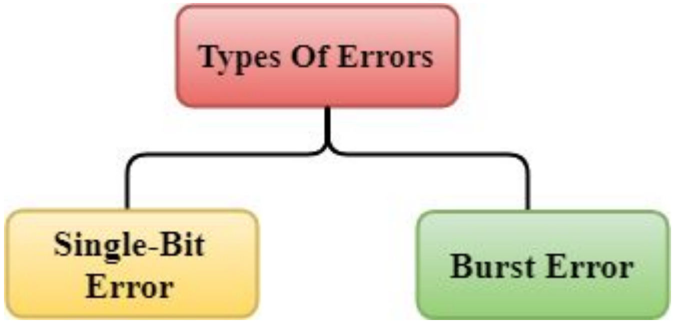
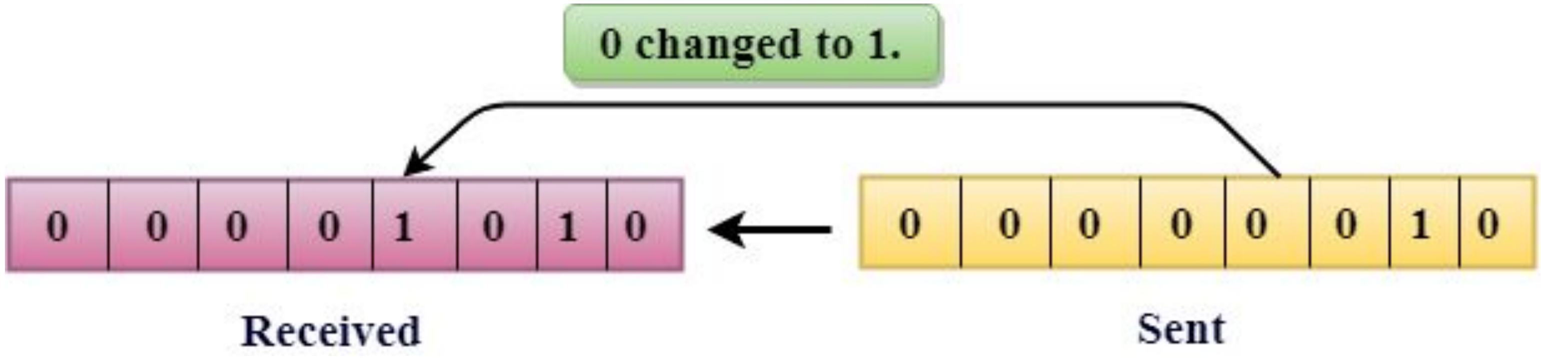
Error Detection

- When data is transmitted from one device to another device, the system does not guarantee whether the data received by the device is identical to the data transmitted by another device. An Error is a situation when the message received at the receiver end is not identical to the message transmitted.



Single-Bit Error:

- The only one bit of a given data unit is changed from 1 to 0 or from 0 to 1.



0 changed to 1.

0	0	0	0	1	0	1	0

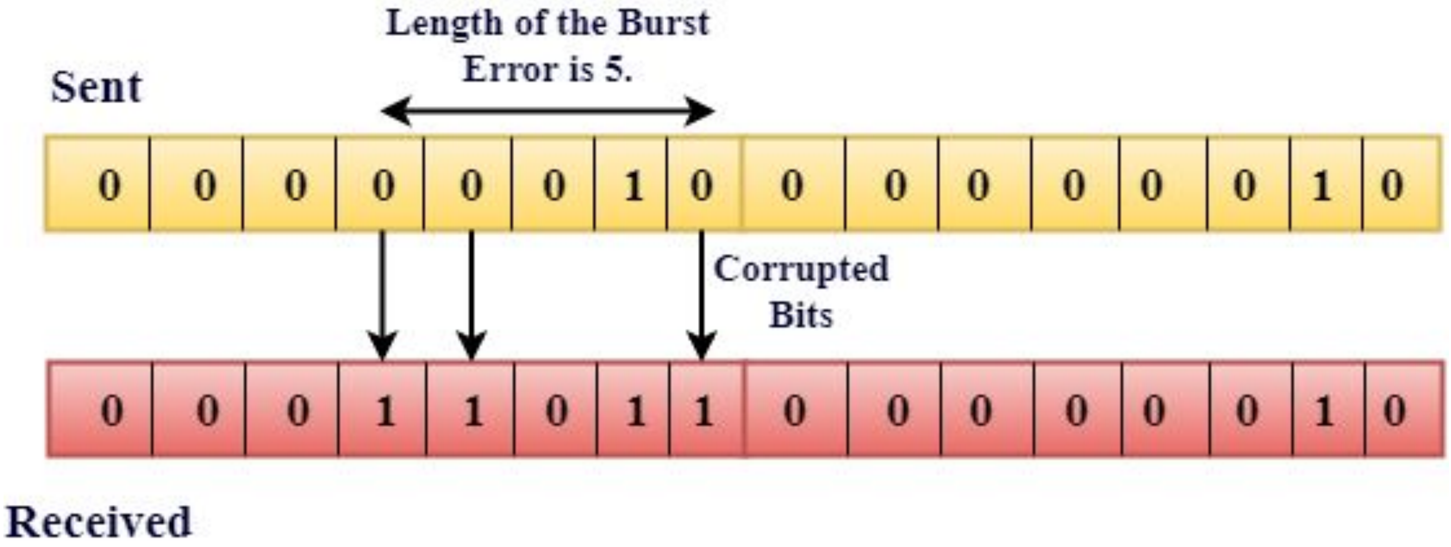
Received

0	0	0	0	0	0	1	0

Sent

Burst Error:

- The two or more bits are changed from 0 to 1 or from 1 to 0 is known as Burst Error.
- The Burst Error is determined from the first corrupted bit to the last corrupted bit.



Length of the Burst Error is 5.

Sent

0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0

Corrupted Bits

0	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0

Received

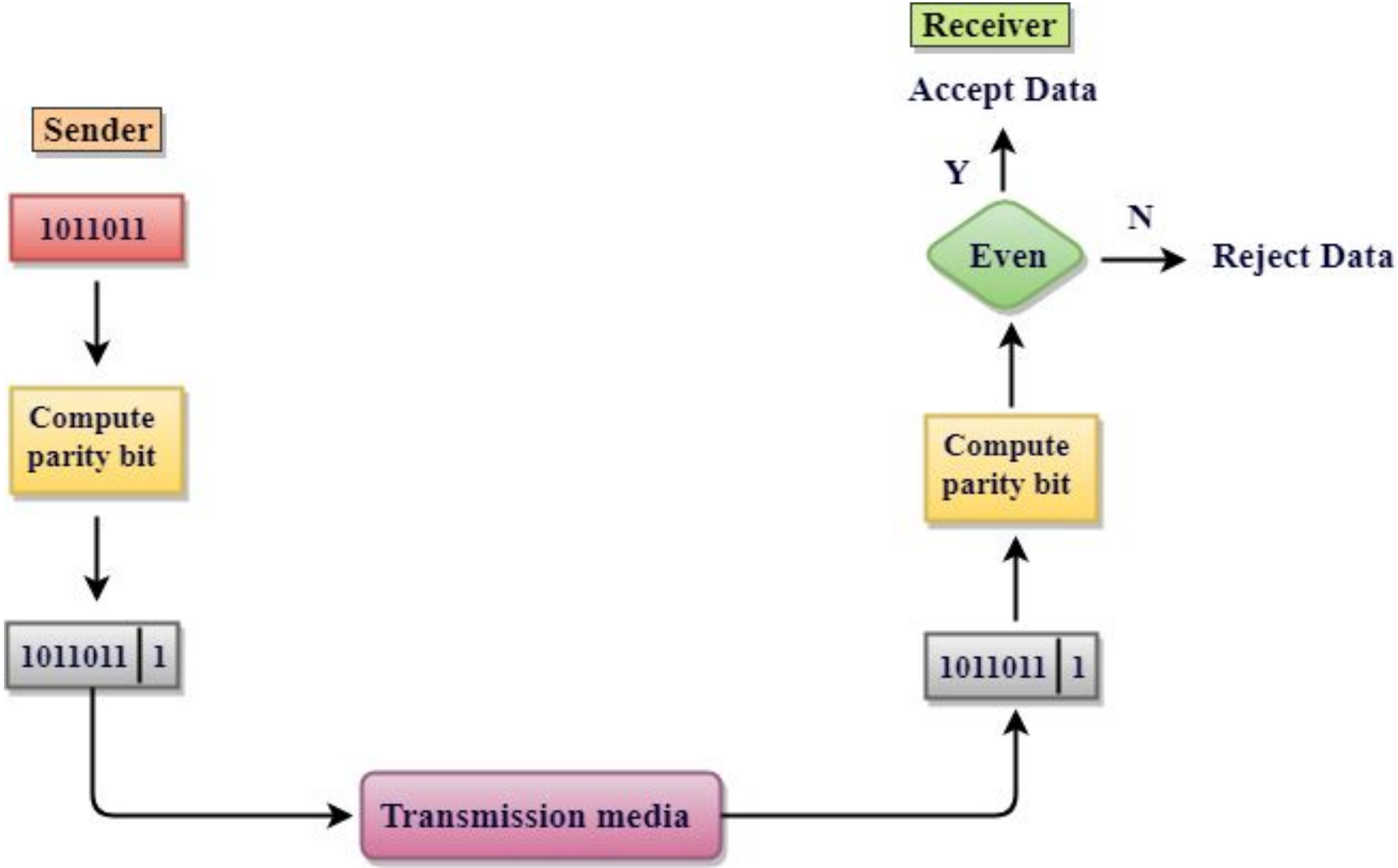
Error Detecting Techniques:

The most popular Error Detecting Techniques are:

- Single parity check
- Two-dimensional parity check
- Checksum
- Cyclic redundancy check

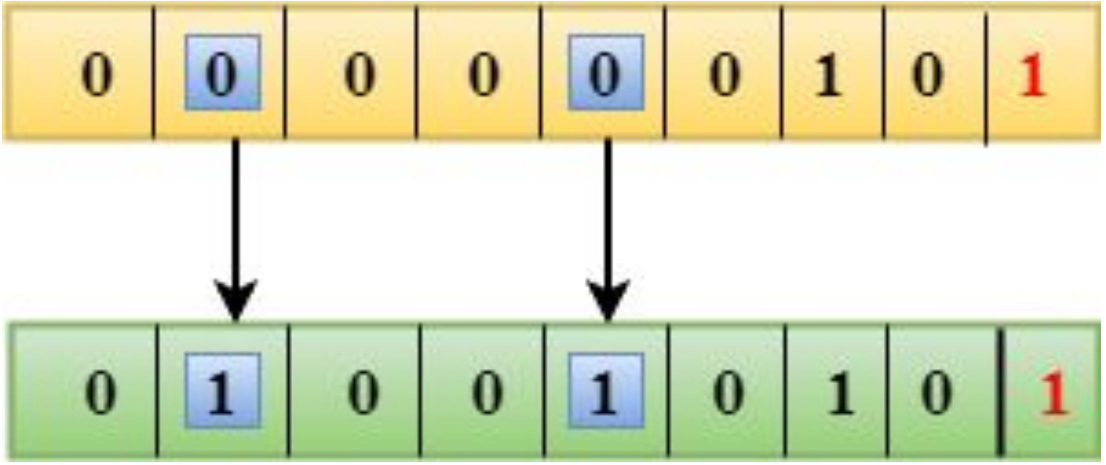
Single Parity Check

- Single Parity checking is the simple mechanism and inexpensive to detect the errors.
- In this technique, a redundant bit is also known as a parity bit which is appended at the end of the data unit so that the number of 1s becomes even. Therefore, the total number of transmitted bits would be 9 bits.
- If the number of 1s bits is odd, then parity bit 1 is appended and if the number of 1s bits is even, then parity bit 0 is appended at the end of the data unit.
- At the receiving end, the parity bit is calculated from the received data bits and compared with the received parity bit.
- This technique generates the total number of 1s even, so it is known as even-parity checking.


Receiver
Accept Data
Y
N
Even
Reject Data
Sender
1011011
Compute parity bit
1011011 | 1
Compute parity bit
1011011 | 1
Transmission media

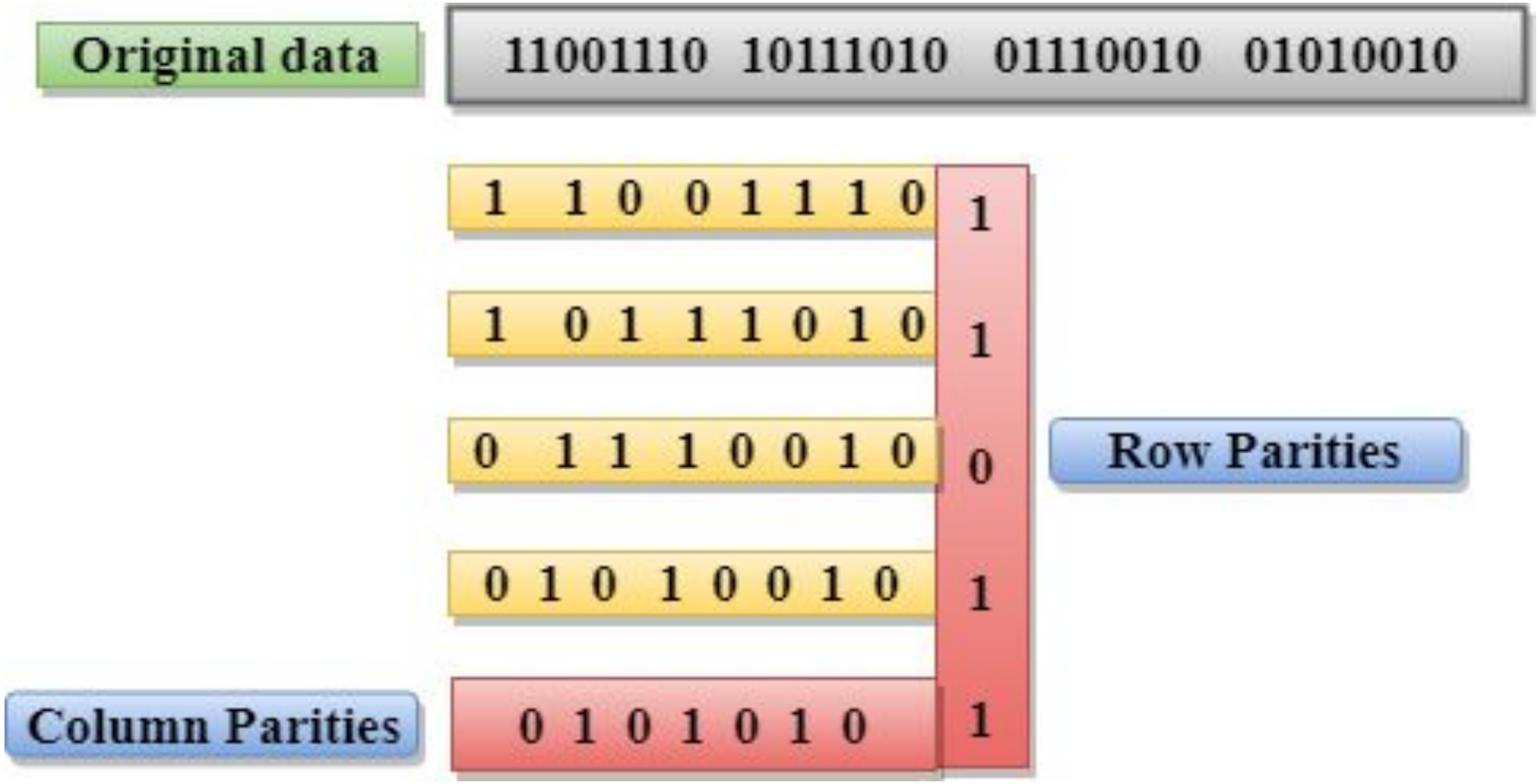
Drawbacks Of Single Parity Checking

- It can only detect single-bit errors which are very rare.
- If two bits are interchanged, then it cannot detect the errors.



Two-Dimensional Parity Check

- Performance can be improved by using **Two-Dimensional Parity Check** which organizes the data in the form of a table.
- 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.



Original data

11001110 10111010 01110010 01010010

1	1	0	0	1	1	1	0	1
1	0	1	1	1	0	1	0	1
0	1	1	1	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	0	1	0	1	0		1

Row Parities

Column Parities

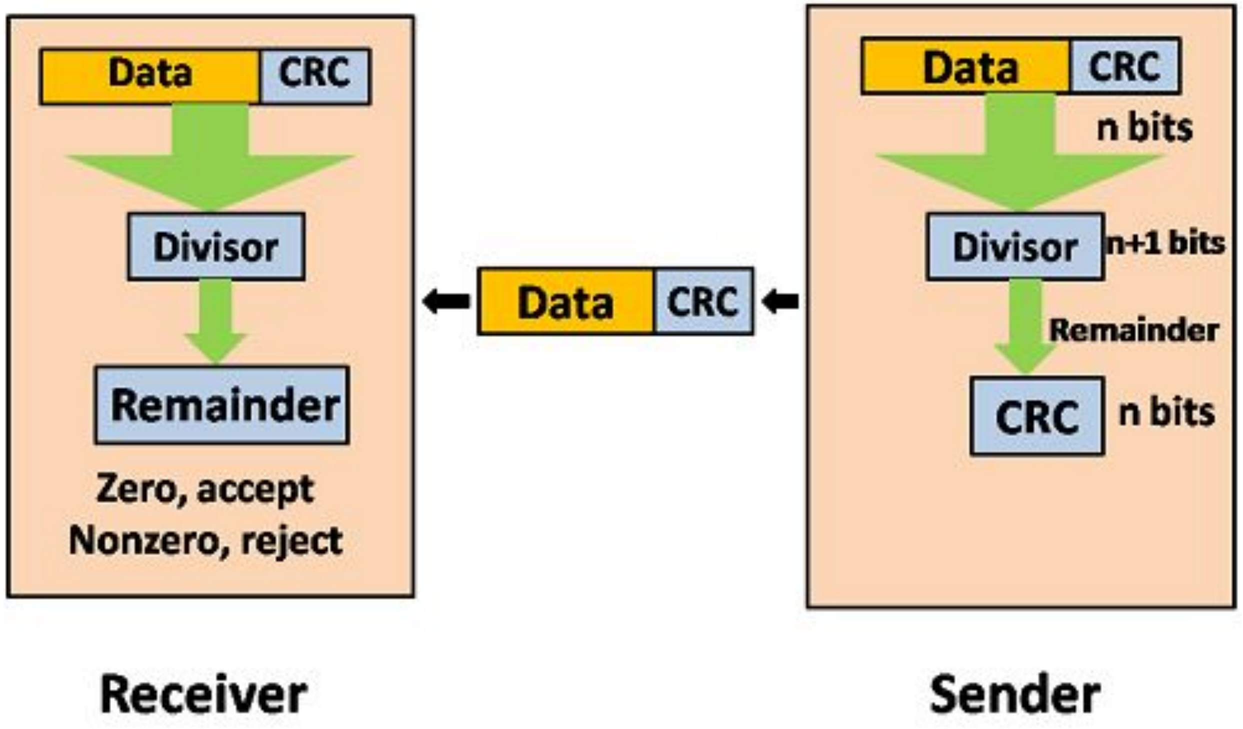
Drawbacks Of 2D Parity Check

- If two bits in one data unit are corrupted and two bits exactly the same position in another data unit are also corrupted, then 2D Parity checker will not be able to detect the error.
- This technique cannot be used to detect the 4-bit errors or more in some cases.

Cyclic Redundancy Check (CRC)

- CRC is a redundancy error technique used to determine the error.
- **Following are the steps used in CRC for error detection:**
- In CRC technique, a string of n 0s is appended to the data unit, and this n number is less than the number of bits in a predetermined number, known as division which is n+1 bits.
- Secondly, the newly extended data is divided by a divisor using a process is known as binary division. The remainder generated from this division is known as CRC remainder.
- Thirdly, the CRC remainder replaces the appended 0s at the end of the original data. This newly generated unit is sent to the receiver.
- The receiver receives the data followed by the CRC remainder. The receiver will treat this whole unit as a single unit, and it is divided by the same divisor that was used to find the CRC remainder.

- If the resultant of this division is zero which means that it has no error, and the data is accepted.
- If the resultant of this division is not zero which means that the data consists of an error. Therefore, the data is discarded.

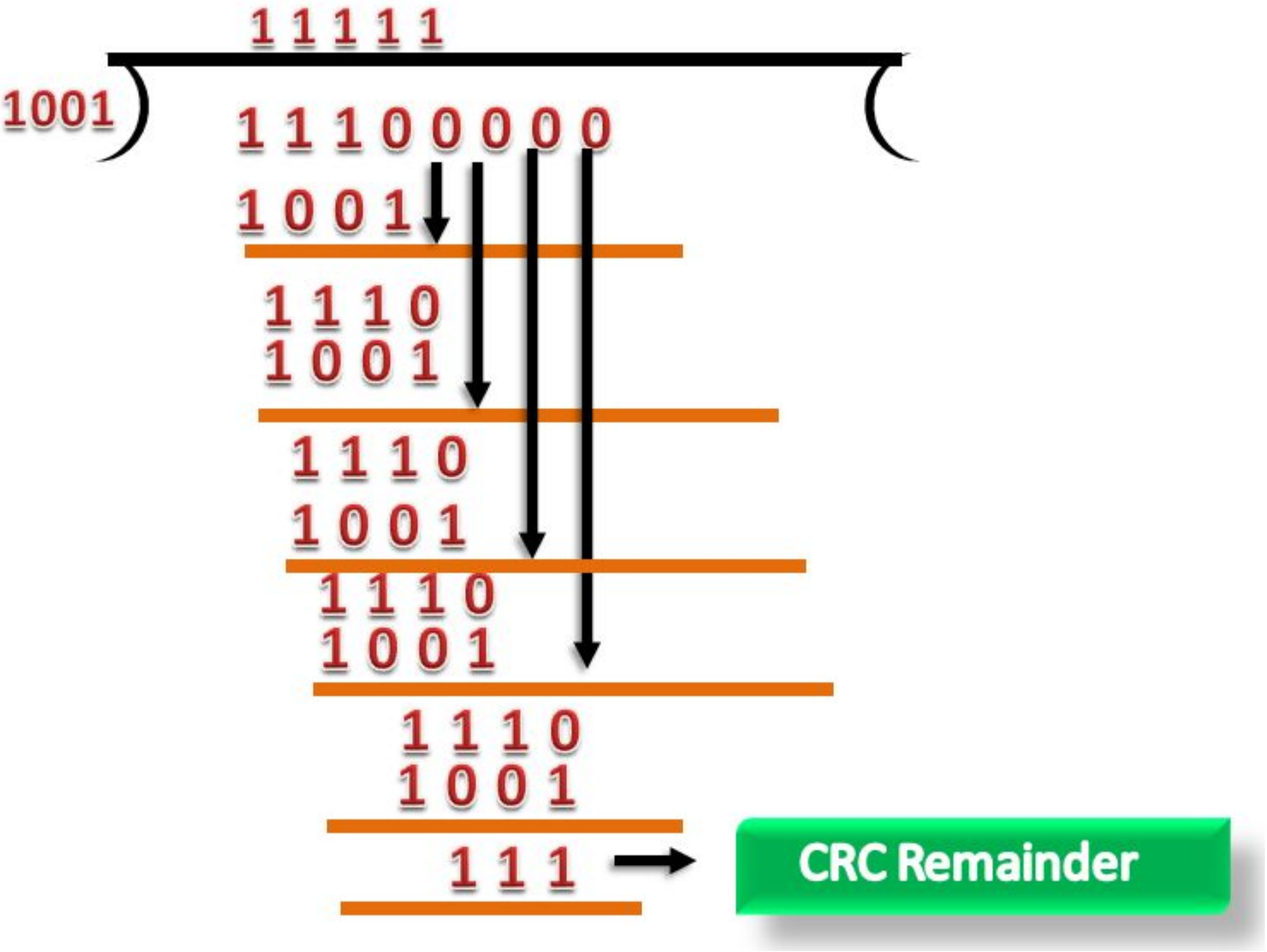


Let's understand this concept through an example:

Suppose the original data is 11100 and divisor is 1001.

CRC Generator

- A CRC generator uses a modulo-2 division. Firstly, three zeroes are appended at the end of the data as the length of the divisor is 4 and we know that the length of the string 0s to be appended is always one less than the length of the divisor.
- Now, the string becomes 11100000, and the resultant string is divided by the divisor 1001.
- The remainder generated from the binary division is known as CRC remainder. The generated value of the CRC remainder is 111.
- CRC remainder replaces the appended string of 0s at the end of the data unit, and the final string would be 11100111 which is sent across the network.

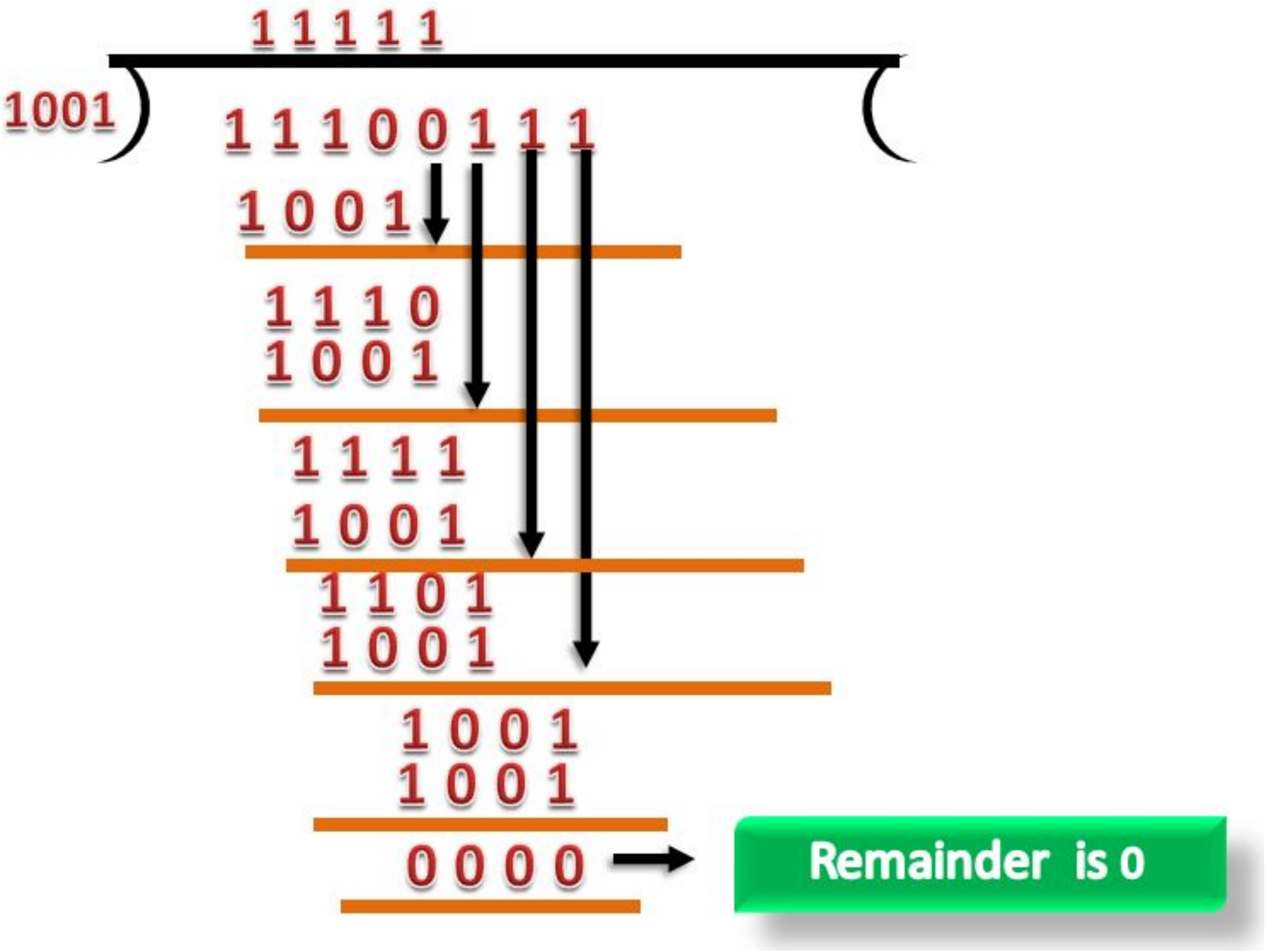


```
          1 1 1 1 1
1001 ) 1 1 1 0 0 0 0 0
       1 0 0 1
         1 1 1 0
         1 0 0 1
           1 1 1 0
           1 0 0 1
             1 1 1 0
             1 0 0 1
               1 1 1 0
               1 0 0 1
                 1 1 1
```

CRC Remainder

CRC Checker

- The functionality of the CRC checker is similar to the CRC generator.
- When the string 11100111 is received at the receiving end, then CRC checker performs the modulo-2 division.
- A string is divided by the same divisor, i.e., 1001.
- In this case, CRC checker generates the remainder of zero. Therefore, the data is accepted.



11111

1001) 11100111

1001

1110

1001

1111

1001

1101

1001

1001

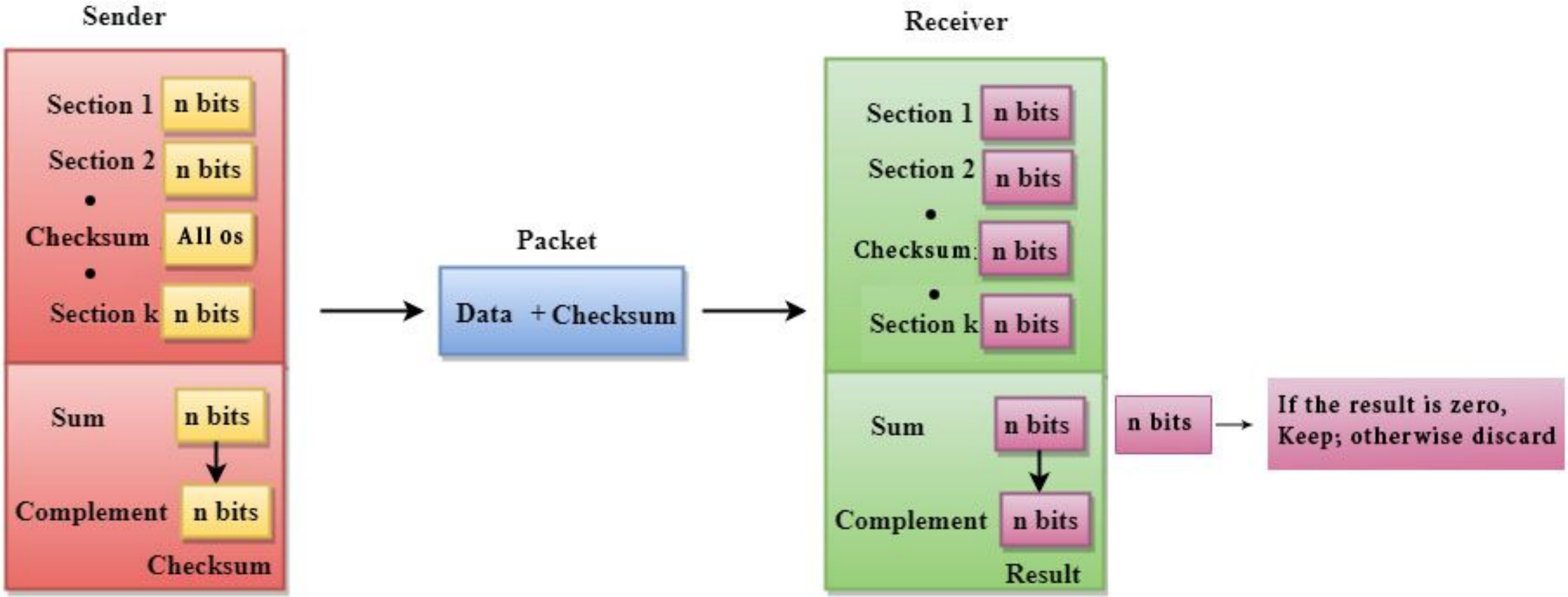
1001

0000

Remainder is 0

Checksum

- A Checksum is an error detection technique based on the concept of redundancy.
- **It is divided into two parts:**
- Checksum Generator
- A Checksum is generated at the sending side. Checksum generator subdivides the data into equal segments of n bits each, and all these segments are added together by using one's complement arithmetic. The sum is complemented and appended to the original data, known as checksum field. The extended data is transmitted across the network.
- Suppose L is the total sum of the data segments, then the checksum would be ?L


Sender
Section 1
n bits
Section 2
n bits
Checksum
All 0s
Section k
n bits
Sum
n bits
Complement
n bits
Checksum
Packet
Data + Checksum
Receiver
Section 1
n bits
Section 2
n bits
Checksum
n bits
Section k
n bits
Sum
n bits
Complement
n bits
Result
n bits
If the result is zero,
Keep; otherwise discard

The Sender follows the given steps:

- The block unit is divided into k sections, and each of n bits.
- All the k sections are added together by using one's complement to get the sum.
- The sum is complemented and it becomes the checksum field.
- The original data and checksum field are sent across the network.

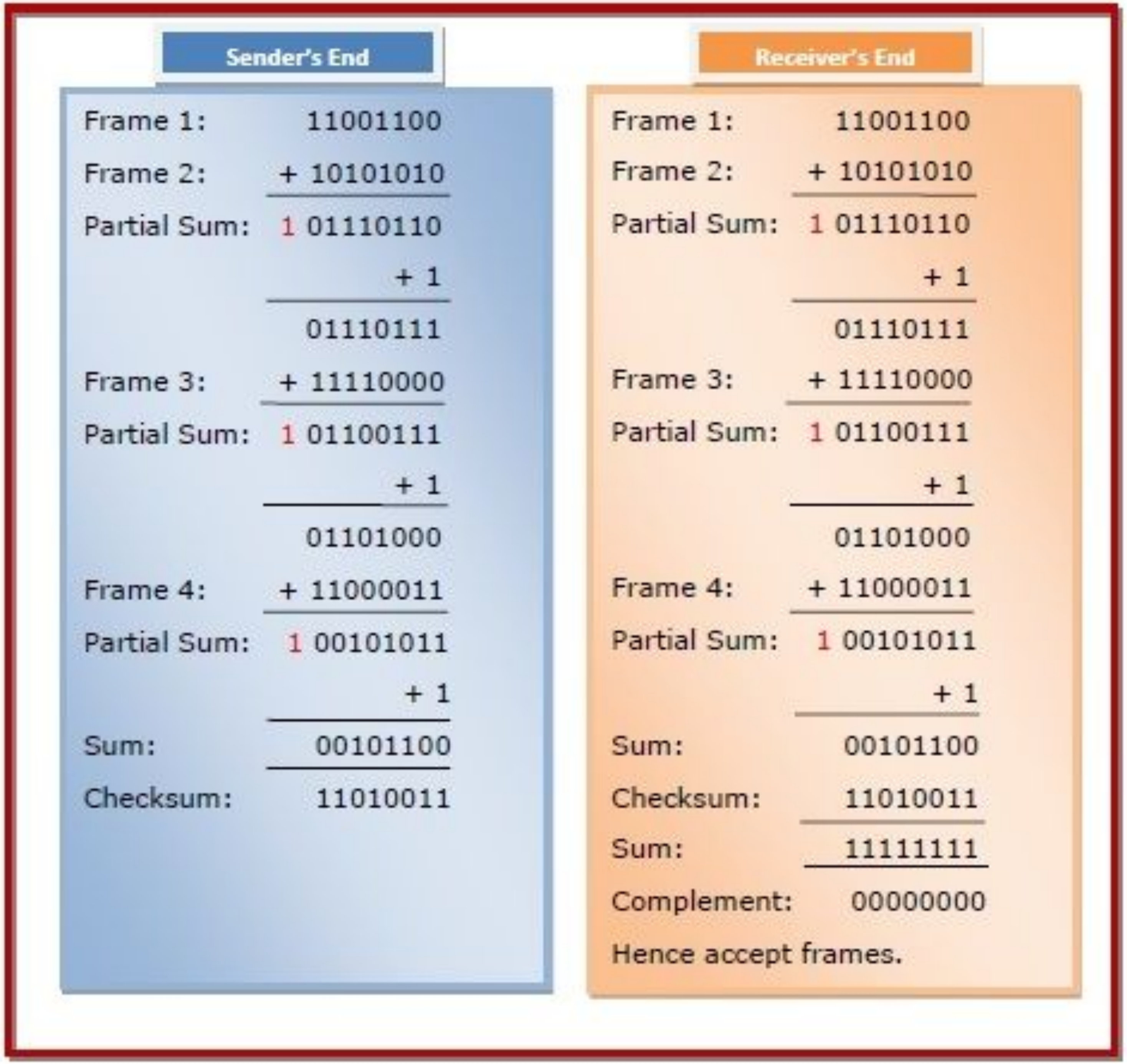
Checksum Checker

- A Checksum is verified at the receiving side. The receiver subdivides the incoming data into equal segments of n bits each, and all these segments are added together, and then this sum is complemented. If the complement of the sum is zero, then the data is accepted otherwise data is rejected.

The Receiver follows the given steps:

- The block unit is divided into k sections and each of n bits.
- All the k sections are added together by using one's complement algorithm to get the sum.
- The sum is complemented.
- If the result of the sum is zero, then the data is accepted otherwise the dat a is discarded.

- For this given data 11001100 10101010 11110000 11000011, perform check sum operation at sender site and receiver site and verify the data at receiver site.
- The sender adds the bits using 1s complement arithmetic. While adding two numbers using 1s complement arithmetic, if there is a carry over, it is added to the sum.
- After adding all the 4 frames, the sender complements the sum to get the checksum, 11010011, and sends it along with the data frames.
- The receiver performs 1s complement arithmetic sum of all the frames including the checksum. The result is complemented and found to be 0. Hence, the receiver assumes that no error has occurred.



Sender's End

```
Frame 1:          11001100
Frame 2:        + 10101010
Partial Sum:    1 01110110
                       + 1
                  01110111
Frame 3:        + 11110000
Partial Sum:    1 01100111
                       + 1
                  01101000
Frame 4:        + 11000011
Partial Sum:    1 00101011
                       + 1
Sum:              00101100
Checksum:         11010011
```

Receiver's End

```
Frame 1:          11001100
Frame 2:        + 10101010
Partial Sum:    1 01110110
                       + 1
                  01110111
Frame 3:        + 11110000
Partial Sum:    1 01100111
                       + 1
                  01101000
Frame 4:        + 11000011
Partial Sum:    1 00101011
                       + 1
Sum:              00101100
Checksum:         11010011
Sum:              11111111
Complement:       00000000
```

Hence accept frames.