Computer Network(CSC 503)

Shilpa Ingoley

Lecture 14

Different error-detecting codes

1. Parity Check
   - VRC
   - LRC
2. Checksums
3. Cyclic Redundancy Checks (CRCs)

Parity Check

Two sets of parity bits generated known as

- **Vertical redundancy bits (VRC)**
- **Longitudinal redundancy bits (LRC)**

Parity check : Vertical Redundancy Check (VRC)

- **Vertical Redundancy Check** is also known as **Parity Check.**
- A **redundant bit** also called **parity bit** is added to each data unit. This method includes even **parity** and **odd parity**
- **One extra bit** is sent along with the **original bits** to make number of **1s** either **even** in case of **even parity**

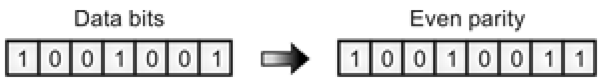
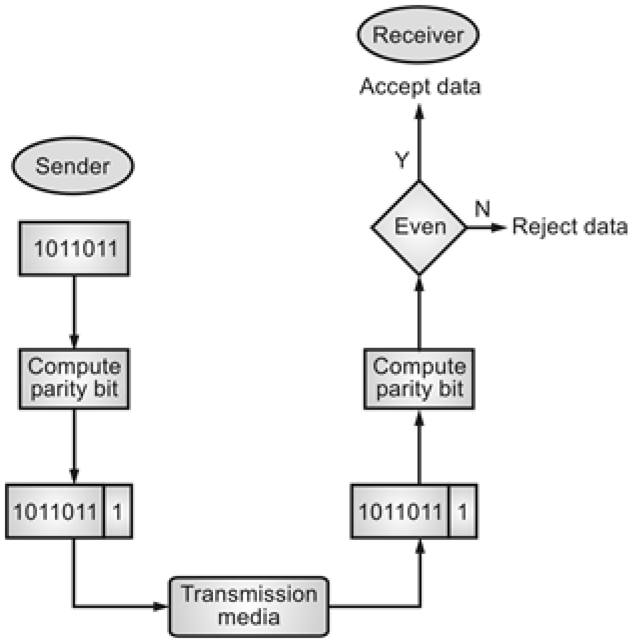


Fig: Simple parity check

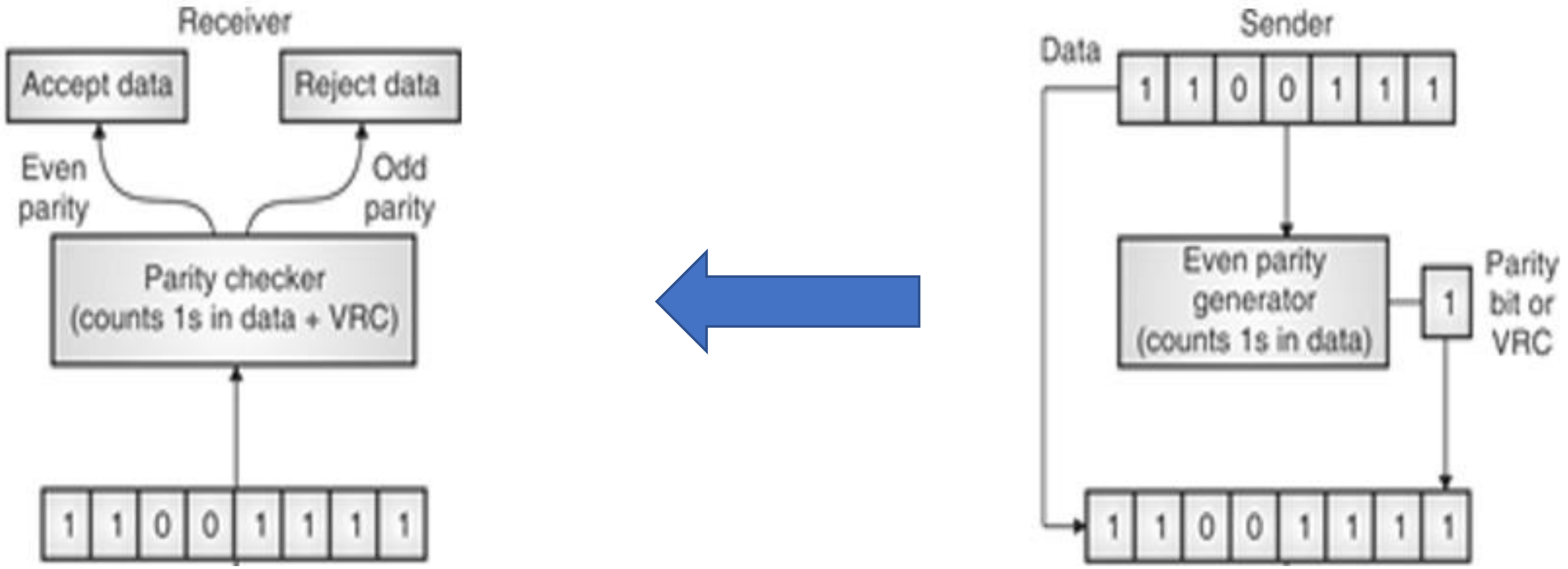


Adv/Disadvantages of Simple Parity Check

- 1. It can only **detect single-bit errors .**
- 2. If **two bits are interchanged**, then it **cannot detect the errors.**
- 3. It can also **detect burst errors** but only in those cases where **number of bits changed is odd**, i.e. 1, 3, 5, 7,…. etc.

Contd…

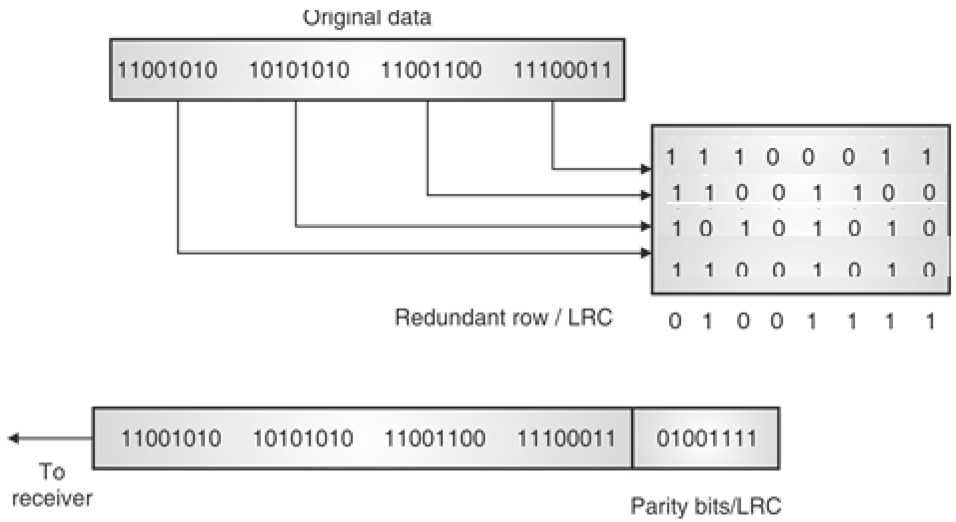
- **Vertical Redundancy Check** is also known as **Parity Check.**
- A **redundant bit** also called **parity bit** is added to each data unit. This method includes **even parity** and **odd parity.**



2. Longitudinal redundancy bits (LRC)

Longitudinal Redundancy Check (**LRC**) is also known as **2-D parity check(**Two-Dimensional Parity Check)**.**

- **Example :** If a block of 32 bits is to be transmitted, it is divided into matrix of four rows and eight columns



- In this matrix of bits, a **parity bit** (odd or even) **is calculated for each column**. It means **32 bits data plus 8** redundant bits are transmitted to receiver.
- Whenever data reaches at the destination, receiver uses **LRC to detect error in data**.

Contd…

- Organizes the data in the form of a table

Example: (MSB)10011001 11100010   0010010010000100(LSB)

Data :   10011001   11100010   00100100   10000100

Arrange the data in rows and columns

10000100
00100100
11100010
10011001

11011011

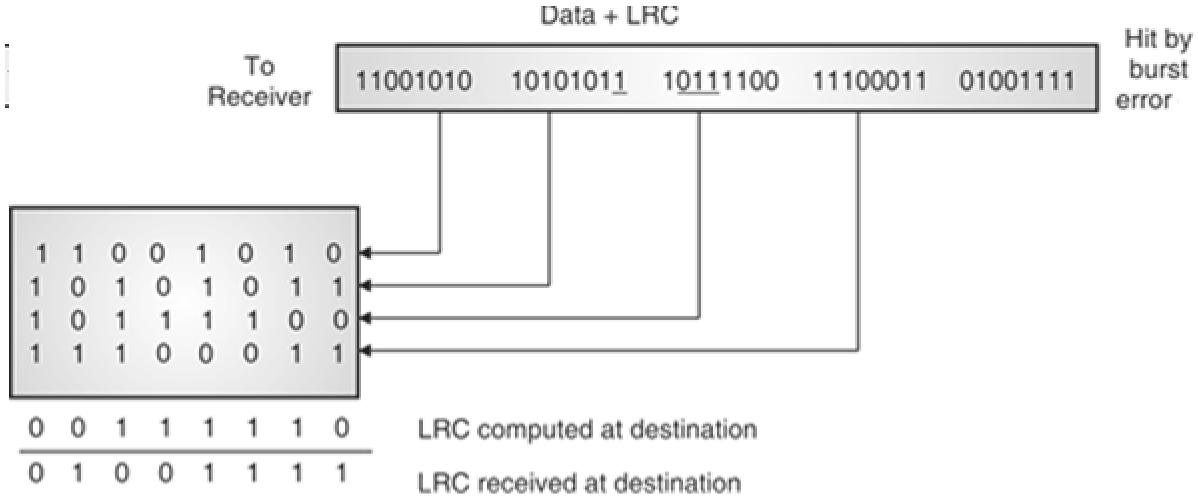
Transmitted Data : **11011011** 10011001 11100010 00100100 10000100

--LRC---

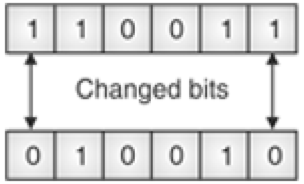
- At the receiving end, the parity bits are compared with the parity bits computed from the received data.

Contd...

- LRC is used to **detect burst errors.**

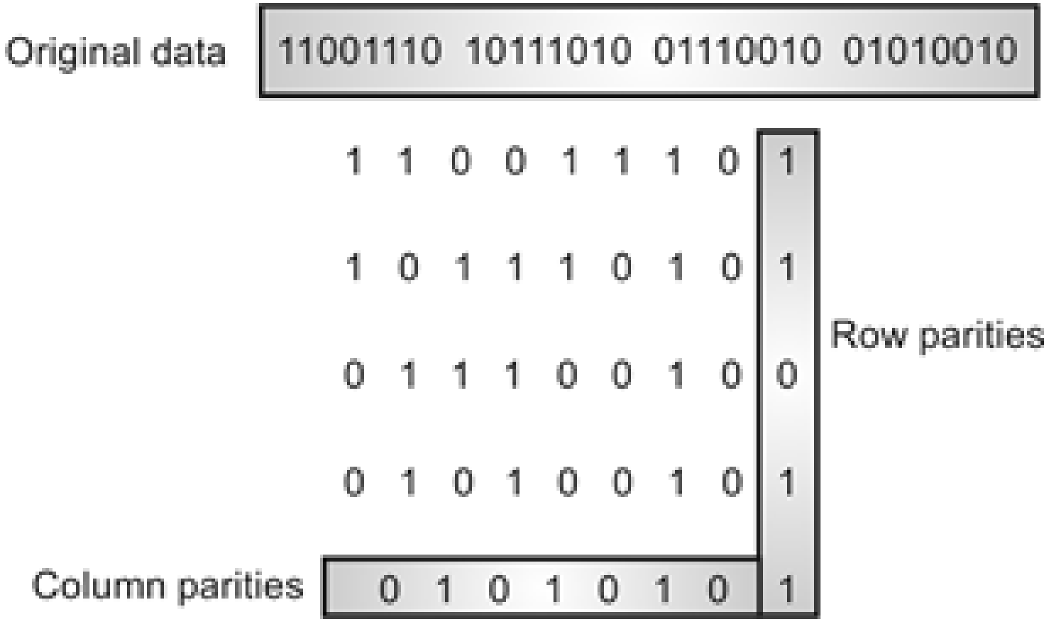


- The LRC received by the **destination does not match** with **newly corrupted LRC**. The **destination** comes to know that the data is erroneous, so it **discards the data**.
- The main problem with LRC is that, it is **not able to detect error if two bits in a data unit are damaged** and two **bits in exactly the same position**



Contd..

- Performance can be improved by using **Two-Dimensional Parity Check** which organizes the data in the form of a table computing and computing both **VRC** and **LRC** on data



- Parity check bits are **computed for each row**, which is equivalent to the single-parity check
- In Two-Dimensional Parity check, **a block of bits is divided into rows, and the redundant row** of bits is added to the whole block.
- At the receiving end, the parity bits are compared with the parity bits computed from the received data.

Checksum

At sender side

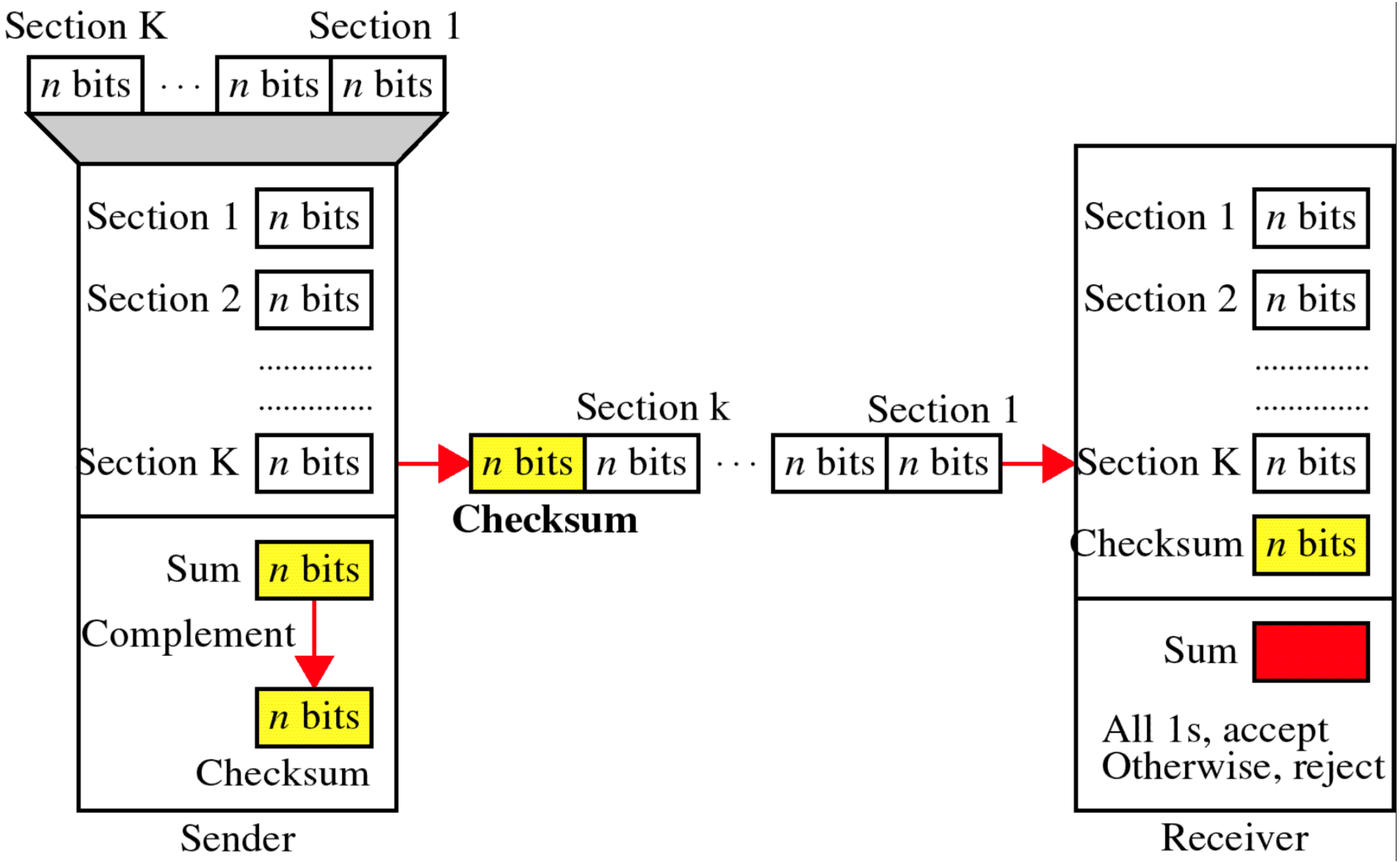
- If m bit checksum is used, the data unit to be transmitted is divided into segments of m bits.
- All the m bit segments are added.
- The result of the sum is then complemented using 1’s complement arithmetic.
- The value so obtained is called as **checksum**.
- The data along with the checksum value is transmitted to the receiver.

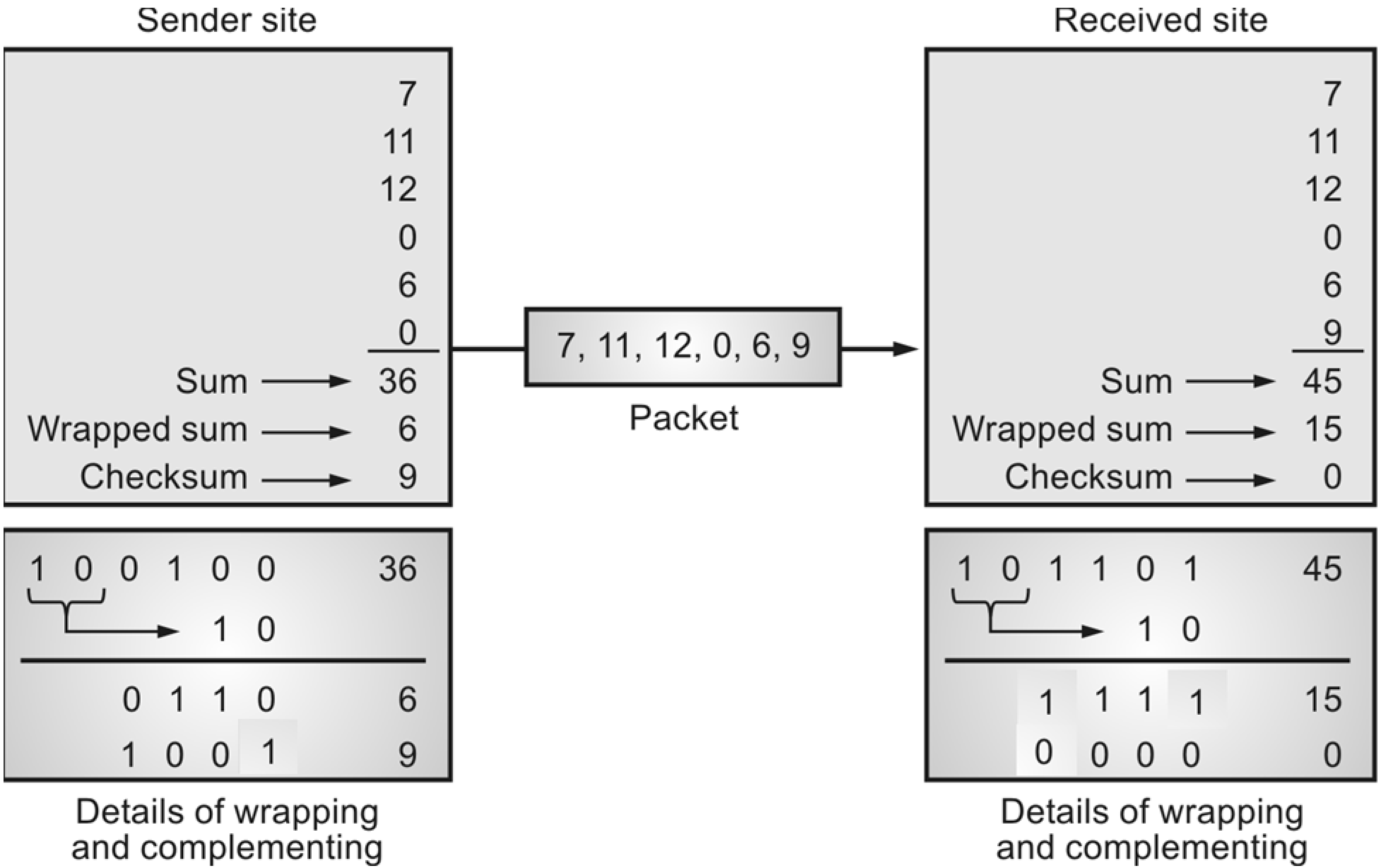
Contd…

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**
- To make the job of the receiver easy, 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.

Contd...


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


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

Contd…

- **At receiver side:**
- ▶If **m** bit checksum is being used, the received data unit is divided into segments of **m** bits.
- ▶All the m bit segments are added along with the checksum value.
- ▶The value so obtained is complemented and the result is checked.

- **Case-01: Result = 0**
- ▶Receiver assumes that no error occurred in the data during the transmission.
- ▶Receiver accepts the data.

- **Case-02: Result≠ 0**
- ▶Receiver assumes that error occurred in the data during the transmission.
- ▶Receiver discards the data and asks the sender for retransmission.

Contd…

Example 1:

At a sender

Original data : 10101001 00111001

10101001
00111001

11100010 Sum
00011101 Checksum

Data for transmission after the appending checksum

00011101 10101001 00111001

Contd...

Example 2:

At a receiver

Received data : 10101001 00111001 00011101

10101001
00111001
00011101

11111111 ←Sum
00000000 ←Complement

Contd…

- **Example 3:**

- Consider the data unit to be transmitted is-
- 10011001 11100010 00100100 10000100
- Consider 8 bit checksum is used.
- ▶Step-01:
- ▶At sender side,
- ▶The given data unit is divided into segments of 8 bits as-

10011001	11100010	00100100	10000100

Contd…

- Now, all the segments are added and the result is obtained as-
- 10011001 +11100010 + 00100100 + 10000100 =1000100011

- Since the result consists of 10 bits, so extra 2 bits are wrapped around.
- 00100011 + 10 = 00100101 (8 bits)
- Now, 1's complement is taken which is 11011010.
- Thus, checksum value =11011010

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.