

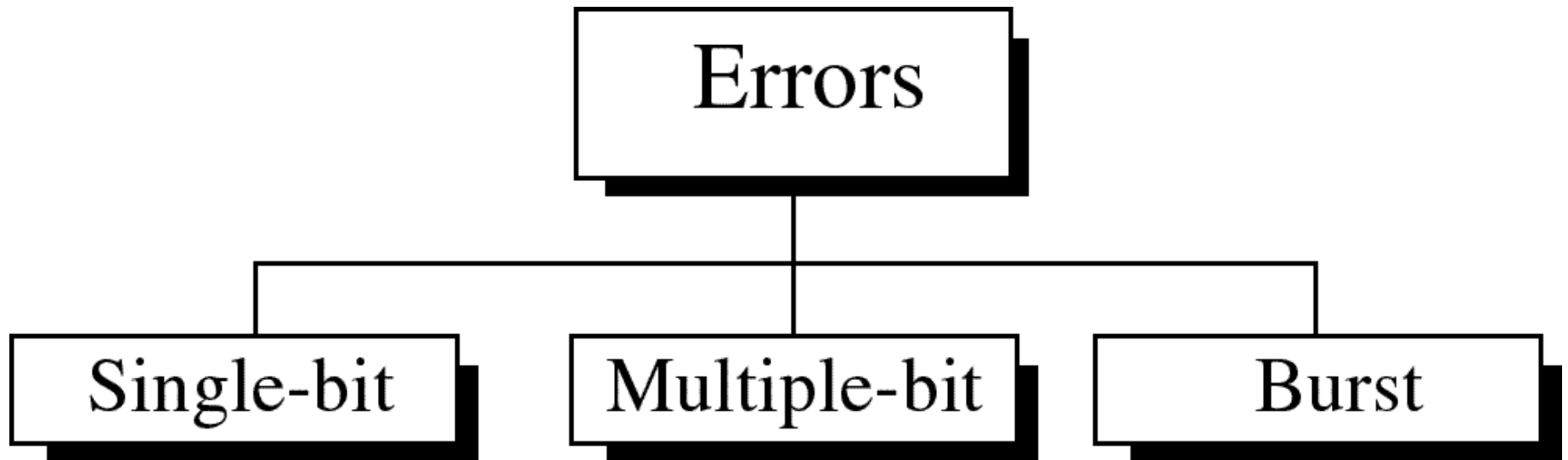
Error Detection and Correction

- **Types of Errors**
- **Detection**
- **Correction**

Basic concepts

- ★ 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 (open

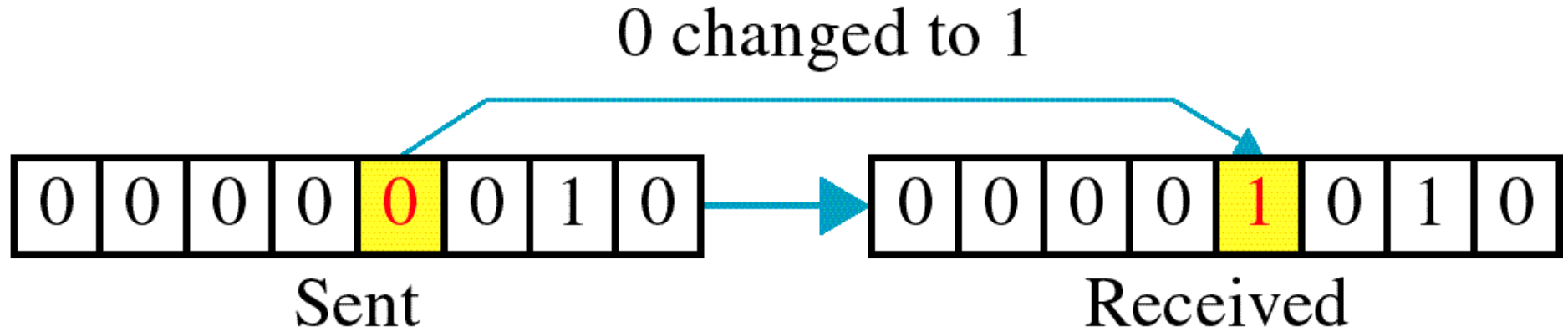
Types of Errors



Note

In a single-bit error, only 1 bit in the data unit has changed.

Single-bit error

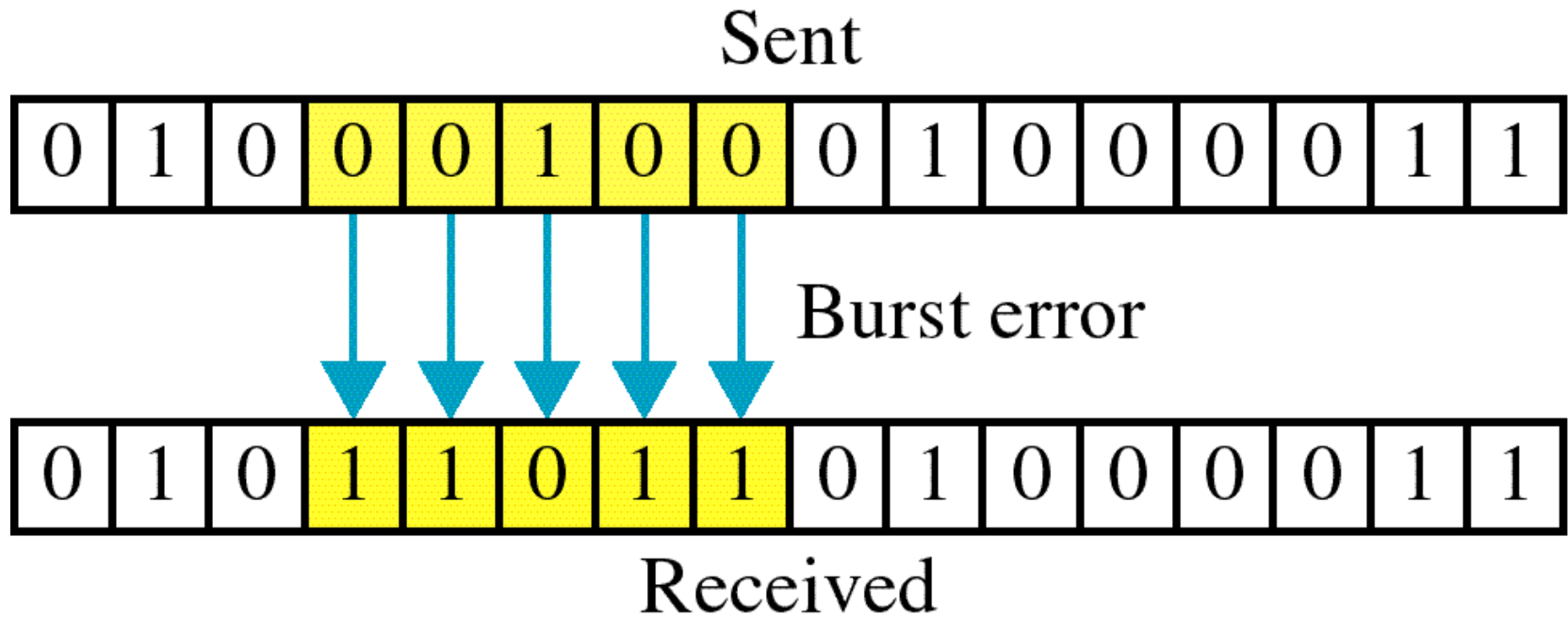


Single bit errors are the **least likely** type of errors in serial data transmission because the noise must have a very short duration which is very rare. However this kind of errors can happen in **parallel transmission**.

Example:

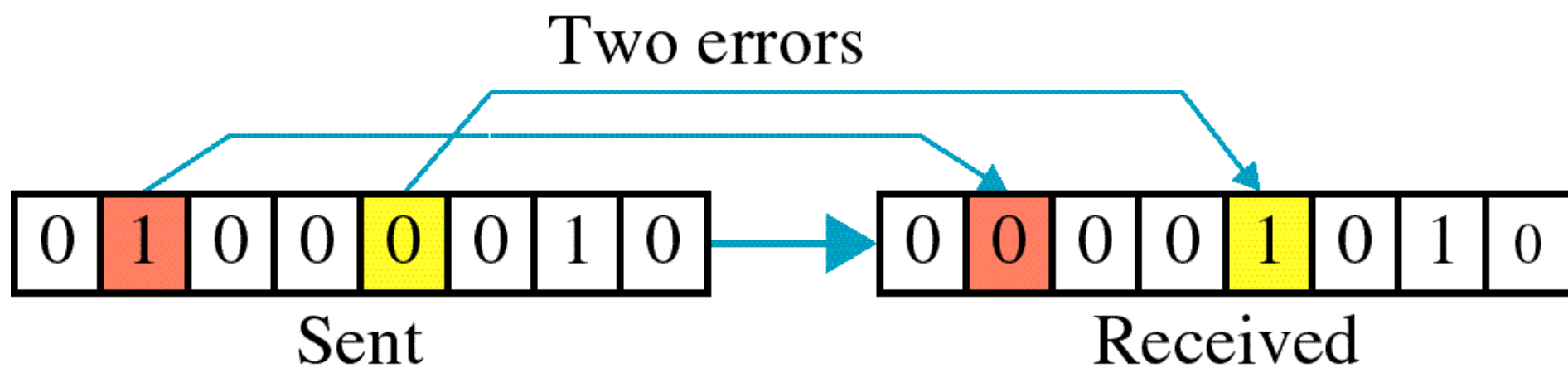
- ★ If data is sent at 1Mbps then each bit lasts only $1/1,000,000$ sec. or $1\ \mu\text{s}$.
- ★ For a single-bit error to occur, the noise must have a duration of only $1\ \mu\text{s}$, which is very rare.

Burst error



Note

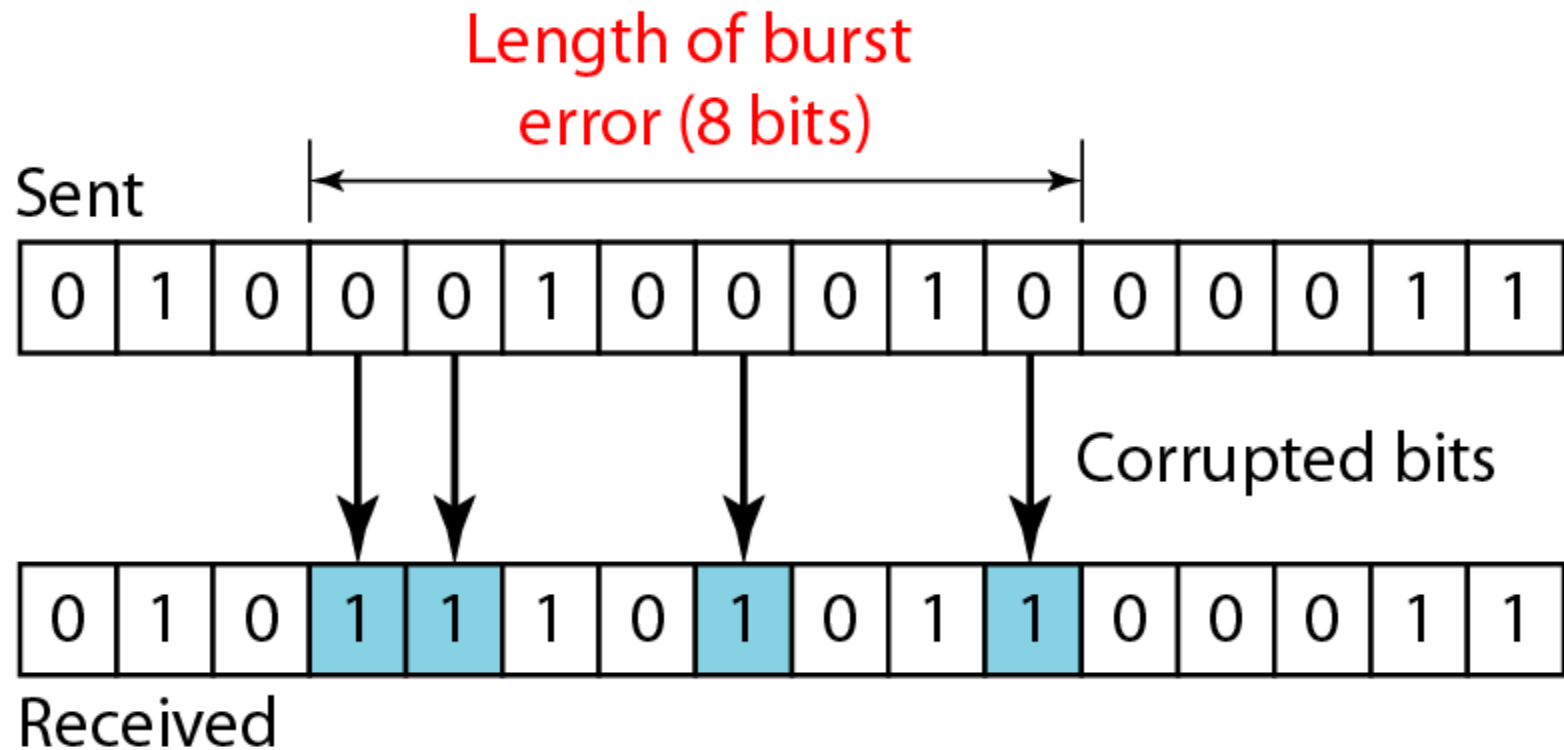
A burst error means that 2 or more bits in the data unit have changed.



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 of length 8



- ★ **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.

Example:

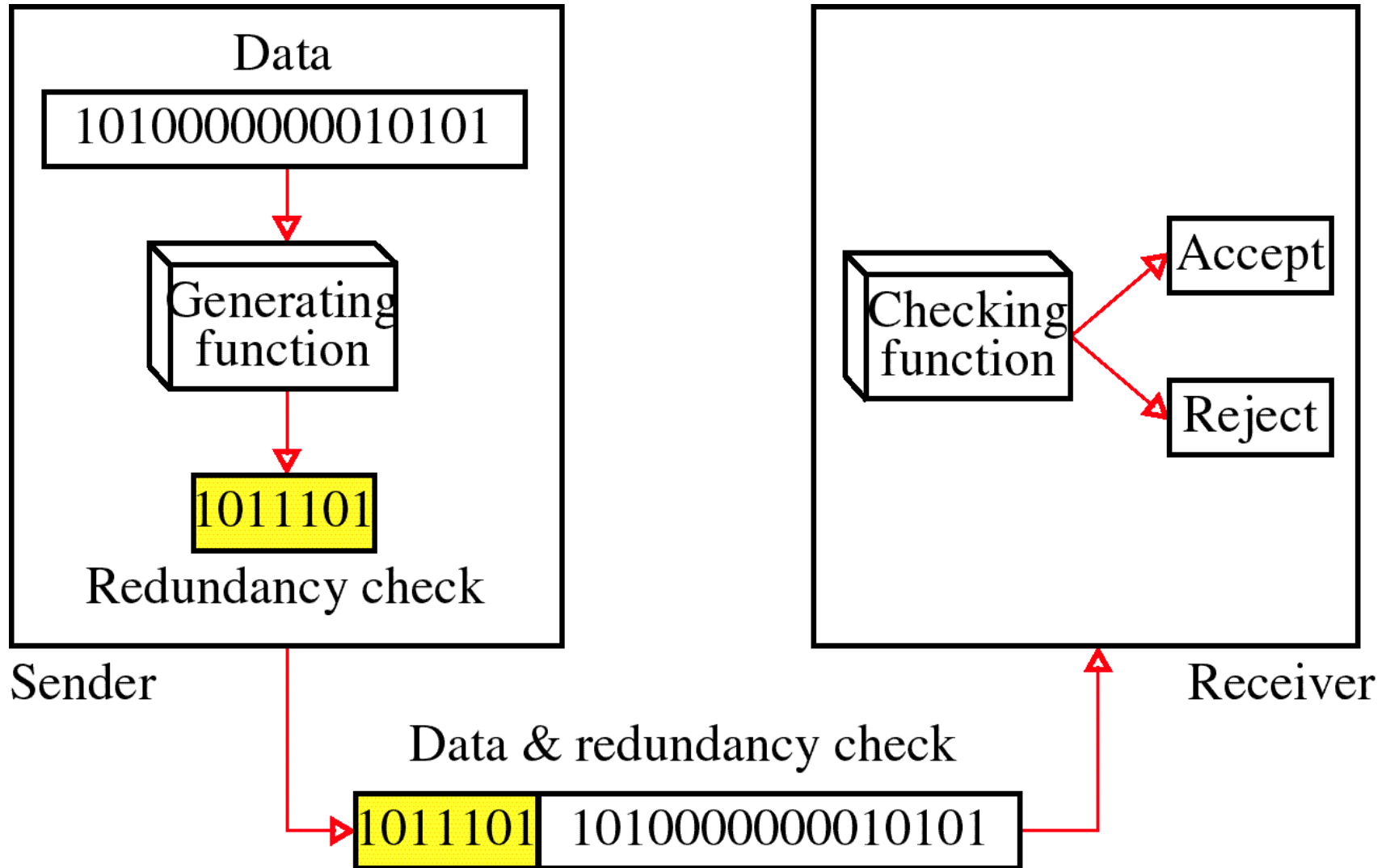
- ⇒ If data is sent at rate = 1Kbps then a noise of 1/100 sec can affect 10 bits. $(1/100 * 1000)$
- ⇒ If same data is sent at rate = 1Mbps then a noise of 1/100 sec can affect 10,000 bits. $(1/100 * 10^6)$

Error detection

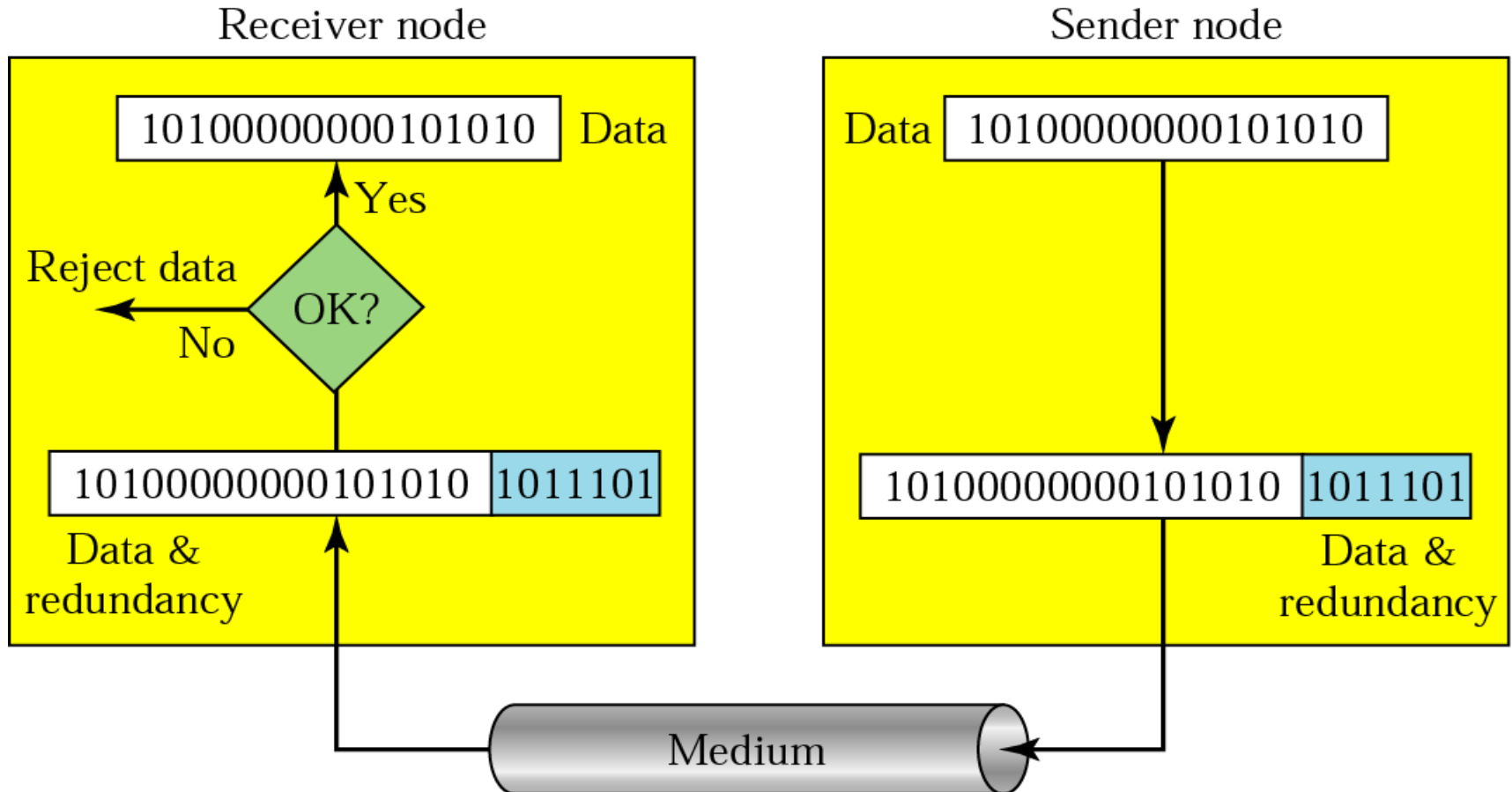
Error detection means to decide whether the received data is correct or not without having a copy of the original message.

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

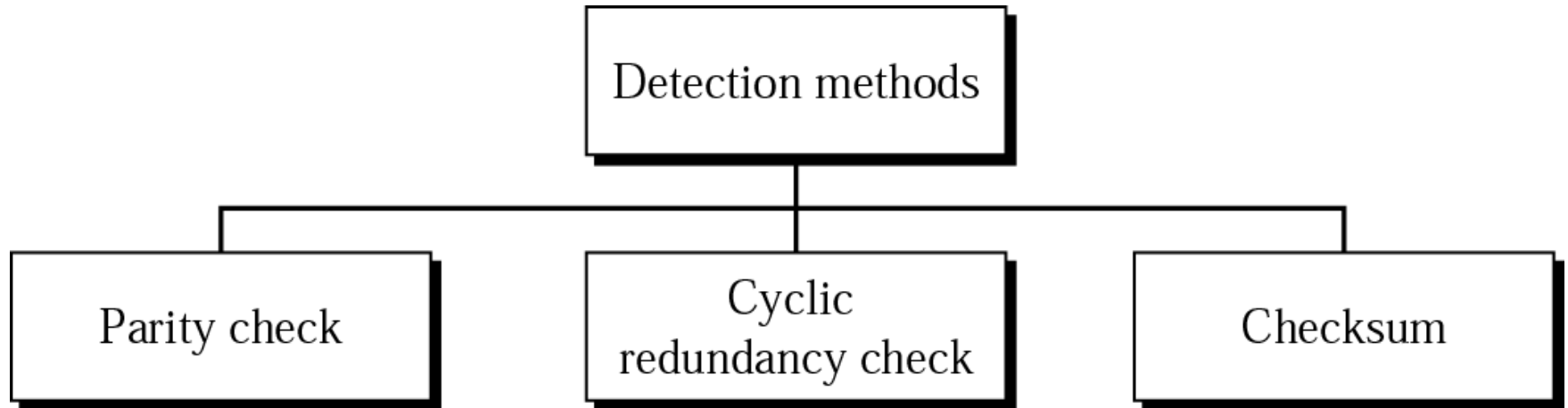
Redundancy



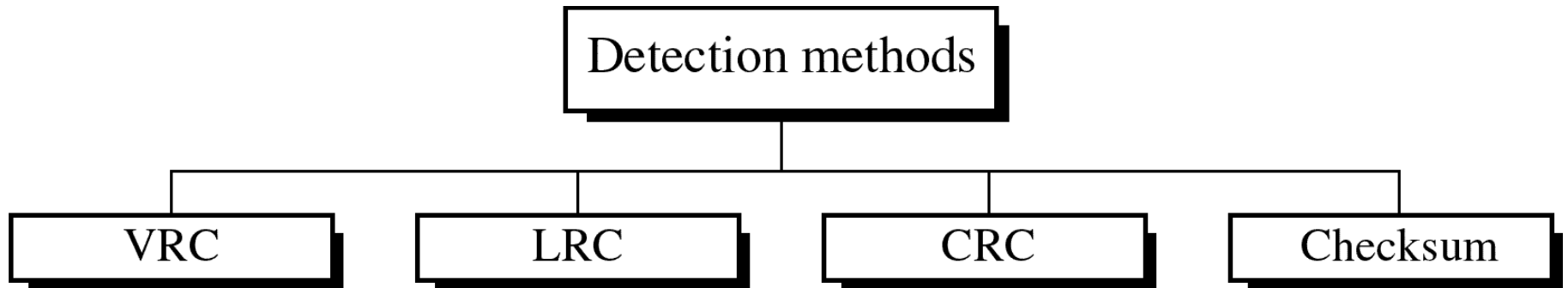
Redundancy (cont'd)



Detection methods



**Four types of redundancy checks are used
in data communications**



Single Parity Check(VRC)

Vertical Redundancy Check

- In Single parity check, a parity bit is added to every data unit so that the total number of 1s is even or odd.

1 1 0 1 0 0 1 1 

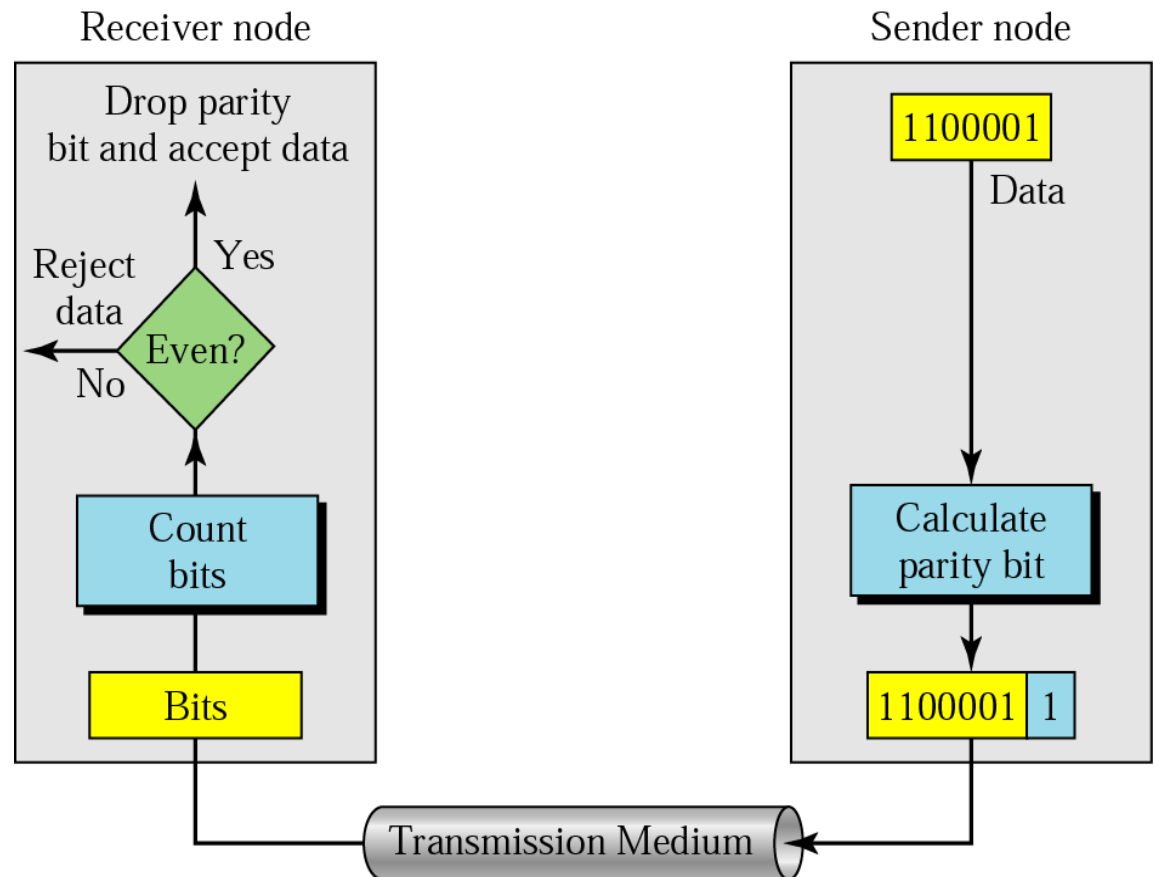
**Additional Bit
added(PARITY BIT)**

**Information(1 byte -
8 bits)**

- Therefore, the total number of bits transmitted would be 9 bits.
(8 bits of information + 1 bit parity = n data bits into n+1 bits)

Parity Check

- A parity bit is added to every data unit so that the total number of 1s(including the parity bit) becomes even for even-parity check or odd for odd-parity check
- Simple parity check



How is the **even parity** bit generated?

Total number of '1's should be even.

If the byte that we want to transmit is:

- **Step 1: count the number of 1's in the byte.**
• Answer: **5**

1 0 1 0 1 1 0

- **Step 2: compute the parity value.**
• Since the total number of **1's is 5**, the **even parity** bit will have a value of **1**.

1 0 1 0 1 1 **1** 0

- **If the number of bits are already even, the parity bit will be '0'.**

How is the **odd parity** bit generated?

Total number of '1's should be odd.

If the byte that we want to transmit is:

- **Step 1: count the number of 1's in the byte.**

1 0 1 0 1 1 0
0

- **Step 2: compute the parity value.**
- Since the total number of 1's is 4, the **odd parity** bit will have a value of **1**.
- **If the number of bits are already odd, the parity bit will be '0'.**

1 0 1 0 1 1 1 0
0

Detection -examples

Example 1

Suppose the sender wants to send the word *world*. In ASCII the five characters are coded as

**1110111 1101111 1110010
1101100 1100100**

The following shows the actual bits sent

**11101110 11011110 11100100 11011000
11001001**

Detection – examples

Example 2

Now suppose the word world in Example 1 is received by the receiver without being corrupted in transmission.

11101110 11011110 11100100
11011000 11001001

The receiver counts the 1s in each character and comes up with even numbers (6, 6, 4, 4, 4). The data are accepted.

Detection – examples

Example 3

Now suppose the word world in Example 1 is corrupted during transmission.

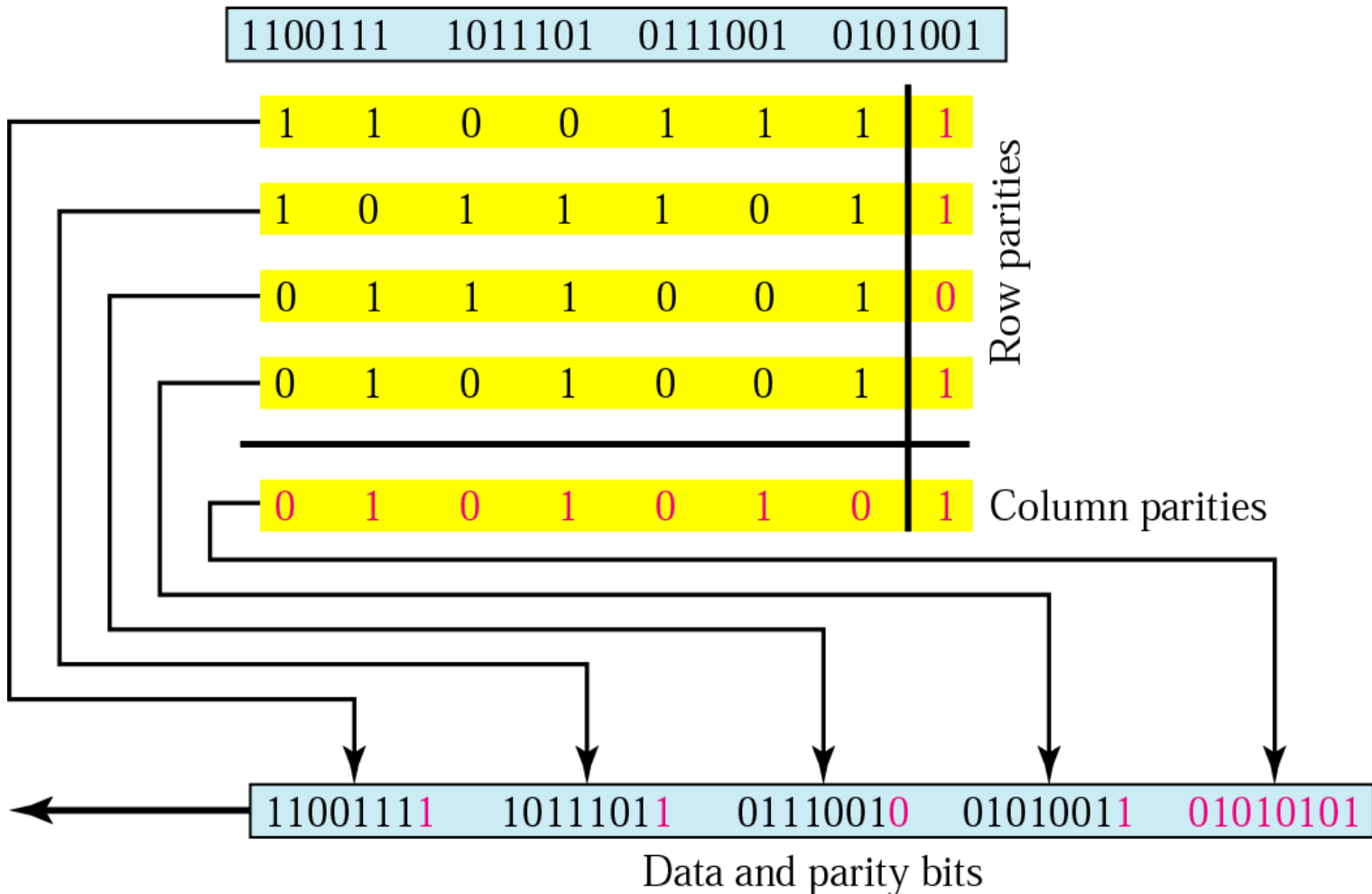
11111110 11011110 11101100
11011000 11001001

The receiver counts the 1s in each character and comes up with even and odd numbers (7, 6, 5, 4, 4). The receiver knows that the data are corrupted, discards them, and asks for retransmission.

Two Dimensional Parity Check(LRC)

Longitudinal Redundancy Check

Original data



Example 4 Detection - example

Suppose the following block is sent:

```
10101001 00111001 11011101 11100111
10101010
```

However, it is hit by a burst noise of length 8, and some bits are corrupted.

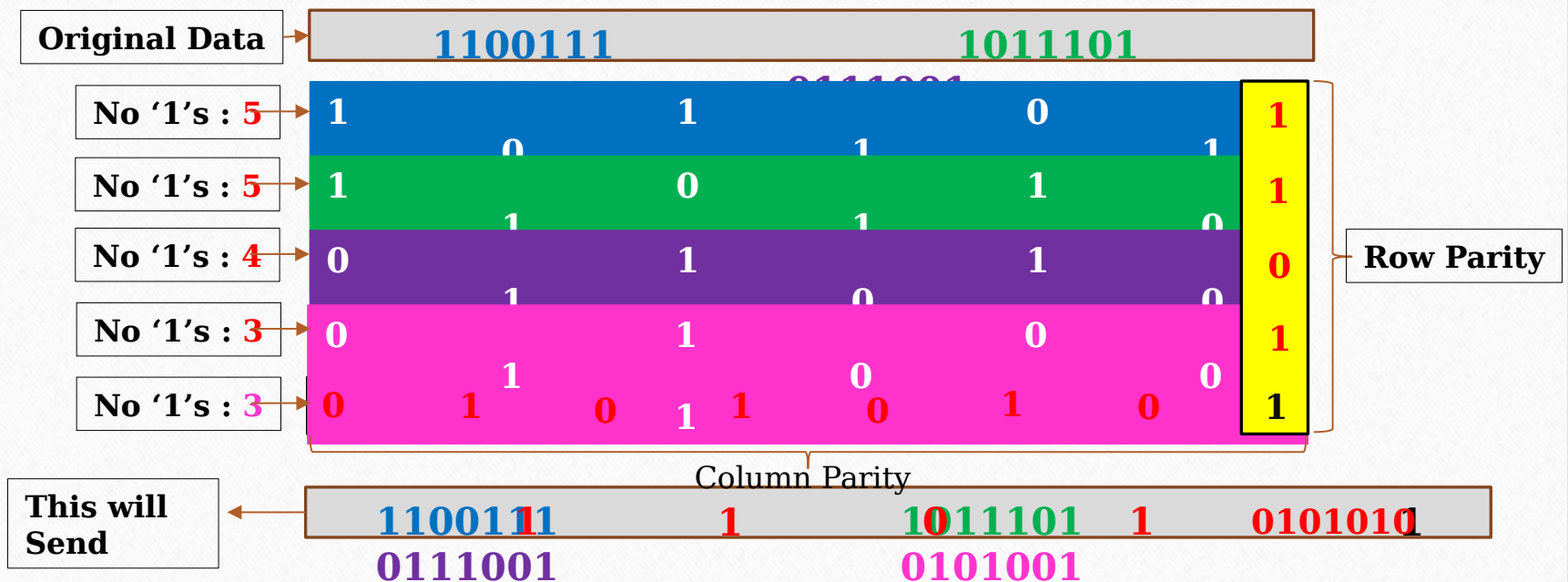
```
10100011 10001001 11011101 11100111
10101010
```

When the receiver checks the parity bits, some of the bits do not follow the even-parity rule and the whole block is discarded.

```
10100011 10001001 11011101 11100111
10101010
```

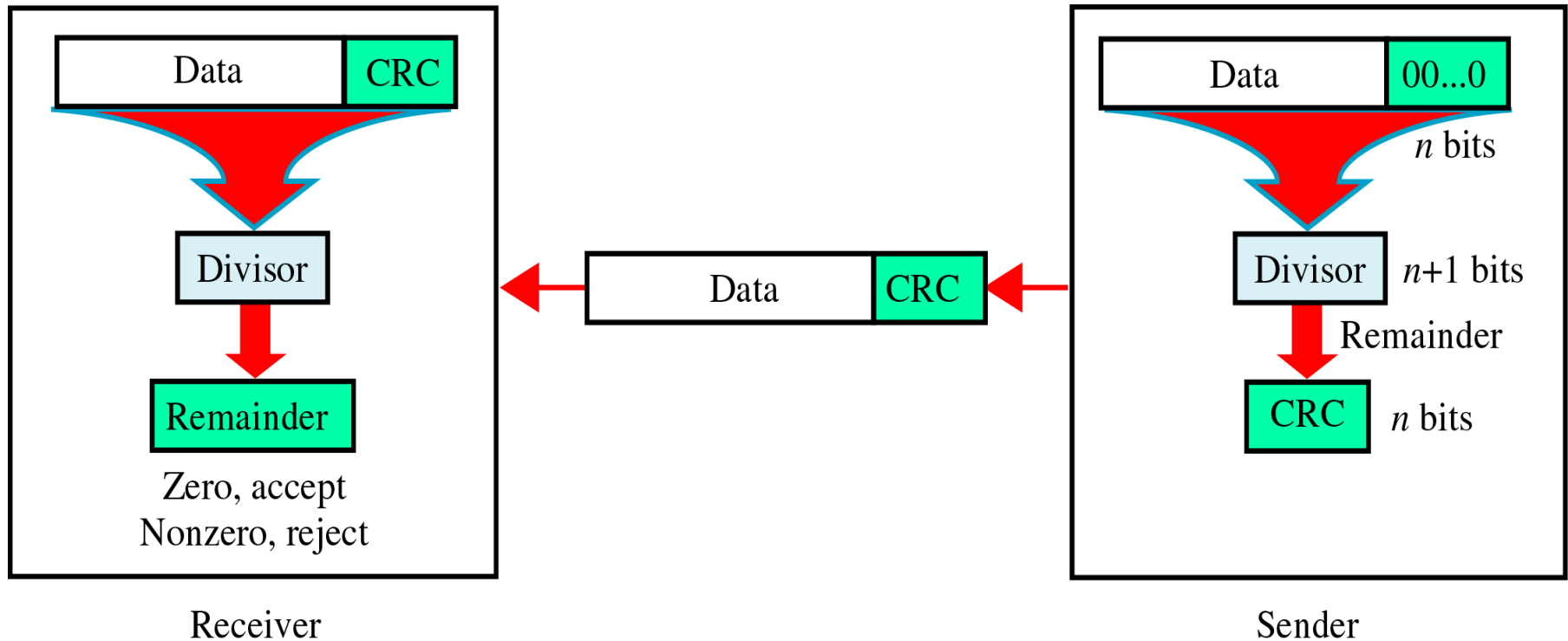
Two Dimensional Parity Generation

Even Parity Concept

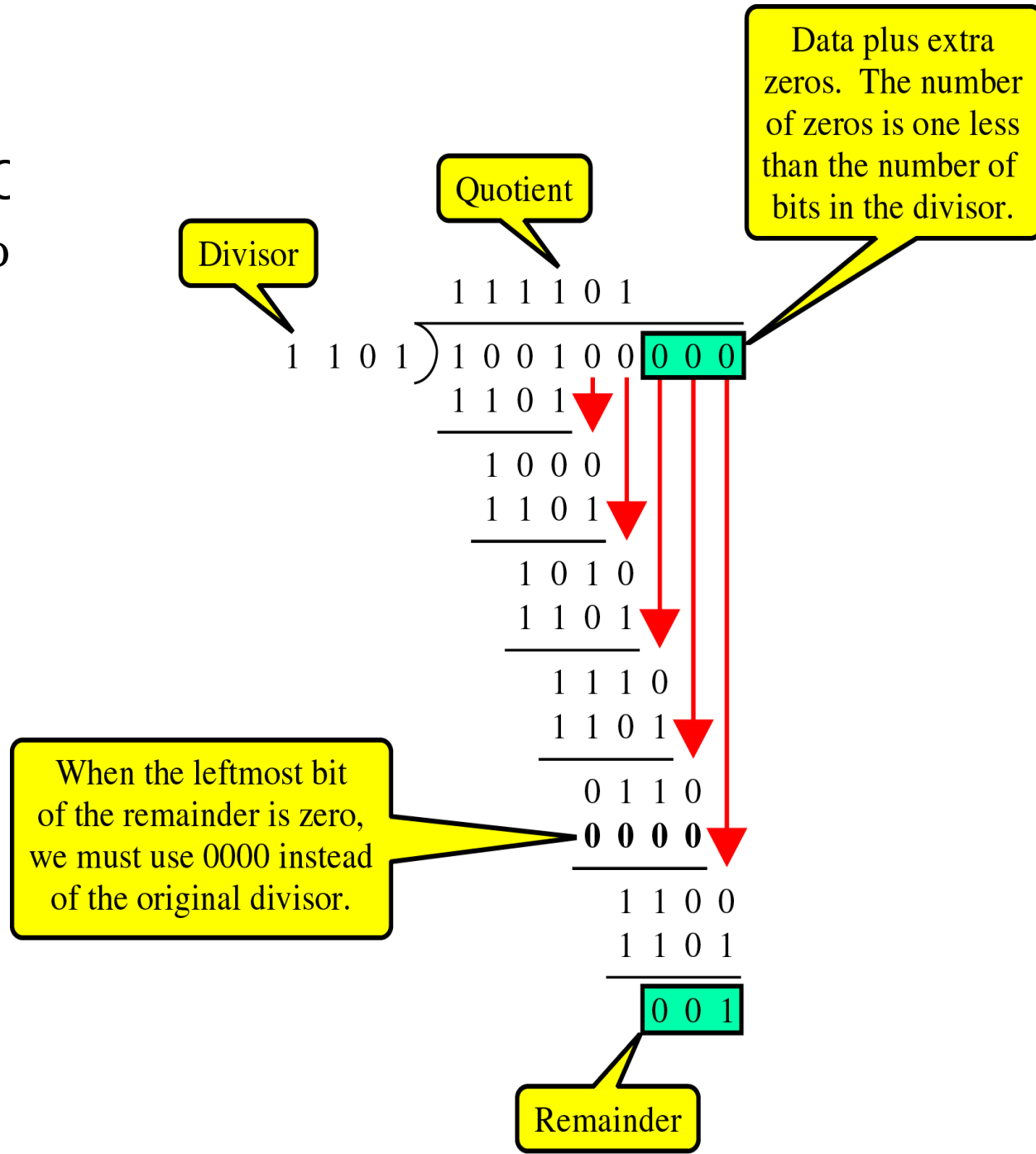


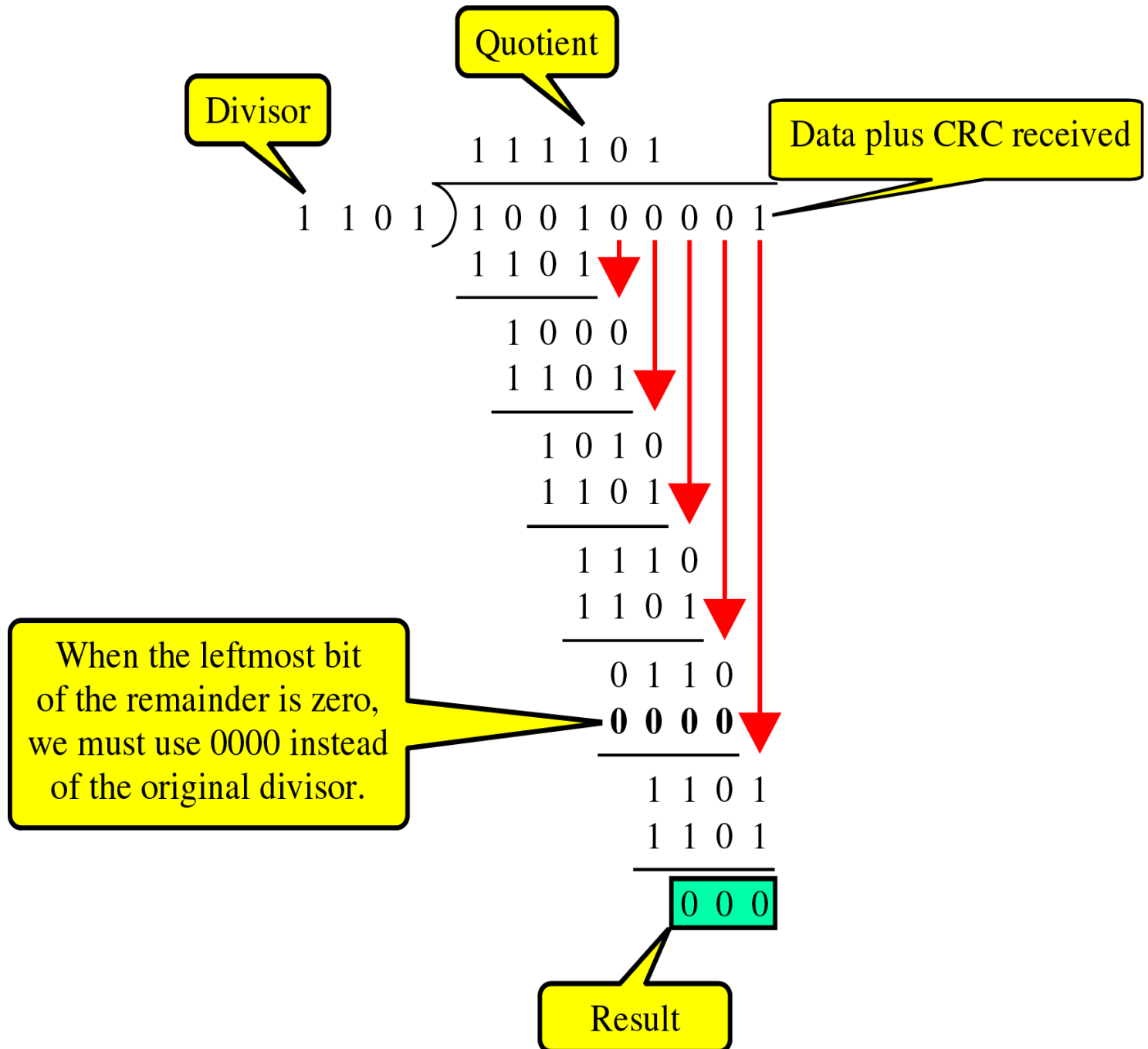
Detection(cont'd)

- CRC(Cyclic Redundancy Check)



- CRC generatc
- ~ uses modular-2 divisio



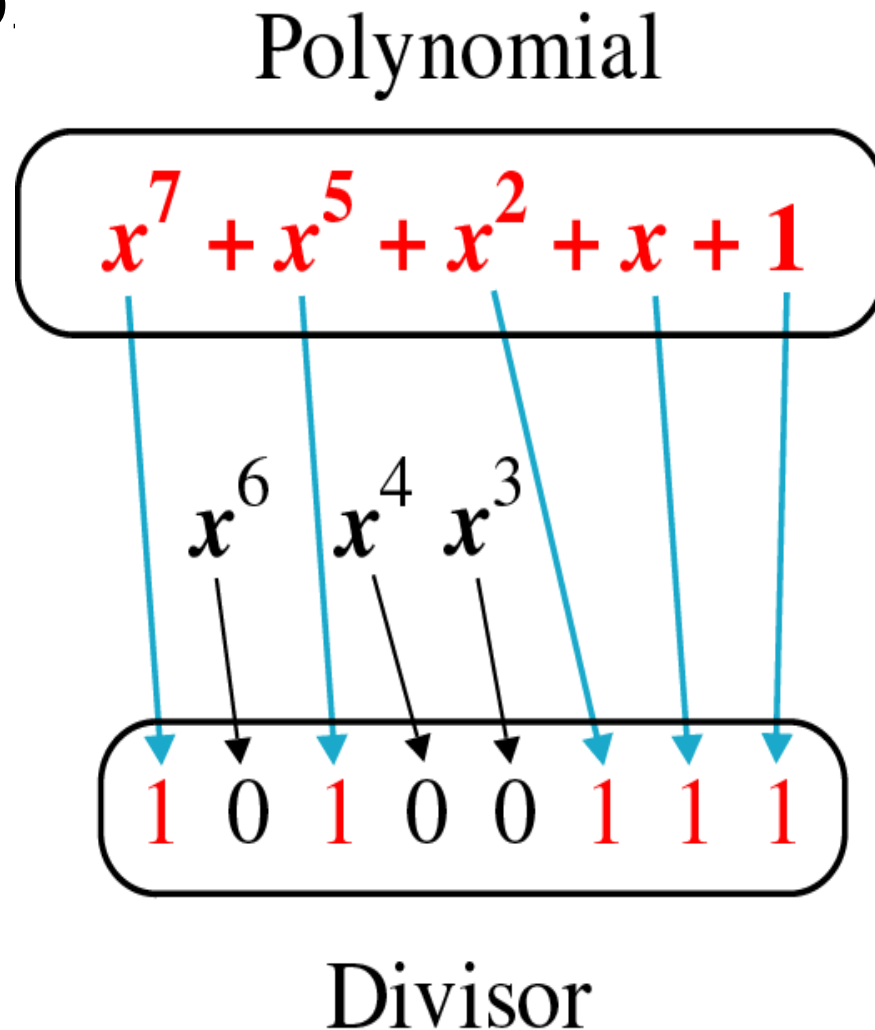


Detection(cont'd)

- Polynomials
 - CRC generator(divisor) is most often

$$x^7 + x^5 + x^2 + x + 1$$

- A polynomial representing a divisor



Detection(cont'd)

CRC-12

$$x^{12} + x^{11} + x^3 + x + 1$$

CRC-16

$$x^{16} + x^{15} + x^2 + 1$$

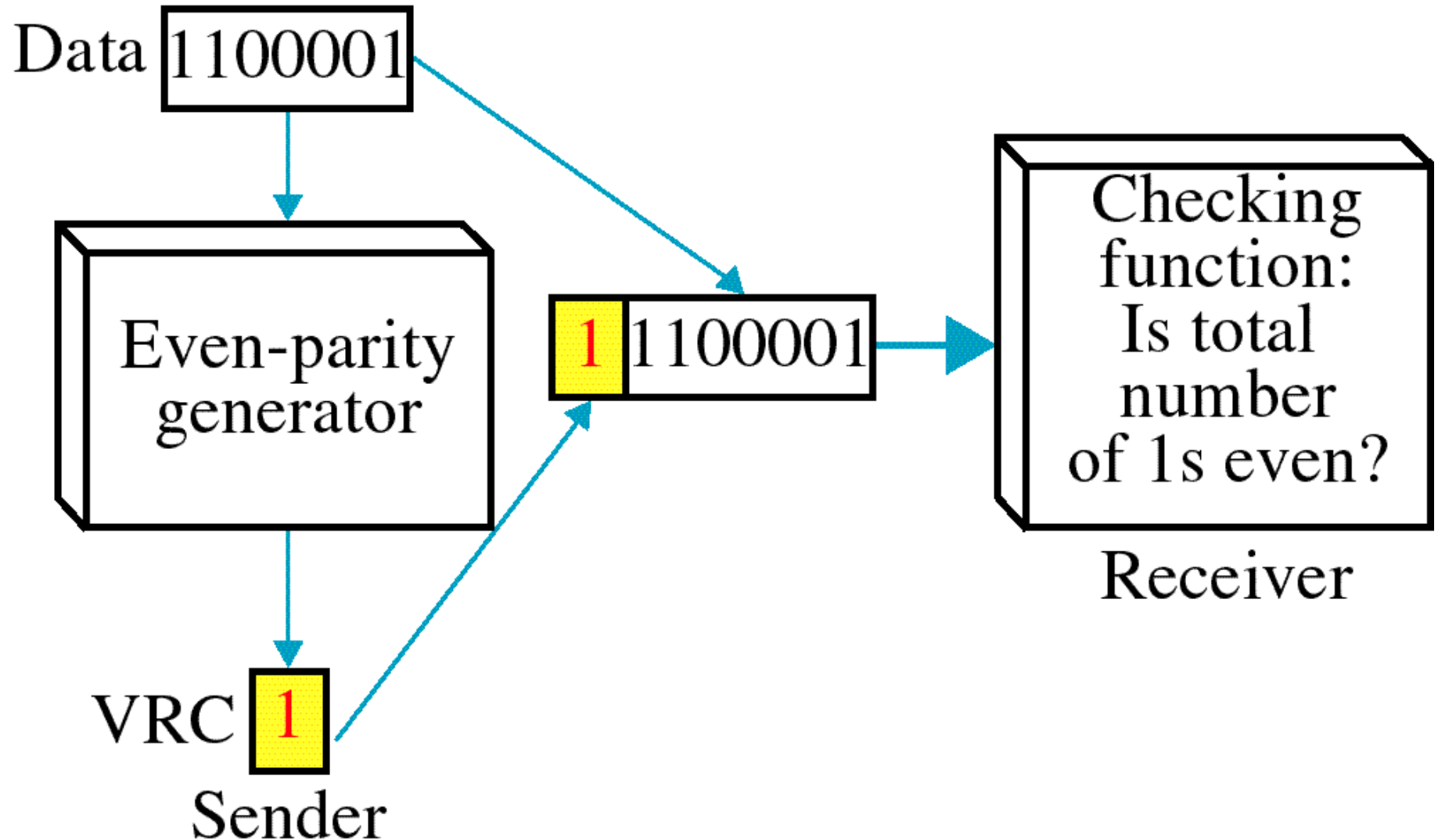
CRC-ITU-T

$$x^{16} + x^{12} + x^5 + 1$$

CRC-32

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

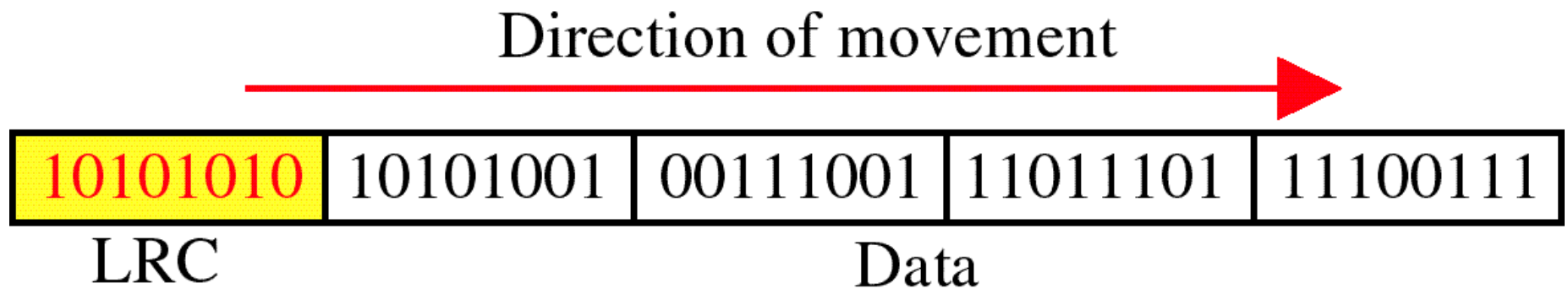
Vertical Redundancy Check VRC



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



Performance

- ⇒ LCR 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.

VRC and LRC

