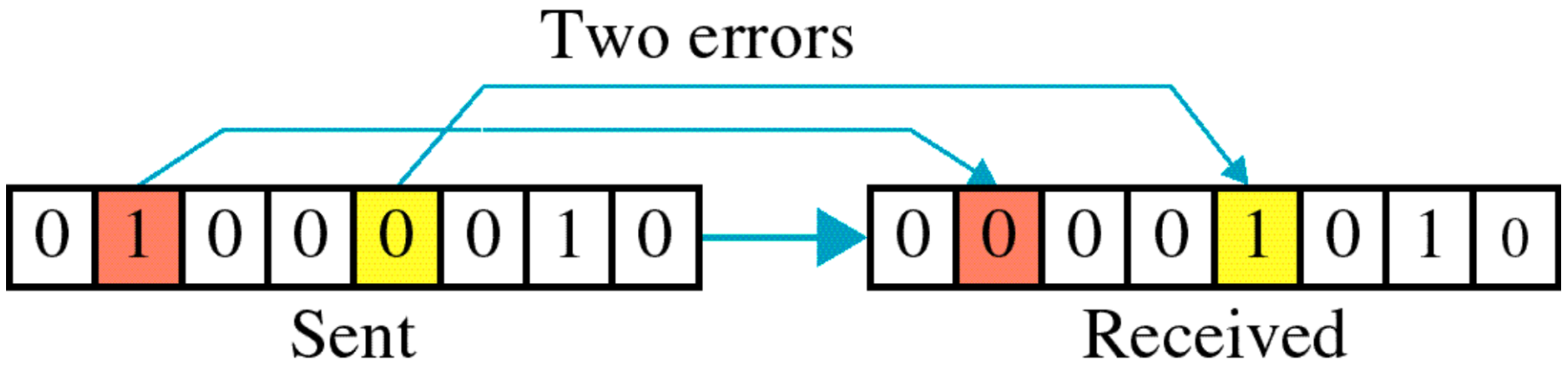


- **Single bit errors** are the **least likely** type of errors in serial data transmission because the noise must have a very short duration which is very rare.

- However this kind of errors can happen in parallel transmission.

Burst error


Sent
0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
Burst error
0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 1
Received


Two errors
0 1 0 0 0 0 1 0
0 0 0 0 1 0 1 0
Sent
Received

- The term burst error means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

- Burst errors does not necessarily mean that the errors occur in consecutive bits, the length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

- Burst error is most likely to happen in serial transmission since the duration of noise is normally longer than the duration of a bit.

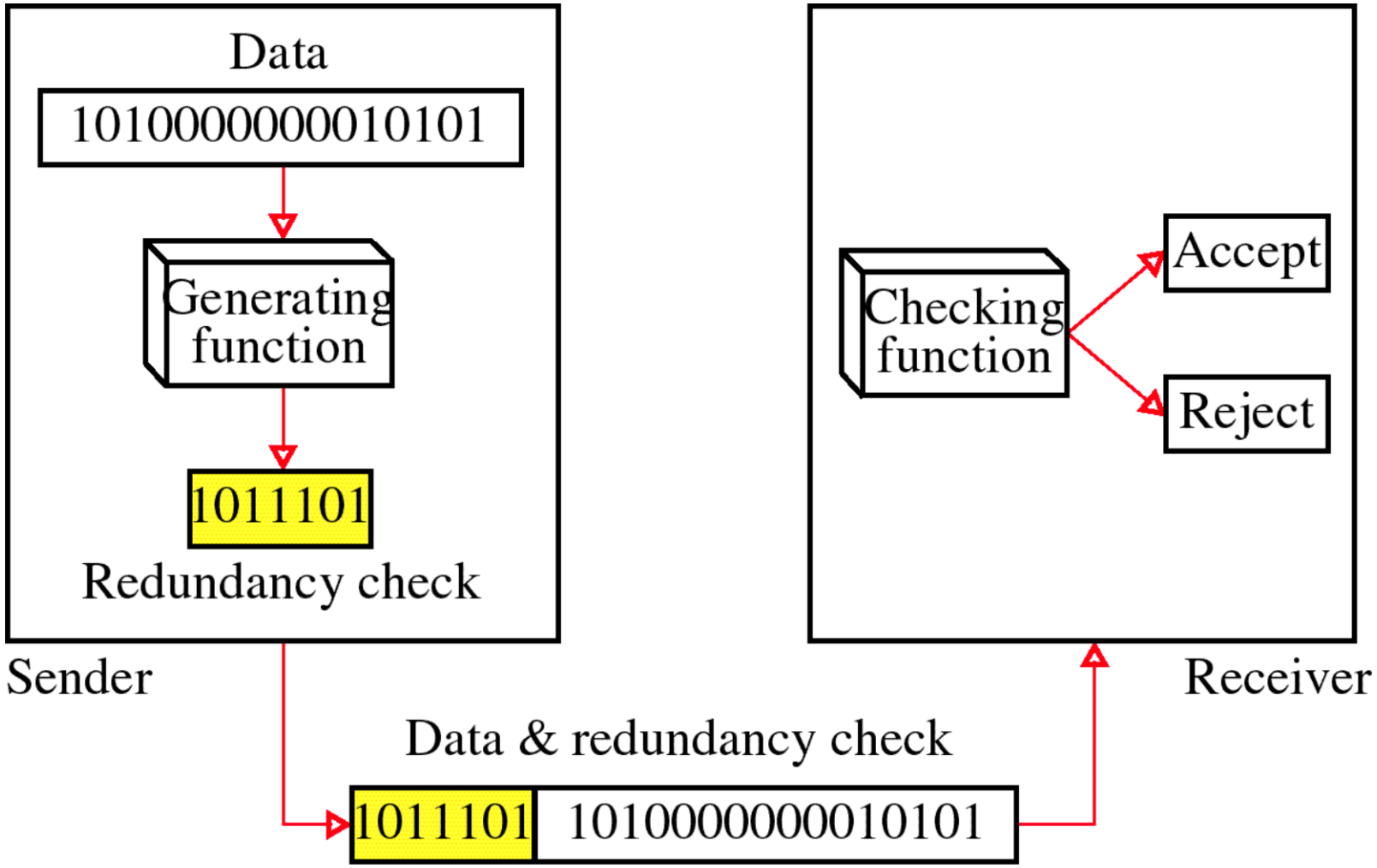
- The number of bits affected depends on the data rate and duration of noise.

Error detection

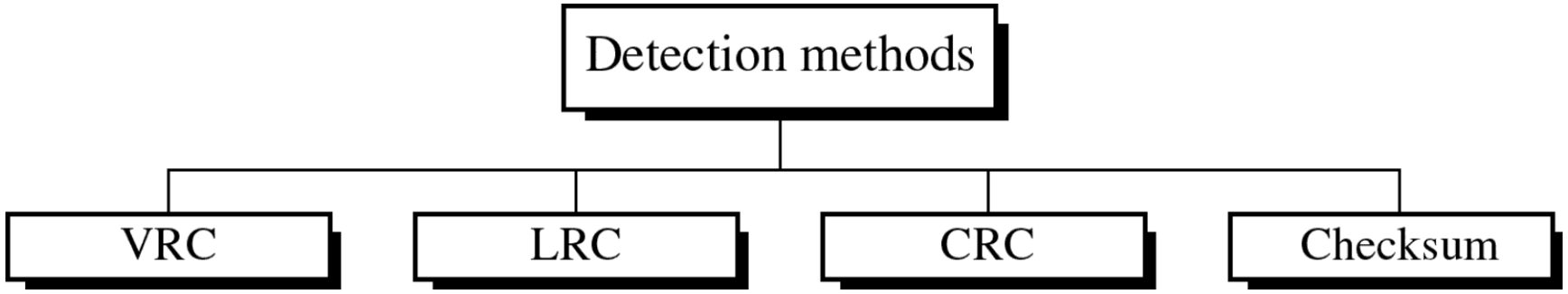
Error detection means to decide whether the received data is correct or not without having a copy of the original message.

Error detection **uses the concept of redundancy**, **which means** adding extra bits for detecting errors at the destination.

Redundancy


Data
1010000000010101
Generating function
1011101
Redundancy check
Sender
Checking function
Accept
Reject
Receiver
Data & redundancy check
1011101
1010000000010101

Four types of redundancy checks are used in data communications



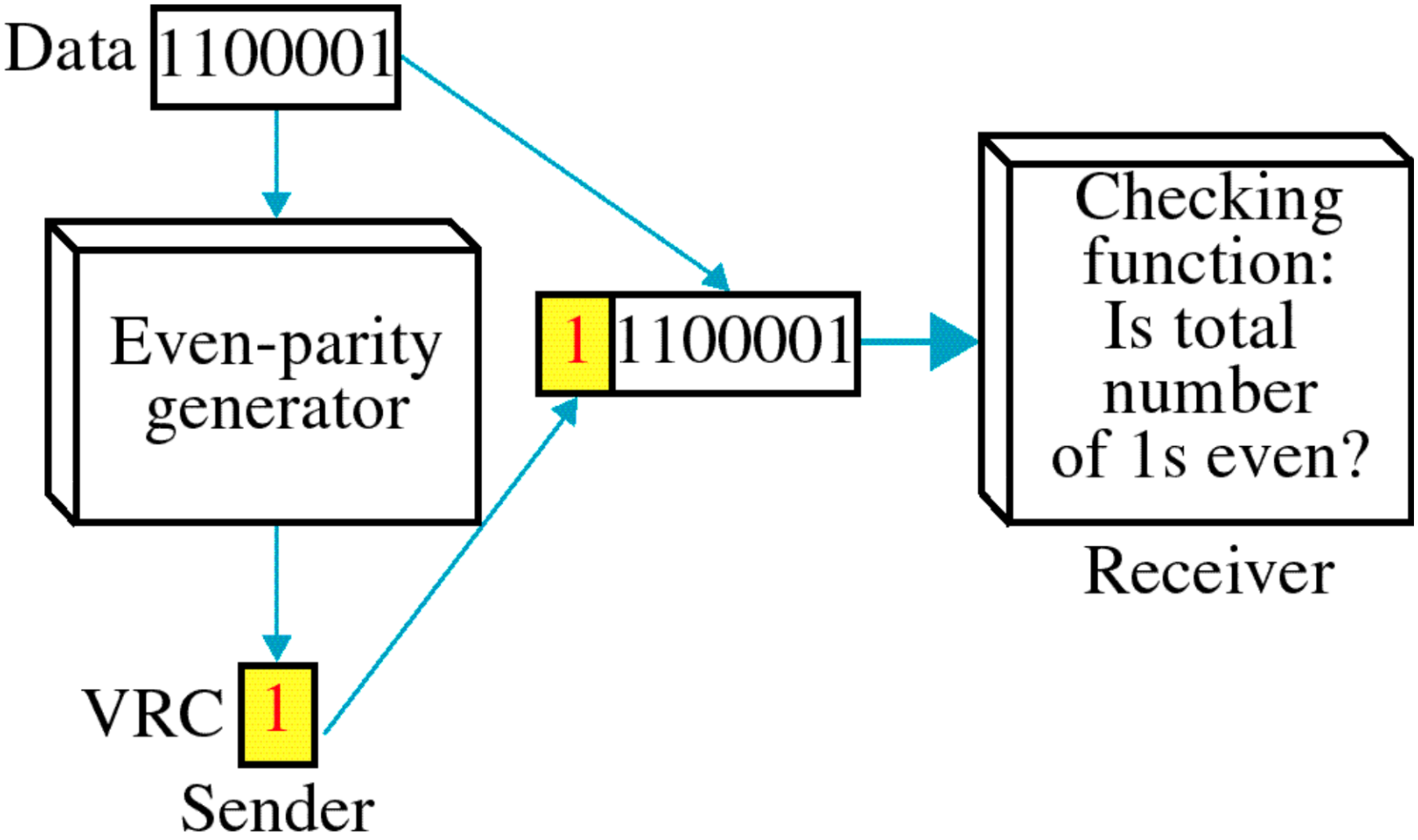
Vertical Redundancy Check (VRC)

Another name of vertical redundancy check is parity check.

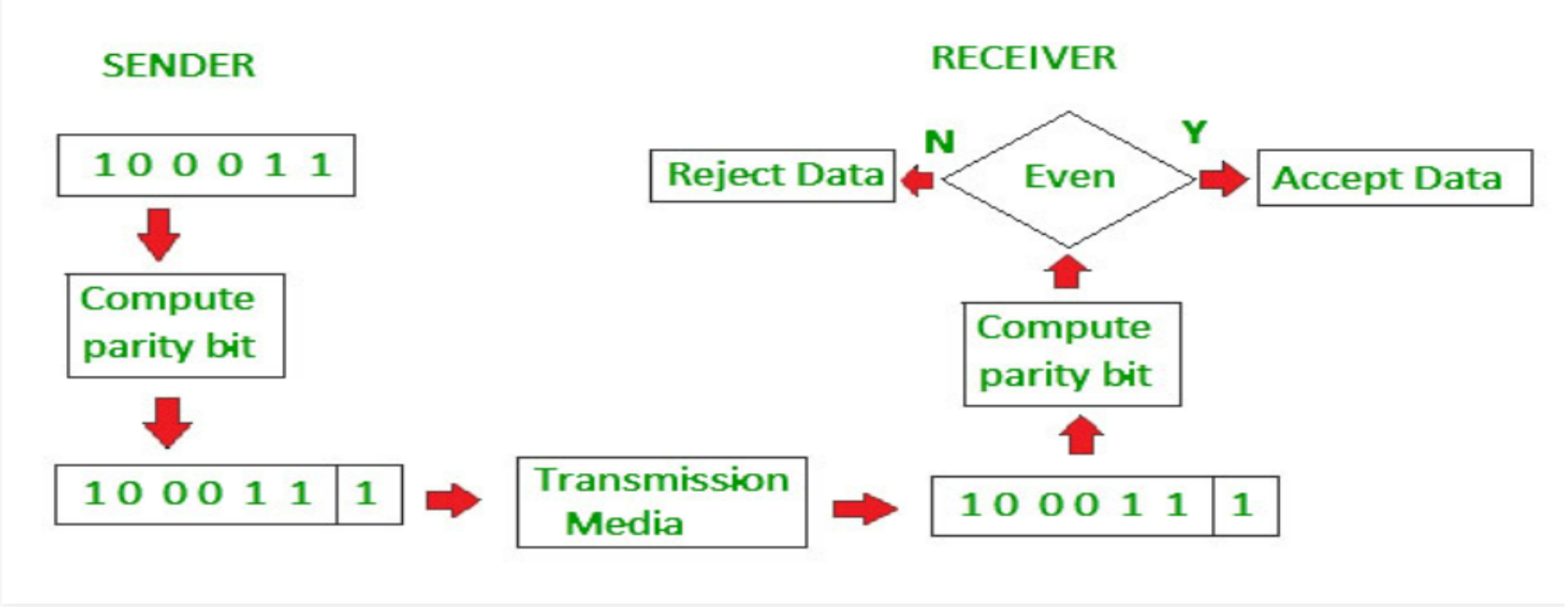
Least Expensive Mechanism.

So in this mechanism when an error detected by this mechanism a parity bit is append to the data unit so the total number of bits becomes **even** including parity bit.

Vertical Redundancy Check



Vertical Redundancy Check



Performance

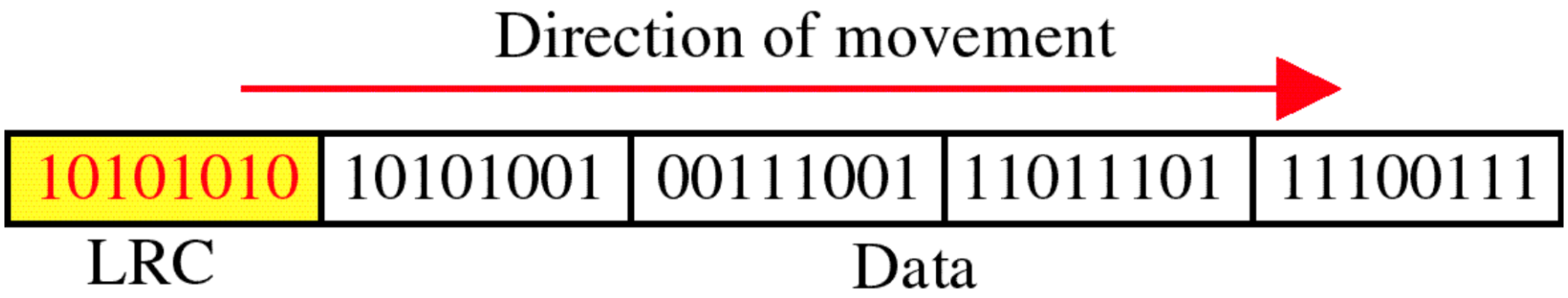
- It can detect single bit error.
- It can detect burst errors only if the total number of errors is odd.

Longitudinal Redundancy Check (LRC)

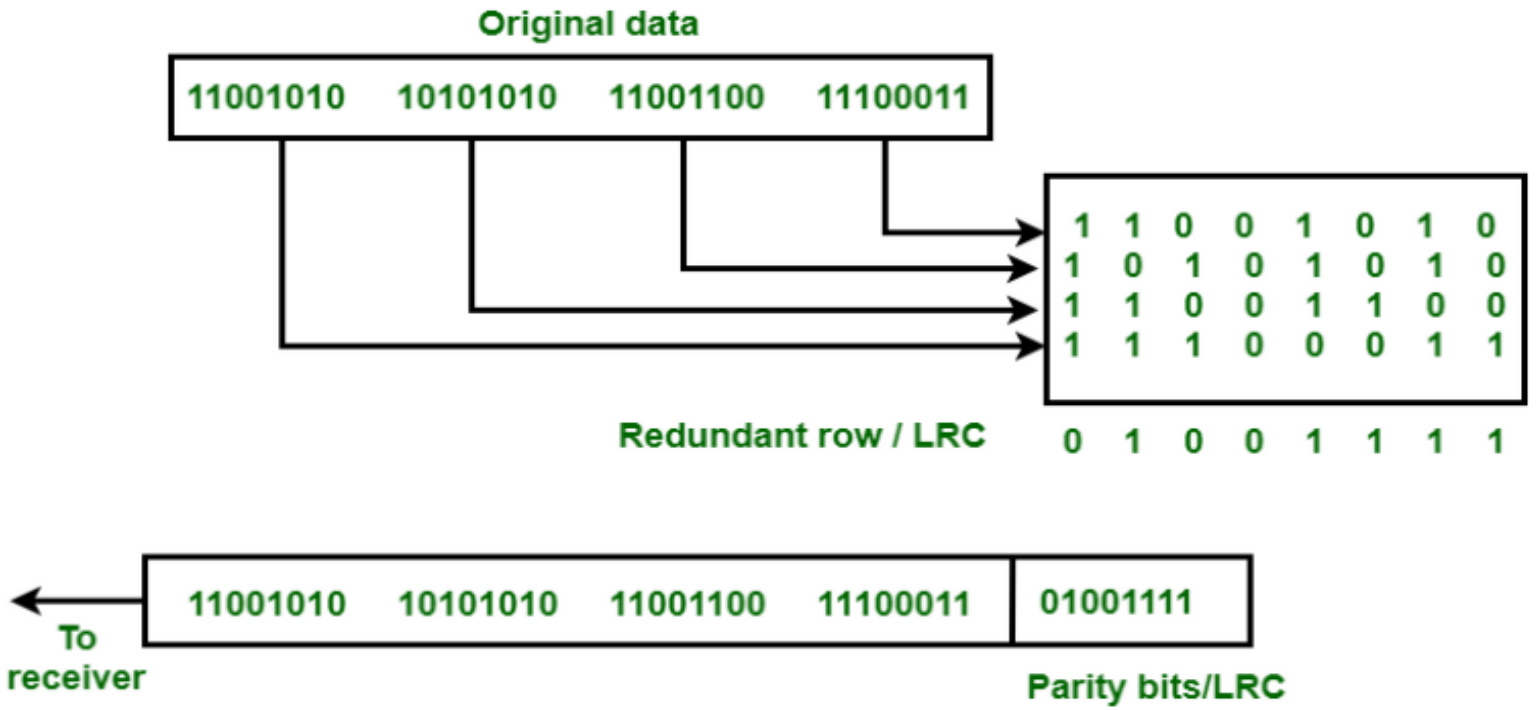
In this method Block of bits organized in tables e.g rows and columns.

calculate the parity bit for each column and the set of this parity bit is also sending with original data

Longitudinal Redundancy Check (LRC)



Longitudinal Redundancy Check (LRC)



Performance

- LRC increases the likelihood of detecting burst errors.
- If two bits in one data units are damaged and two bits in exactly the same positions in another data unit are also damaged, the LRC checker will not detect an error.

Example : If data 110011 010101 is changed to 010010110100.

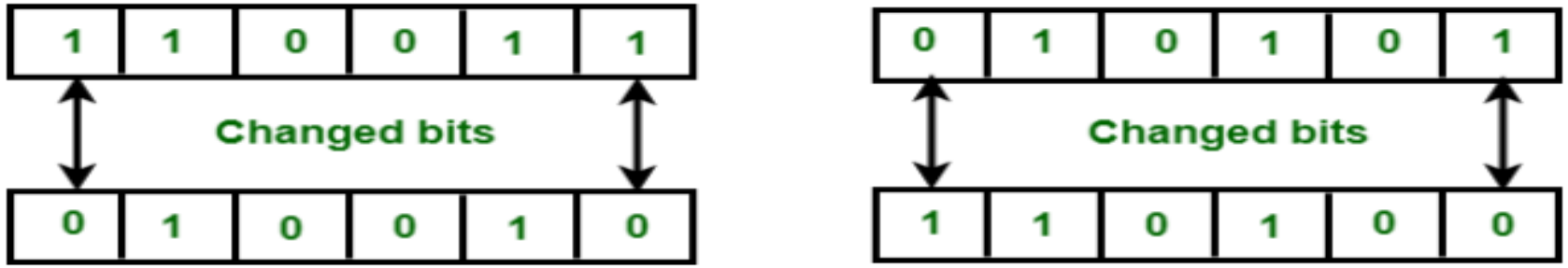


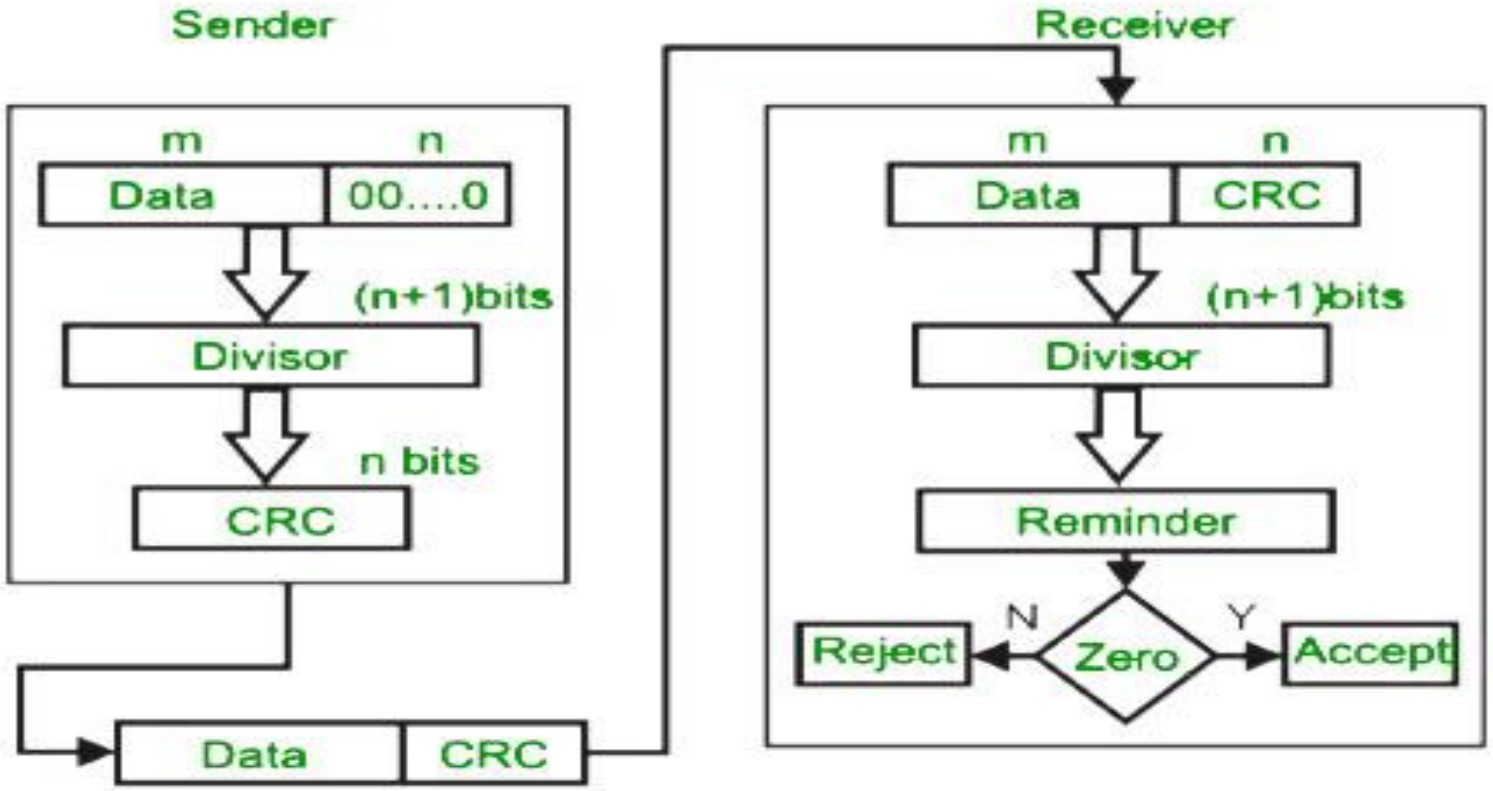
Figure : *Two bits at same bit position damaged in 2 data units*

In this example 1st and 6th bit in one data unit is changed . Also the 1st and 6th bit in second unit is changed.

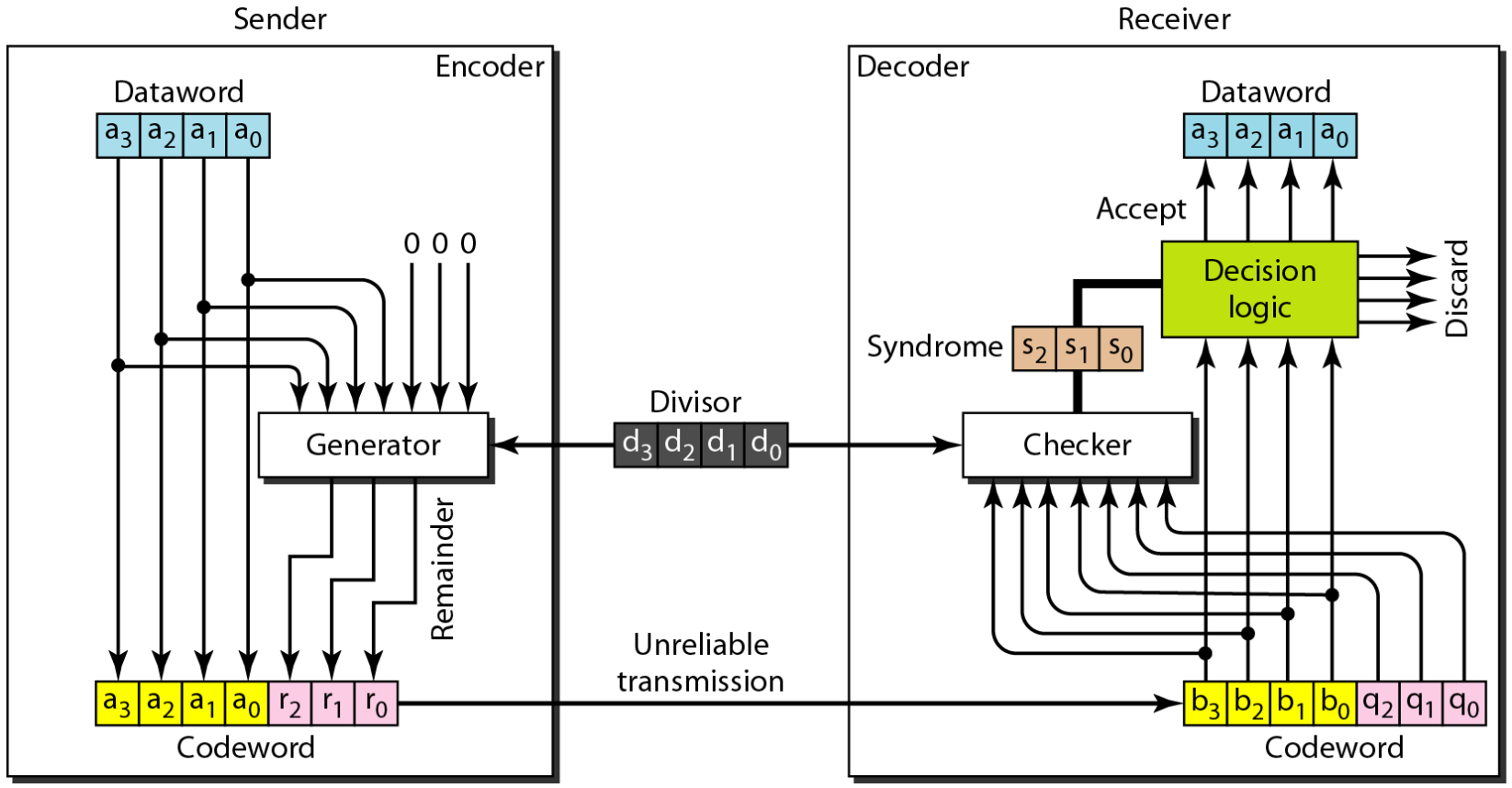
Cyclic redundancy check (CRC)

- CRC is based on binary division.
- 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 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 and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

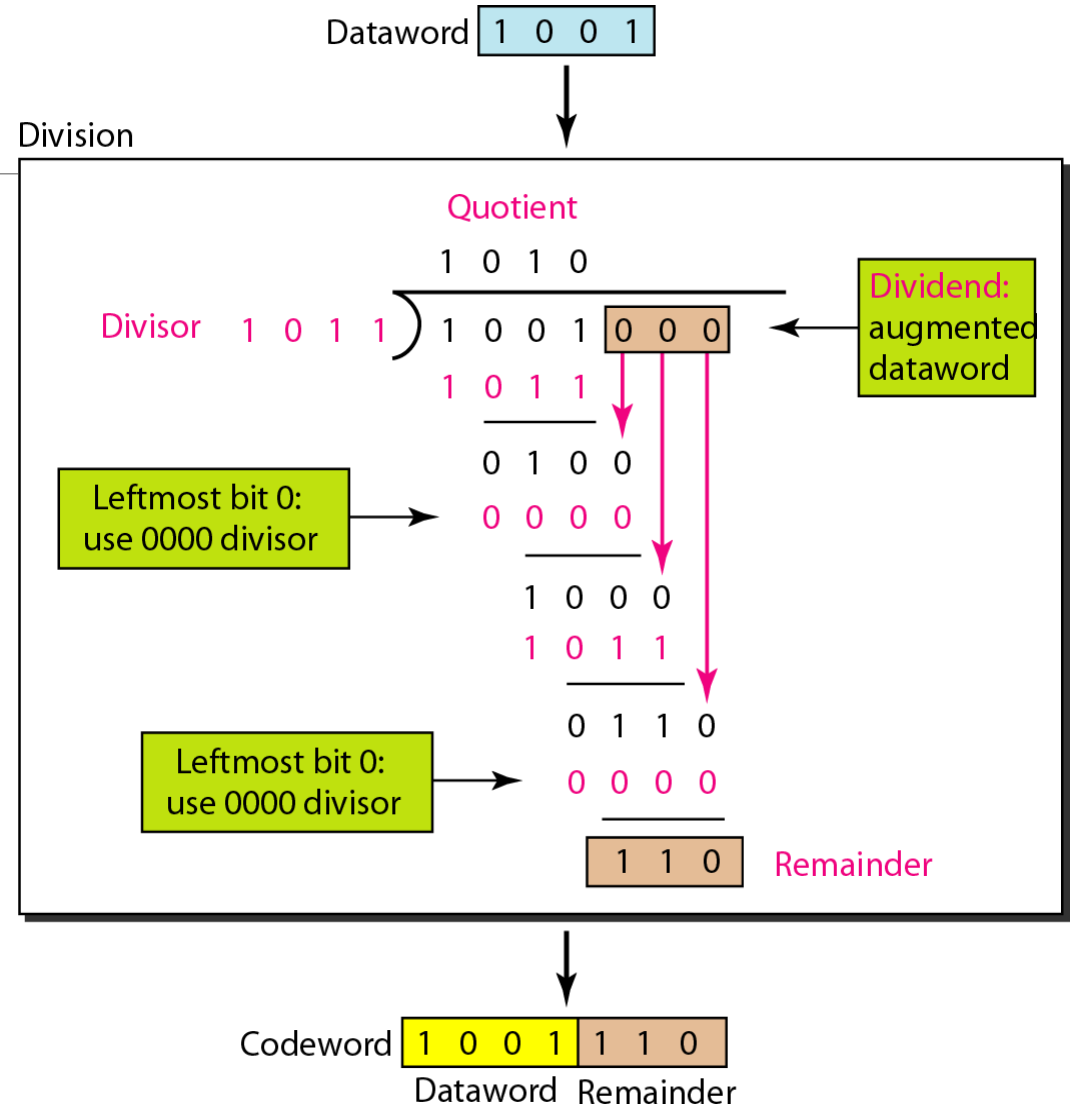
CRC


Sender
m
n
Data
00....0
(n+1)bits
Divisor
n bits
CRC
Data
CRC
Receiver
m
n
Data
CRC
(n+1)bits
Divisor
Reminder
N
Y
Reject
Zero
Accept

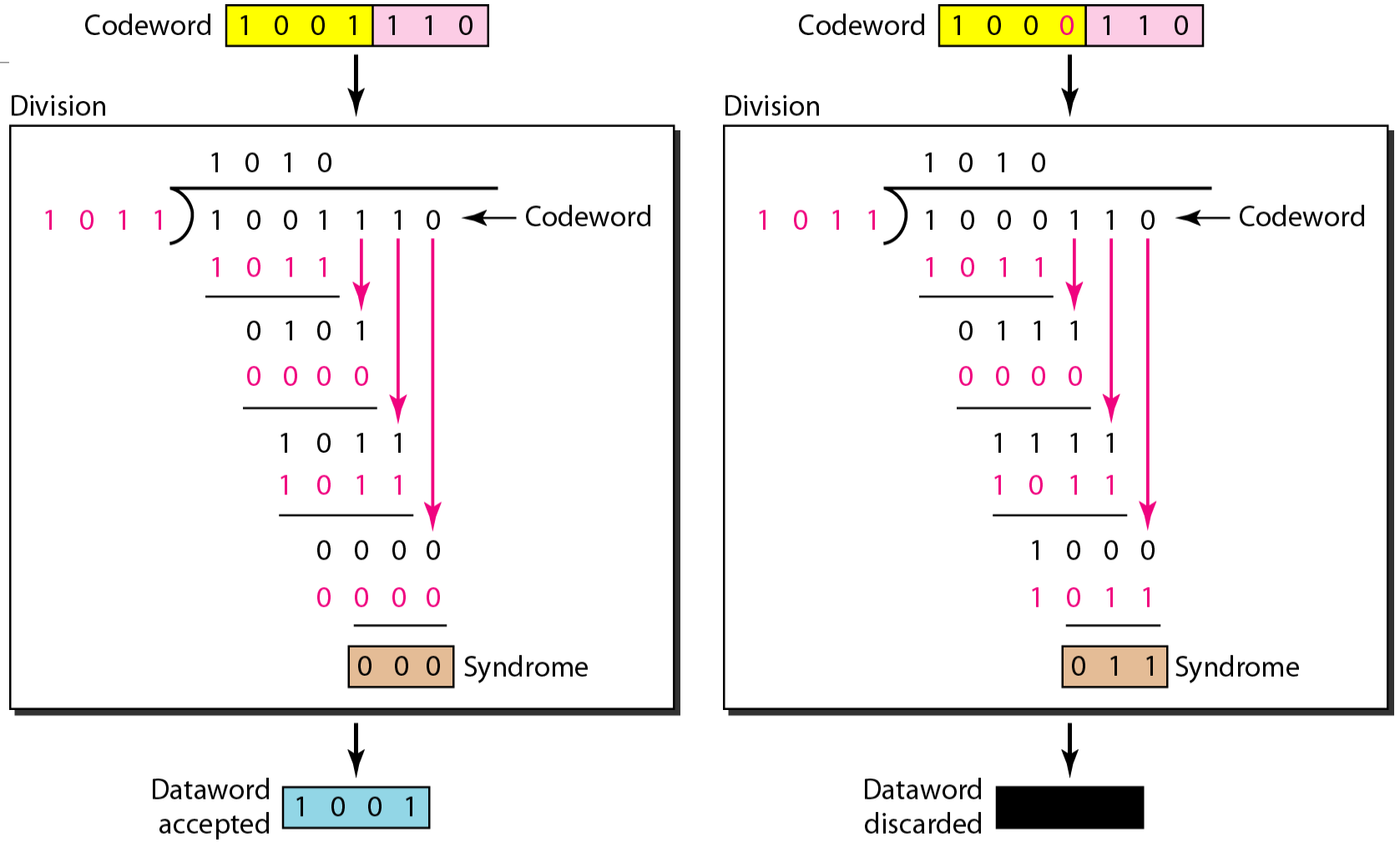
CRC encoder and decoder



Division in CRC encoder



Division in the CRC decoder for two cases



Example

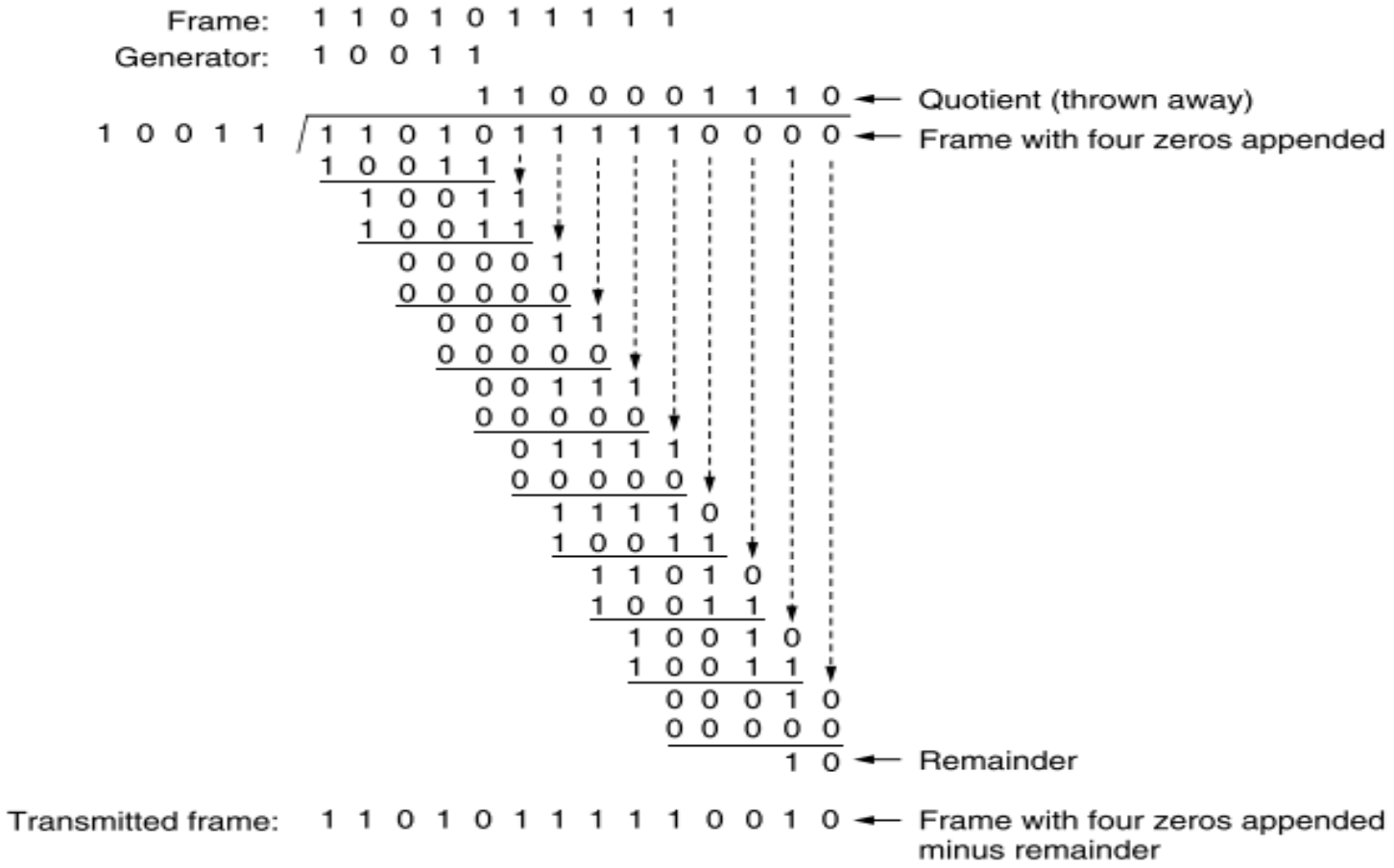
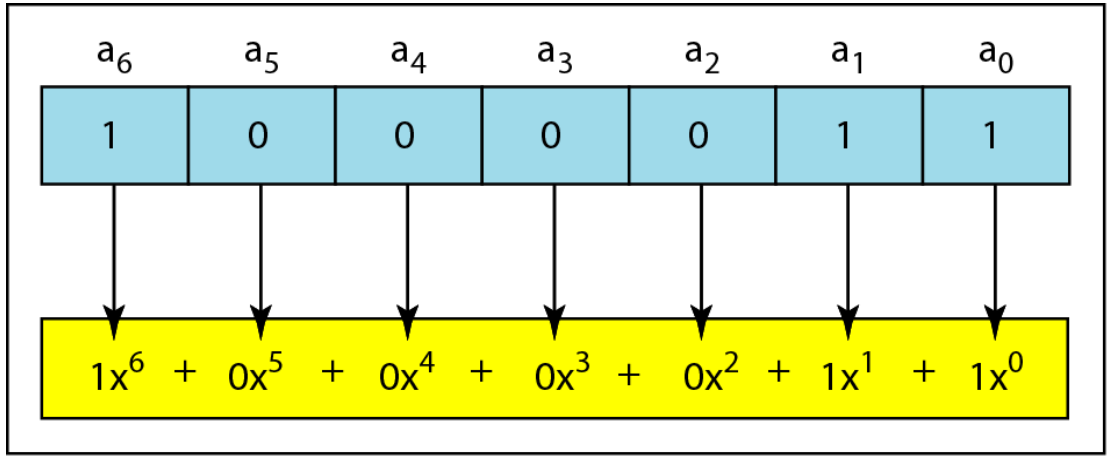

Frame: 1 1 0 1 0 1 1 1 1 1
Generator: 1 0 0 1 1
1 1 0 0 0 0 1 1 1 0 ← Quotient (thrown away)
1 0 0 1 1 | 1 1 0 1 0 1 1 1 1 1 0 0 0 0 ← Frame with four zeros appended
1 0 0 1 1
1 0 0 1 1
1 0 0 1 1
0 0 0 0 1
0 0 0 0 0
0 0 0 1 1
0 0 0 0 0
0 0 1 1 1
0 0 0 0 0
0 1 1 1 1
0 0 0 0 0
1 1 1 1 0
1 0 0 1 1
1 1 0 1 0
1 0 0 1 1
1 0 0 1 0
1 0 0 1 1
0 0 0 1 0
0 0 0 0 0
1 0 ← Remainder
Transmitted frame: 1 1 0 1 0 1 1 1 1 1 0 0 1 0 ← Frame with four zeros appended minus remainder

Figure: *A polynomial to represent a binary word*

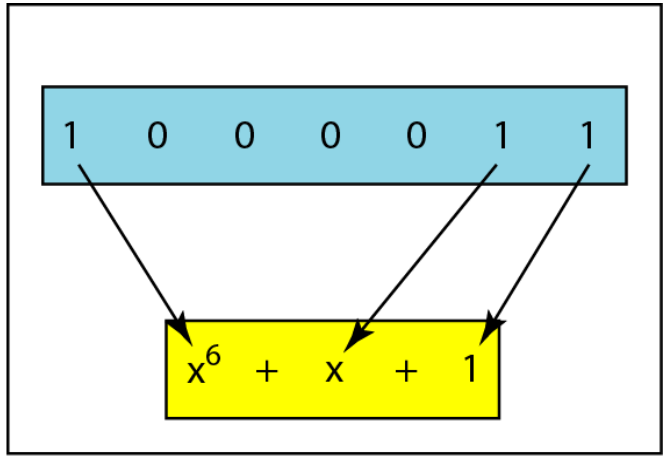
A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1.

The power of each term shows the position of the bit;

the coefficient shows the value of the bit.

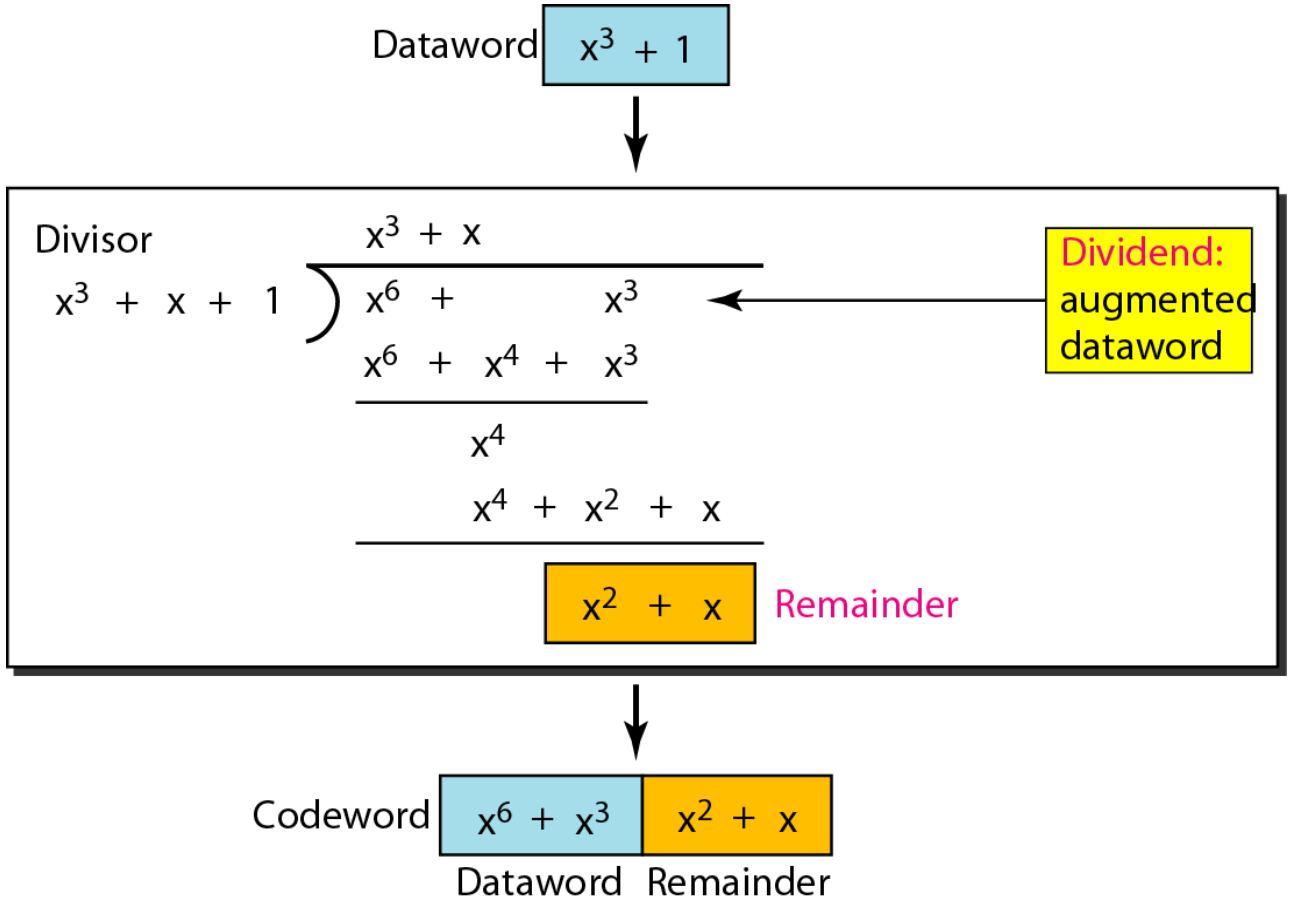


a. Binary pattern and polynomial



b. Short form

Figure *CRC division using polynomials*



Degree of a Polynomial

The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial x6 + x + 1 is 6.

Note that the degree of a polynomial is 1 less that the number of bits in the pattern.

Note

The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

A bit stream 1101011011 is transmitted using the standard CRC method. The generator polynomial is x^4+x+1. What is the actual bit string transmitted?

Benefit of Polynomial

The benefit is even more when we have a polynomial such as x23 + X3 + 1. Here the bit pattern is 24 bits in length (three Is and twenty-one Os) while the polynomial is just three terms.

Checksum

The checksum is used in the Internet by several protocols although not at the data link layer.

Example

Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

Example

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 11, 12, 0, 6, −36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

Example

How can we represent the number 21 in one's complement arithmetic using only four bits?

Solution

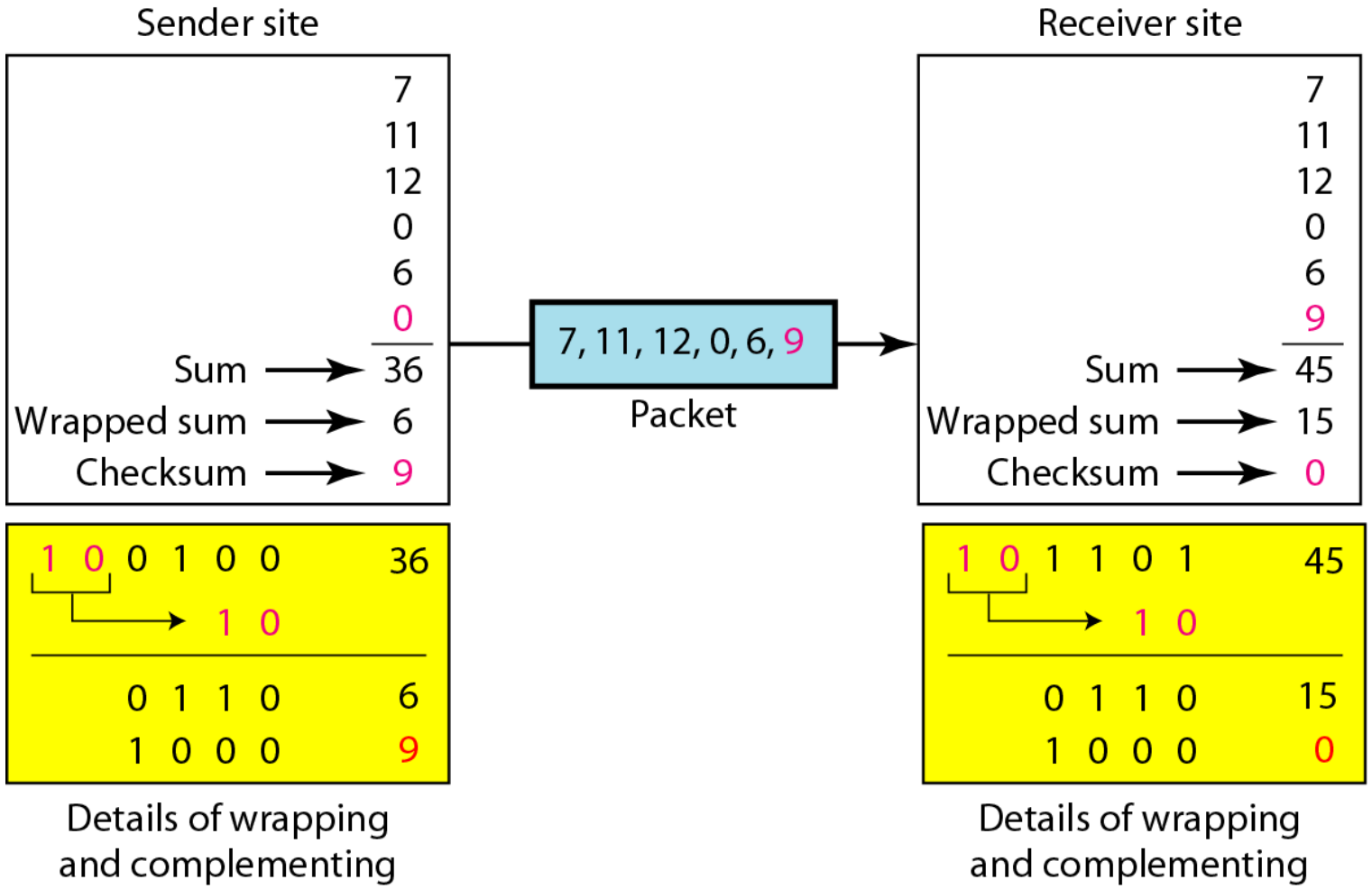
The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

the process at the sender and at the receiver. The sender initializes the checksum to 0 and adds all data items and the checksum (the checksum is considered as one data item and is shown in color). The result is 36. However, 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6. In the figure, we have shown the details in binary. The sender now sends six data items to the receiver including the checksum 9.

Example

The receiver follows the same procedure as the sender. It adds all data items (including the checksum); the result is 45. The sum is wrapped and becomes 15. The wrapped sum is complemented and becomes 0. Since the value of the checksum is 0, this means that the data is not corrupted. The receiver drops the checksum and keeps the other data items. If the checksum is not zero, the entire packet is dropped.

Figure Process



Note

Sender site:

1. The message is divided into 16-bit words.
2. The value of the checksum word is set to 0.
3. All words including the checksum are added using one's complement addition.
4. The sum is complemented and becomes the checksum.
5. The checksum is sent with the data.

Note

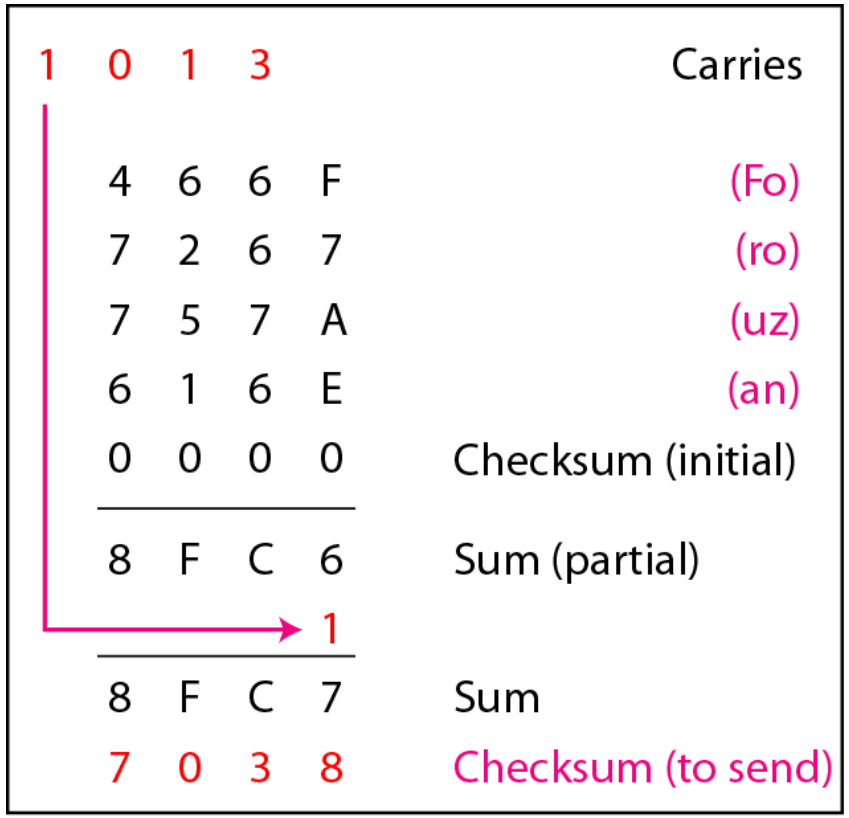
Receiver site:

1. The message (including checksum) is divided into 16-bit words.
2. All words are added using one's complement addition.
3. The sum is complemented and becomes the new checksum.
4. If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

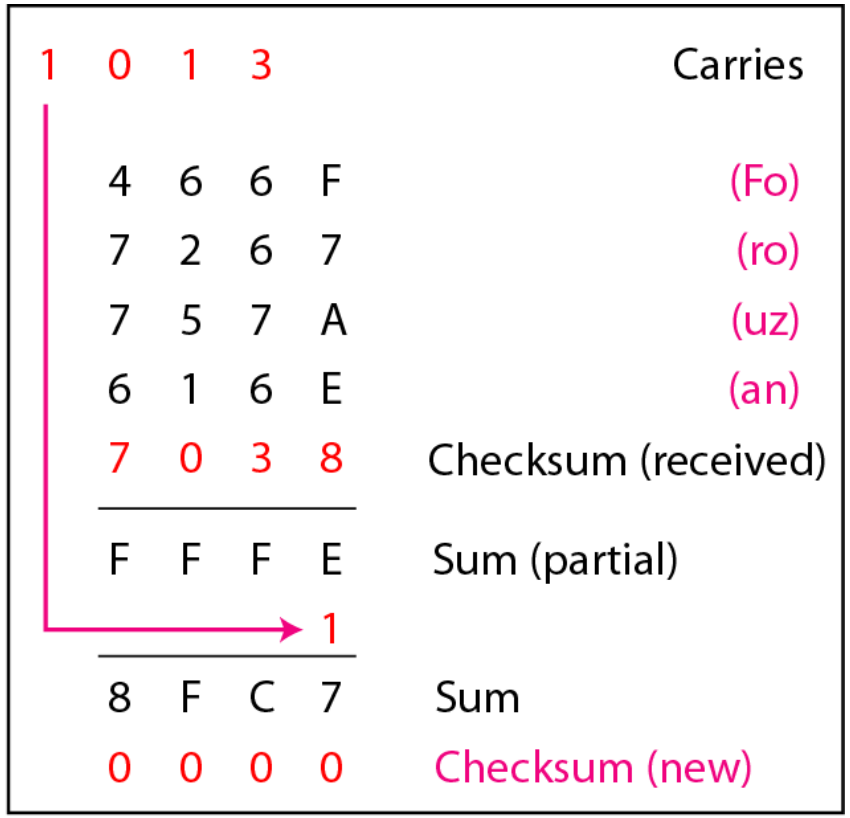
Example

Let us calculate the checksum for a text of 8 characters ("Forouzan"). The text needs to be divided into 2-byte (16-bit) words. We use ASCII to change each byte to a 2-digit hexadecimal number. For example, F is represented as 0x46 and o is represented as 0x6F. Figure shows how the checksum is calculated at the sender and receiver sites. In part a of the figure, the value of partial sum for the first column is 0x36. We keep the rightmost digit (6) and insert the leftmost digit (3) as the carry in the second column. The process is repeated for each column. Note that if there is any corruption, the checksum recalculated by the receiver is not all 0s. We leave this an exercise.

Figure



a. Checksum at the sender site



a. Checksum at the receiver site

Performance

- The checksum detects all errors involving an odd number of bits.
- 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 sums of those columns will not change and the receiver will not detect a problem.