Error detection

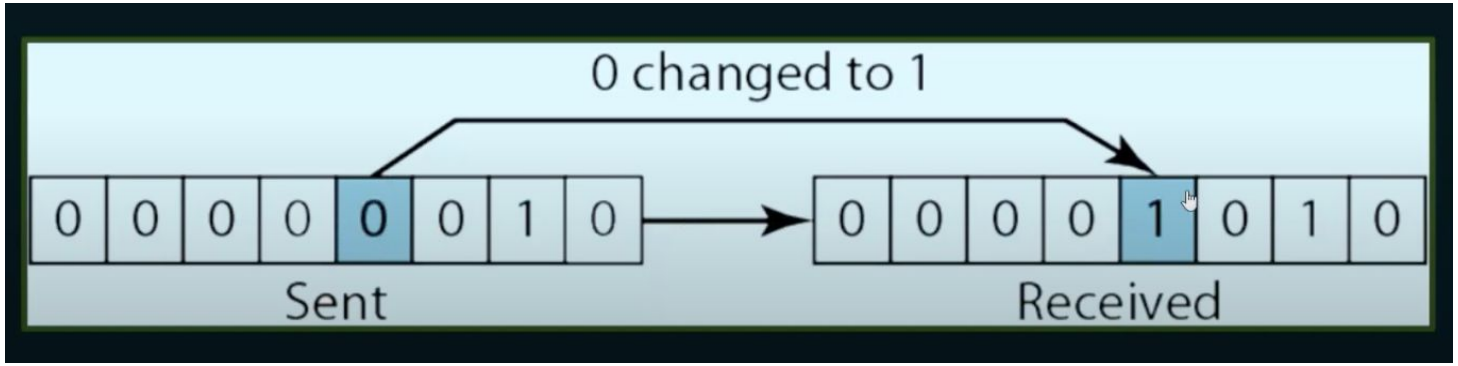
Error

- Data are transmitted in several network setups from one device to another
- Might be corrupted during transmission
- Transmission error
- For reliable communication, errors must be detected and corrected
- It happens in Transport Layer and Data link layer of OSI model

Types of errors: Single bit error

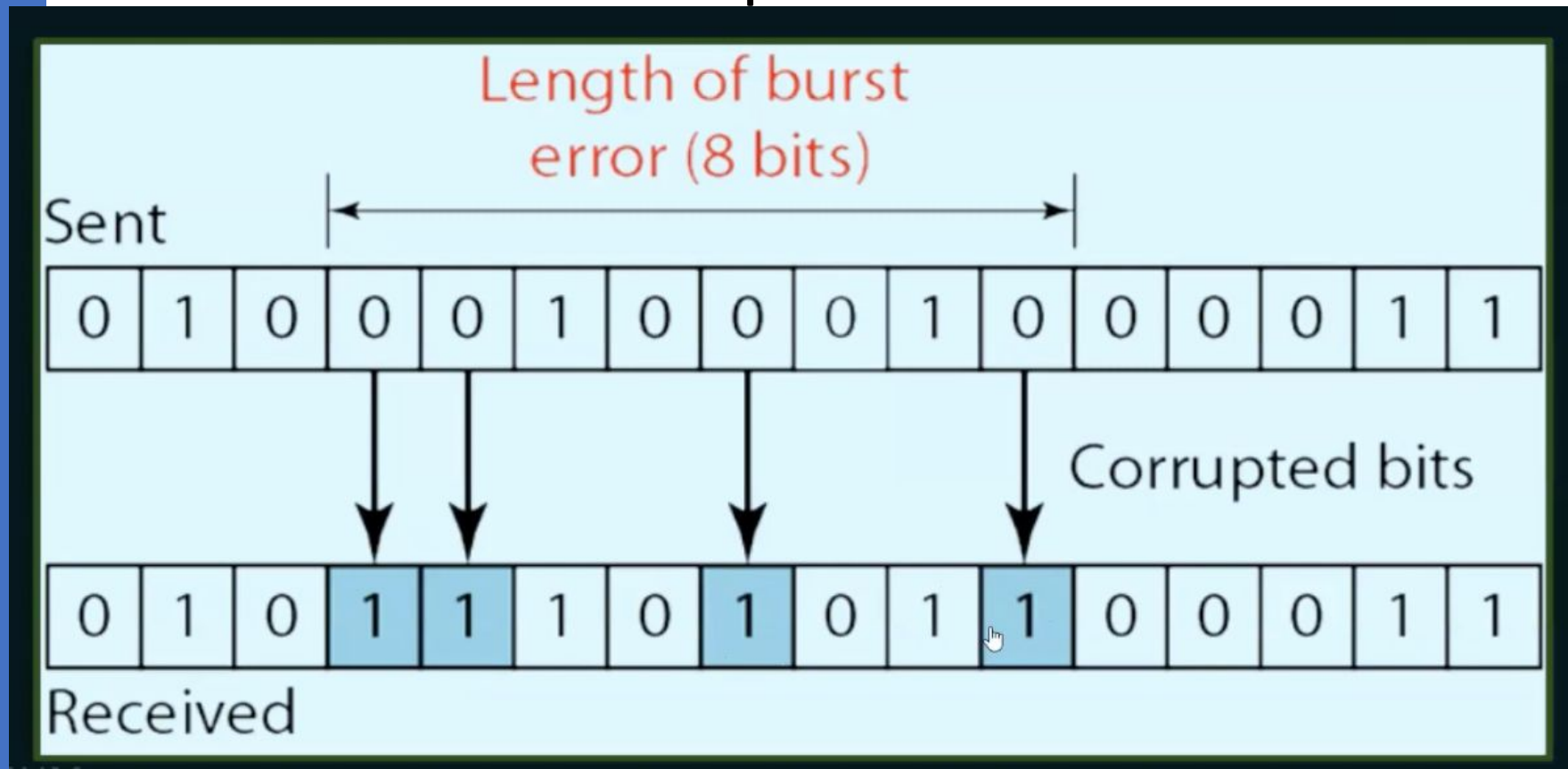
Sender sends 00000010 (2) , reiever recives 00001010 (4)

- Long distance attenuation, natural causes- lighting , machine error



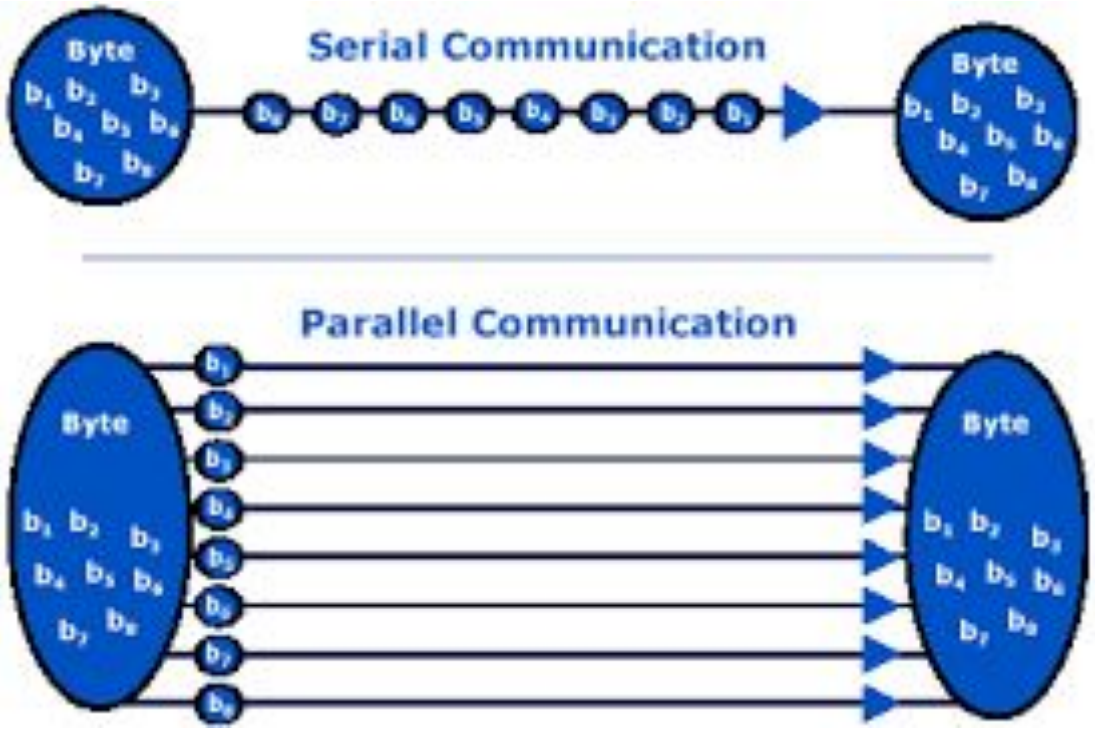
Burst error

- More than one bit is channged
  - 101010---------111011 change in 2 bits or more
  - lenght of the error 5 bits
    - Another example:



Side notes

Sr. No.	Factor	Serial	Parallel
1.	Number of bits transmitted at one clock pulse	One bit	*n* bits
2.	No. of lines required to transmit *n* bits	One line	*n* lines
3.	Speed of data transfer	Slow	Fast
4.	Cost of transmission	Low as one line is required	Higher as *n* lines are required.
5.	Application	Long distance communication between two computers	Short distance communication. like computer to printer.



2.2 Serial and Parallel Transmissions

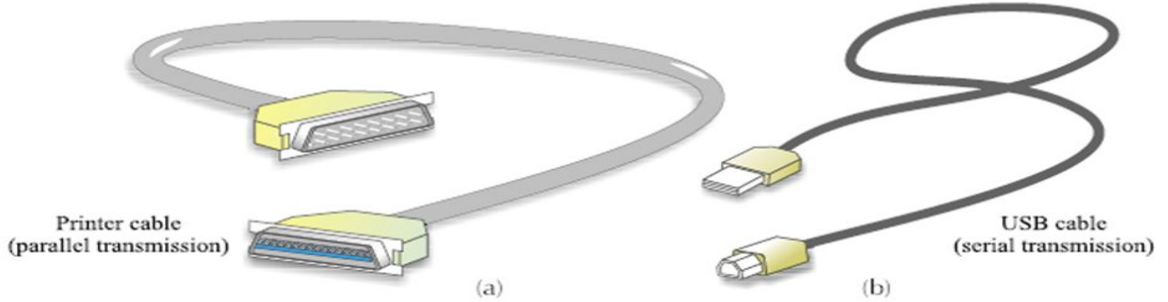


Fig.2.1 Parallel cables are in general thicker than serial cables.

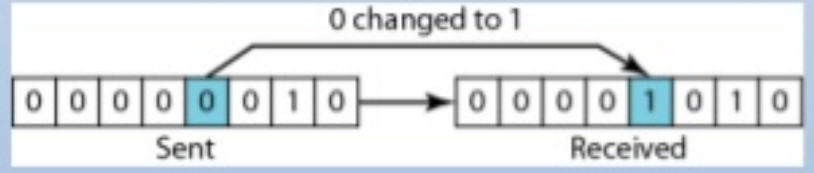
- Both serial and parallel transmissions have advantages and disadvantages.
- Parallel transmission is used for shorter distances and provides greater speed, while serial transmission is reliable for transferring data over longer distances.
- Both serial and parallel transmissions are individually essential for transferring data.

Difference

Type of errors

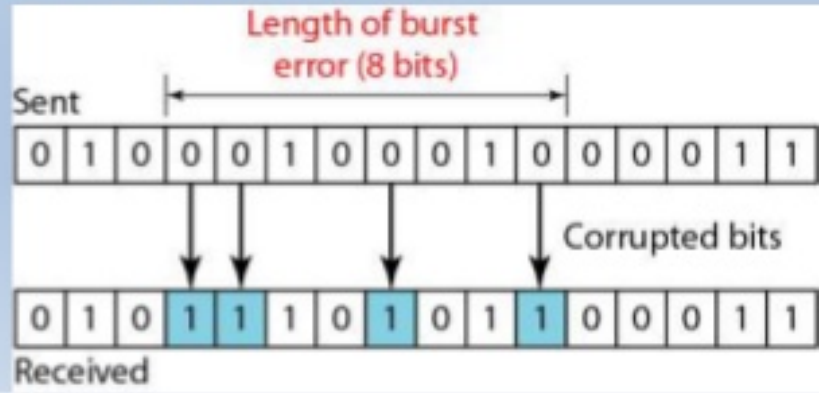
Single-Bit Error

- Only one bit of a given data unit is changed
- The least likely type of error in serial transmission
- Single-bit error can happen in parallel transmission



Burst Error

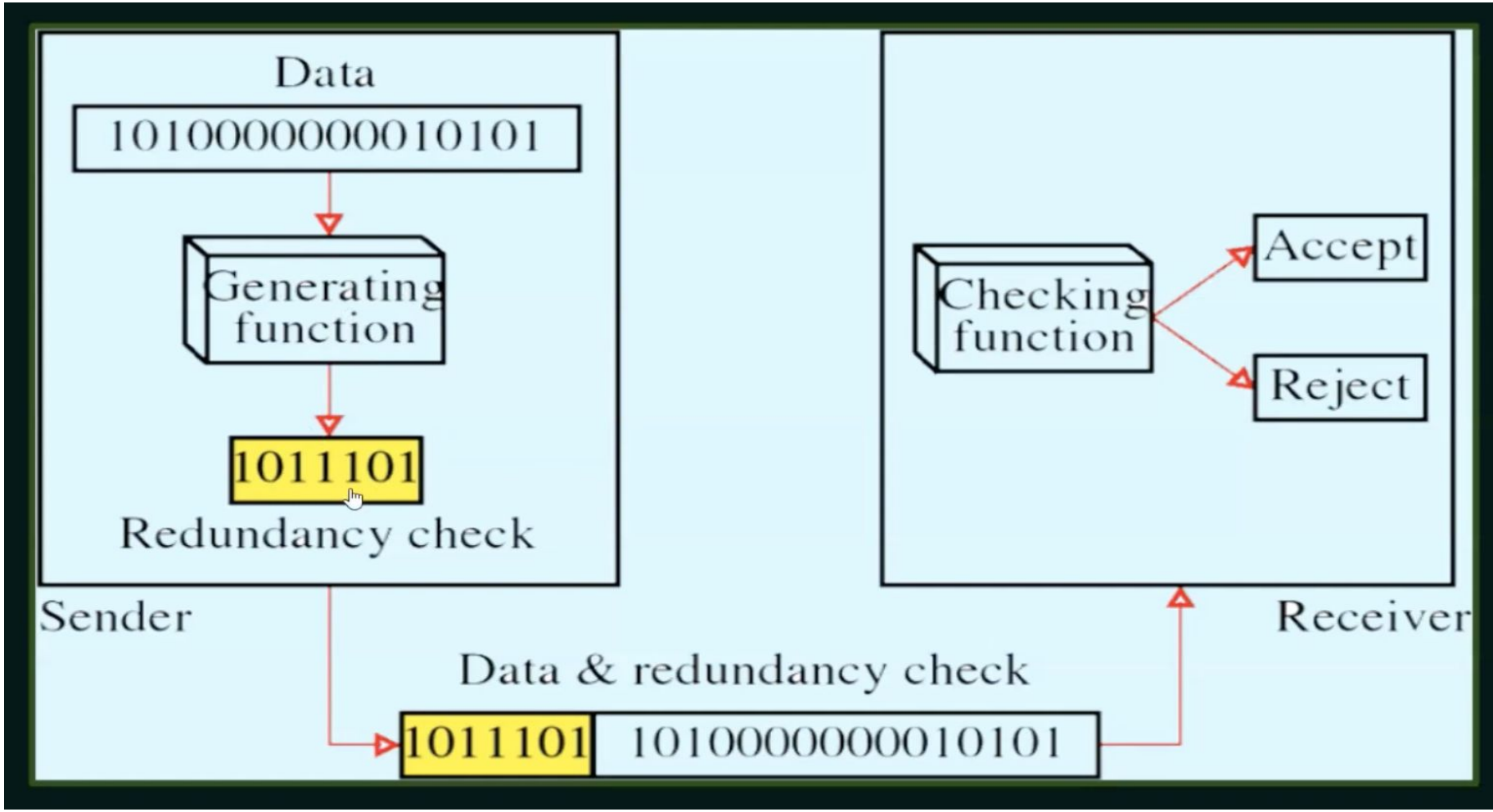
- Two or more bits in the data unit have changed
- Burst error does not necessarily mean that the errors occur in consecutive bits
- Most likely to happen in a serial transmission
- Number of bits affected depends on the data rate and duration of noise



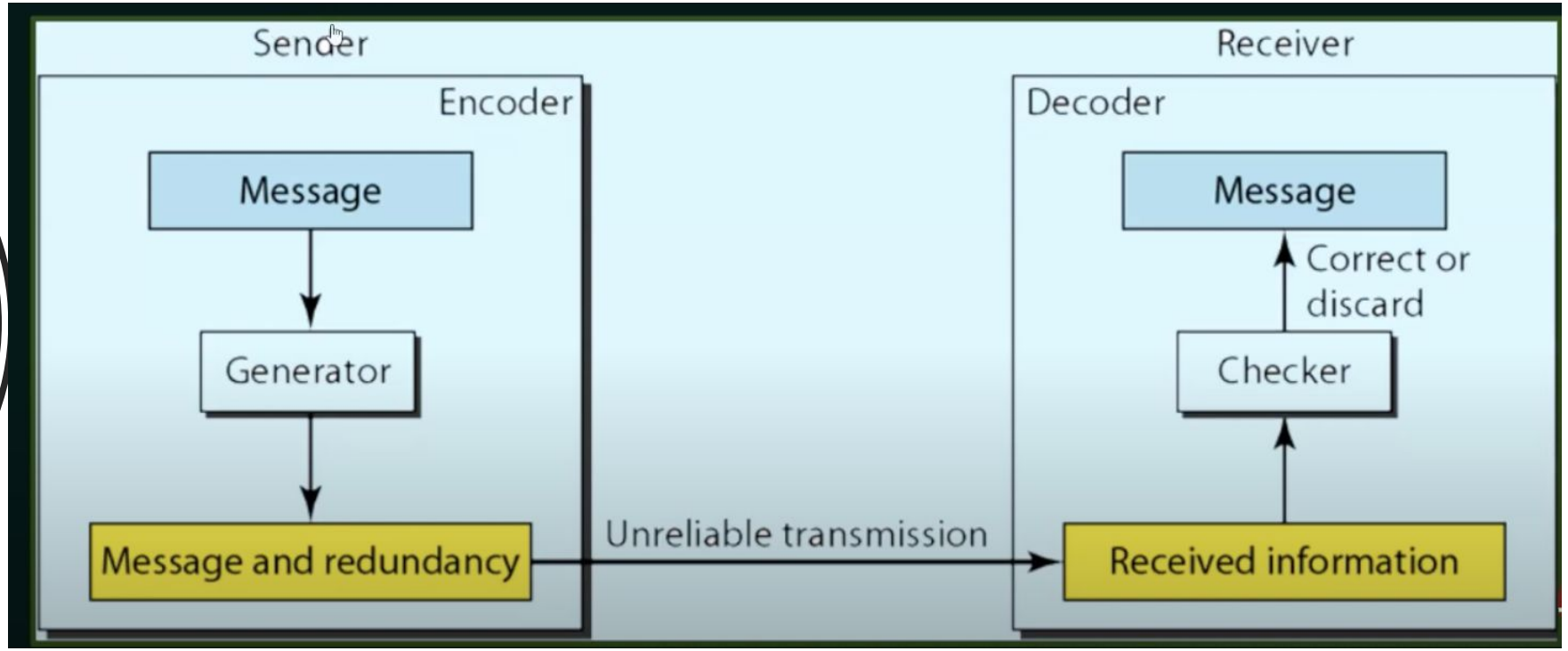
How to detect errors

- Error detection means to decide whether the received data is correct or not without having a copy of the original message
- If only data is transmitted, errors cannot be detected
- Send more information with data that satisfies a special relationship
- The extra bits are called redundant bits

Redundancy flowchart



Error detection steps



Error Detection methods

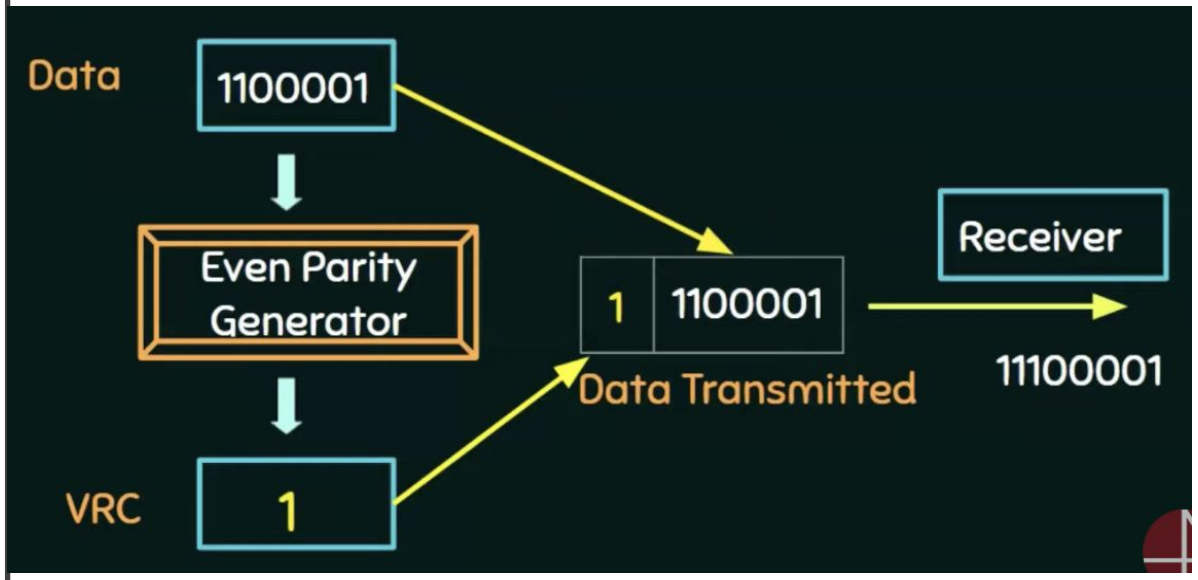
Detection

- Virtual Redundancy `check (VRC)
- Longitudinal Redundancy `check (LRC)
- Checksum
- CRC

Single parity –simple method +least expensive

- Want to send more data, less redundant bits
- m+1 bits (m=message bits)
  - Even parity no of 1 s should be even
  - 1010 0
  - 1110 1
  - It can detect all single bit errors in code words
  - 11101 is changed 01101 It can detect but cant correct
  - Can detect bburst errors only if the numnber of errors is odd
  - 11101---00101 can not detect

VRC / Parity check



ODD parity –self study

- example
- 000 1
- 001 0
- 010 0
- 011 1
- 100 0
- 101. 1
- 110. 1
- 111. 0

`Test=even/odd

- 1st 1110110 2nd 1101111 3d 1110010

LRC

- In LRC a block of bits is organized in rows and columns
- A.k.a= Two dimensional parity
- The parity bit is calculated for each column and sent along with the date
- The block of parity acts as the redundant its

Odd no. of 1's	1
Even no. of 1's	0

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

LRC →

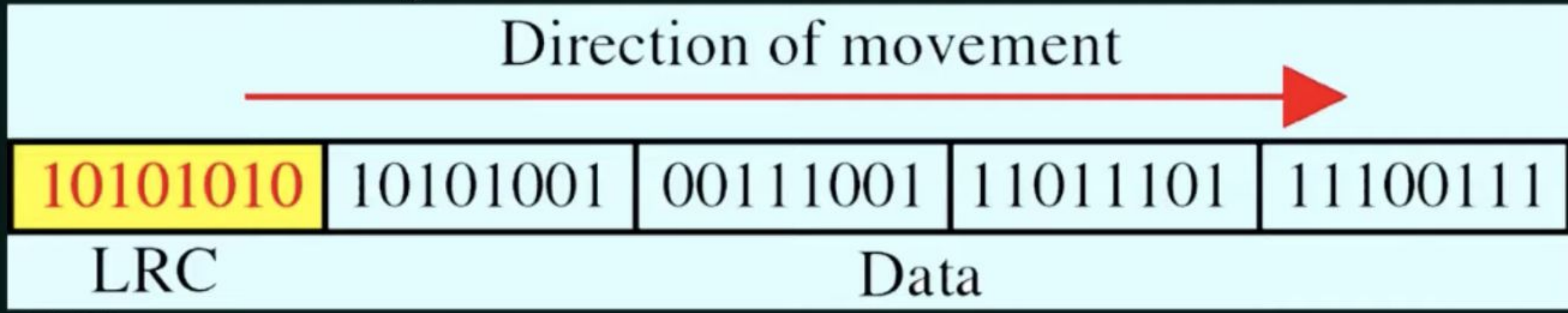
LRC -example

- Find the LRC for the data blocks 11100111 11011101 00111001 10101001 and determine the data that is transmitted.

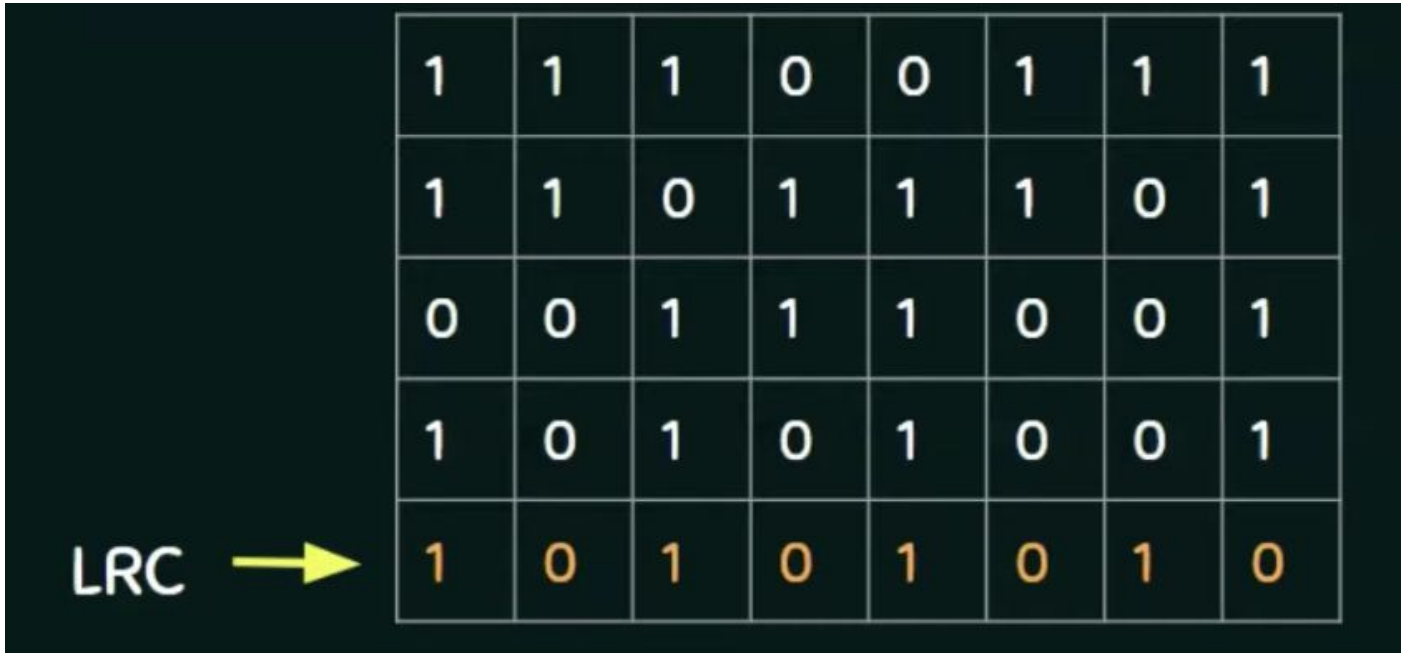
LRC – Example

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

LRC →



PERFORMANCE OF LRC



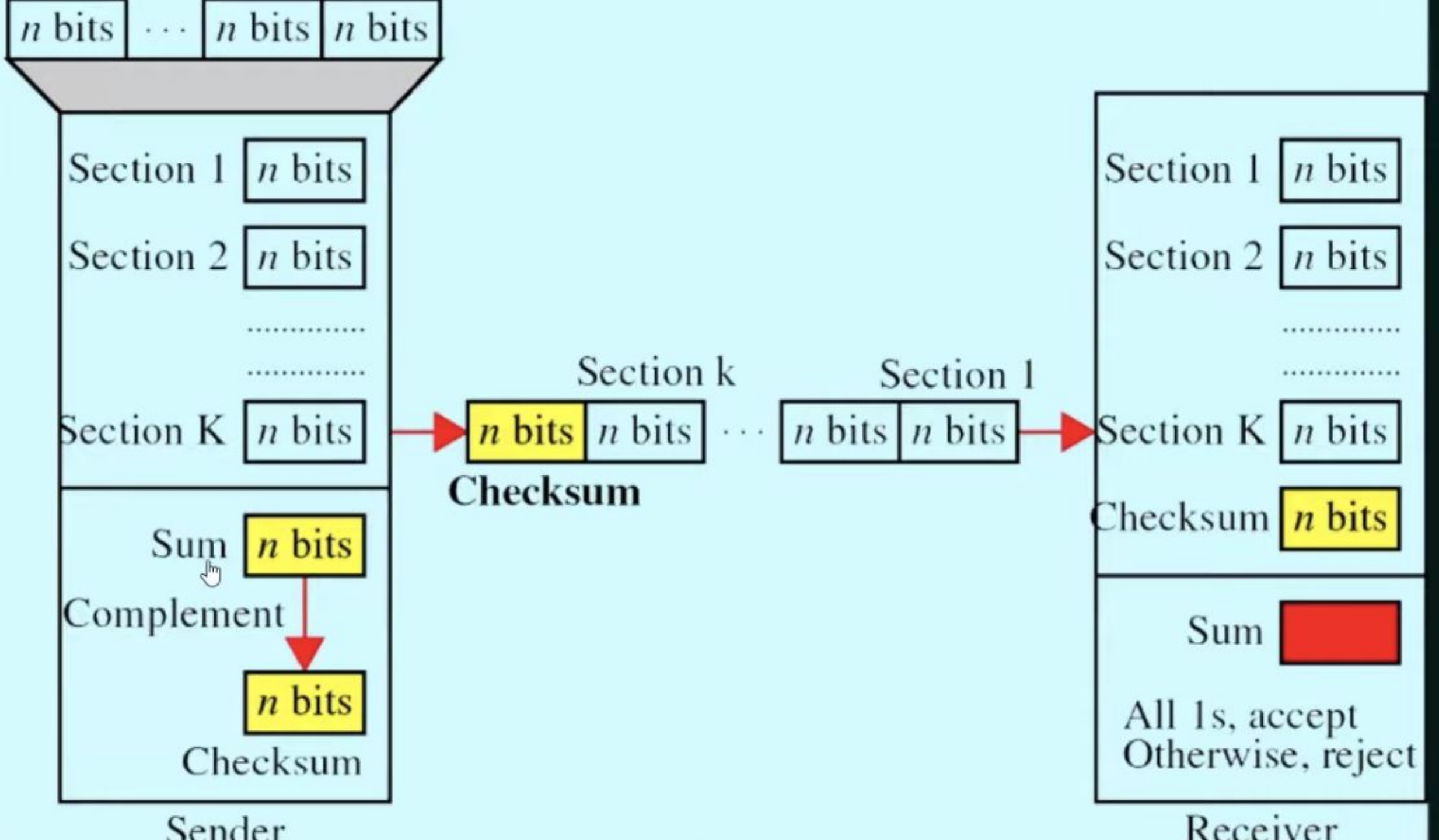
- INCREASES THE LIKELIHOOD OF DETECTING BURST ERRORS
- IF 2 BITS IN ONE DATA UNIT ARE DAMAGED AND 2 BITS ARE EXACTLY IN THE SAME POSITION IN ANOTHER DATA UNIT ARE ALSO DAMAGED , LRC CHECKER WILL NOT BE ABLE TO DETECT THE ERROR

CHECKSUM

- Checksum=check+sum
- Sender side= creation
- Reciever side = validation

Operation at sender side

- Break the original message into ‘k’ number of blocks with ‘n’ bits in ech block
- Sum all the k data blocks
- Add the carry to the sum, if any
- Do 1’s complement to the sum


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

CONSIDR THE DATA UNIT TO BE TRASMITTED

10011001111000100010010010000100

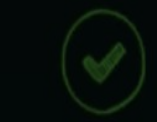
10011001 11100010 00100100 10000100

11011010	10011001		11100010		00100100		10000100	
Carry	1	1	1	1	1			
	1	0	0	0	0	1	0	0
	0	0	1	0	0	1	0	0
	1	1	1	0	0	0	1	0
	1	0	0	1	1	0	0	1
	0	0	1	0	0	0	1	1
							1	0
	0	0	1	0	0	1	0	1
CHECKSUM	1	1	0	1	1	0	1	0

Operation at reciever side

- Collect all the data blocks inculiding the cheksum
- Sum all the data blocks and checksum
- If the result is all 1’s ACCEPT., ELSE , REJECT

11011010	10011001	11100010	00100100	10000100
Carry	1 1 1 1 1 1			
	1 0 0 0 0 1 0 0			
	0 0 1 0 0 1 0 0			
	1 1 1 0 0 0 1 0			
	1 0 0 1 1 0 0 1			
	1 1 0 1 1 0 1 0			
	1 1 1 1 1 1 0 1			
	1 0			
	1 1 1 1 1 1 1 1			



Performance

- Detects all errors involving an odd number of bits
- It detects most errors involving an even number of bits