Error Control

Error Control

- Positive and Negative feedback
- Timers : what happens when a frame completely vanishes : receiver neither sends a +ack nor –ack … then timer comes to help.
  - It may result in a frame being sent more than once and received more than once :
  - solution : assign sequence numbers to frames

Error Detection and Correction

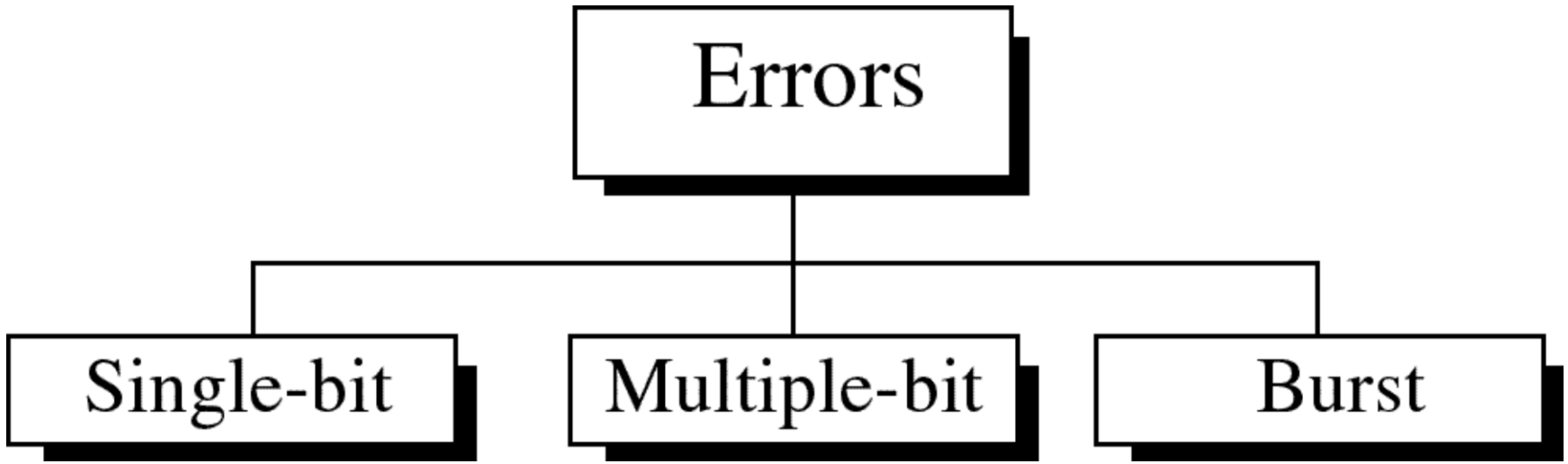
- In some cases it is sufficient to detect an error and in some, it requires the errors to be corrected also. For eg.
  - On a reliable medium : ED is sufficient where the error rate is low and asking for retransmission after ED would work efficiently
  - In contrast, on an unreliable medium : Retransmission after ED may result in another error and still another and so on. Hence EC is desirable.

Error Detection and Correction

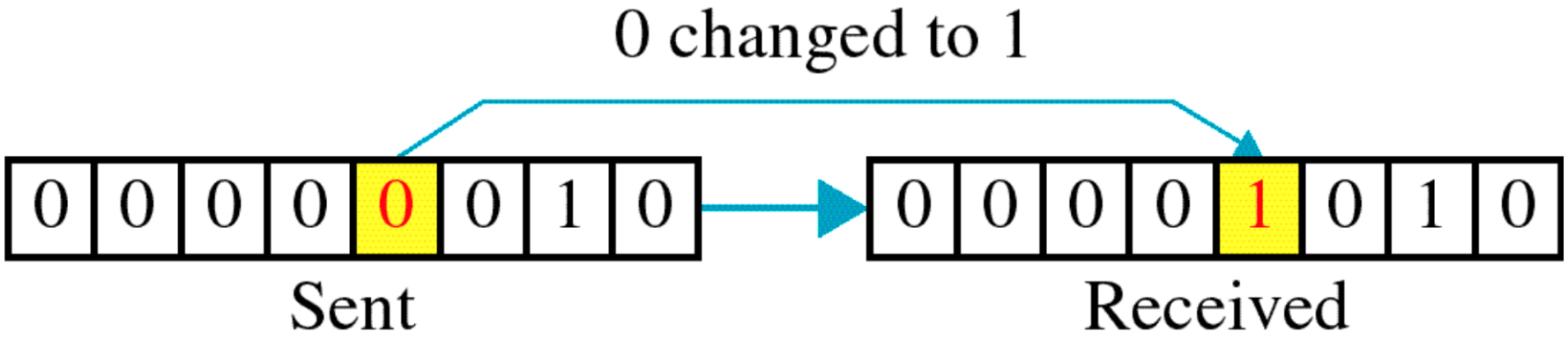
- Networks must be able to transfer data from one device to another with complete accuracy.
- Data can be corrupted during transmission.
- For reliable communication, errors must be detected and corrected.
- Error detection and correction are implemented either at the data link layer or the transport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

- Network designers have developed two basic strategies for dealing with errors. Both add redundant information to the data that is sent.
- One strategy is to include enough redundant information to enable the receiver to deduce what the transmitted data must have been.
- The other is to include only enough redundancy to allow the receiver to deduce that an error has occurred (but not which error) and have it request a retransmission.
- The former strategy uses error-correcting codes and the latter uses error-detecting codes.
- Use of error-correcting codes is called as Forward Error Correction

Types of Errors



Single-bit error

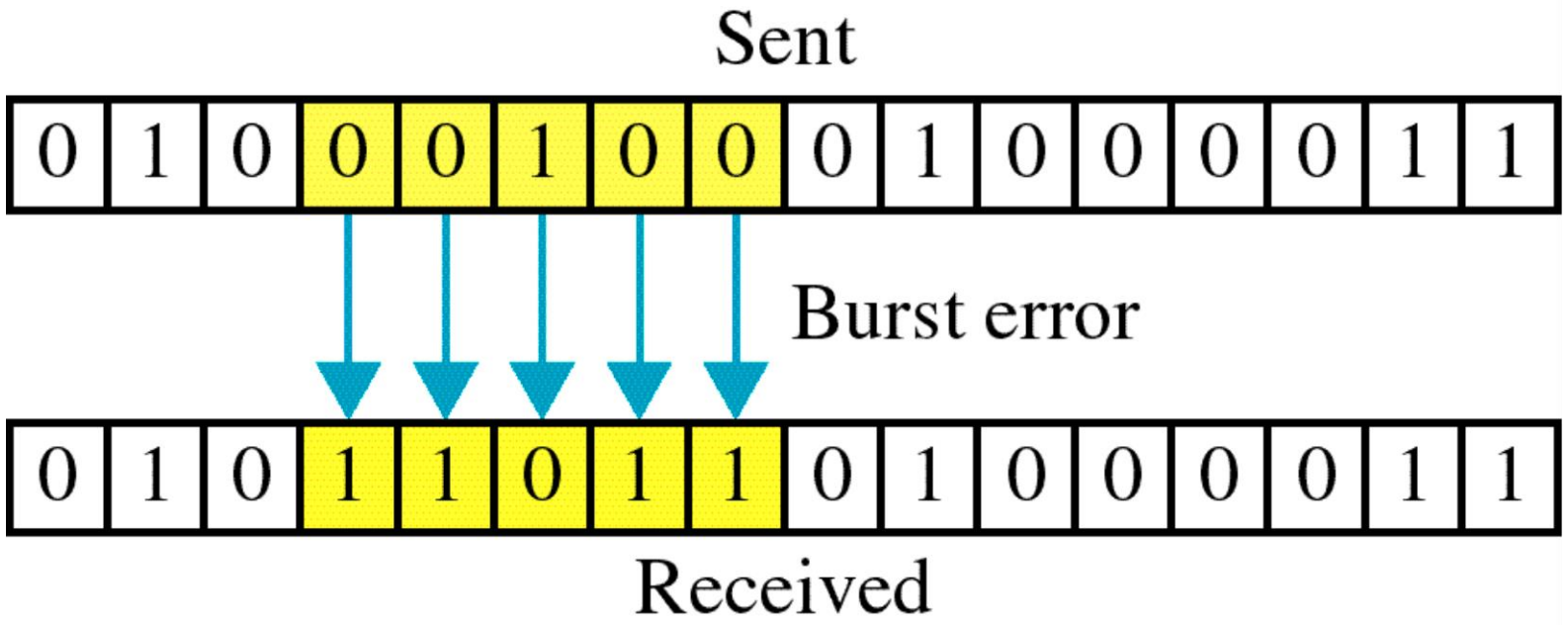


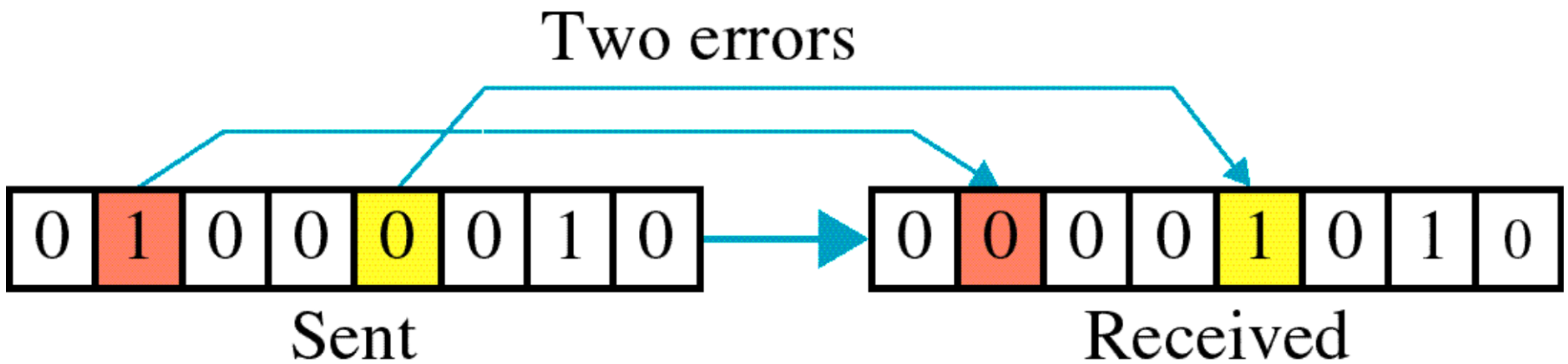
Single bit errors are the least likely type of errors in serial data transmission because the noise must have a very short duration which is very rare. However this kind of errors can happen in parallel transmission.

Example:

- If data is sent at 1Mbps then each bit lasts only 1/1,000,000 sec. or 1 µs.
- For a single-bit error to occur, the noise must have a duration of only 1 µs, which is very rare.

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
Sent
0 0 0 0 1 0 1 0
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.

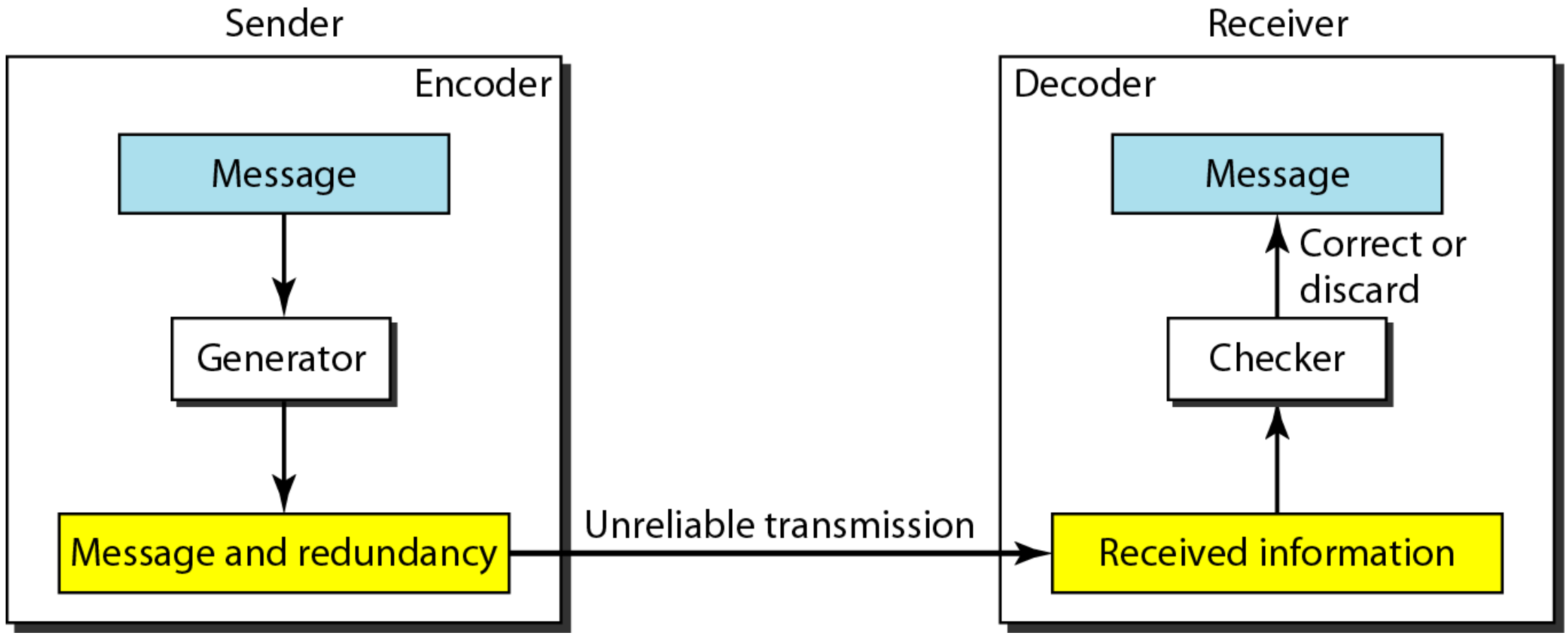
- Burst error is most likely to happen in serial transmission since the duration of noise is normally longer than the duration of a bit.
- The number of bits affected depends on the data rate and duration of noise.

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)

- Redundancy : is the central concept in detecting & correcting errors. We need to send some extra bits with our data.
- These redundant bits are added by the sender and removed by the receiver .
- To detect or correct errors, we need to send extra (redundant) bits with data.

The structure of encoder and decoder



Detection Versus Correction

- In error detection, we are looking only to see if any error has occurred. A single-bit error is the same for us as a burst error.
- In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message. So the number of errors and the size of the message are important factors.
- correction of errors is more difficult than the detection

Forward Error Correction Versus Retransmission

- Tow main methods of error correction
- Forward error correction FEC: is the process in which the receiver tries to guess the message by using redundant bits.
- Retransmission: is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message.
- FEC is used if the number of errors is small.

Coding

- Redundancy is achieved through various coding schemes. The sender adds redundant bits through a **process** that creates a relationship between the redundant bits and the actual data bits.
- The receiver checks the relationships between the two sets of bits to detect or correct the errors.
- The ratio of redundant bits to the data bits and the robustness of the process are important factors in any coding scheme

Coding schemes is divided into two categories :

- 1-block coding .
- 2-convolution coding.
- convolution coding is more complex than block coding.

Modular Arithmetic

- In modular arithmetic, we use only a limited range of integers. We define an upper limit, called a modulus N. We then use only the integers 0 to N -1.
- For example, if the modulus is 12, we use only the integers 0 to 11.
- In a modulo-N system, if a number is greater than N, it is divided by N and the remainder is the result.
- Addition and subtraction in modulo arithmetic are simple. There is no carry when you add two digits in a column. There is no carry when you subtract one digit from another in a column

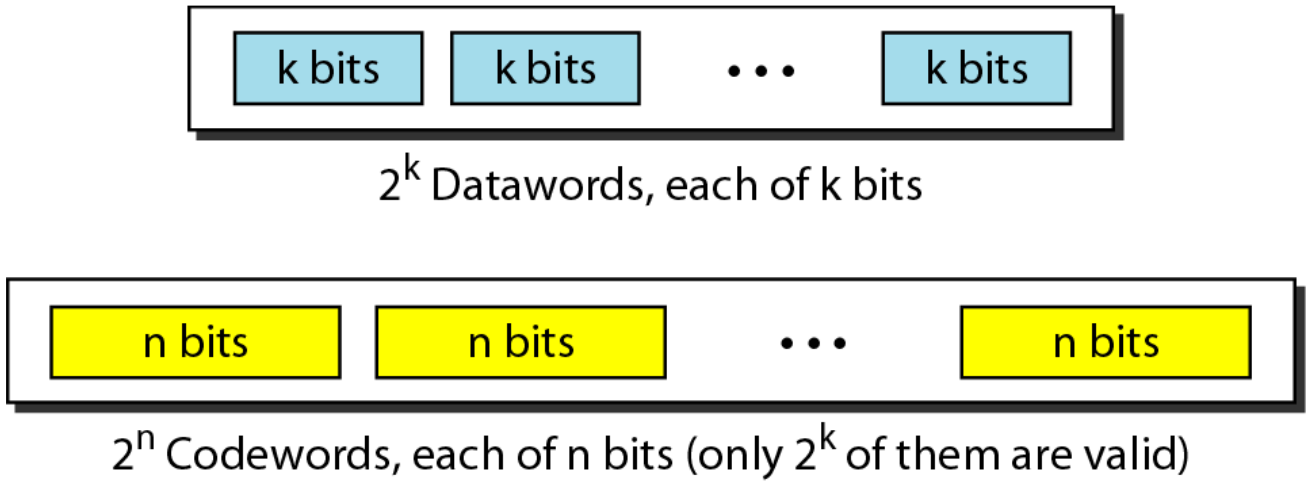
Modulo-2 Arithmetic

- Of particular interest is Modulo-2arithmetic. In this arithmetic ,the modulus is 2. We can use only 0 and 1. Operations in this arithmetic are very simple. The following shows how we can add or subtract 2 bits.
- Adding: 0+0=0 0+1=1 1+0=1 1+1=0
- Subtracting: 0 -0=0 0 -1=1 1-0=1 1 -1=0
- **Use** the XOR (exclusive OR) operation for both addition and subtraction.

BLOCK CODING

- In block coding, we divide our message into blocks, each of kbits, called data words.
- We add redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called code words.
- With k bits, we can create a combination of 2k data words; with n bits, we can create a combination of 2ncodewords.
- The block coding process is one-to-one; the same data word is always encoded as the same code word. This means that we have *2n-2k* code words that are not used.

Data words and code words in block coding



Error Detection

- How can errors be detected by using block coding? If the following two conditions are met, the receiver can detect a change in the original code word.

1. The receiver has (or can find) a list of valid code words.
2. The original code word has changed to an invalid one.

- An error-detecting code can detect only the types of errors for which it is; designed, other types of errors may remain undetected.

- **Hamming Distance**
- One of the central concepts in coding for error control is the idea of the Hamming distance.
- The Hamming distance can easily be found if we apply the XOR operation (Θ) on the two words and count the number of 1’sin the result. Note that the Hamming distance is a value greater than zero.

- The Hamming distance between two words is the number of differences between corresponding bits.

- ***The Hamming distance d(000, 011) is 2 because***

$$000 \oplus 011 \text{ is } 011 \text{ (two 1s)}$$

- ***The Hamming distance d(10101, 11110) is 3 because***

$$10101 \oplus 11110 \text{ is } 01011 \text{ (three 1s)}$$

- Minimum Hamming Distance - the measurement that is used for designing a code is the minimum Hamming distance.
- We used min to define the minimum Hamming distance in a coding scheme.
- To find this value, we find the Hamming distances between all words and select the smallest one.
- The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Datawords	Codewords
00	000
01	011
10	101
11	110

$d(000, 011) = 2$ $d(000, 101) = 2$ $d(000, 110) = 2$ $d(011, 101) = 2$
$d(011, 110) = 2$ $d(101, 110) = 2$

The d_{min} in this case is 2.

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

$d(00000, 01011) = 3$ $d(00000, 10101) = 3$ $d(00000, 11110) = 4$
$d(01011, 10101) = 4$ $d(01011, 11110) = 3$ $d(10101, 11110) = 3$

The d_{min} in this case is 3.

Three Parameters

- any coding scheme needs to have at least three parameters: the code word size n, the data word size k, and the minimum Hamming distance dmin.
- A coding scheme C is written as C(n, k) with a separate expression for dmin.
- For example, we can call our first coding scheme C(3, 2)
- with dmin=2 and our second coding scheme C(5, 2) with dmin= 3.

Hamming Distance and Error

- Relationship between the Hamming distance and errors occurring during transmission.
- When a code word is corrupted during transmission, the Hamming distance between the sent and received code words is the number of bits affected by the error.
- In other words, the Hamming distance between the received code word and the sent code word is the number of bits that are corrupted during transmission.
- For example, if the code word 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is d(00000, 01101) =3.
- To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be dmin= s + 1.

LINEAR BLOCK CODES

- *Almost all block codes used today belong to a subset called linear block codes. A linear block code is a code in which the exclusiveOR (additionmodulo-2) of two valid code words creates another valid code word.*

- In a linear block code, the exclusive OR (XOR) of any two valid code words creates another valid code word.

- Minimum Distance for Linear Block Codes
- It is simple to find the minimum Hamming distance for a linear block code.
- The minimum Hamming distance is the number of 1sin the nonzero valid code word with the smallest number of 1s.

Error Detection

- Error detection is the ability to detect the presence of errors caused by noise or other impairments during transmission from the transmitter to the receiver.
- 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.

- Three different error-detecting codes. They are all linear, systematic block codes:

1. Parity.
2. Checksums.
3. Cyclic Redundancy Checks (CRCs).

Parity

- The stream of data is broken up into blocks of bits, and the number of 1 bits is counted.Then, a "parity bit" is set (or cleared) if the number of one bits is odd (or even).
- The parity bit is chosen so that the number of 1 bits in the codeword is even (or odd).
- Doing this is equivalent to computing the (even) parity bit as the modulo 2 sum or XOR of the data bits.
- For example, when 1011010 is sent in even parity, a bit is added to the end to make it 10110100. With odd parity 1011010 becomes 10110101.

- A code with a single parity bit has a distance of 2, since any single-bit error produces a codeword with the wrong parity. This means that it can detect single-bit errors.
- If the tested blocks overlap, then the parity bits can be used to isolate the error, and even correct it if the error affects a single bit.

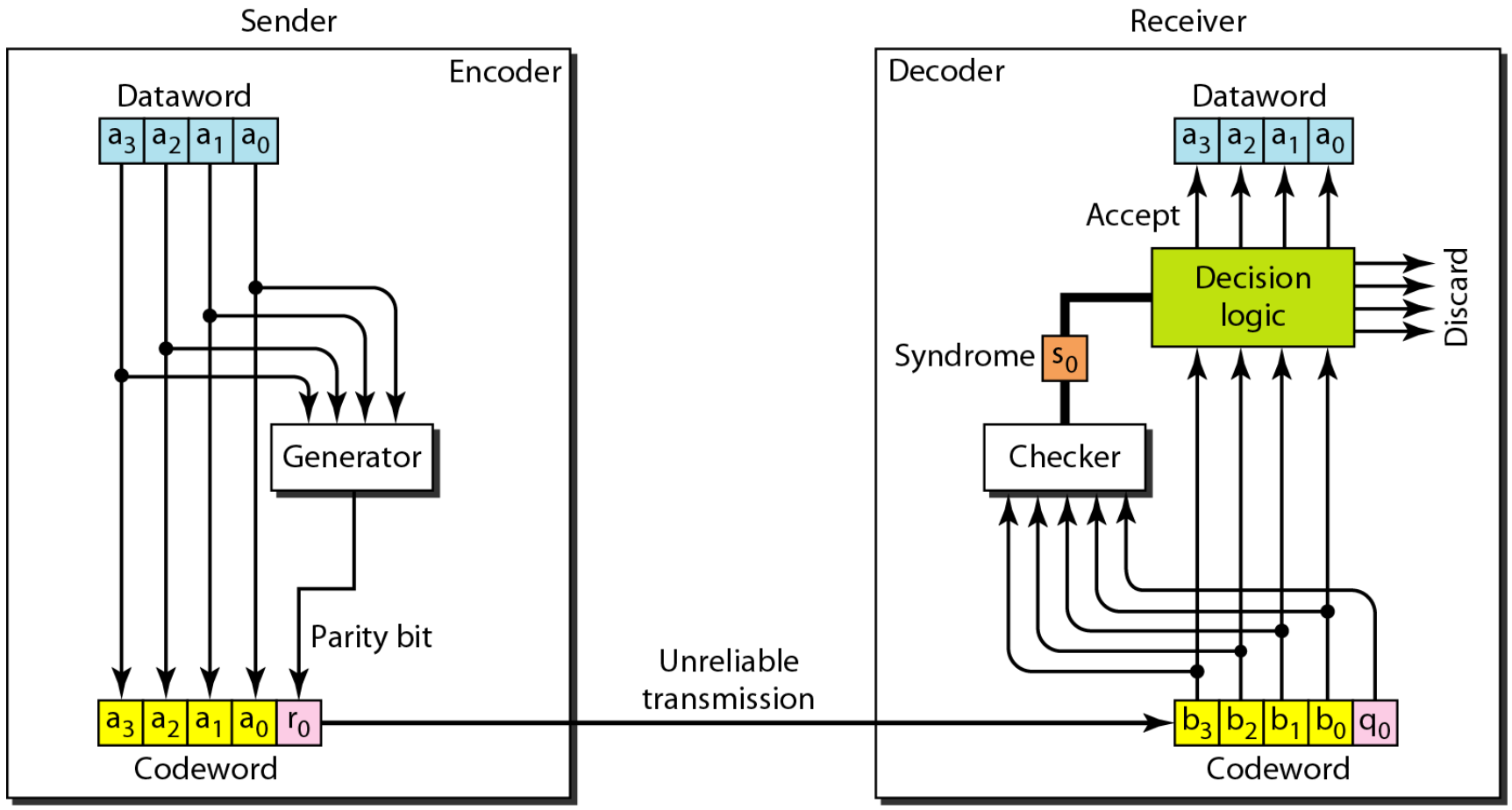
- A single parity bit can only reliably detect a single-bit error in the block. If the block is badly garbled by a long burst error, the probability that the error will be detected is only 0.5, which is hardly acceptable.
- Solution :
- **Interleaving** - compute the parity bits over the data in a different order than the order in which the data bits are transmitted.
- We compute a parity bit for each of the *n* columns and send all the data bits as *k* rows, sending the rows from top to bottom and the bits in each row from left to right in the usual manner. At the last row, we send the *n* parity bits.

- Simple Parity-Check Code: the most familiar error-detecting code is the simple parity-check code. In this
- code, a k-bit data word is changed to an n-bit code word where n = k + 1. The extra bit, called the parity bit, is selected to make the total number of 1s in the code word even.
- A simple parity-check code is a single-bit error-detecting code in which n = k + 1 with dmin= 2.10.46

Simple parity-check code C(5, 4)

Datawords	Codewords	Datawords	Codewords
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Encoder and decoder for simple parity-check code



- Some transmission scenarios. Assume the sender sends the data word 1011.
- The code word created from this data word is 10111, which is sent to the receiver.
- We examine five cases:

1.No error occurs; the received code word is 10111. The syndrome is 0. The data word 1011 is created.

2.One single-bit error changes a1. The received code word is 10011. The syndrome is 1. No data word is created.

3.One single-bit error changes r0. The received code word is 10110. The syndrome is 1. No data word is created.

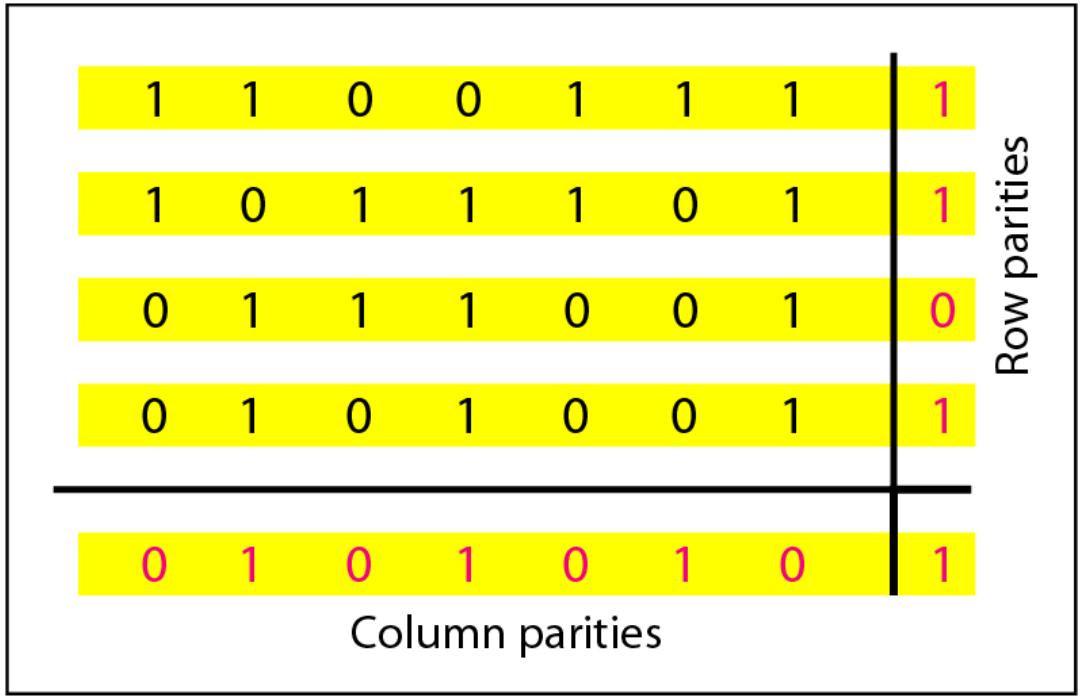
4.An error changes r0 and a second error changes a3.

- The received code word is 00110. The syndrome is 0.The data word 0011is created at the receiver. Note that here the data word is wrongly created due to the syndrome value.

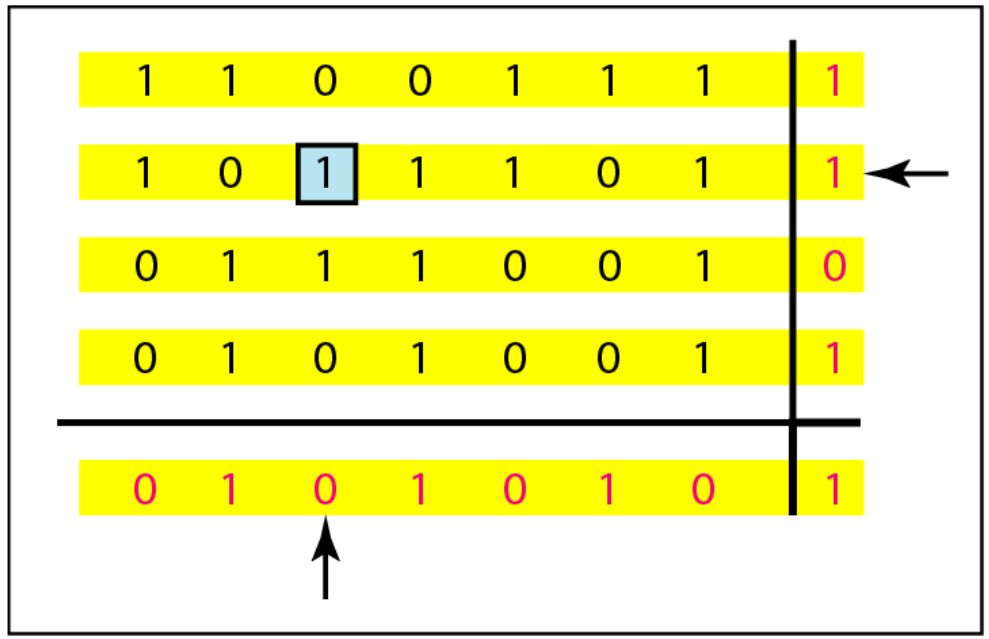
5. Three bits—a3, a2, and a1—are changed by errors. The received code word is (01011). The syndrome is 1.The data word is not created.

This shows that the simple parity check, guaranteed to detect one single error ,can also find any odd number of errors.

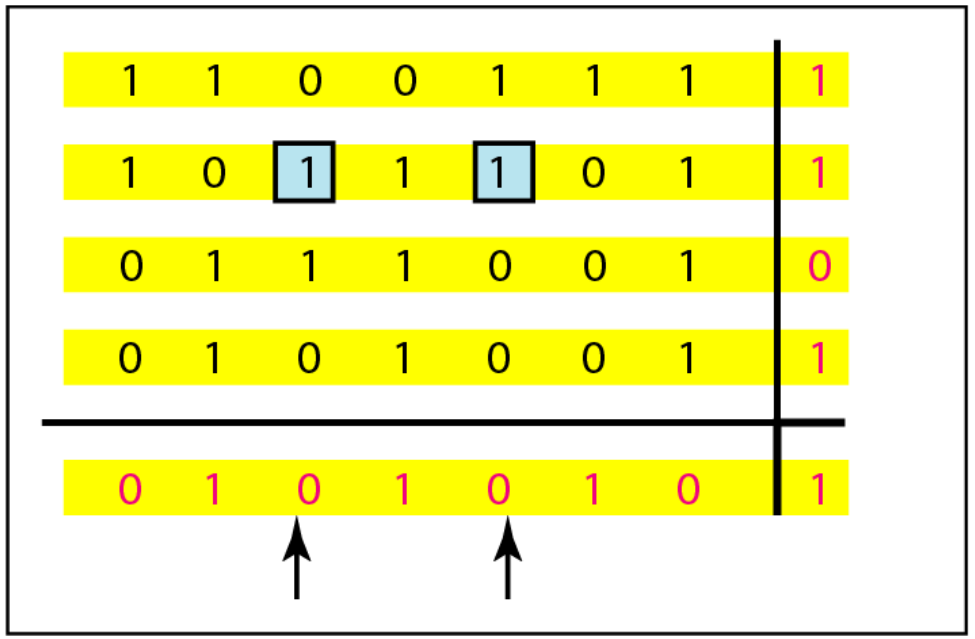
Two-dimensional parity-check code



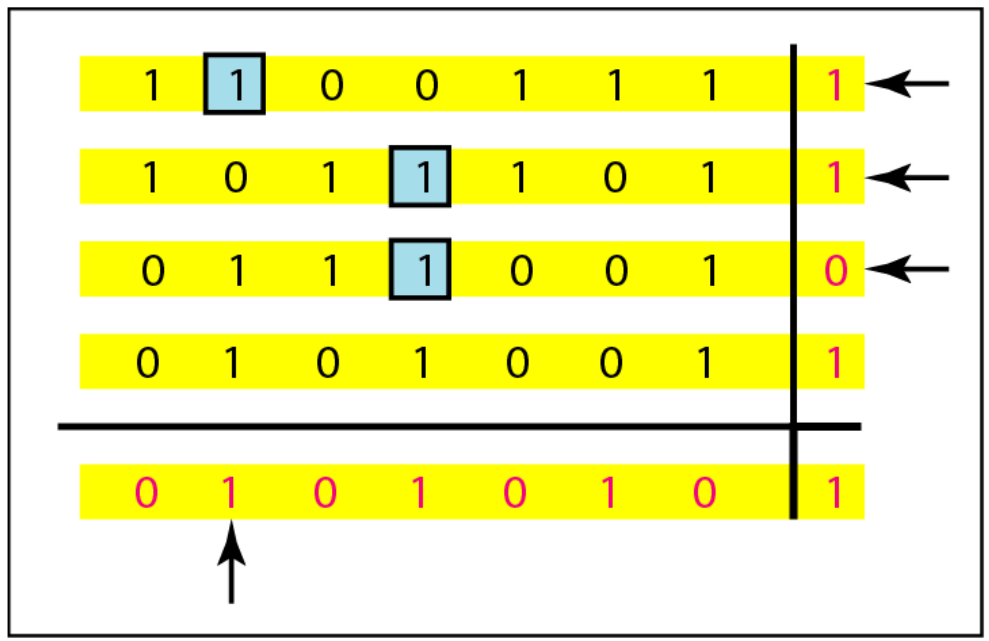
a. Design of row and column parities



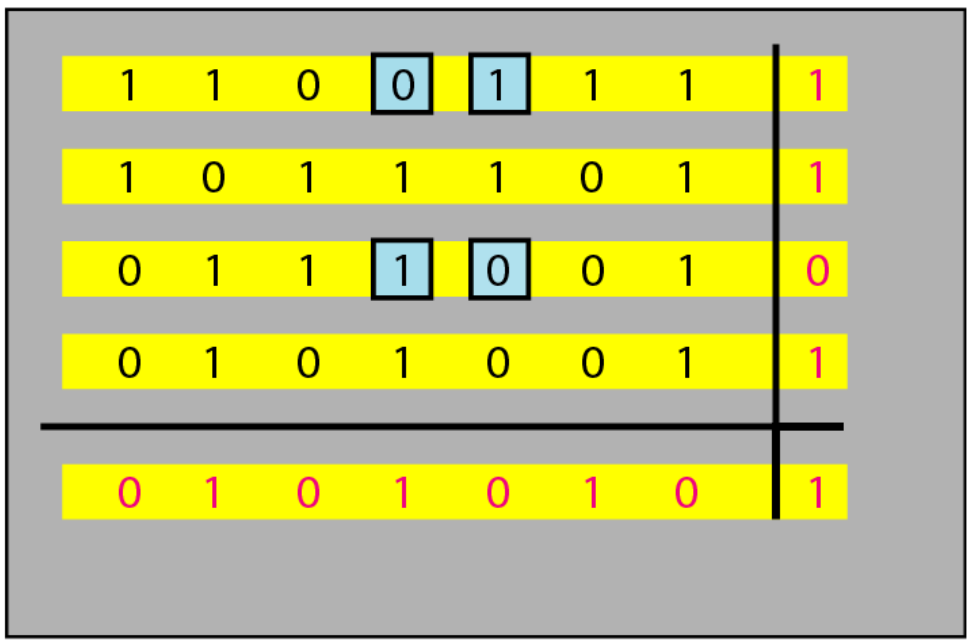
b. One error affects two parities



c. Two errors affect two parities



d. Three errors affect four parities



e. Four errors cannot be detected

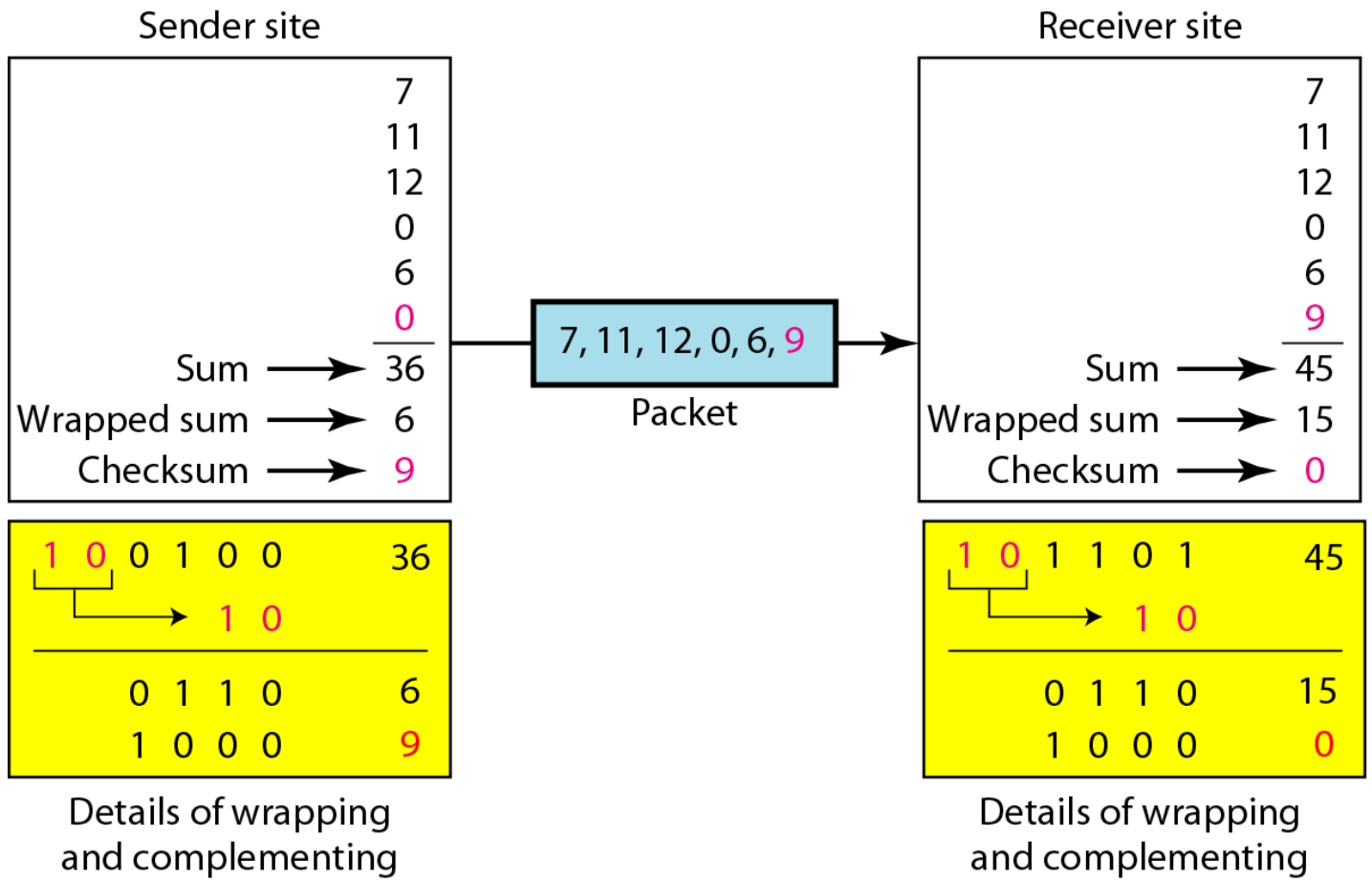
Checksum

- The second kind of error-detecting code, the **checksum**, is closely related to groups of parity bits.
- The word ‘‘checksum’’ is often used to mean a group of check bits associated with a message, regardless of how are calculated. A group of parity bits is one example of a checksum.
- However, there are other, stronger checksums based on a running sum of the data bits of the message. The checksum is usually placed at the end of the message, as the complement of the sum function.
- This way, errors may be detected by summing the entire received codeword, both data bits and checksum. If the result comes out to be zero, no error has been detected.

- A checksum of a message is an arithmetic sum of message code words of a certain word length, for example byte values, and their carry value. The sum is negated by means of ones-complement, and stored or transferred as an extra code word extending the message.
- On the receiver side, a new checksum may be calculated from the extended message. If the new checksum is not 0, an error has been detected.

- The sender initializes the checksum to 0 and adds all dataitems and the checksum (the checksum is considered as one dataitem and is shown in color).The result is 36. However, 36 cannot be expressed in 4bits.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 sum is then complemented, resulting in the checksum value 9(15−6=9).The sender now sends six dataitems to the receiver including the checksum 9.

- The receiver follows the same procedure as the sender. It adds all dataitems (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 dataitems. If the checksum is not zero, the entire packet is dropped.


Sender site
7
11
12
0
6
0
Sum
36
Wrapped sum
6
Checksum
9
7, 11, 12, 0, 6, 9
Packet
Receiver site
7
11
12
0
6
9
Sum
45
Wrapped sum
15
Checksum
0
1 0 0 1 0 0 36
1 0
0 1 1 0 6
1 0 0 0 9
Details of wrapping and complementing
1 0 1 1 0 1 45
1 0
0 1 1 0 15
1 0 0 0 0
Details of wrapping and complementing

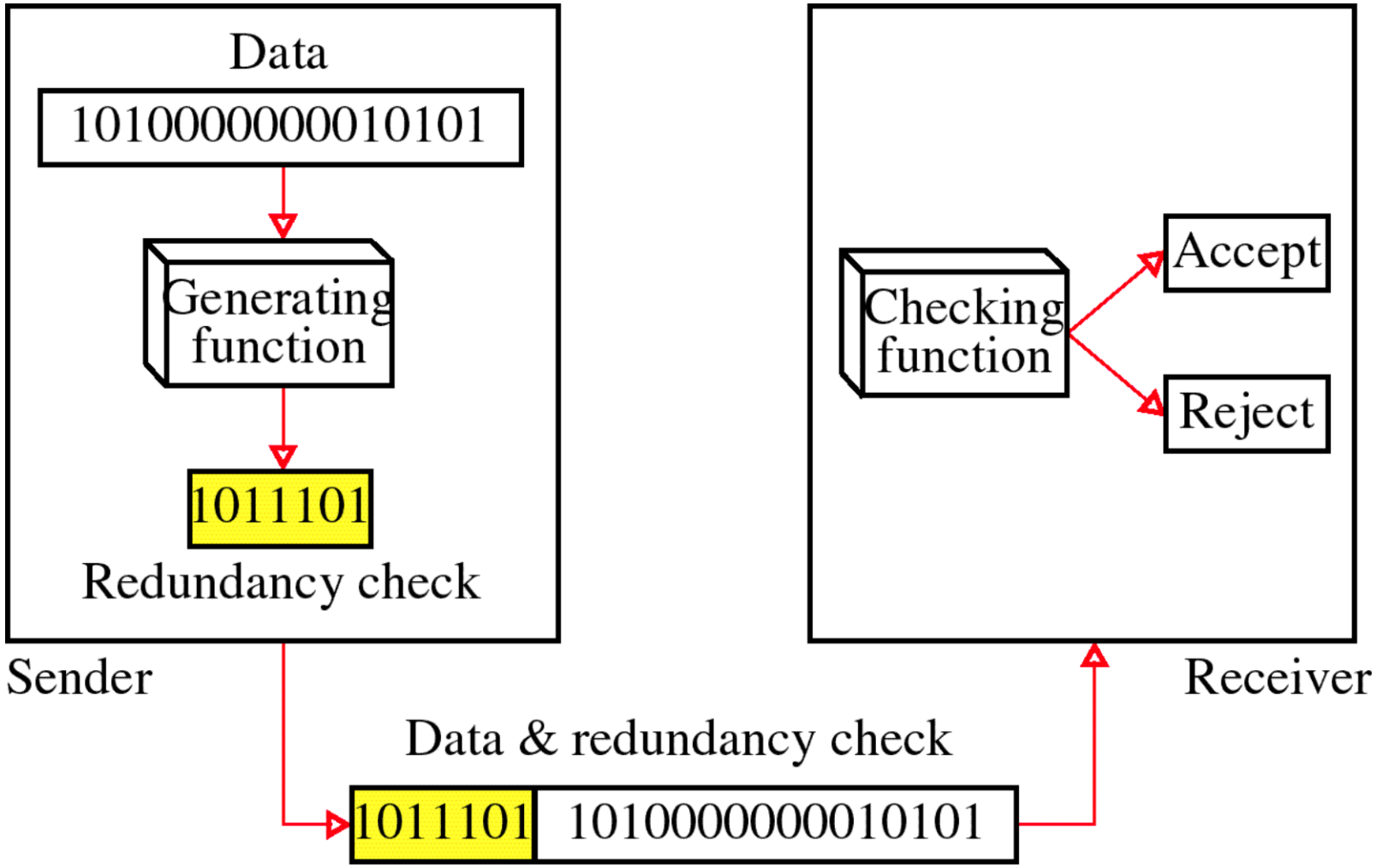
- **Sender site:**
- **1.The message is divided into 16-bit words.**
- **2.The value of the checksum word is set to 0.**
- **3.All words including the checksum areadded using one’s complement addition.**
- **4.The sum is complemented and becomes thechecksum.**
- **5.The checksum is sent with the data.**

- **Receiver site:**
- **1.The message (including checksum) isdivided into 16-bit words.**
- **2.All words are added using one’scomplement addition.**
- **3.The sum is complemented and becomes thenew checksum.**
- **4.If the value of checksum is 0, the messageis accepted; otherwise, it is rejected.**

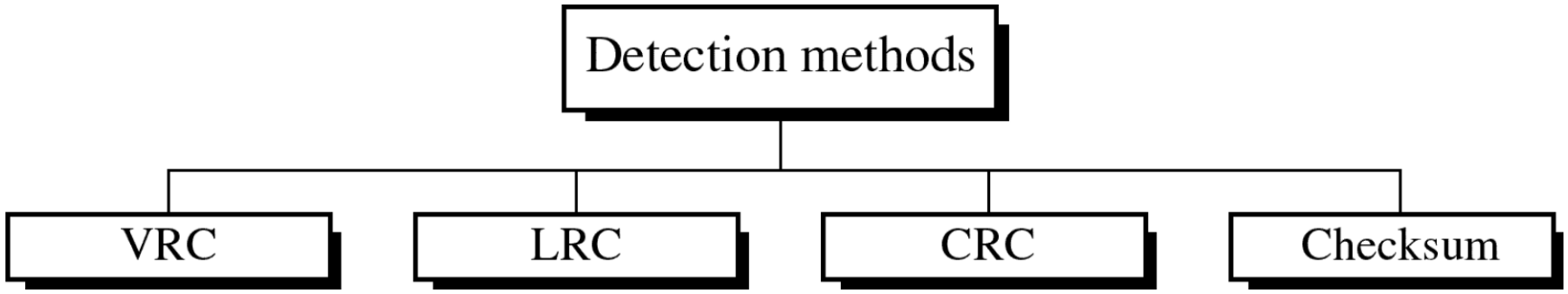
Cyclic redundancy check

- The cyclic redundancy check considers a block of data as the coefficients to a polynomial and then divides by a fixed, predetermined polynomial. The coefficients of the result of the division is taken as the redundant data bits, the CRC.
- On reception, one can recompute the CRC from the payload bits and compare this with the CRC that was received. A mismatch indicates that an error occurred.

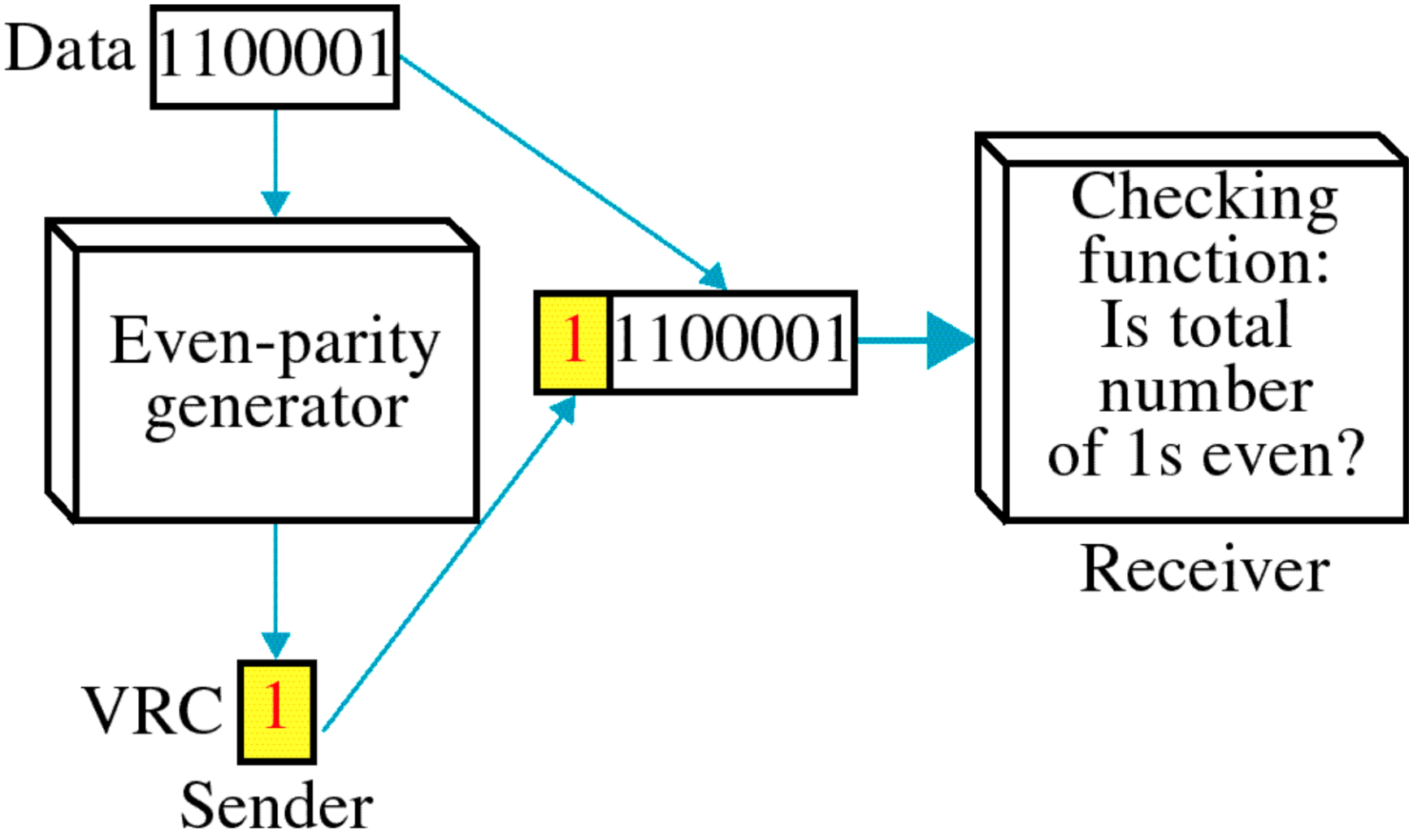
Redundancy


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



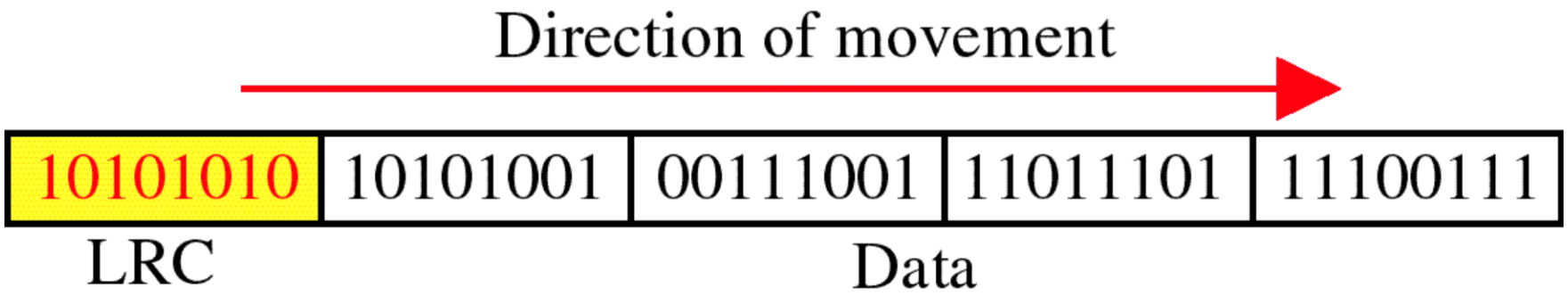
Vertical Redundancy Check
VRC



It can detect single bit error

It can detect burst errors only if the total number of errors is odd.

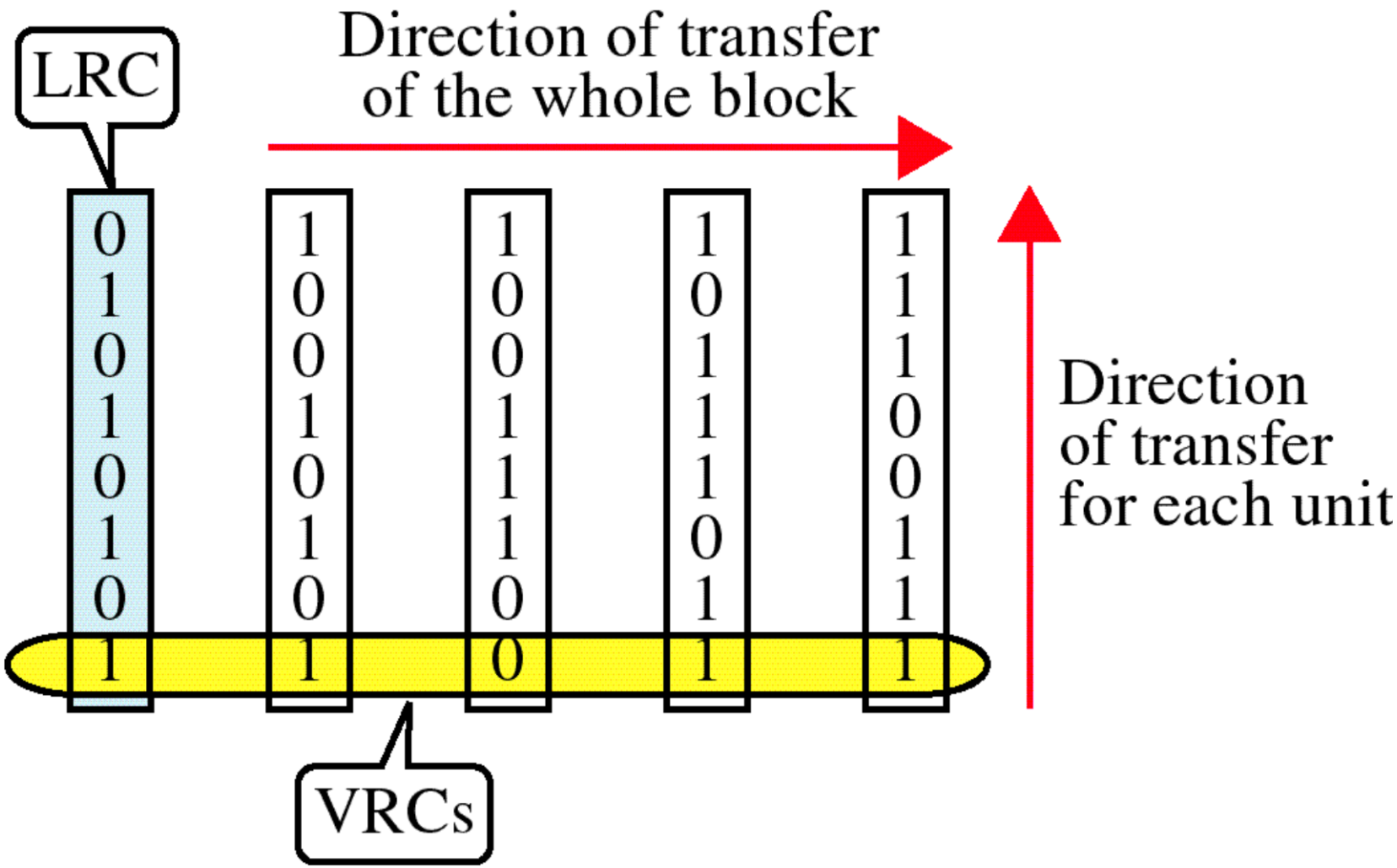
Longitudinal Redundancy Check
LRC



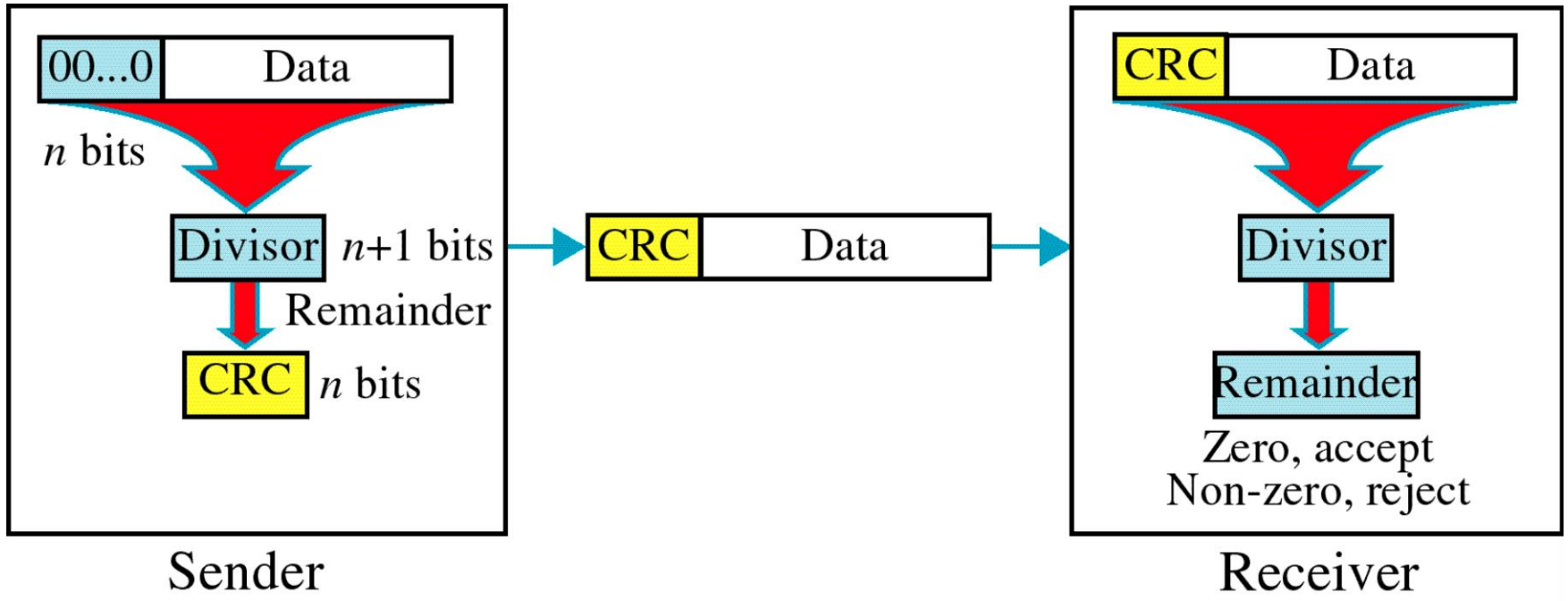
LCR increases the likelihood of detecting burst errors.

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


LRC
Direction of transfer
of the whole block
0 1 0 1 0 1 0 1
1 0 0 1 0 1 0 0
1 0 0 1 1 1 1 0
1 0 1 1 1 0 1 1
1 1 1 0 0 1 1 1
Direction
of transfer
for each unit
VRCs

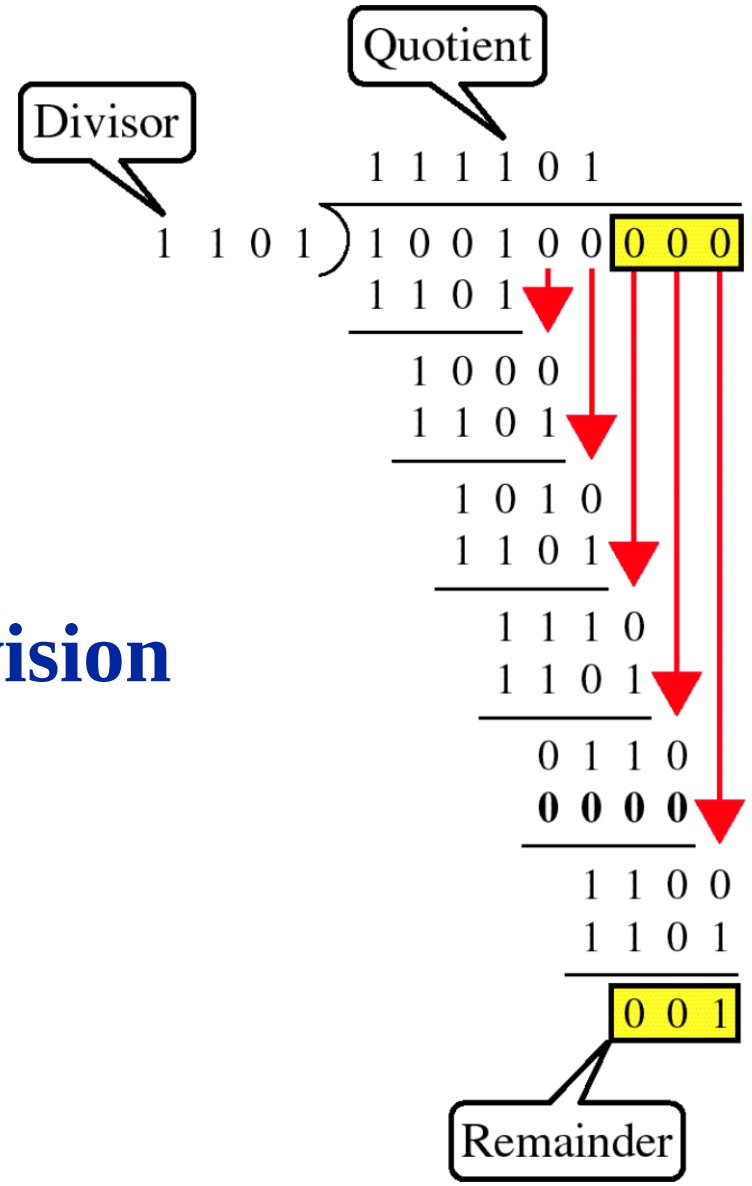
Cyclic Redundancy Check
CRC



Cyclic Redundancy Check

- Given a k-bit frame or message, the transmitter generates an n-bit sequence, known as a **frame check sequence (FCS)**, so that the resulting frame, consisting of (k+n) bits, is exactly divisible by some predetermined number.
- The receiver then divides the incoming frame by the same number and, if there is no remainder, assumes that there was no error.

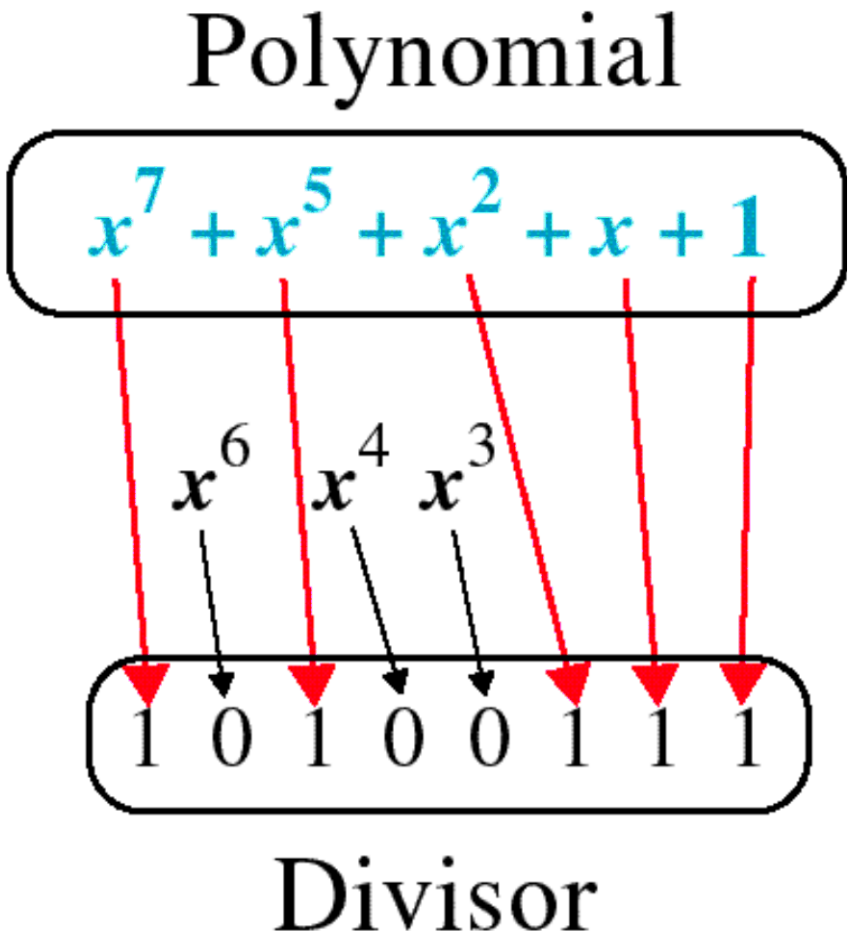
Binary Division



Polynomial

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

Polynomial and Divisor



Standard Polynomials

CRC-12

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

CRC-16

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

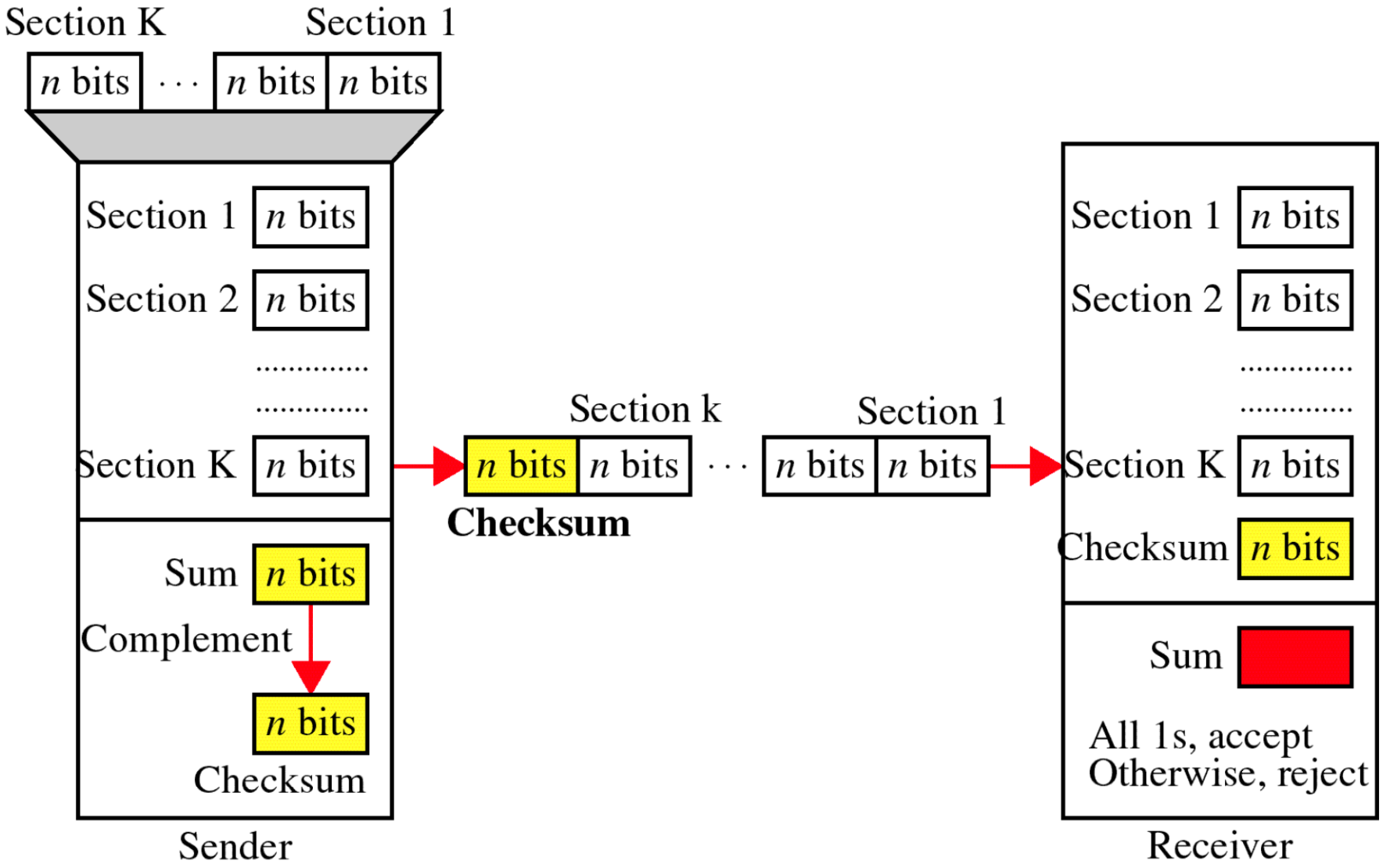
CRC-ITU

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

CRC-32

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

Checksum


Section K
Section 1
n bits
n bits
n bits
Section 1
n bits
Section 2
n bits
Section K
n bits
Sum
n bits
Complement
n bits
Checksum
Sender
Section k
Section 1
n bits
n bits
n bits
n bits
Checksum
Section 1
n bits
Section 2
n bits
Section K
n bits
Checksum
n bits
Sum
All 1s, accept
Otherwise, reject
Receiver

At the sender

The unit is divided into *k* sections, each of *n* bits.

All sections are added together using one's complement to get the sum.

The sum is complemented and becomes the checksum.

The checksum is sent with the data

At the receiver

The unit is divided into *k* sections, each of *n* bits.

All sections are added together using one's complement to get the sum.

The sum is complemented.

If the result is zero, the data are accepted: otherwise, they are rejected.

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.