1. **Explain the different type of transmission medias and its characteristics.**
2. **Prepare notes on MODEMS and its standards.**
3. **Write short note on X.21 digital interfaces.**

DATA LINK LAYER - FUNCTIONS

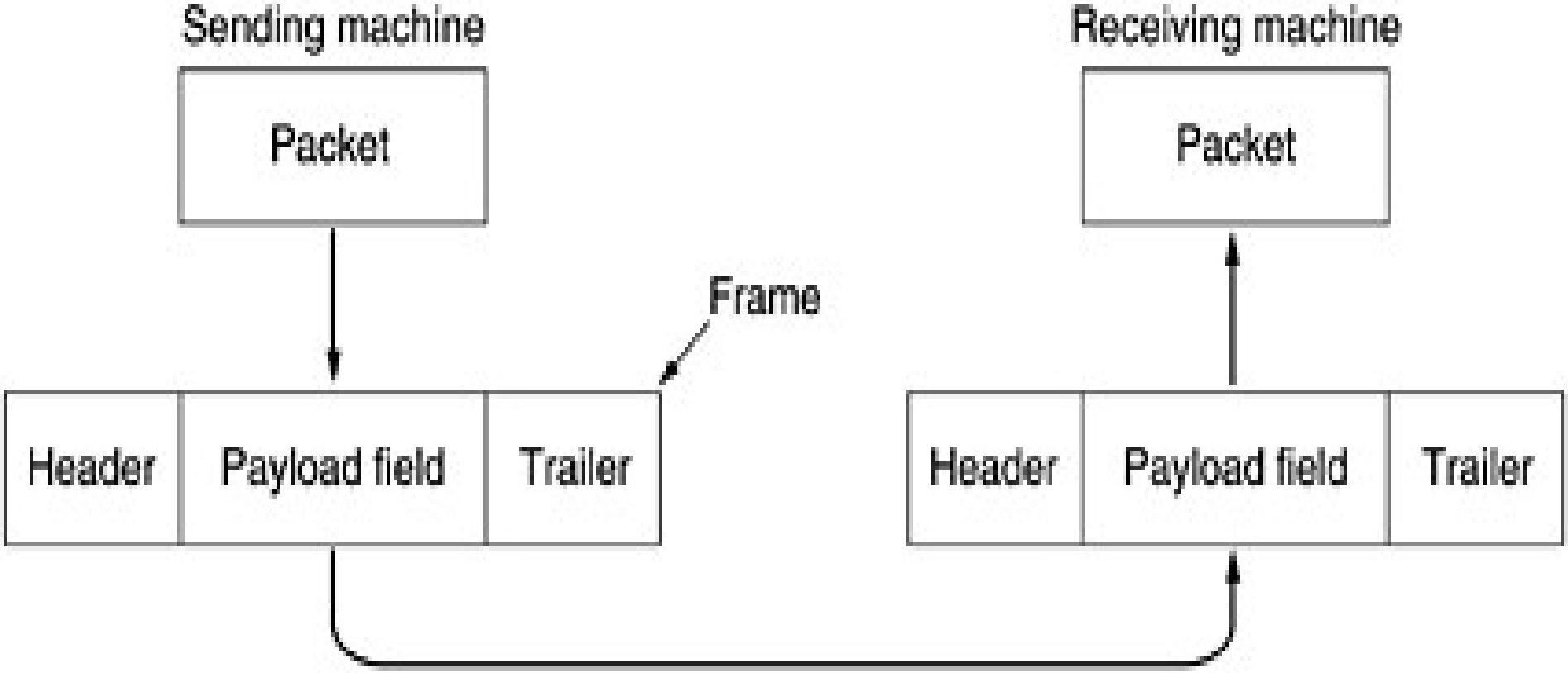
1. Providing a well-defined service interface to the network layer.
2. Provides services for the reliable interchange of data across a data link established by the physical layer.
3. Link layer protocols manage the establishment, maintenance, and release of data-link connections.

4. **Data-link protocols control the flow of data, dealing with transmission errors, and supervise data error recovery.**
5. **Regulating the flow of data so that slow receivers are not swamped by fast senders.**
6. **Recovery from abnormal conditions.**

To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission.

Each frame contains a frame header, a payload field for holding the packet, and a frame trailer.

RELATIONSHIP BETWEEN PACKETS AND FRAMES

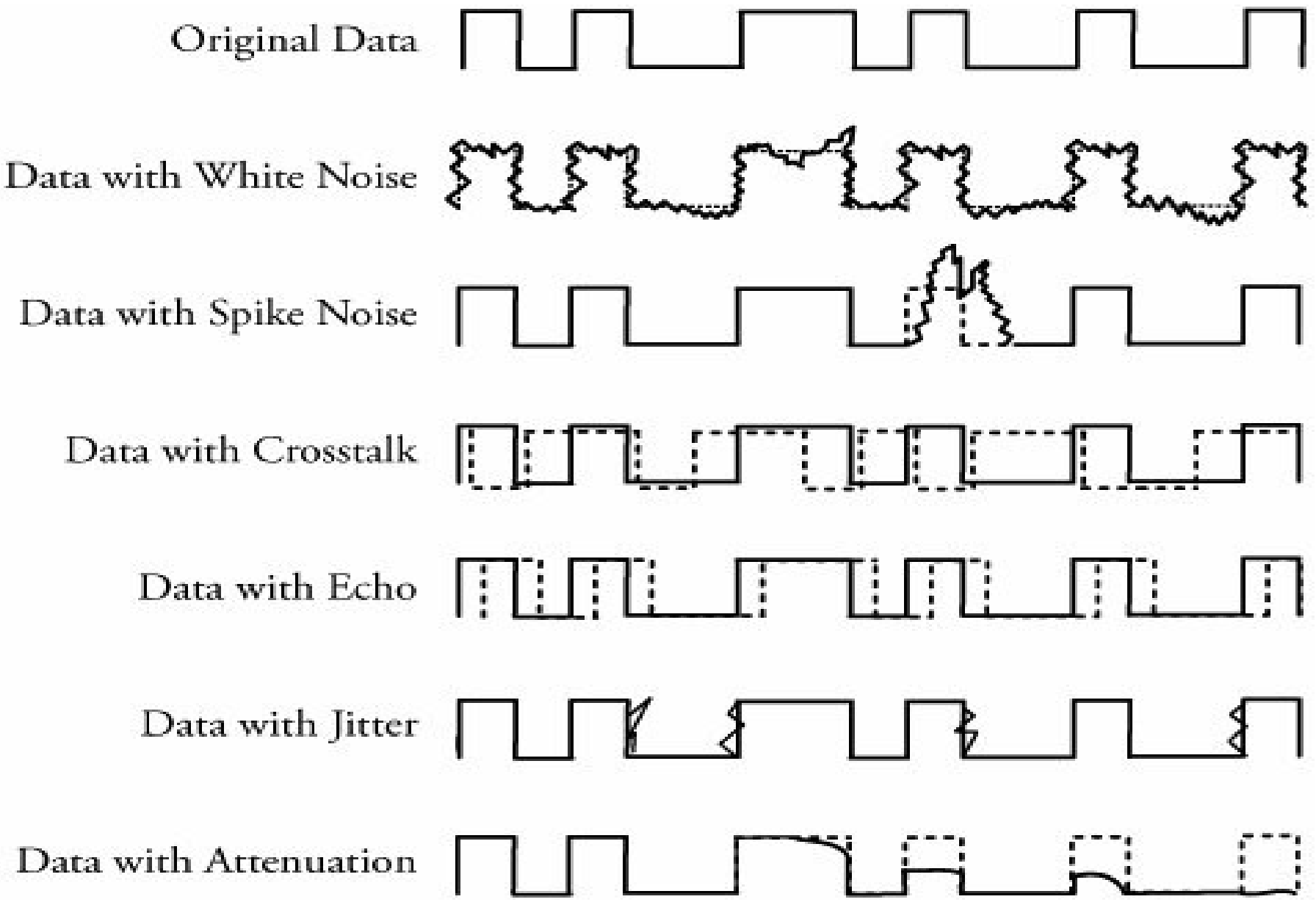


ERROR DETECTION AND CORRECTION

Error detection is applied mostly in the data-link layer but is also performed in other layers.

When a packet arrives at the destination, the destination may extract an error-checking code from the transport header and perform error detection.

LINK LEVEL


Original Data
Data with White Noise
Data with Spike Noise
Data with Crosstalk
Data with Echo
Data with Jitter
Data with Attenuation

TYPES OF ERRORS

The interference in transmission can change the shape of the signal; leads to an error in data transmission.

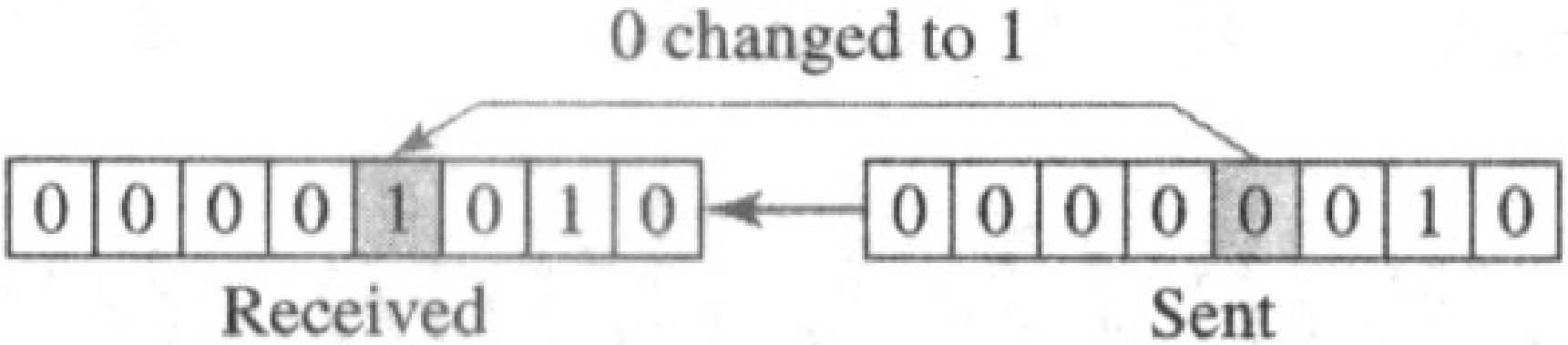
Basic types of errors are

1. Single-Bit Error
2. Burst Error

1. SINGLE-BIT ERROR

only one bit of a given data unit (such as a byte, character, data unit, or packet) is changed from 1 to 0 or from 0 to I.

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



SINGLE-BIT ERROR

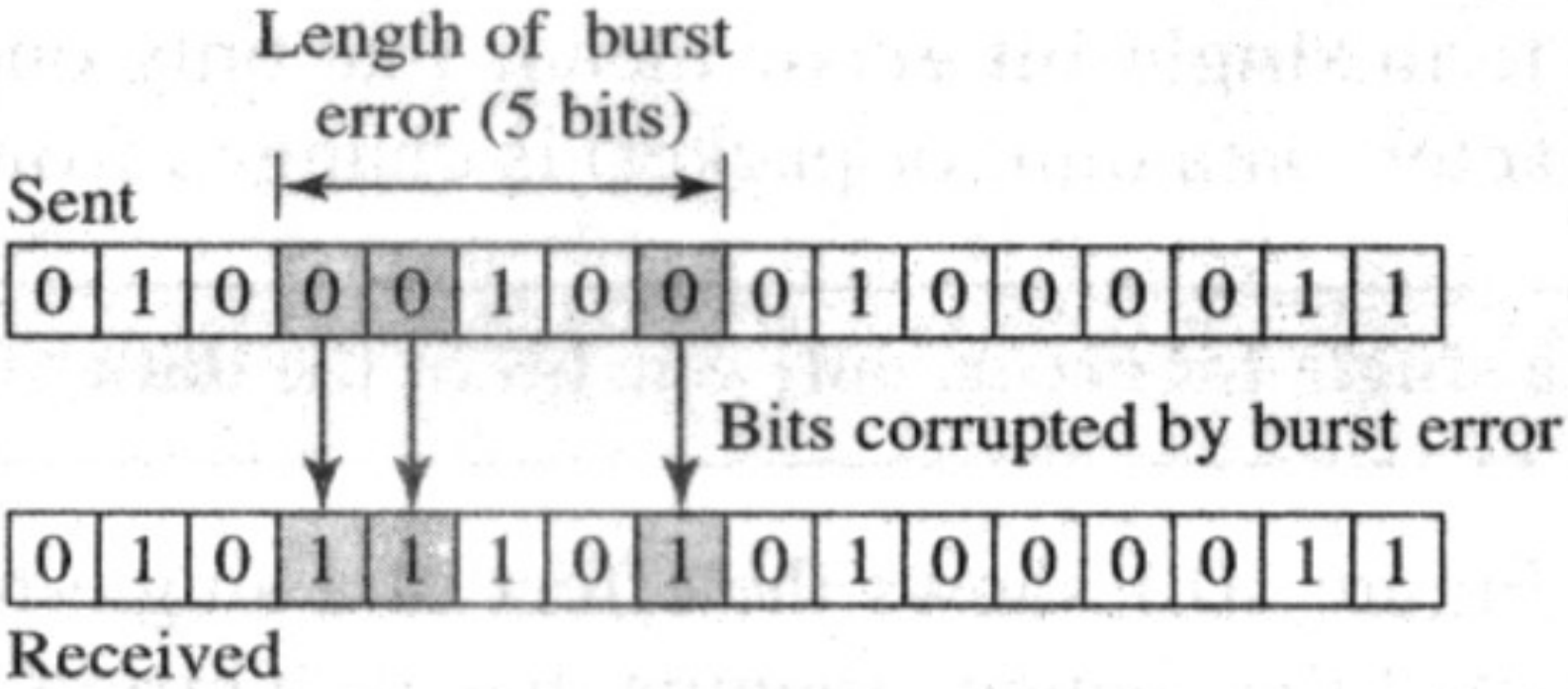
A single-bit error can happen if we are sending data using parallel transmission.

For example, if eight wires are used to send all 8 bits of 1 byte at the same time and one of the wires is noisy, one bit can be corrupted in each byte.

Single-bit errors are less in serial data transmission.

2. BURST ERROR

The term burst error means that 2 or more bits in the data unit have



Burst error is most likely to occur in a serial transmission.

The duration of noise is normally longer than the duration of one bit, which means that when noise affects data, it affects a set of bits.

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

ERROR DETECTION

Most networking equipment at the data-link layer inserts some type of error-detection code.

When a frame arrives at the next hop in the transmission sequence, the receiving hop extracts the error-detection code and applies it to the frame.

When an error is detected, the message is normally discarded.

The most common approaches to error detection are based on Redundancy

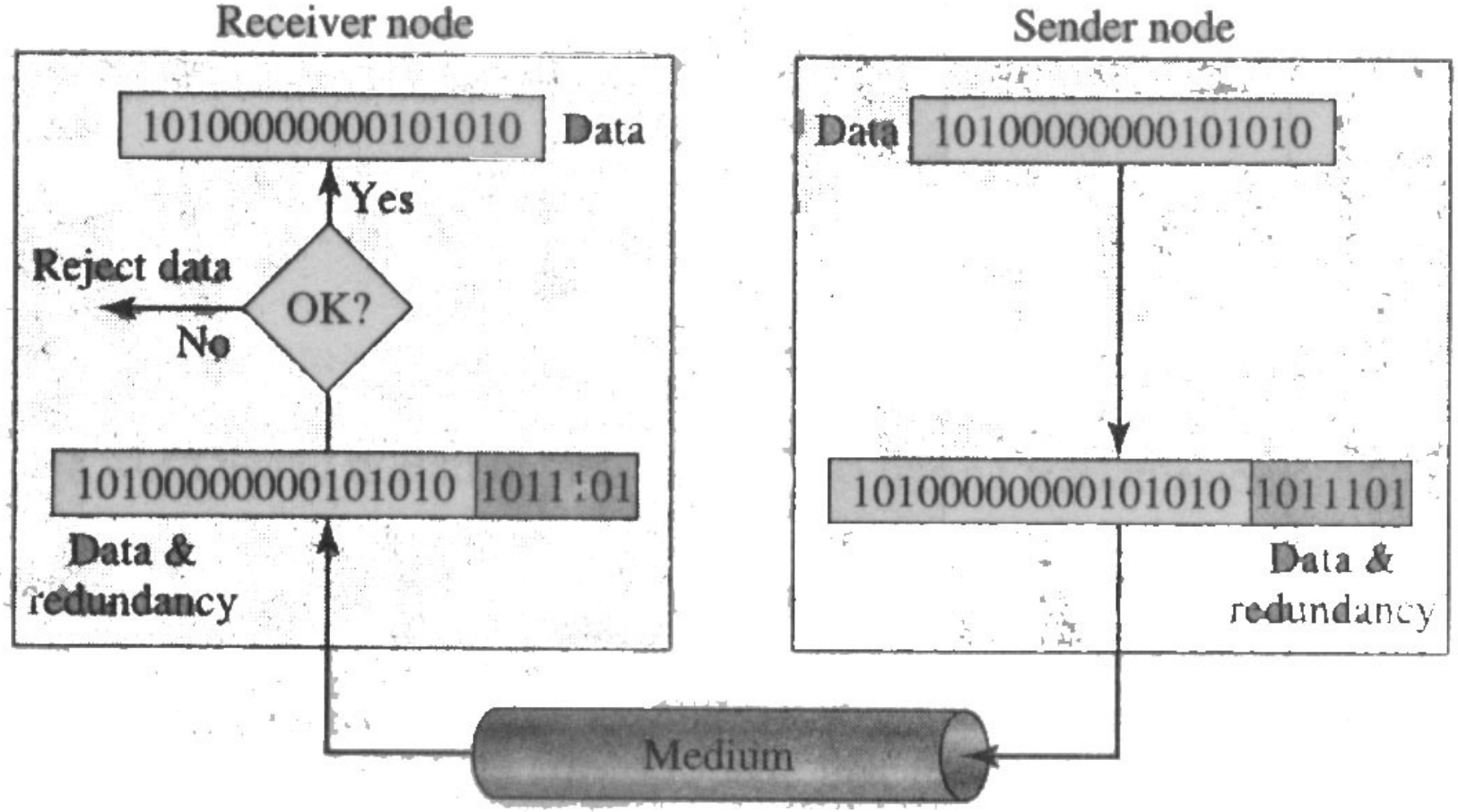
send every data unit twice.

The receiving device would then able to do a bit for bit comparison between the two versions of the data.

Any discrepancy would indicate an error, and an appropriate correction mechanism could be set in place.

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

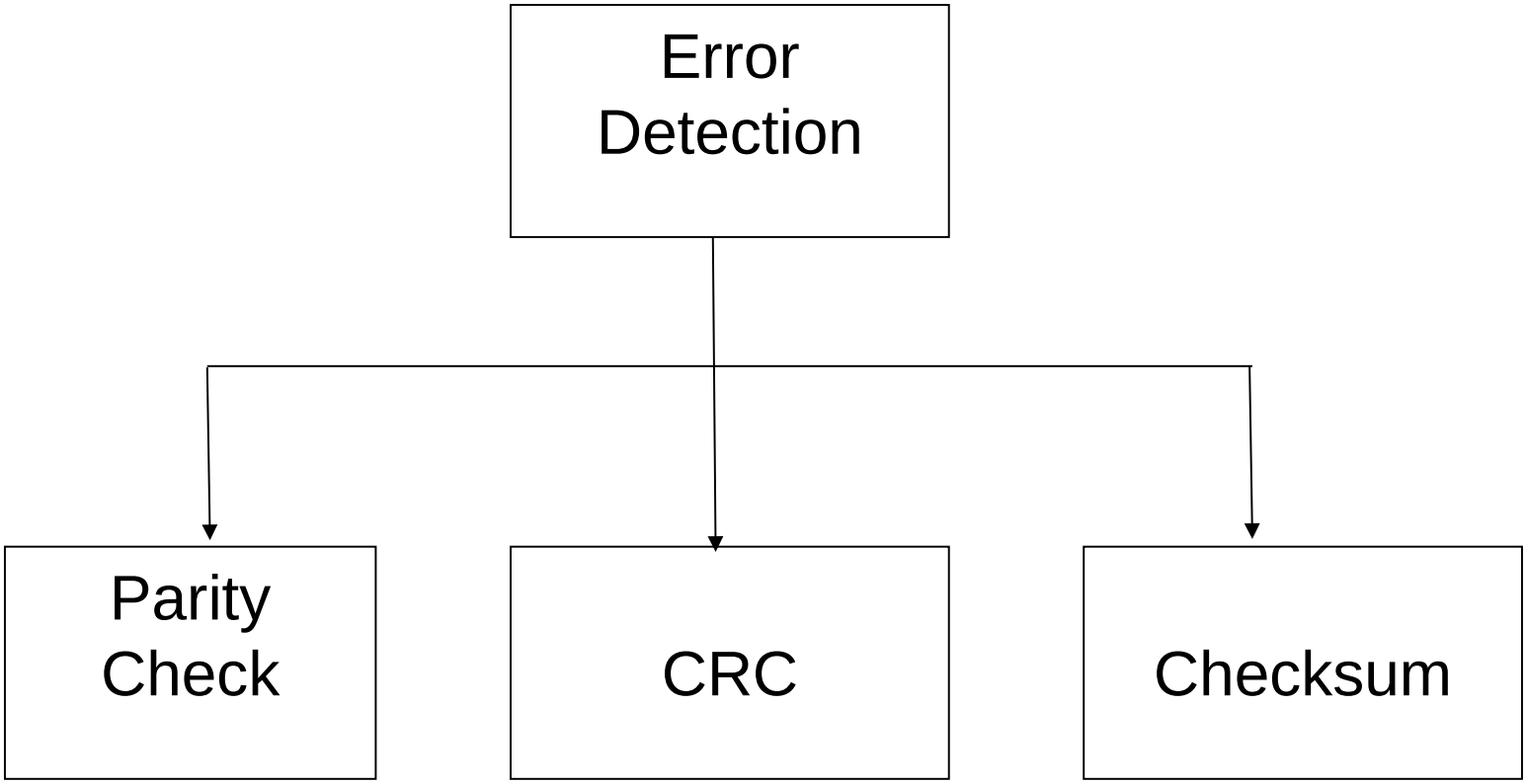
This technique is called redundancy because the extra bits are redundant to the information; they are discarded as soon as the accuracy of the transmission has been determined.


Receiver node
1010000000101010
Data
Yes
Reject data
OK?
No
1010000000101010
1011101
Data &
redundancy
Sender node
Data
1010000000101010
1010000000101010
1011101
Data &
redundancy
Medium

ERROR DETECTION

Three types of redundancy checks are common in data communications:

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


Error Detection
Parity Check
CRC
Checksum

1. PARITY CHECK

The most common and least expensive mechanism for error detection is the parity check.

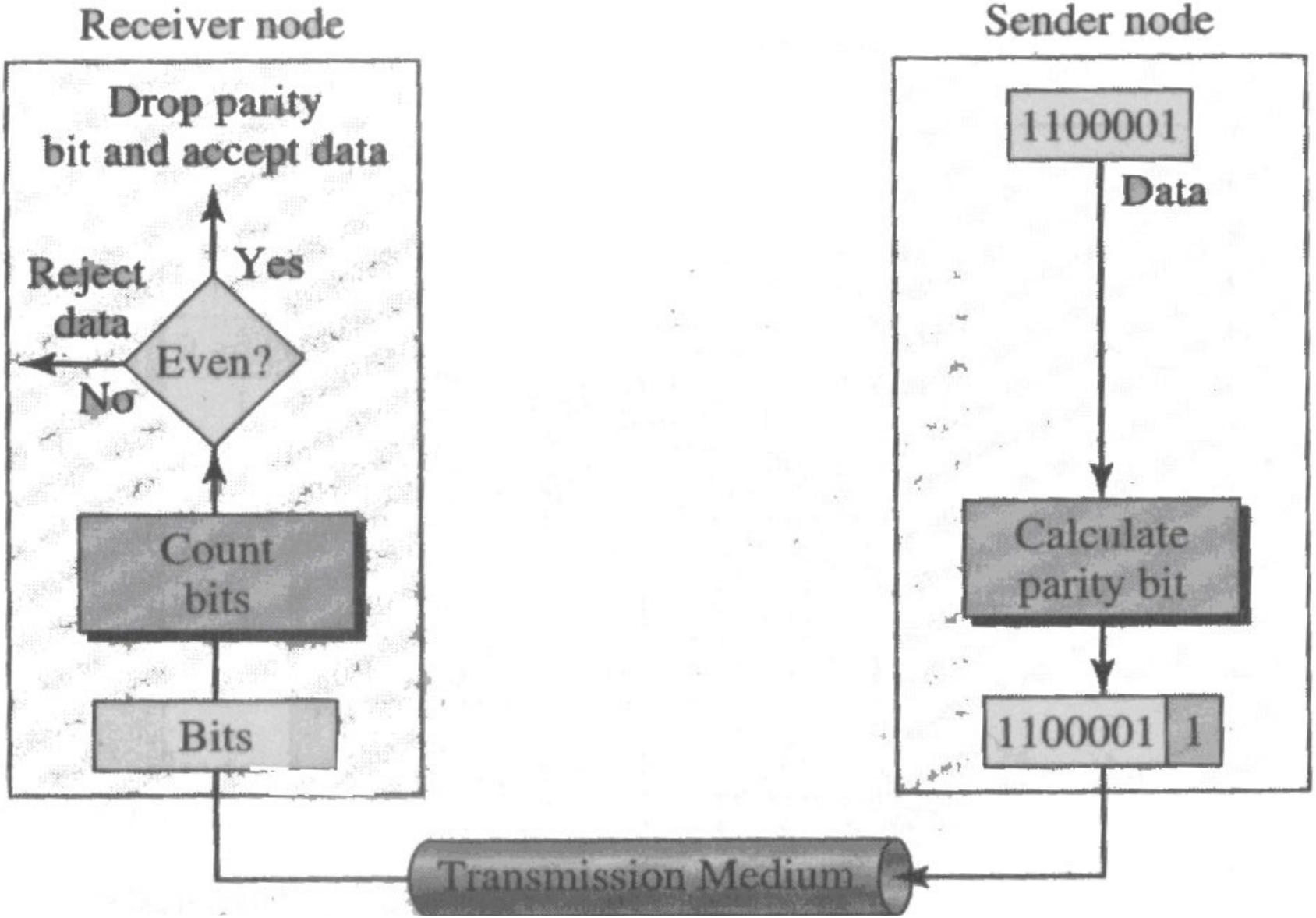
Parity checking can be simple or two dimensional.

A. SIMPLE PARITY CHECK

In this technique, a redundant bit, called a parity bit, added to the unit so that the total number of 1's in the unit (including the parity bit) becomes even (or odd).

Suppose we want to transmit the binary data unit 1100001

EVEN-PARITY CHECKING FUNCTION



SIMPLE PARITY CHECK - PERFORMANCE

Simple parity check can detect all single bit error.

It can also detect burst errors as long as the total number of bits change to odd (1, 3, 5, etc.).

Suppose, that 2 bits of the data unit are changed….????

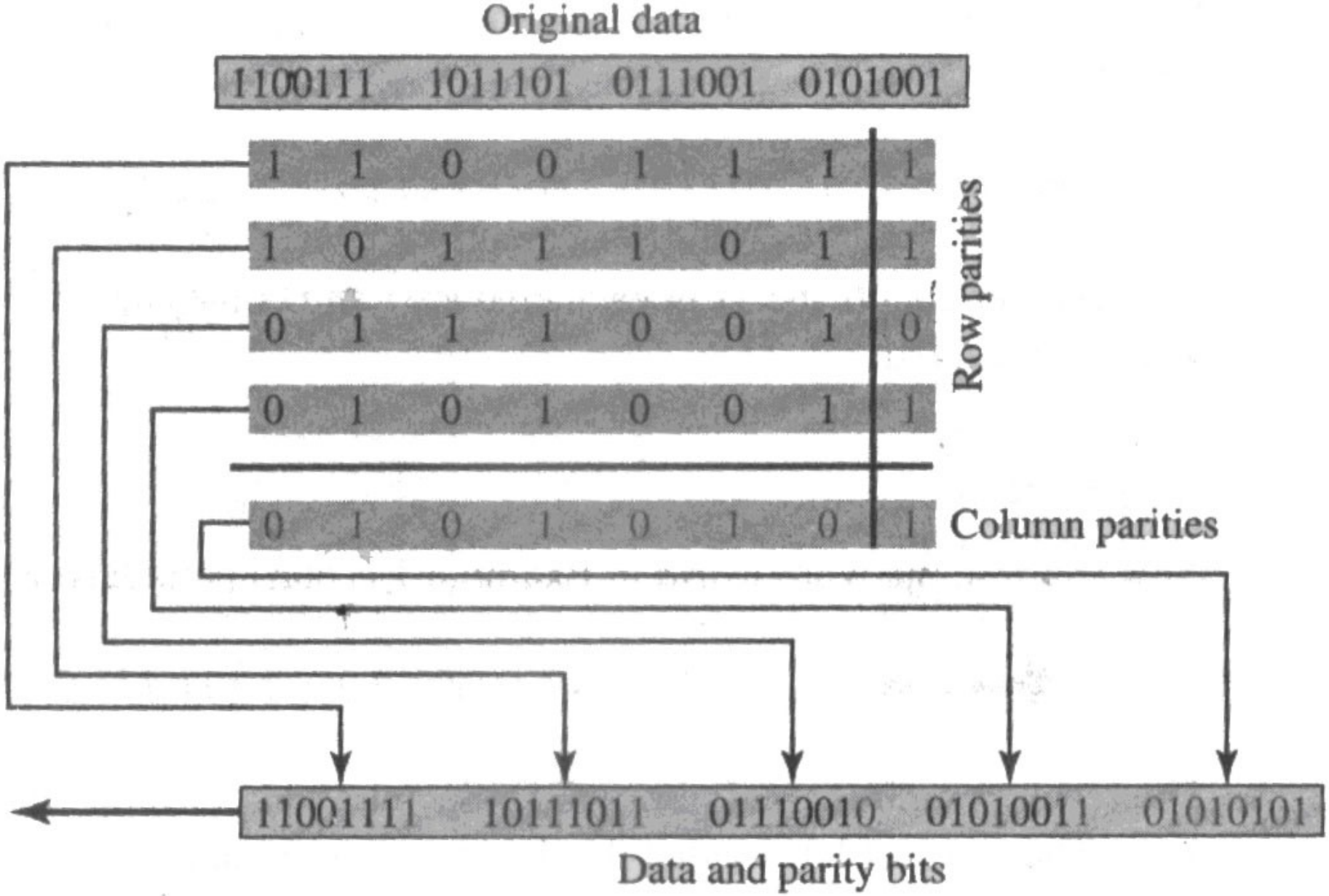
TWO DIMENSIONAL PARITY CHECK

In this method, block of bits is organized as a table (rows and columns).

First we calculate the parity bit for each data unit.

Then we organize them into a table format.

TWO DIMENSIONAL



PERFORMANCE

Two-dimensional parity check increases the chance of detecting burst errors.

A redundancy of *n* bits can easily detect a burst error of n bits.

If 2 bits in one data unit are damaged and two bits in exactly the same positions in another data unit are also damaged, the checker will not detect an error.

2. CYCLIC REDUNDANCY CHECK (CRC)

The most powerful of the redundancy checking techniques is the cyclic redundancy check (CRC).

Unlike the parity check which is based on addition, CRC is based on binary division.

CRC…

In CRC, a sequence of redundant bits, called the CRC or the CRC remainder, is appended to the end of a data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.

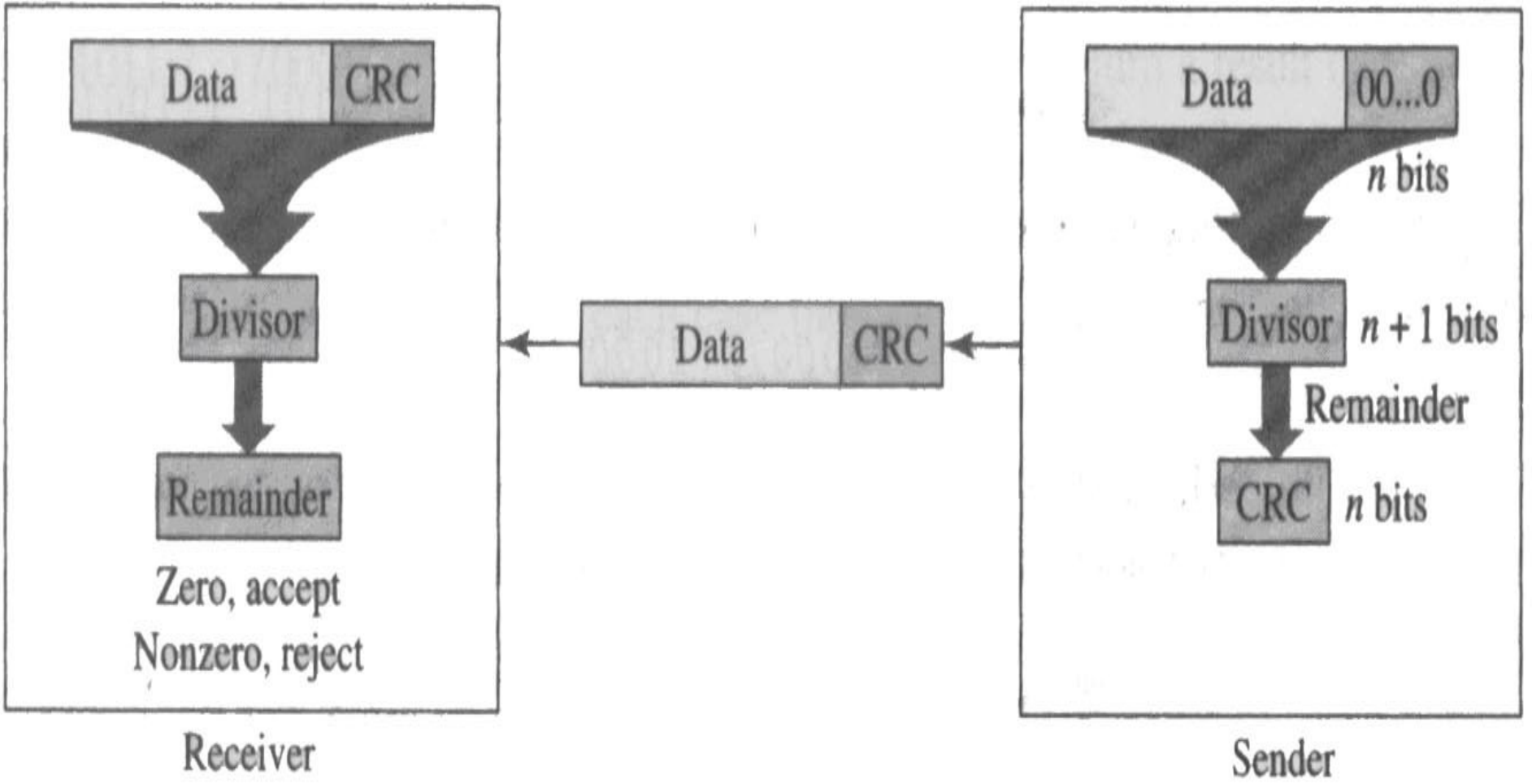
CRC…

At its 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 intact and is therefore accepted.

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

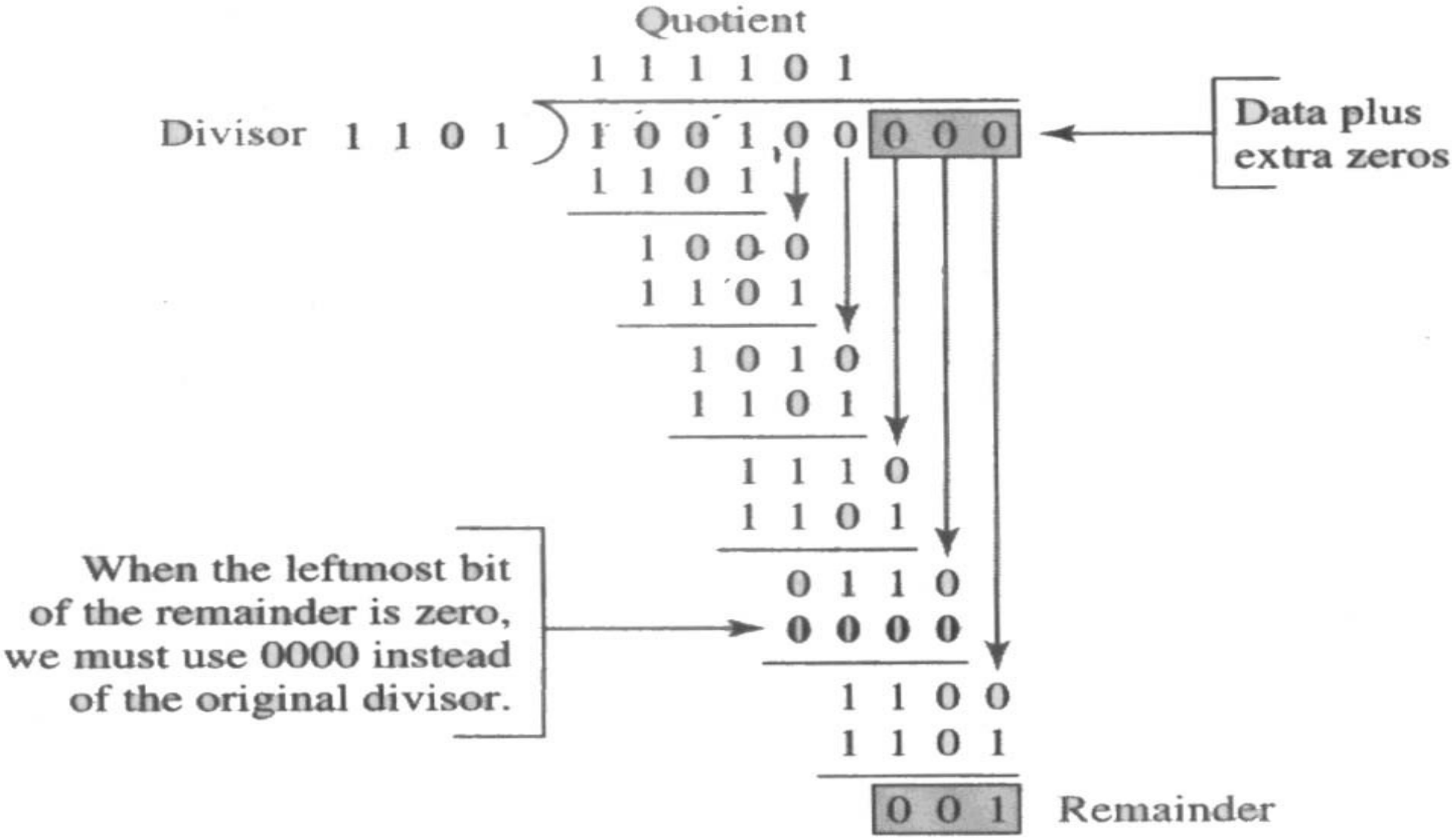
CRC…

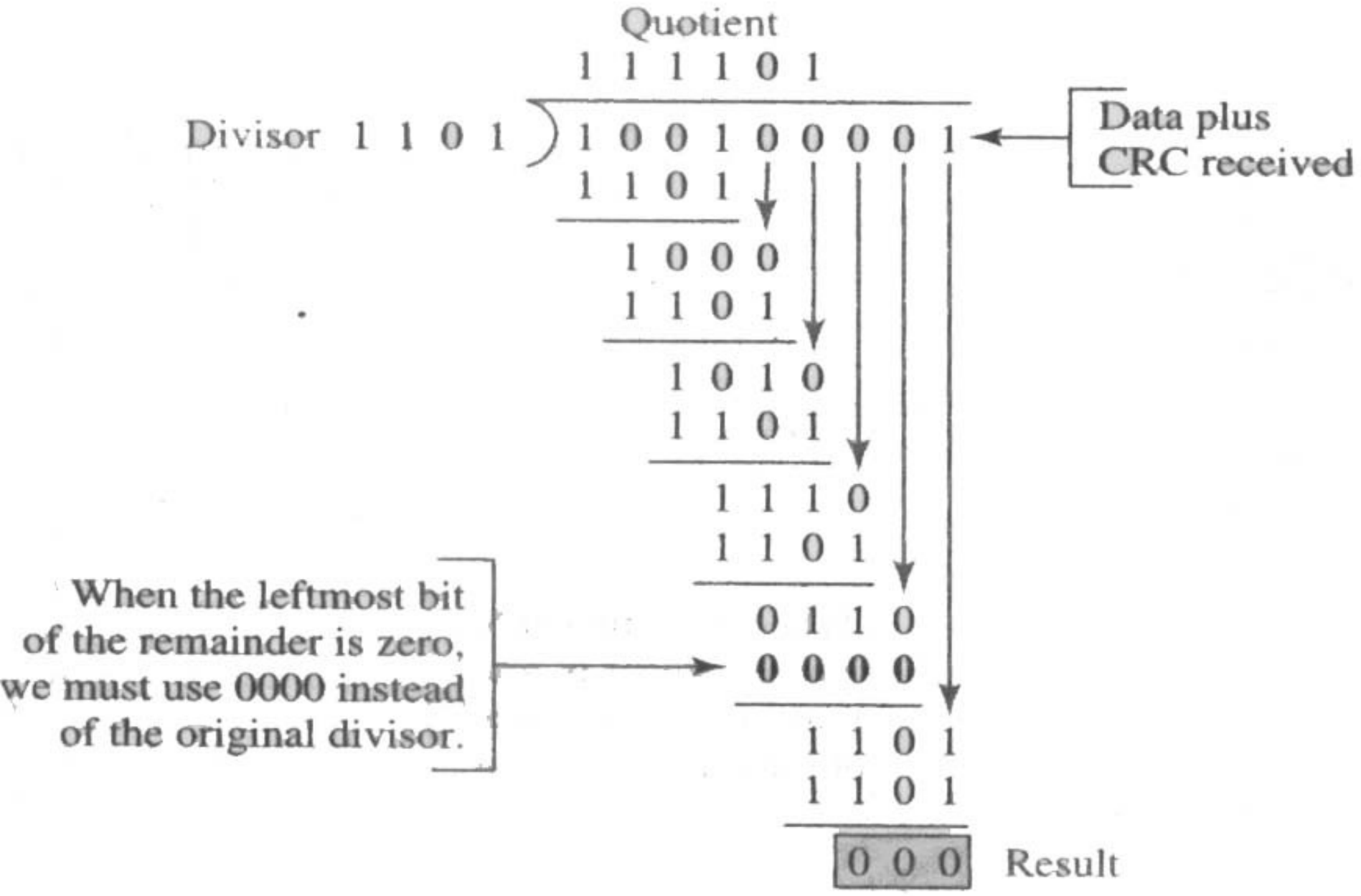

Data
CRC
Divisor
Remainder
Zero, accept
Nonzero, reject
Receiver
Data
CRC
Data
00...0
n bits
Divisor
n + 1 bits
Remainder
CRC
n bits
Sender

THREE BASIC STEPS IN CRC.

1. **A string of n 0s is appended to the data unit. The number n is 1 less than the number of bits in the predetermined divisor, which is n + 1 bit.**
2. **The newly elongated data unit is divided by the divisor, using a process called binary division. The remainder resulting from this division is the CRC.**
3. **The CRC of n bits derived in step 2 replaces the appended 0s at the end of the data unit. Note that the CRC may consist of all 0s.**

THE CRC


Quotient
1 1 1 1 0 1
Divisor 1 1 0 1
1 0 0 1 0 0 0 0 0
1 1 0 1
1 0 0 0
1 1 0 1
1 0 1 0
1 1 0 1
1 1 1 0
1 1 0 1
0 1 1 0
0 0 0 0
1 1 0 0
1 1 0 1
0 0 1
Remainder
Data plus extra zeros
When the leftmost bit of the remainder is zero, we must use 0000 instead of the original divisor.

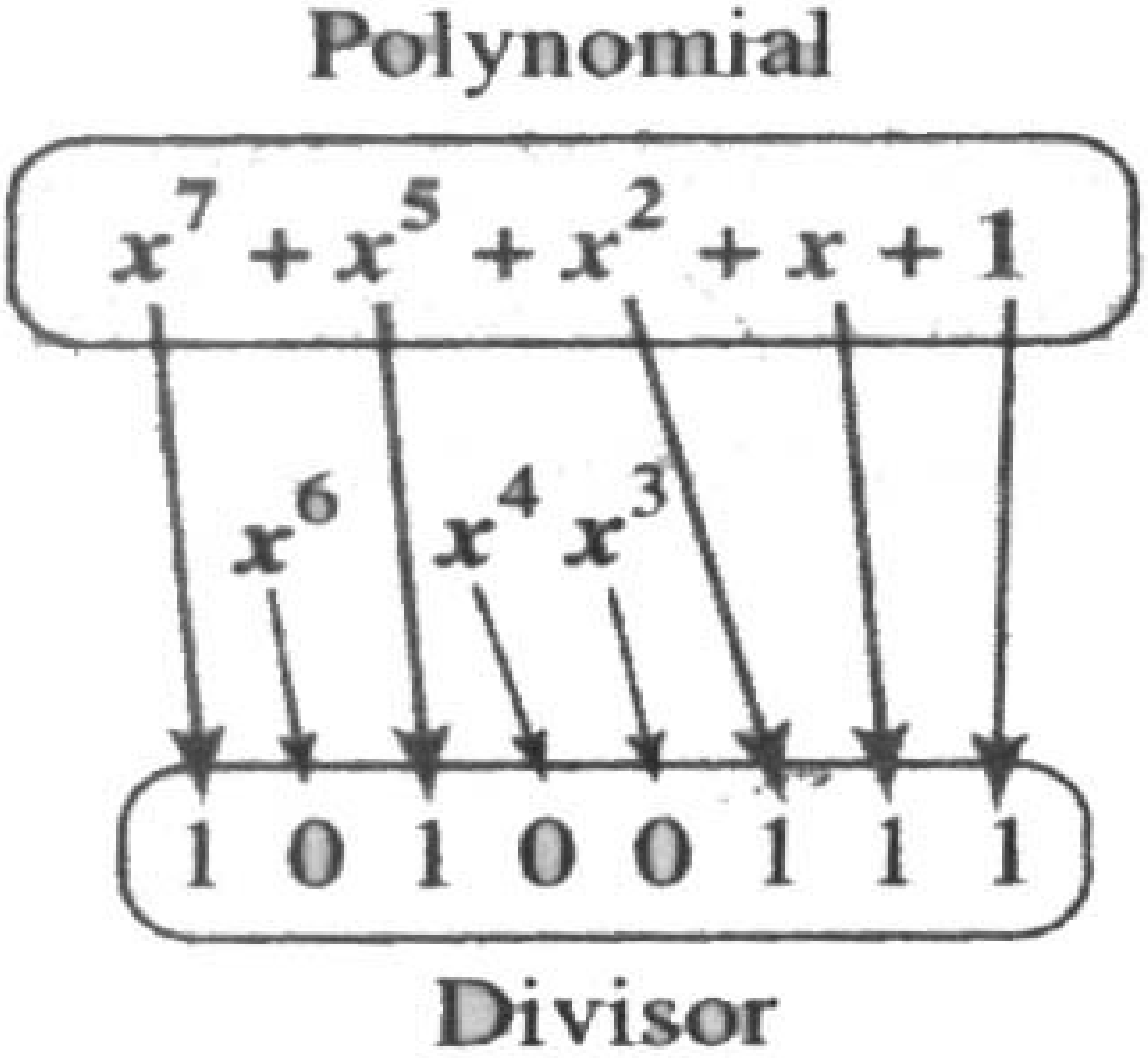

Quotient
1 1 1 1 0 1
Divisor 1 1 0 1
1 0 0 1 0 0 0 0 1
Data plus CRC received
1 1 0 1
1 0 0 0
1 1 0 1
1 0 1 0
1 1 0 1
1 1 1 0
1 1 0 1
0 1 1 0
0 0 0 0
1 1 0 1
1 1 0 1
0 0 0
Result
When the leftmost bit of the remainder is zero, we must use 0000 instead of the original divisor.

POLYNOMIALS

The divisor in the CRC generator is most often represented *not as a string* of 1s and 0s, but as an algebraic polynomial, like $x^7+x^5+x^2+x+1$

The polynomial format is useful for two reasons: It is short, and it can be used to prove the concept mathematically.

cor



on

POLYNOMIAL - PROPERTIES

- It should not be divisible by x.
- It should be divisible by x + 1.

The first condition guarantees that all burst errors of a length equal to the degree of the polynomial are detected.

The second condition guarantees that all burst errors affecting an odd number of bits are detected.

PERFORMANCE

CRC can detect all burst errors that affect an odd number of bits.

CRC can detect all burst errors of length less than or equal to the degree of the polynomial.

CRC can detect, with a very high probability, burst errors of length greater than the degree of the polynomial.

3. CHECKSUM

Like the parity checks and CRC, the checksum is based on the concept of redundancy.

CHECKSUM GENERATOR - SENDER

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

All these *k* sections are added using ones complement arithmetic in such a way that the tota is also *n* bits long.

The sum is complemented and appended to the end of the original data unit as redundancy bits, called the checksum field

The extended data unit is transmitted across the network. So, if the sum of the data segment is T the checksum will be -T.

EXAMPLE:

The numbers are added using ones complement arithmetic

	10101001
	00111001
Sum	11100010
Checksum	00011101

The pattern sent is: 10101001 00111001 00011101

Data Checksum

CHECKSUM CHECKER– RECEIVER SIDE

The unit is divided into k sections, each of n bits as the checksum generation.

All sections are added using ones complement to get the sum.

The sum is complemented.

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