Error Detection

&

Correction

Introduction

Data can be corrupted during transmission due to…

accidents, sudden increase in electricity and voltage / decrease in signal power over distance

For reliable communication, errors must be detected and corrected

What is an Error ?

- *Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference*

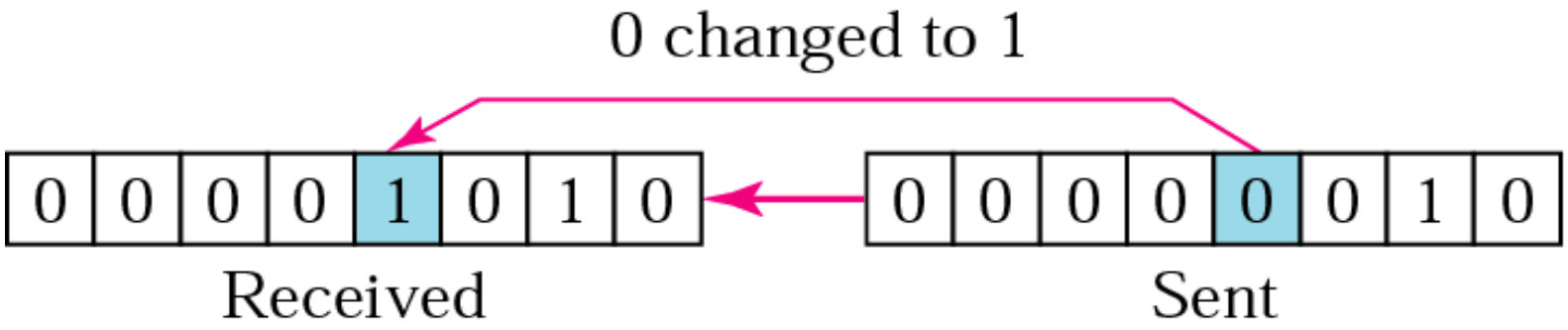
- *The interference can change the shape of the signal, thus the bit value either from "1" to "0" or from "0" to "1"*

Two Types of Errors

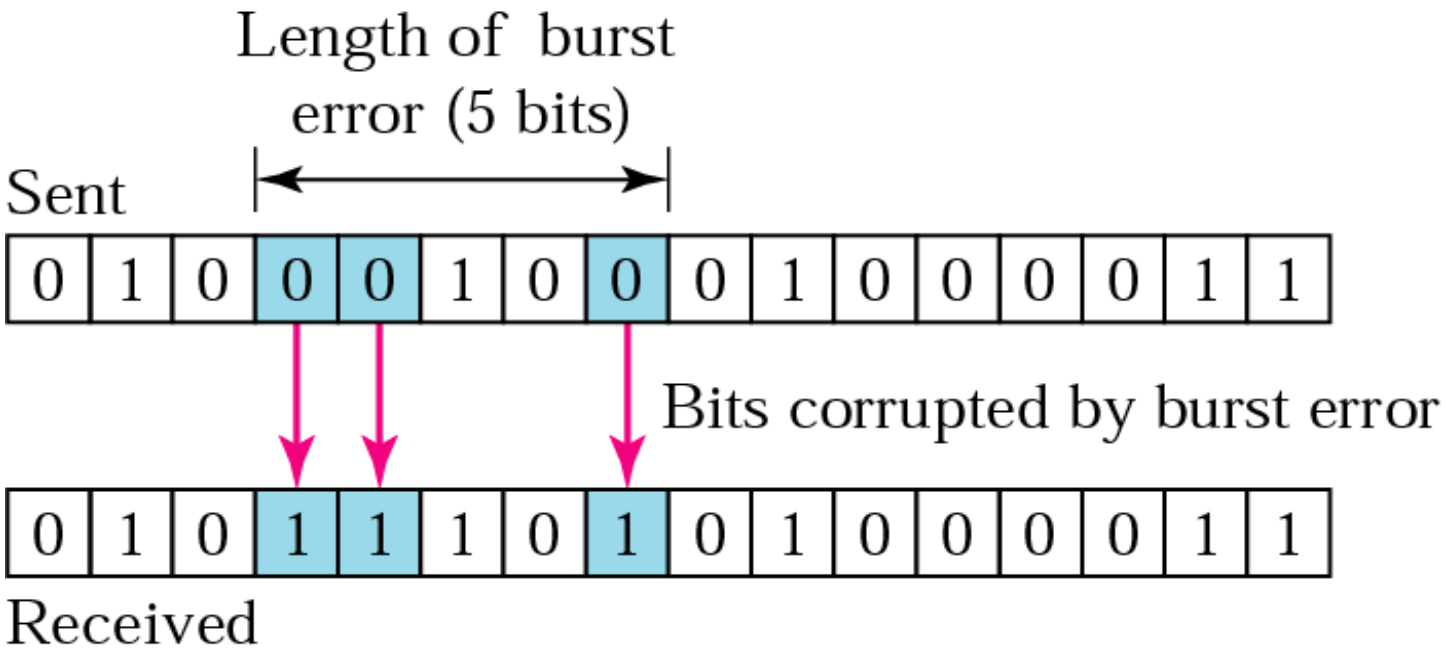
- ***Single-Bit Errors :*** *only one bit in the data unit has changed*

- ***Burst Errors of length 'n' :*** *2 or more bits in the data unit have changed ('n' is the distance between the FIRST and LAST errors in the data block)*

Single-Bit Errors



Burst Errors



Error Detection

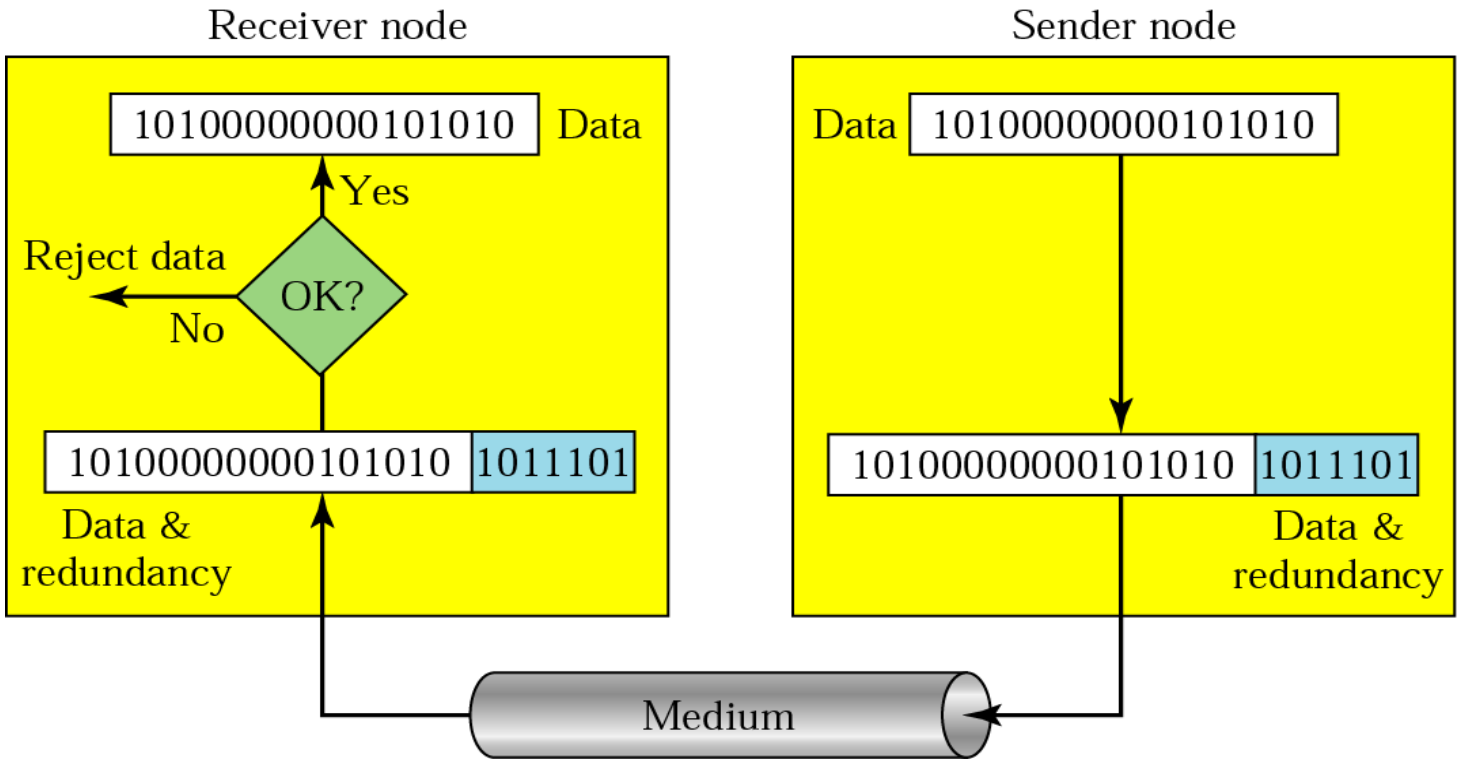
Error Detection-General

- Sender transmits every data unit twice
- Receiver performs bit-by-bit comparison between that two versions of data
- Any mismatch would indicate an error, which needs error correction
- Advantage: very accurate
- Disadvantage: time consuming

Error Detection- Redundanc

- Called as “redundancy” because the extra bits are redundant to the information
- Redundant information will be discarded as soon as the accuracy of the information has been determined

Error Detection- Redundanc



Error Detection-Redundanc

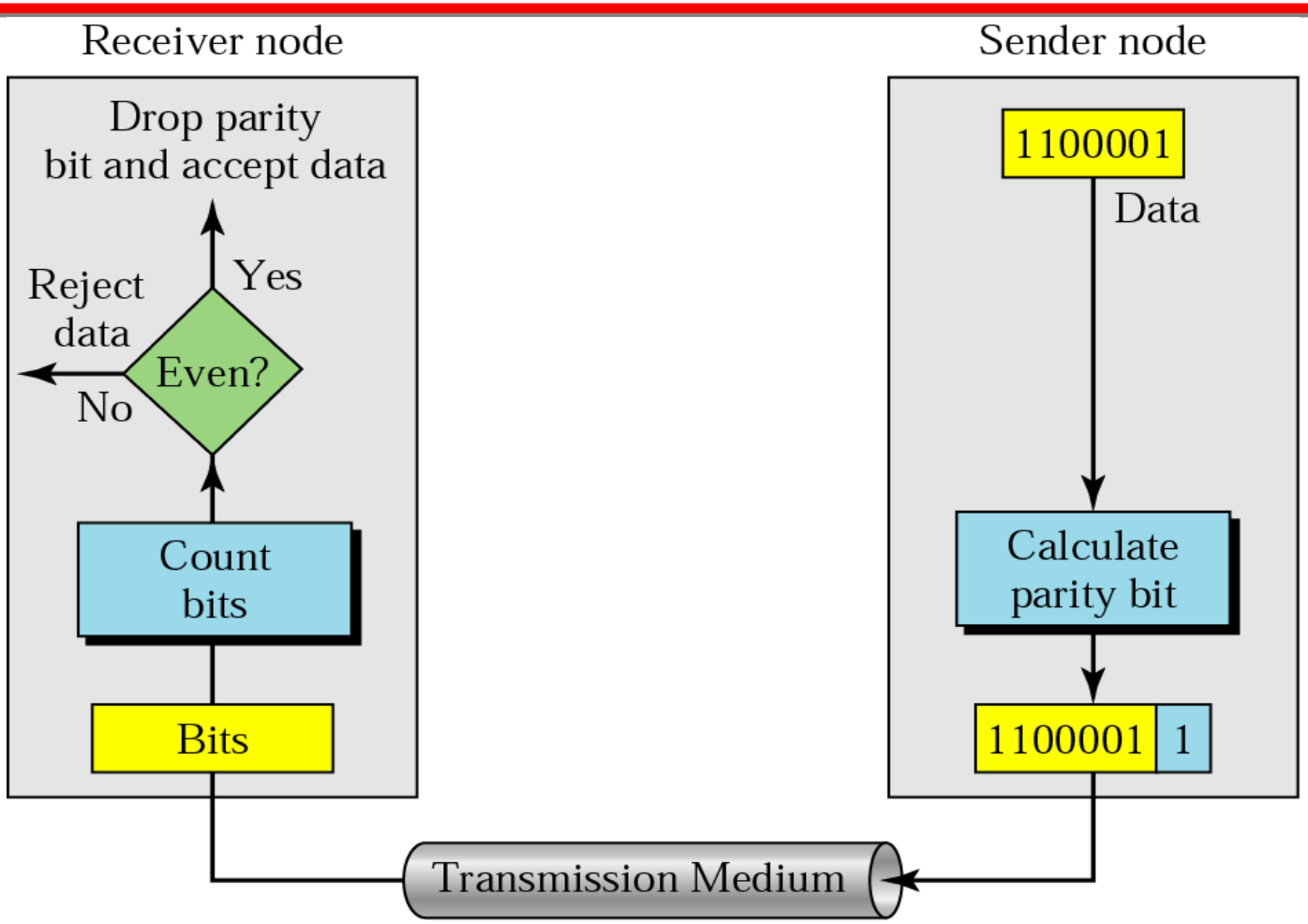
Types of Redundancy Checks

- Parity Check
  - Simple Parity Check
  - Two Dimensional Parity Check / Longitudinal Redundancy Check (LRC)
- Cyclic Redundancy Check (CRC)
- Check Sum

Error Detection-Simple Parity Chec

- A redundant bit called "Parity Bit" is added to every data unit
- Even Parity : total number of 1's in the data unit becomes even
- Odd Parity : total number of 1's in the data unit becomes odd

Error Detection- Simple Parity Che

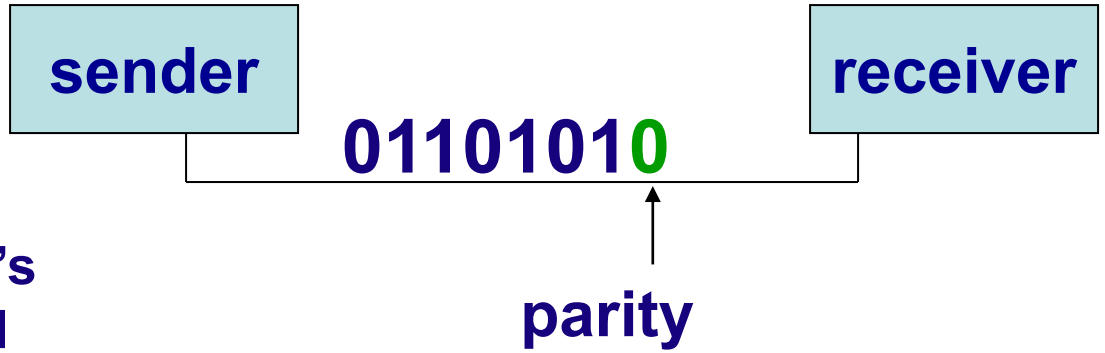


Even Parity Check

Example of Using Parity Bits

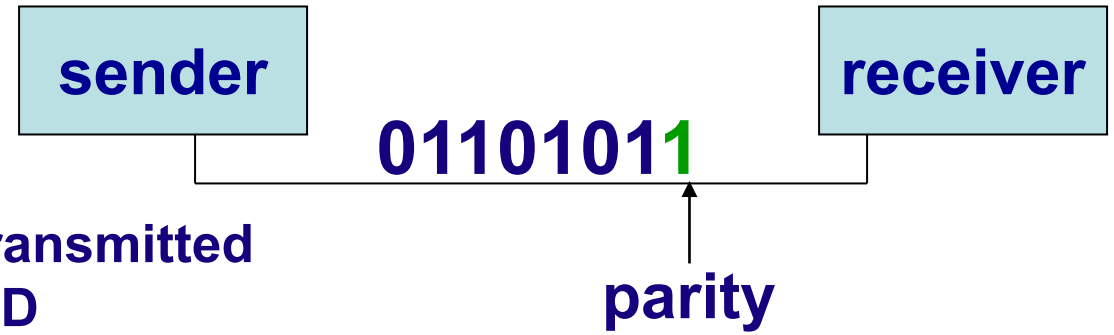
Data to be sent: Letter V in 7-bit ASCII: 0110101

EVEN parity



number of all transmitted 1's remains EVEN

ODD parity



number of all transmitted 1's remains ODD

Simple Parity Check-An Example

Data:

w	o	r	l	d
1110111	1101111	1110010	1101100	1100100

Sent As:

11101110 11011110 11100100 11011000 11001001

Corrupted:

11111110 11011110 11101100 11011000 11001001

Parity Check - Performance

- Can detect all single-bit errors
- Can also detect burst errors if the total number of bits changed is odd (1,3,5,..)
- Cannot detect errors where the total number of bits changed is even
- Detects about 50% of errors

Error Detection- 2D/LRC

- Adds an additional character (instead of a bit)
- A block of bits is organized in a table
- The Parity Bit for each data unit is calculated
- Then Parity Bit for each column is calculated
- Parity Bits are attached to the data unit

Error Detection- LRC

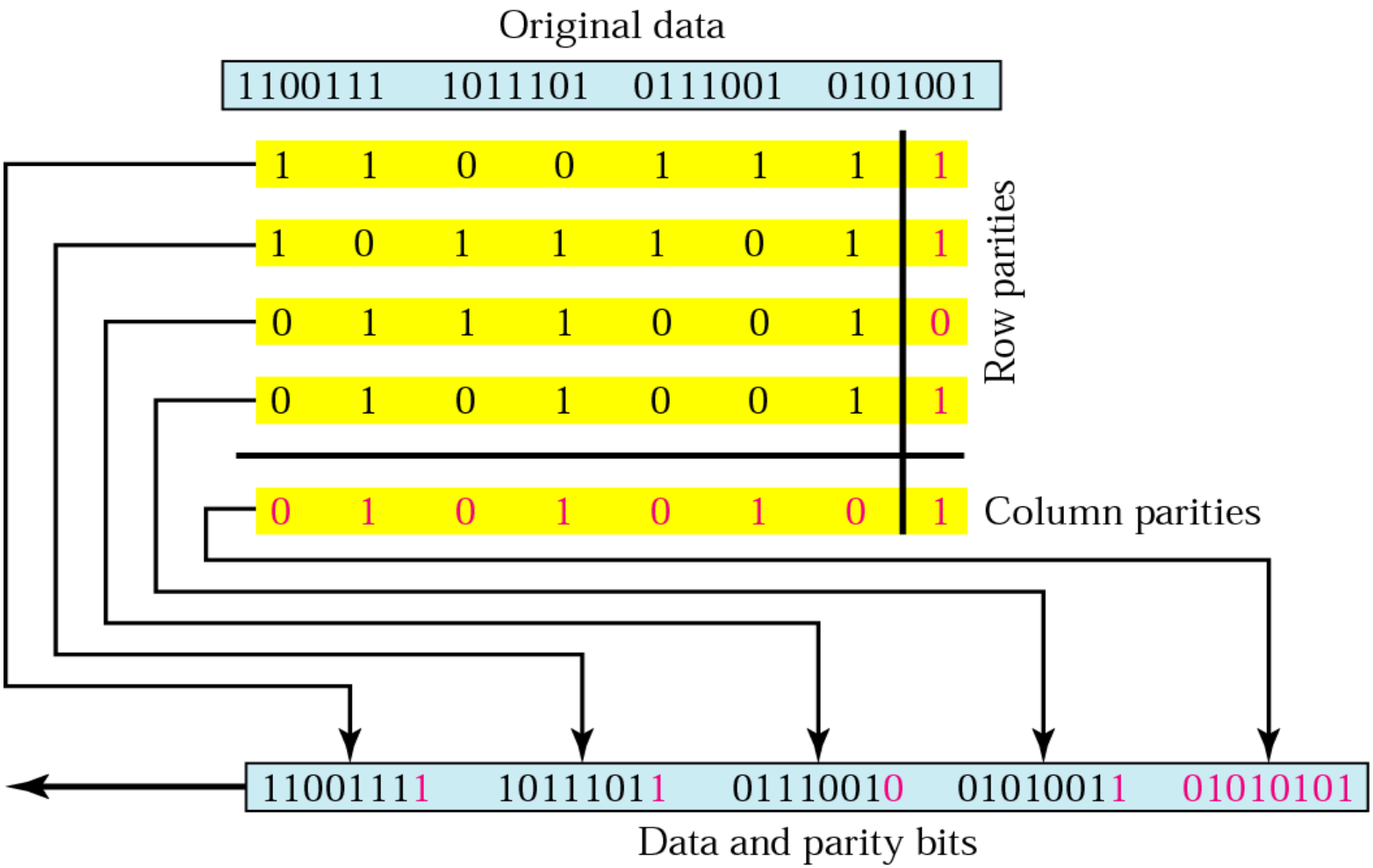
Example:

Send the message "DATA" using ODD parity and LRC

Letter	ASCII	Parity bit
D	1 0 0 0 1 0 0	1
A	1 0 0 0 0 0 1	1
T	1 0 1 0 1 0 0	0
A	1 0 0 0 0 0 1	1
BCC	1 1 0 1 1 1 1	1

Note that the BCC's parity bit is also determined by parity

Error Detection- LRC



LRC - Performance

- Detects all burst errors up to length n (number of columns)
- If two bits in one data unit are damaged and two bits in exactly same positions in another data unit are also damaged, the checker will not detect an error

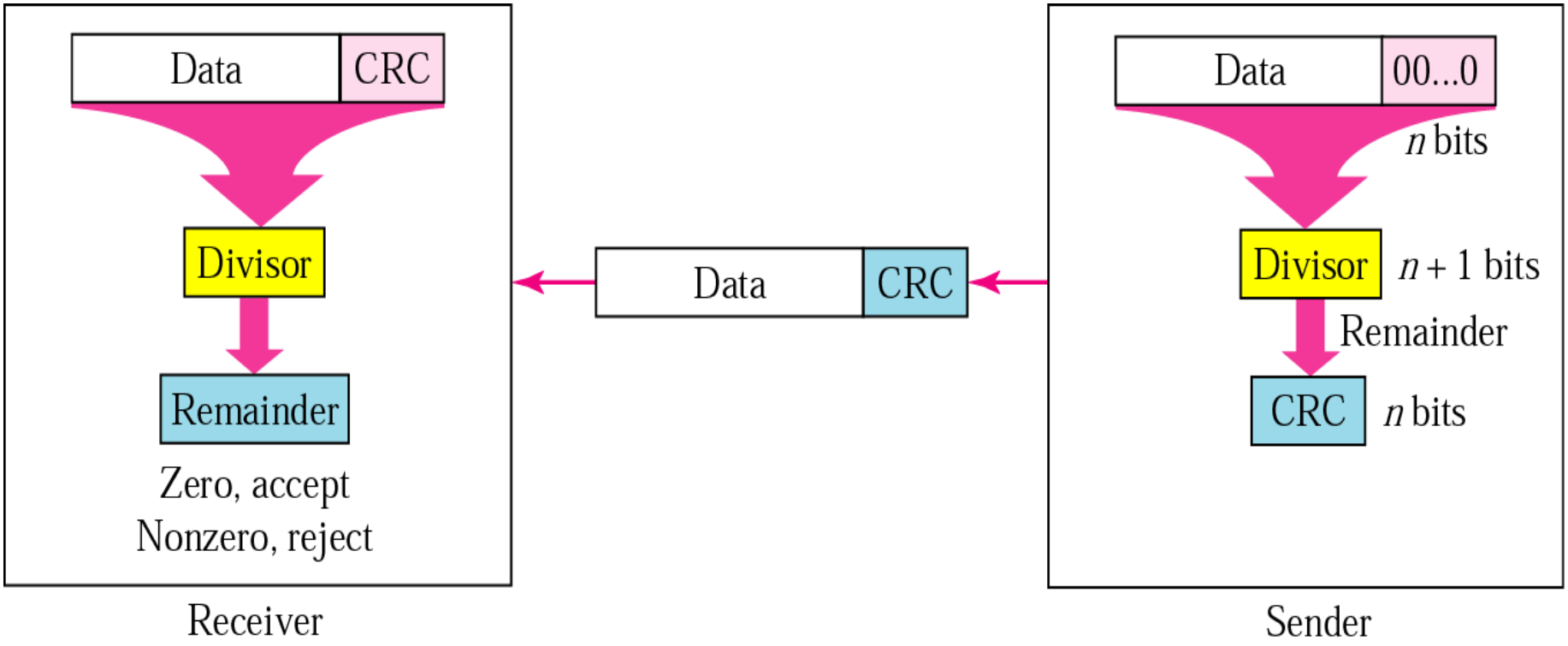
Error Detection- CRC

- Powerful error detection scheme
- Rather than addition, binary division is used
- A sequence of redundant bits, called “CRC” or “CRC remainder” is appended to the data unit, so that the resulting data unit becomes divisible by a predetermined binary number

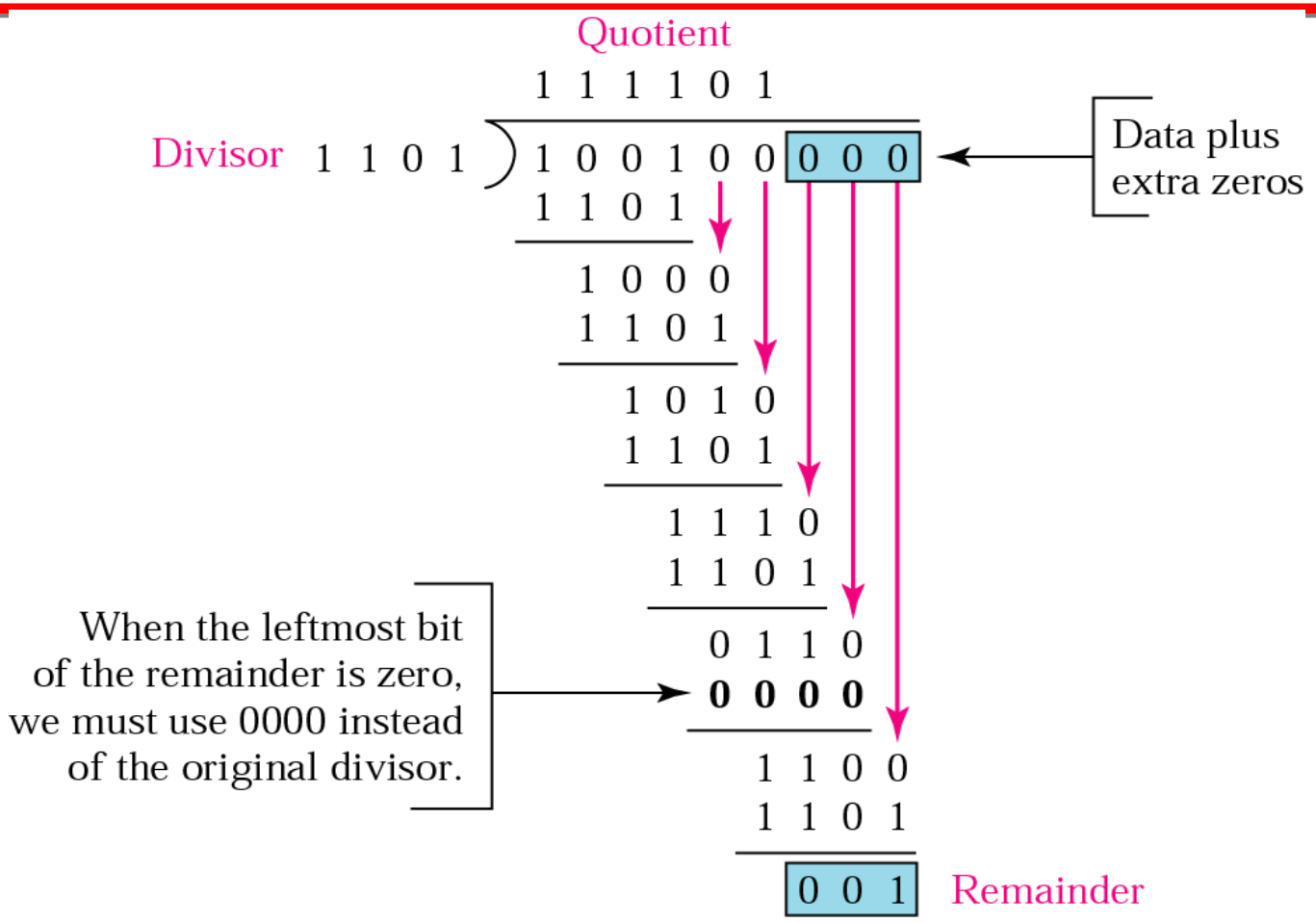
Error Detection- CRC

- At the receiver side, the incoming data unit is divided by the same predetermined number.
- If there is no remainder, the data unit is accepted
- If there is a remainder, the receiver indicates that the data unit has been damaged during transmission

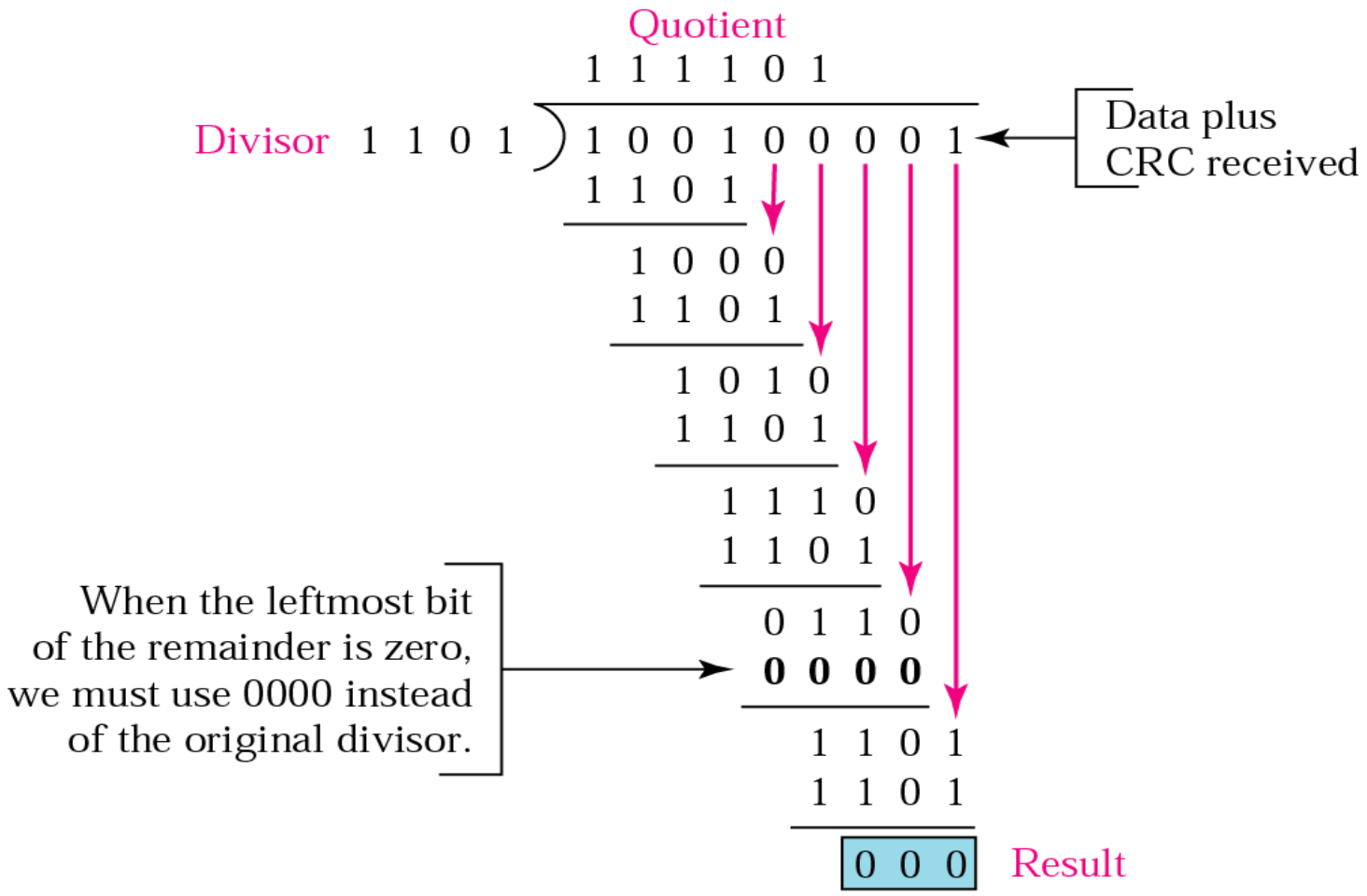
Error Detection- CRC



Error Detection- CRC Generatc



Error Detection- CRC Checker

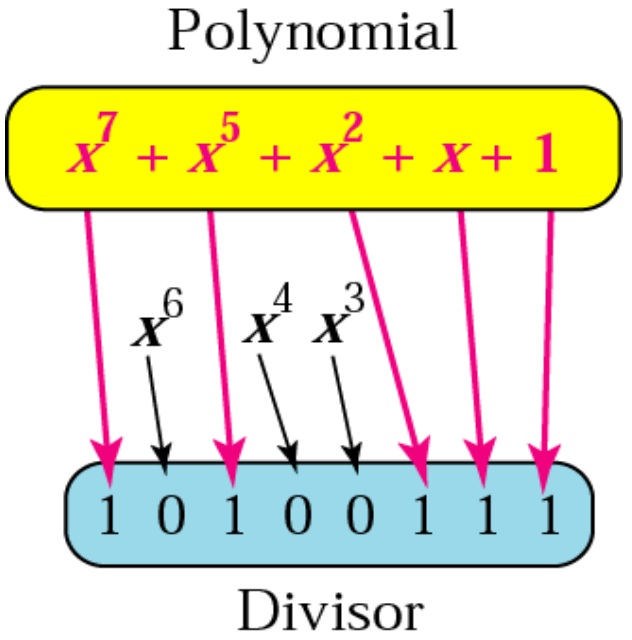


Error Detection- CRC Polynomi

- The divisor in the CRC generator is most often represented as an algebraic *Polynomial.*
- Reasons:
  - It is short

Error Detection- CRC Polynomi

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



Error Detection- CRC Polynomi

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
ITU-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
ITU-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

CRC-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

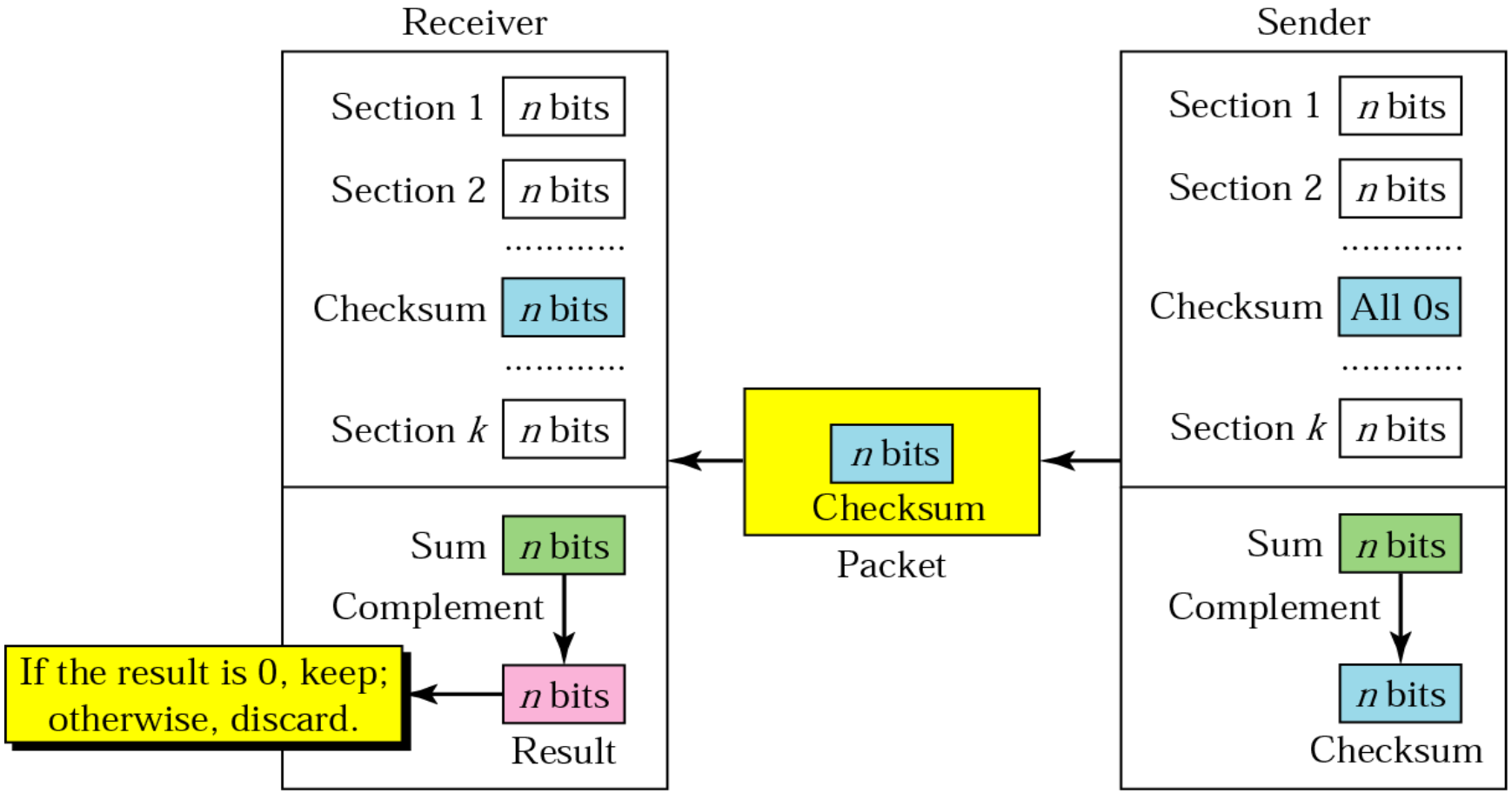
Error Detection- Check Sum

- ***The sender follows these steps:***
  - The data unit is divided into “k” sections, each of “n” bits
  - All sections are added using one’s complement to get the sum
  - The sum is complemented and becomes the checksum.
  - The checksum is appended and sent with the data.

Error Detection- Check Sum

- ***The receiver follows these steps:***
  - The unit is divided into “k” sections, each of “n” bits
  - All sections are added using one’s complement to get the sum.
  - The sum is complemented.
  - If the result is zero, the data are accepted; otherwise, rejected

Error Detection- Check Sum



Check Sum-An Example

Data:

10101001 00111001

Computing Checksum:

10101001

00111001

Sum 11100010

CheckSum 00011101

Data Sent :

10101001 00111001 00011101

Check Sum-An Example

Receiver Side:

10101001

00111001

00011101

Sum 11111111

Complement 00000000

Error Correction

Error Correction Techniques

- Retransmission
- Forward Error Correction
- Burst Error Correction

Error Correction-Retransmissi

When an error is discovered, the receiver can ask the sender to retransmit the entire data unit

Error Correction-Forward Erro Correction

- **A receiver can use an error-correcting code, which automatically corrects certain errors**
- **Single-bit errors:**
  - Can be detected by the addition of parity bit which helps to find “error” or “no error” which is sufficient to detect errors
  - To correct errors the receiver can simply invert 0 to 1 or 1 to 0, but the problem is “locating” the position of error
  - To do so requires enough redundancy bits
  - Condition: $2^r >= m + r + 1$

Error Correction

Number of data bits m	Number of redundancy bits r	Total bits m + r
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

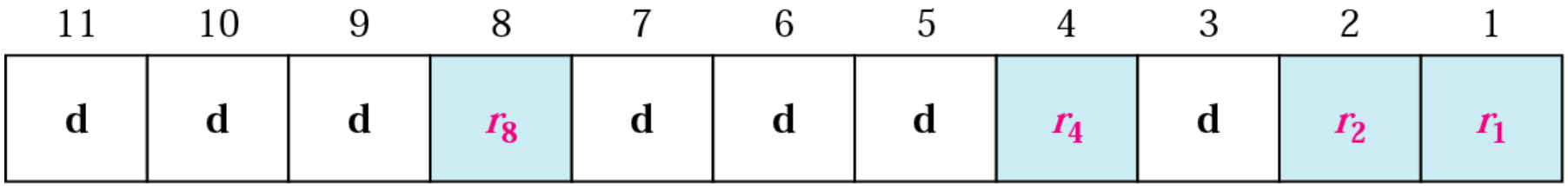
Relationship b/w data and redundancy bits

Error Correction-Hamming Co

- Hamming Code can be applied to data units of any length and uses the relationship between data and redundancy bits

- For example: a 7-bit ASCII code requires 4 redundancy bits that can be added to the end of the data unit or mixed with the original data bits, which are placed in positions 1, 2, 4 and 8 i.e x^0,x^1,x^2,x^3 and so on.

Error Correction-Hamming Co



Positions of Redundancy bits in Hamming Code

Error Correction-Hamming Co

- In the Hamming Code, each “r” bit for one combination of data bits as below:

r1: bits 1, 3, 5, 7, 9, 11
r2: bits 2, 3, 6, 7, 10, 11
r3: bits 4, 5, 6, 7
r4: bits 8, 9, 10, 11

Error Correction-Hamming Co

r_1 will take care of these bits.

11	10	9	8	7	6	5	4	3	2	1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

r_2 will take care of these bits.

11	10	9	8	7	6	5	4	3	2	1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

r_4 will take care of these bits.

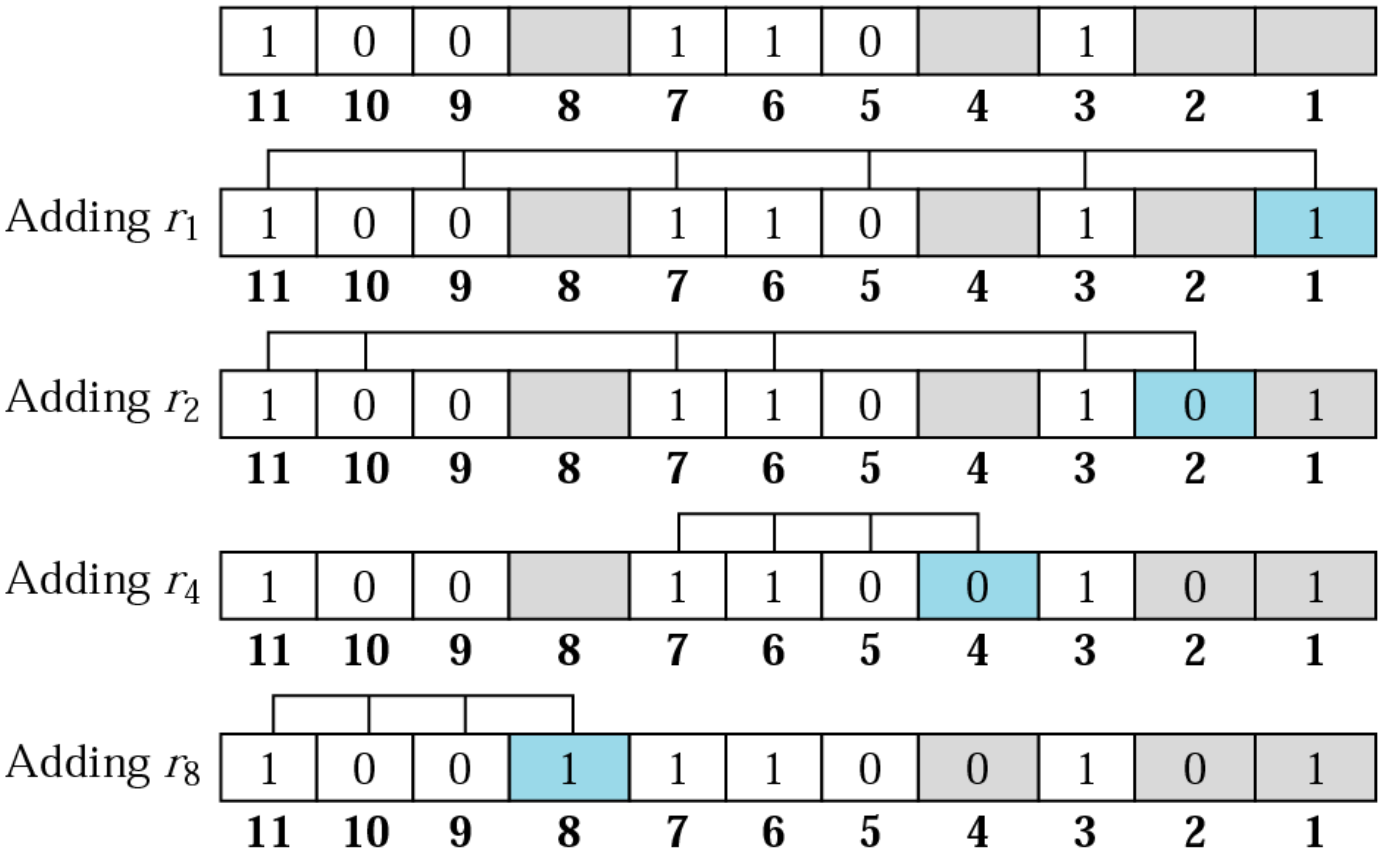
11	10	9	8	7	6	5	4	3	2	1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

r_8 will take care of these bits.

11	10	9	8	7	6	5	4	3	2	1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

Redundancy bit Calculation

Error Correction-Hamming Co

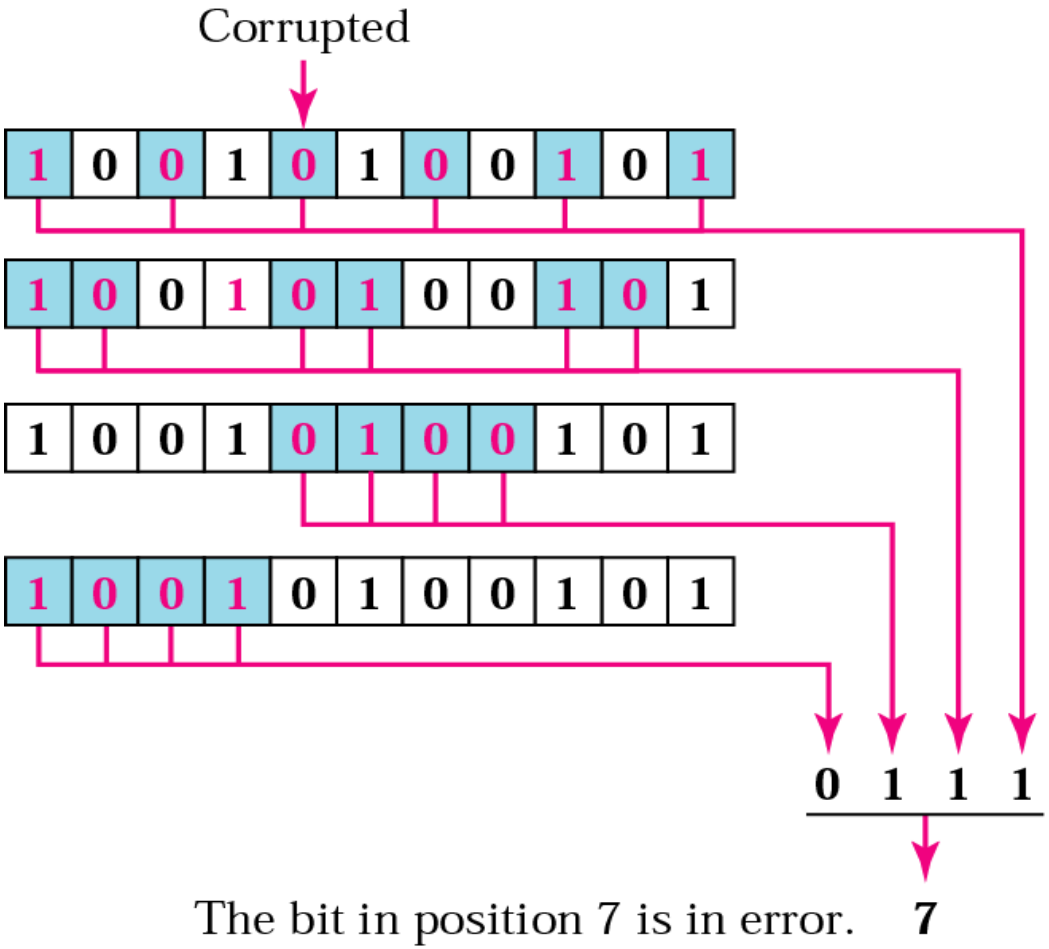


Data:
1 0 0 1 1 0 1

Code:
1 0 0 1 1 1 0 0 1 0 1

Example of Redundancy bit Calculation

Error Detection-Using Hammir Code

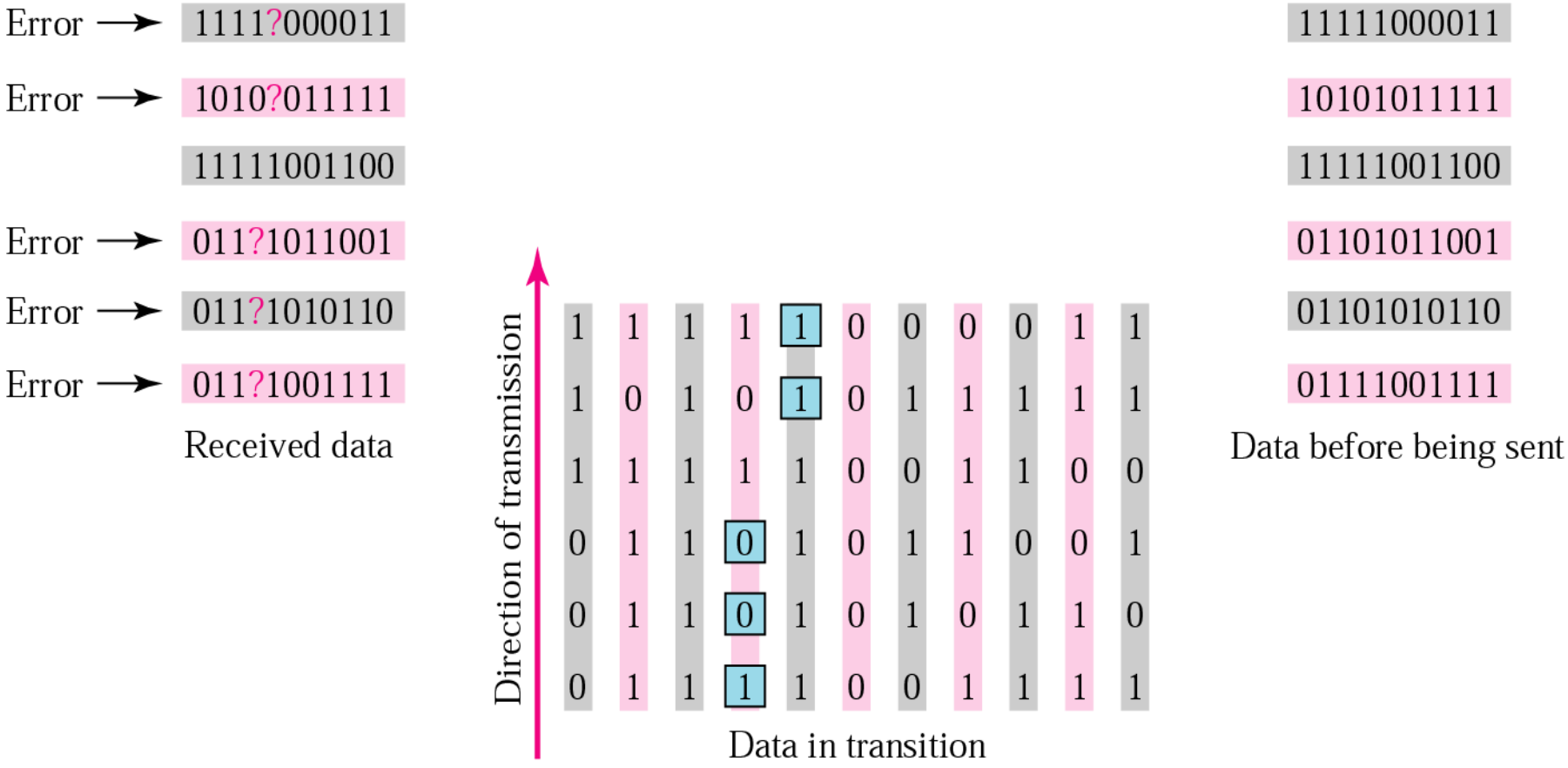


Error Correction-Burst Error Correction

- Instead of sending all the bits in a data unit together, we can organize “N” units in a column and then send the first bit of each , followed by the second bit of each and so on

- In this way, if a burst error of M bits occurs (M<N), then the error does not corrupt M bits of one single unit; it corrupts only 1 bit of a unit.

Burst Error Correction - Example



Thank You