Group 9

19010819-012	Sajjal Farooq
19010819-013	Gulshan Fatima
19010819-021	Nabeel Yousaf
19810819-011	Moughais ul Hassan

MSc Computer Science

CONTENTS COVERED:

Error Detection
&
Types of errors

19010819-012

Sajjal Farooq

Introduction

- Data can be corrupted during transmission

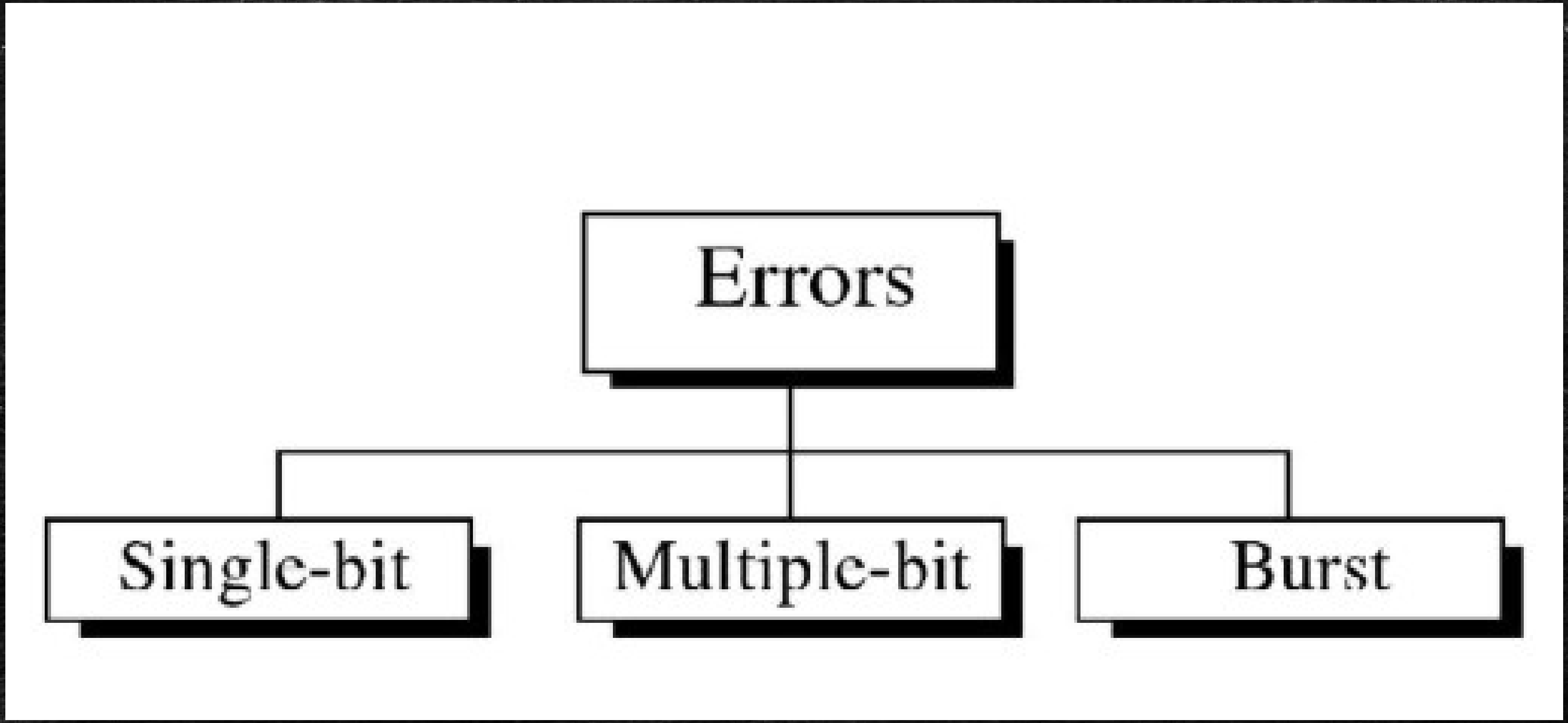
  .
- some applications require that errors be detected and corrected

What is Error

Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0

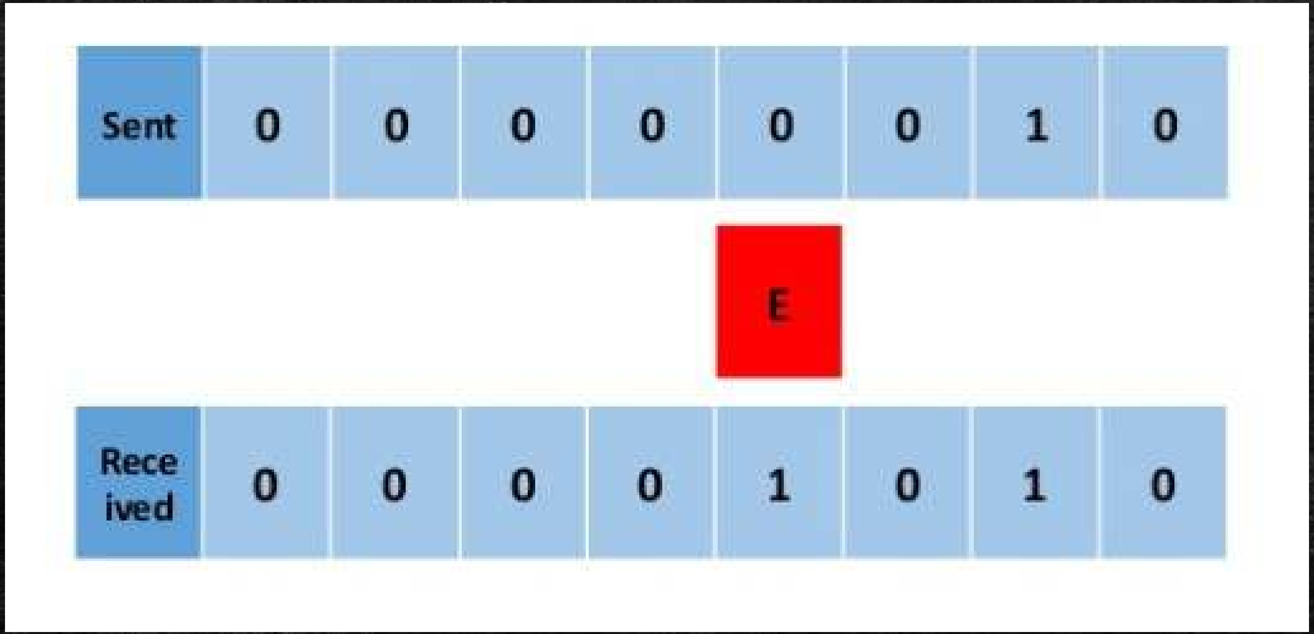
TYPES of Error

There may be three types of errors:



Single Bit Error

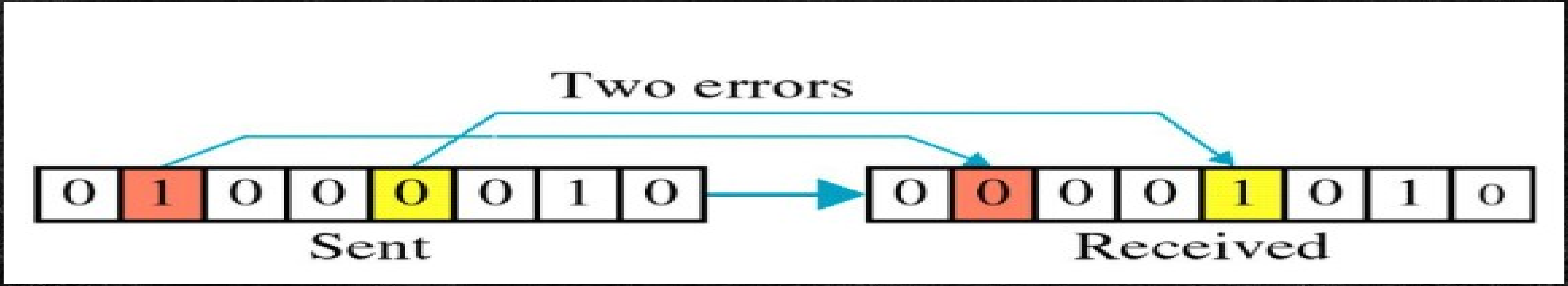
In a frame, there is only one bit, anywhere though, which is corrupt.



Single-bit Error Only 1 bit in the data unit has changed.

Multiple Bit Error

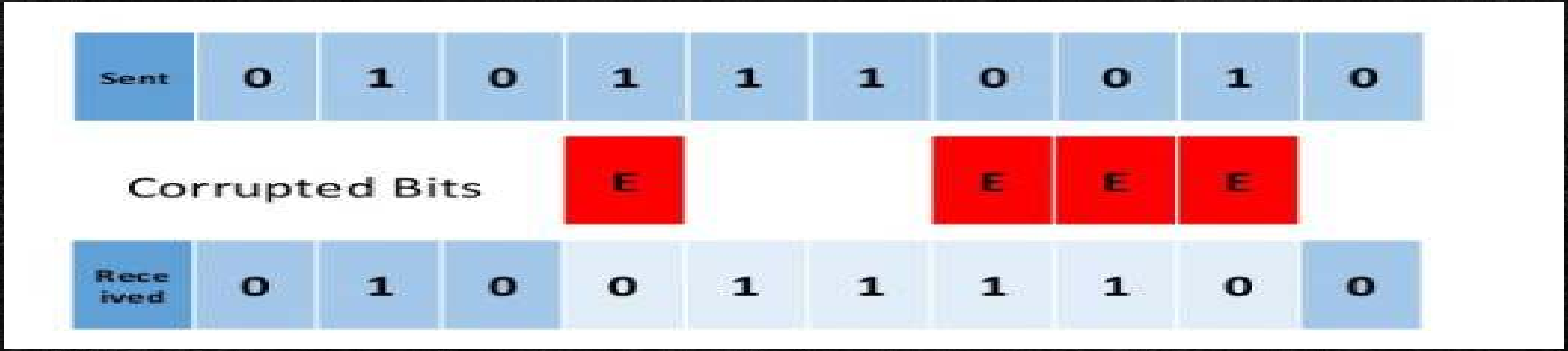
If two or more **bits** are differed between sending and receiving data stream, then the **error** is called **multiple bit error**. The position of the **bits** may differ the **bits** may not be consecutive.



Frame is received with more than one bits in corrupted state.

Burst Error

- 2 or more consecutive bits in the data unit have changed from 1 to 0 or 0 to 1 .
- The length of the burst error is measured from the first corrupted bit to the last corrupted bit.
- Some bits in between may not be corrupted.



Sent	0	1	0	1	1	1	0	0	1	0
Corrupted Bits				E			E	E	E	
Rece ived	0	1	0	0	1	1	1	1	0	0

- **Error detecting** codes are either implemented at the data link layer or transport layer of **OSI model**. Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use **error-detecting** codes which are additional data added to a given digital message to help us detect if an error has occurred during transmission of the message.

Parity Check
&
Cyclic Redundancy Check

19810819-011 Moughais ul Hassan

Parity Checking

In this an extra bit (parity bit) is added to each word before transmitting.

Even Parity:

No.1 of is in given word including parity should be even.

Odd parity:

No .1 Of is in the given word including parity should be odd.

Parity Checking

Data= 1 0 0 1 0 1 1] →

Even Parity

P=0 , | 0 | 1 0 0 1 0 1 1 |

↓

00001011

↓

Error

Receiver: No Error.

ODD Parity

p=1

| 1 | 1 0 0 1 0 1 1 |

↓

| 1 | 1 1 0 1 0 1 1 |

↓

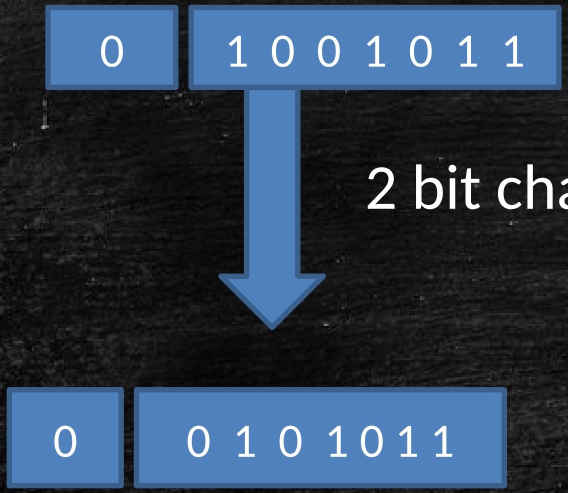
Error.

Limitation of Parity Checking

Not suitable for detection of multiple errors.

Cannot reveal the location of erroneous bit. It cannot correct the error.

Even parity:



{ No. of is=4=Even}

{No. of is=4=Even}

Case (I) One Error,Two, Three Error

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

Two DimensionalParity Checking

In this, the data-word is organized in table (rows and columns)

NOTE:

It checks and detects up to three

(3) errors. Errors affecting 4 bits may not be detected.

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

Row Parities

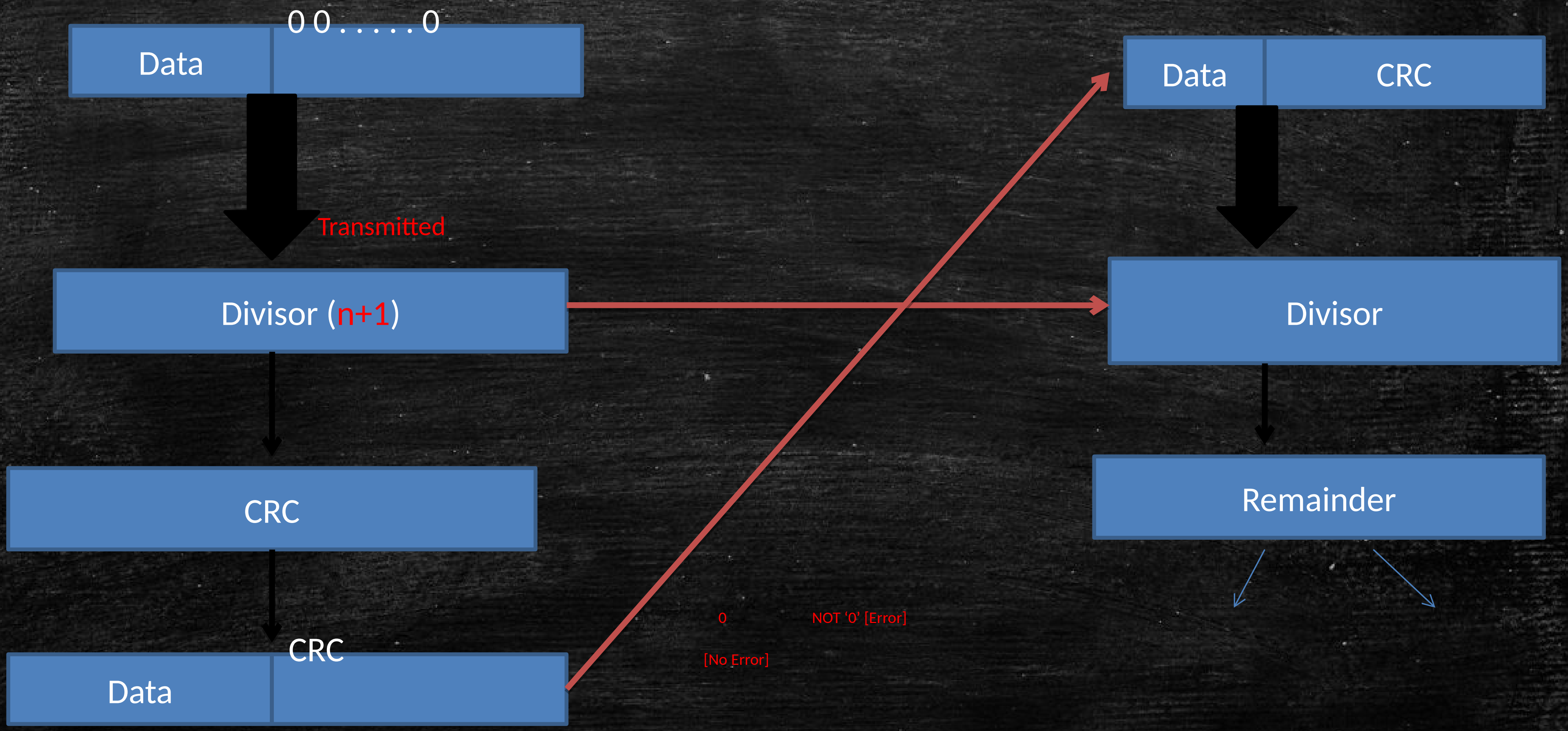
$\longrightarrow$

Column Parities

Cycle Redundancy Check

It is based on the concept of Binary Division.

- CRC Generator: (Data)
-
- Append string of n Os to the data unit.
- Divide newly generated data unit in 1 by the divisor.
- Remainder after is n bit CRC.
- The CRC will replace n Os to get codeword to be transmitted.
- CRC Checker:
- Here the receiver divides the data unit by the same divisor which was used by the transmitter. The reminder of the divisor is then checked
  -
- Remainder is ‘0’ [Accepted]
- Remainder is not ‘0’ [Rejected]


0 0 0
Data
Transmitted
Divisor (n+1)
CRC
CRC
Data
Data
CRC
Divisor
Remainder
0
[No Error]
NOT '0' [Error]

CRC Example

Dataset = 1001, Divisor = 1011.

Data = 1001

Divisor = 1011 [n+1 bits]

```
          1 0 1 0
1011  1 0 0 1 0 0 0
      1 0 1 1
        0 1 0 0
        0 0 0 0
          1 0 0 0
          1 0 1 1
            0 1 1 0         CRC
            0 0 0 0
              1 1 0   ->   Remainder
```

XOR

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0

Receiver Side

1 0 0 1 1 10

```
                1 0 1 0

1 0 1 1      1 0 0 1  1 1 0
             1 0 1 1
               0 1 0  1
               0 0 0 0
                 1 0 1 1
                 1 0 1 1
                   0 0 0 0
                   0 0 0 0
 NO                  0 0 0             Remainder
  Error
```

```
           1 0 1 1

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


          Error
```

Checksum & Flow and Error Control

19010819-013

Gulshan Fatima

What is checksum?

- Checksum is an error-detecting technique that can be applied to a message of any length.
- A checksum “adds” together “chunks” of data.
- The “add” operation may not be normal integer addition.
- The chunk size is typically 8,16, or 32 bits.
- Sender side- Checksum creation
- Receiver side- Checksum validation

Checksum- Operation at sender side

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

Checksum from sender- Example :

Consider the data unit to be transmitted is:

10011001111000100010010010000100

10011001	11100010	00100100	10000100

Checksum from sender- Example :

10011001 11100010 00100100 10000100

$$
\begin{array}{lcccccccccc}
\text{carry} & & & 1 & 1 & 1 & 1 & 1 & & & \\
 & & & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & & & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
 & & & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
 & & + & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
\hline
 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1
\end{array}
$$

Checksum from sender- Example (cont.):

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

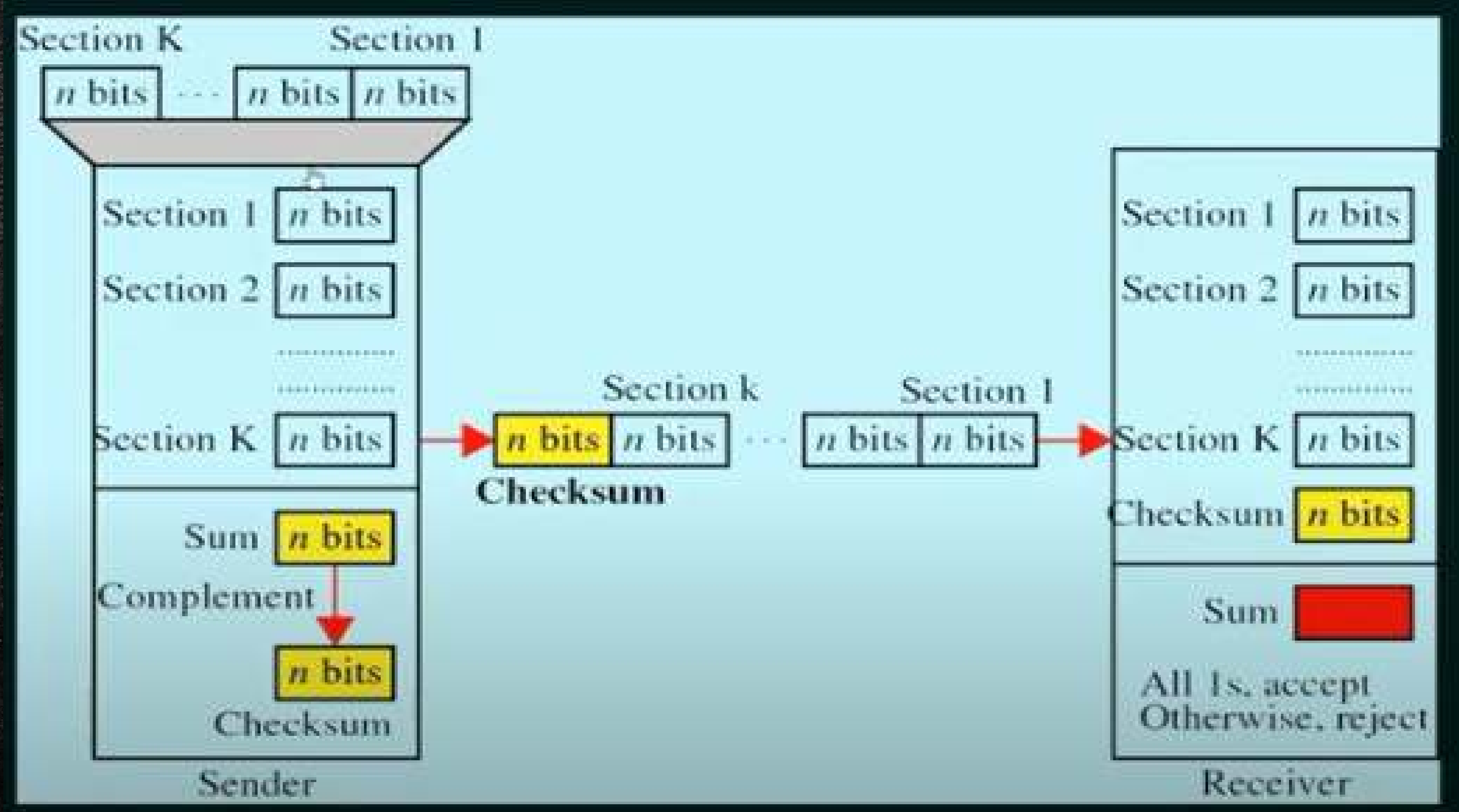
Checksum from sender- Example (cont.):

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

1's compliment

Diagram for further explanation:



Checksum- Operation at receiver side:

- Collect all the data blocks including the checksum.
- Sum all the data blocks and checksum.
- If the result is all 1’s it ACCEPT; Else REJECT.

Checksum from receiver- Example :

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	0	1	1	1	1	1	1	0	1

Checksum from receiver- Example (cont.) :

11011010 10011001 11100010 00100100 10000100

$$\begin{array}{lcccccccc}
\text{Carry} & 1 & 1 & 1 & 1 & 1 & 1 & & \\
 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
+ & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
 & & & & & & & 1 & 0 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}$$

Message accepted

Performance of checksum:

- The checksum detects all the errors involving an odd number of bits.
- It detects most errors on even number of bits.
- If one or more bits of a segment are damaged and the corresponding bit or bits of opposite vale in a second segment are also damaged, the sum of those columns will not change and the receiver will not detect the errors.

Flow control:

- Flow controls refers to a set of procedures used to restrict the amount of data that the sender can send before receiving an acknowledgement from the receiver.
- Flow control tells the sender how much data it can send without congestion the receiver.
- The receiving device has a restricted speed and capacity at which can accept the data.
- If this were exceeded ,data would be lost.

Error control:

- Error control includes both error detection and error correction.
- It allows the receiver to inform the sender if a frame is lost or damaged during transmission and coordinates the re-transmission of those frames by the sender.
- Error control in the data link layer is based on automatic repeat request (ARQ). Whenever an error is detected, specified frames are re-transmitted

Protocols of Noiseless & Noisy Channels

19010819-021

Nabeel Yousaf

Noiseless Channel

An ideal channel in which no frames are lost, duplicated or corrupted is regarded as Noiseless Channel.

There are two types of Protocols :

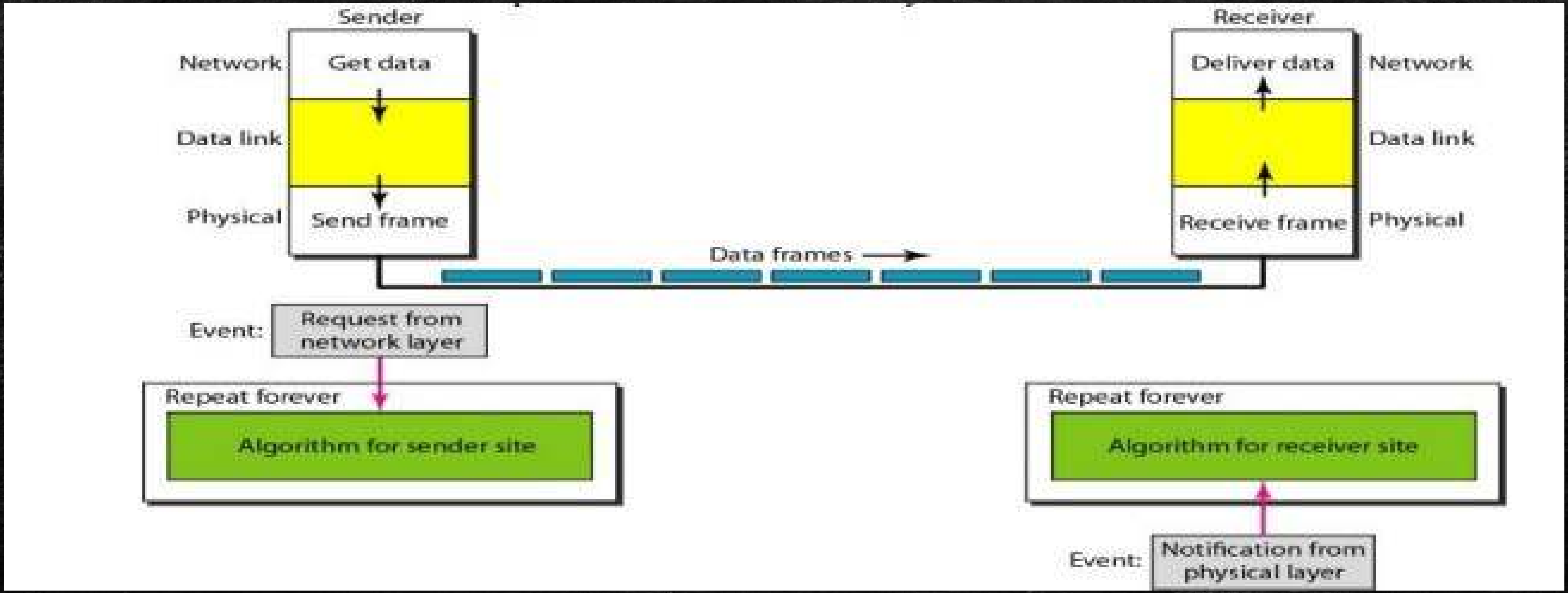
→ Simplest Protocol

→ Stop and Wait Protocol

Simplest Protocol

It has no flow or error control.

It is a unidirectional protocol in which data frames are traveling in only one direction-from the sender to receiver.



Simplest Protocol

- ✓ The sender sends a sequence of frames without even thinking about the receiver.
- ✓
- ✓ To send three frames, three events occur at the sender site and three events at the receiver site.

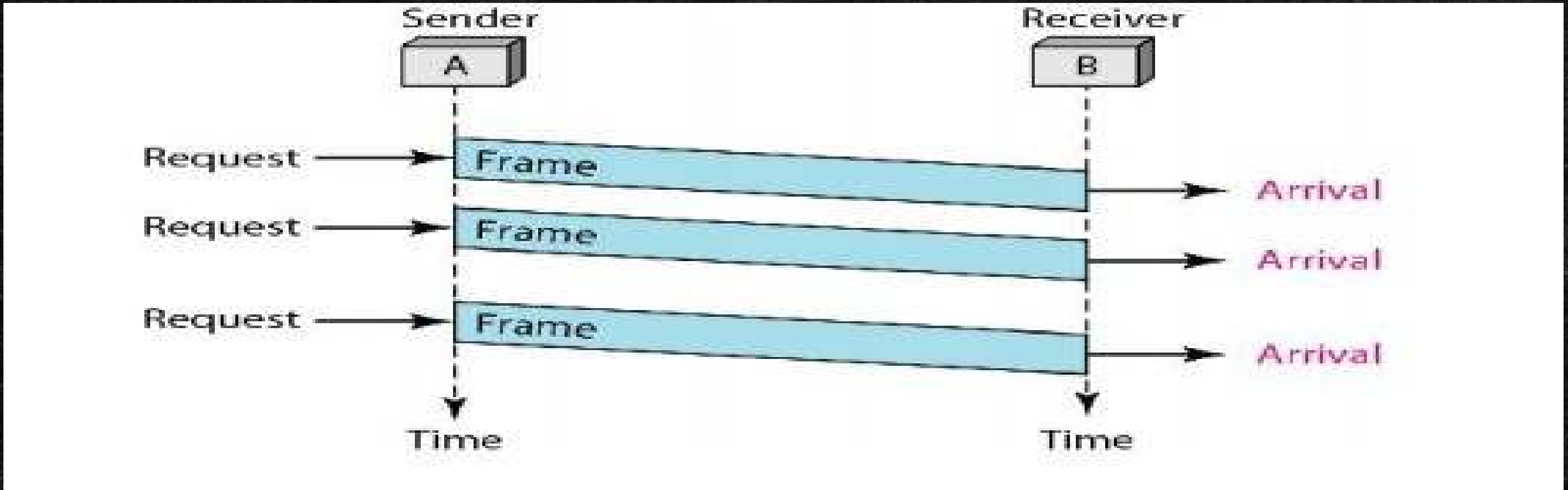
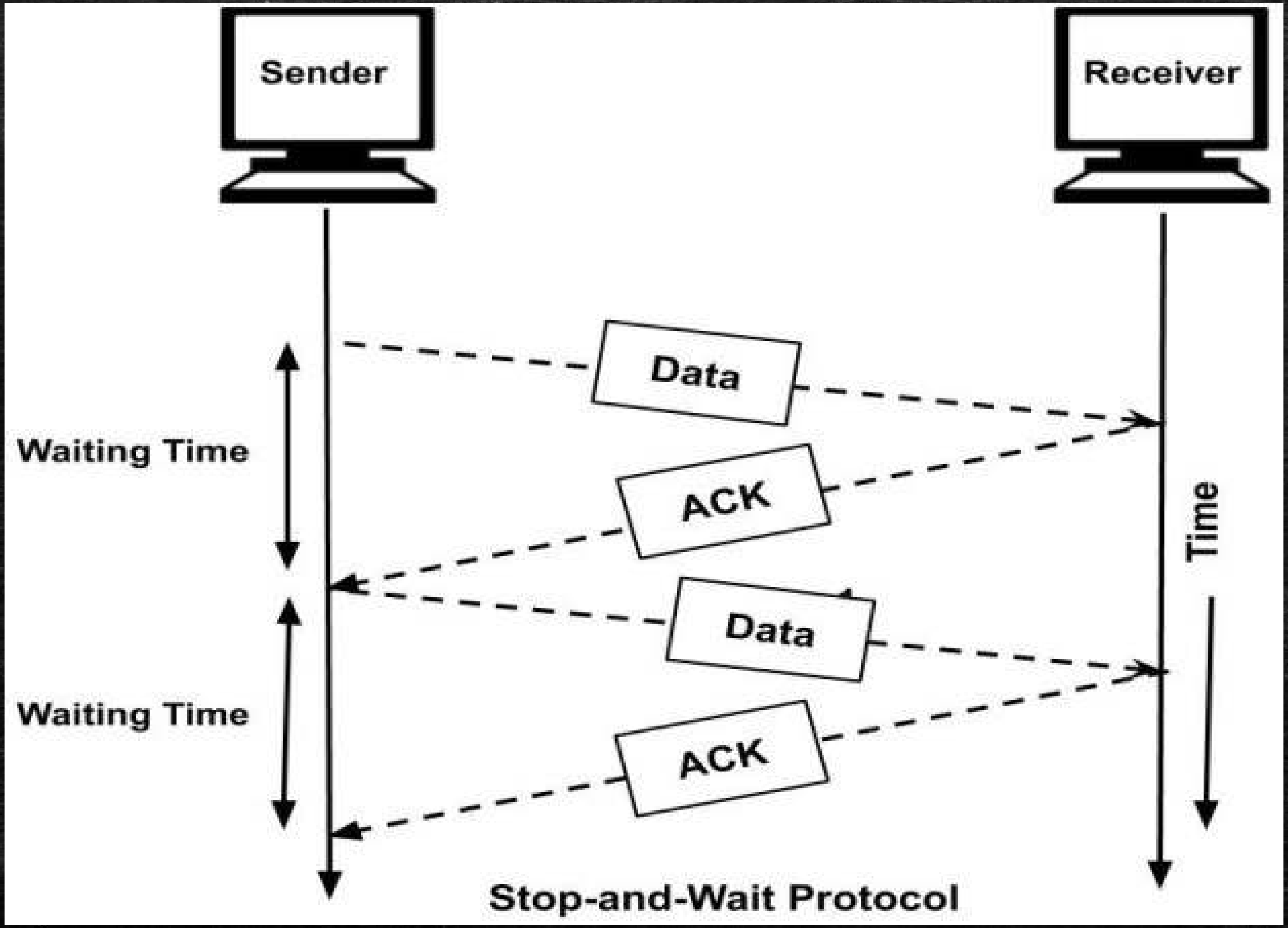


Figure 2.7 Flow diagram for Example 2.1

Stop and Wait Protocol

- ✓ It is the simplest flow control method.
- ✓ In this, the sender will send one frame at a time to the receiver.
- ✓ The sender will stop and wait for the acknowledgment from the receiver.
- ✓ This time(i.e. the time between message sending and acknowledgment receiving) is the waiting time for the sender and the sender is totally idle during this time.
- ✓ When the sender gets the acknowledgment(ACK), then it will send the next data packet to the receiver and wait for the acknowledgment again and this process will continue as long as the sender has the data to send.

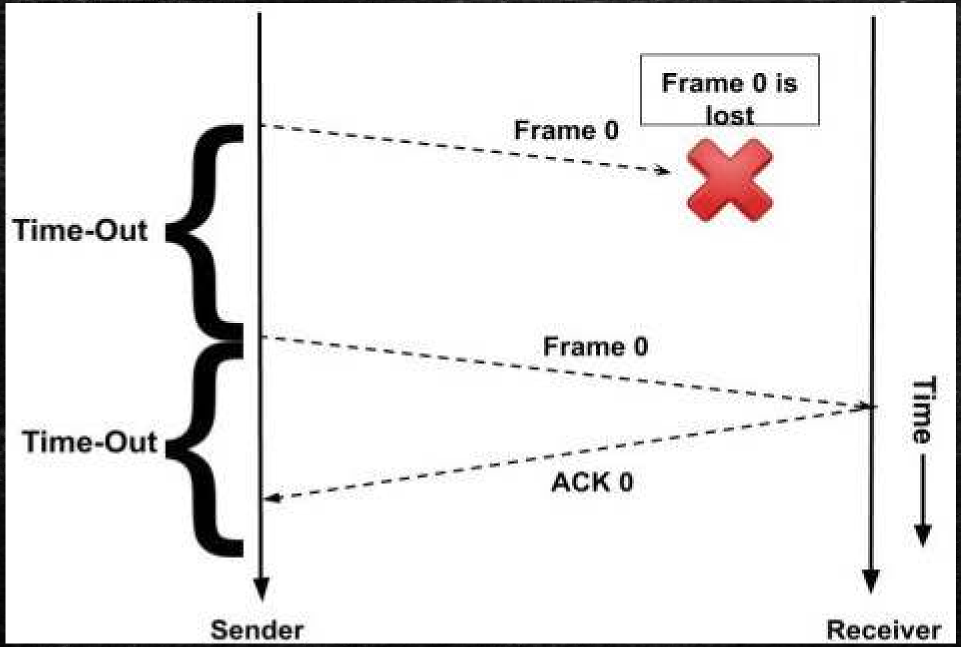
Stop and Wait Protocol (Diagram)



How Stop and Wait Protocol Responds

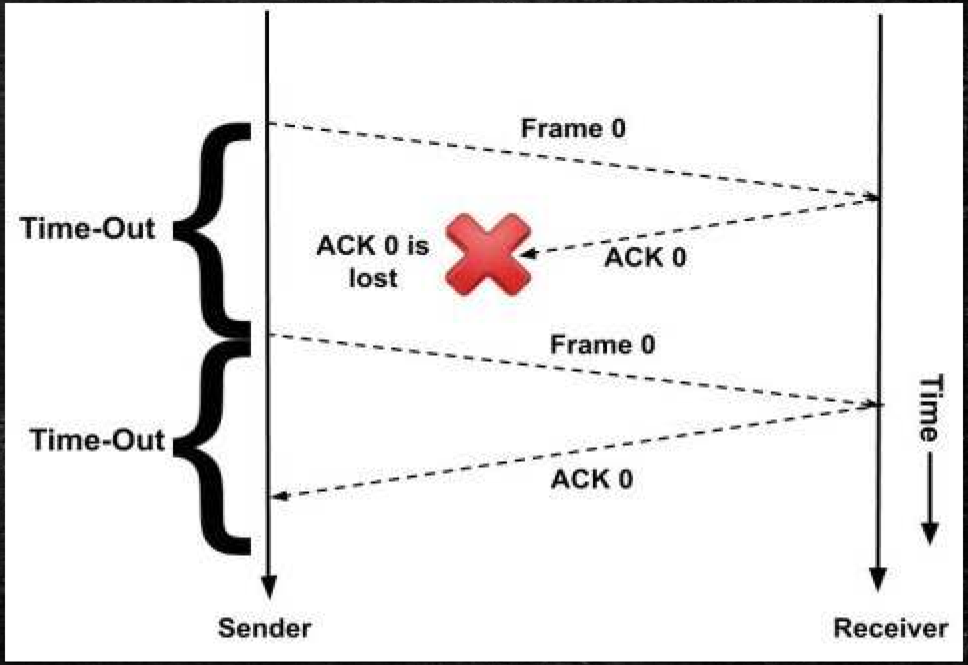
Situation 1

if any frame sent is not received by the receiver and is lost



Situation 2

if the ACK is lost



Advantages and Disadvantages

Advantages

- It is very simple to implement.

- *The main advantage of this protocol is the accuracy. The next frame is sent only when the first frame is acknowledged. So, there is no chance of any frame being lost.*

Disadvantages

- We can send only one packet at a time.
- If the distance between the sender and the receiver is large then the propagation delay would be more than the transmission delay. Hence, efficiency would become very low.
- After every transmission, the sender has to wait for the acknowledgment and this time will increase the total transmission time. This makes the transmission process slow.

Noisy Channel

Consider the normal situation of a communication channel that makes errors. Frames may be either damaged or lost completely.

There are three types of Protocols :

→ Stop and Wait Automatic Repeat Request

→ Go Back N Automatic Repeat Request

→ Selective Repeat Automatic Repeat Request

Stop and Wait ARQ

- It assumes that the communication channel is imperfect and noisy.
- Data packet sent by the sender may get corrupt.
- A negative acknowledgment is sent by the receiver if the data packet is found to be corrupt.
- Sender starts the time out timer after sending the data packet.
- Data packets and acknowledgments are numbered using sequence numbers.

Stop and Wait ARQ (Diagram)

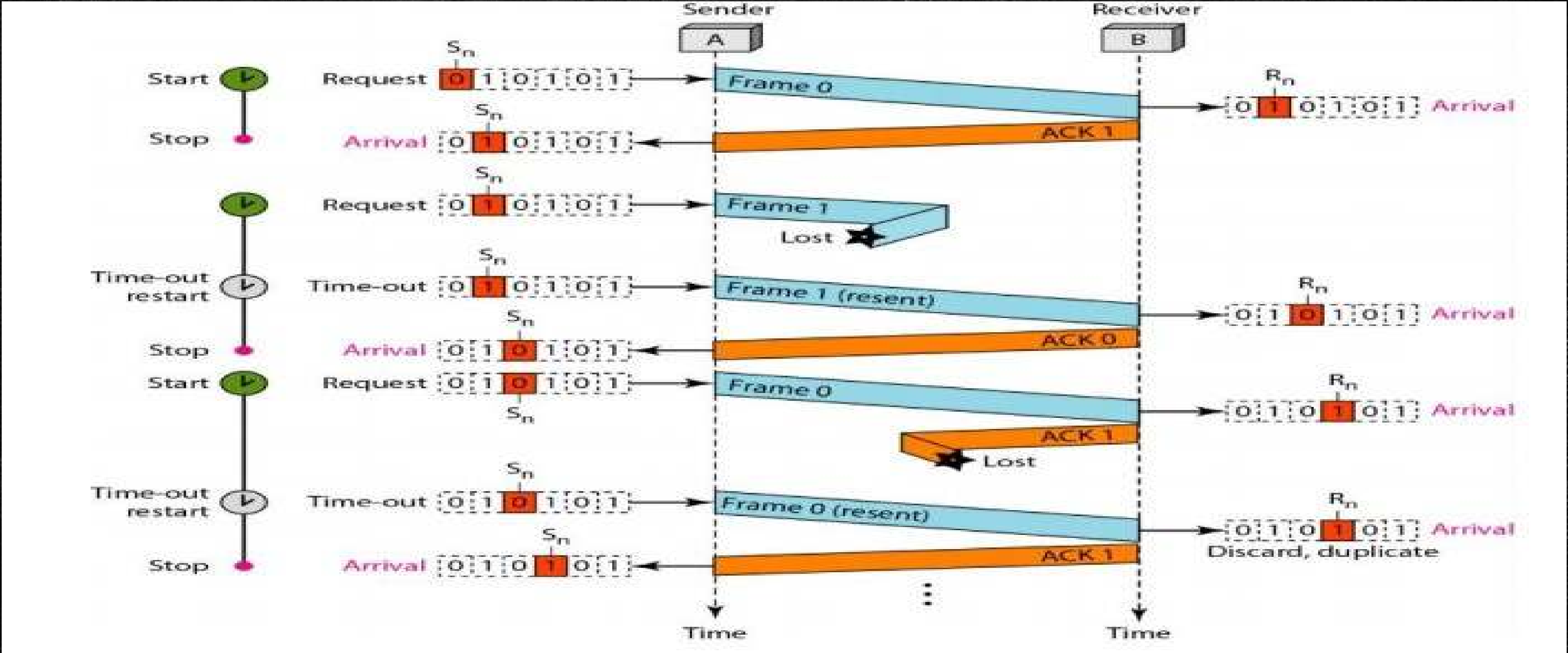


Figure 2.11 Flow diagram for Example 2.3

How Stop and Wait ARQ Solves All Problems?

- ✓ Problems of Lost Data Packet
- ✓ Problems of lost Acknowledgment
- ✓ Problem of delayed Acknowledgment
- ✓ Problem of Damaged Packets

→ Role of sequence numbers on Data Packets

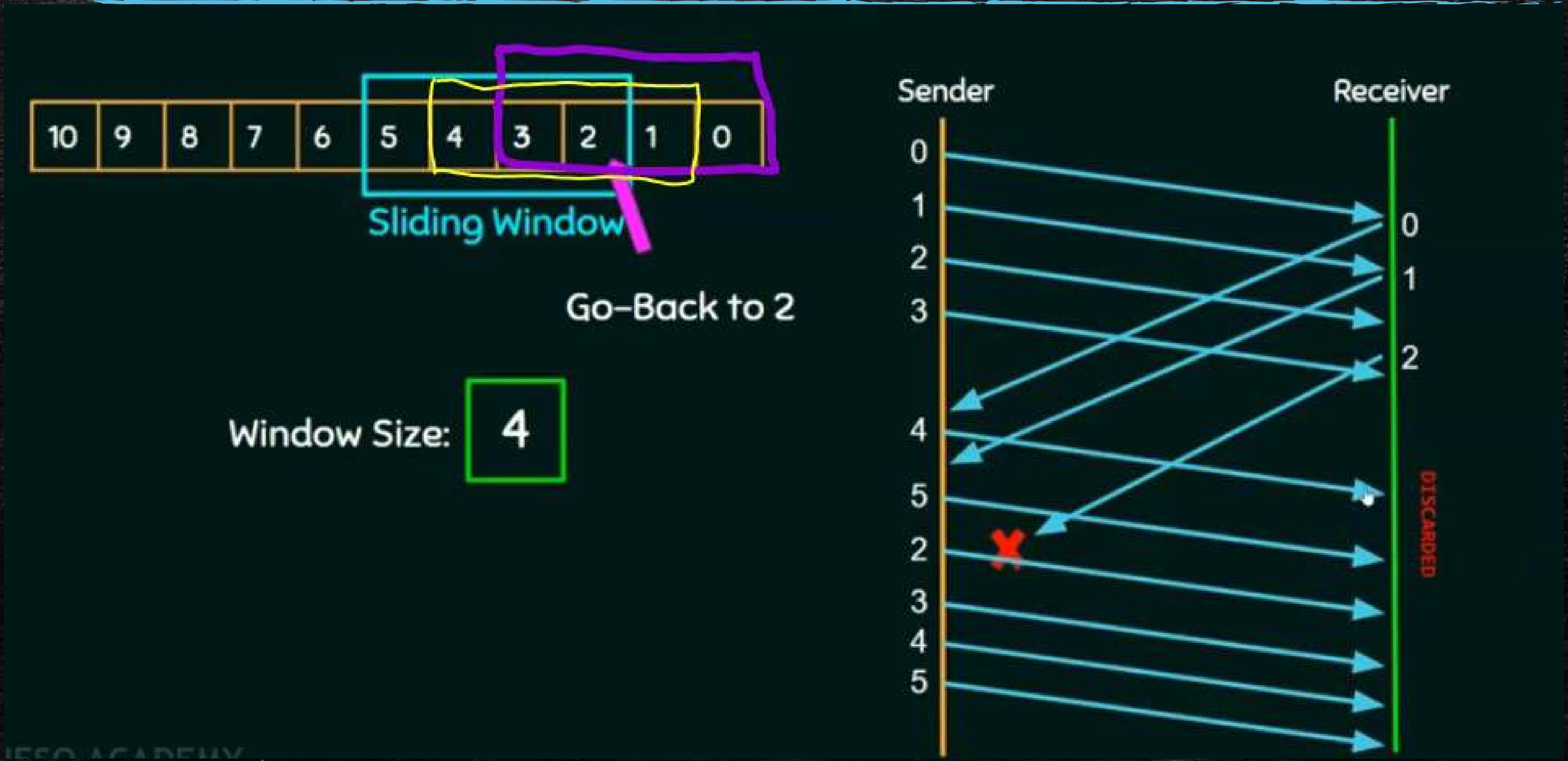
Limitation of Stop and Wait ARQ

- ✓ It has very less efficiency
- ✓ Sender window size is 1
- ✓ Means Sender sends one frame and then waits until the frame gets acknowledged

Go-Back-N ARQ

- ✓ Is a one of the implementation of sliding window protocol
- ✓ It allows the sender to send multiple frames before needing the acknowledgments.(protocol pipe-lining)
- ✓ Sender slides its window on receiving the acknowledgments for the sent frames.
- ✓ This allows the sender to send more frames.
- ✓ Here ‘N’ the sender windows size.
  → for example: If N=3, the 3 frames will be sent before expecting acknowledgment.
- ✓ There are finite number of frames and are numbered in sequential manner.
- ✓ **If the acknowledgment of frame is not received within an agreed time period, all frames in the current window are re-transmitted.**

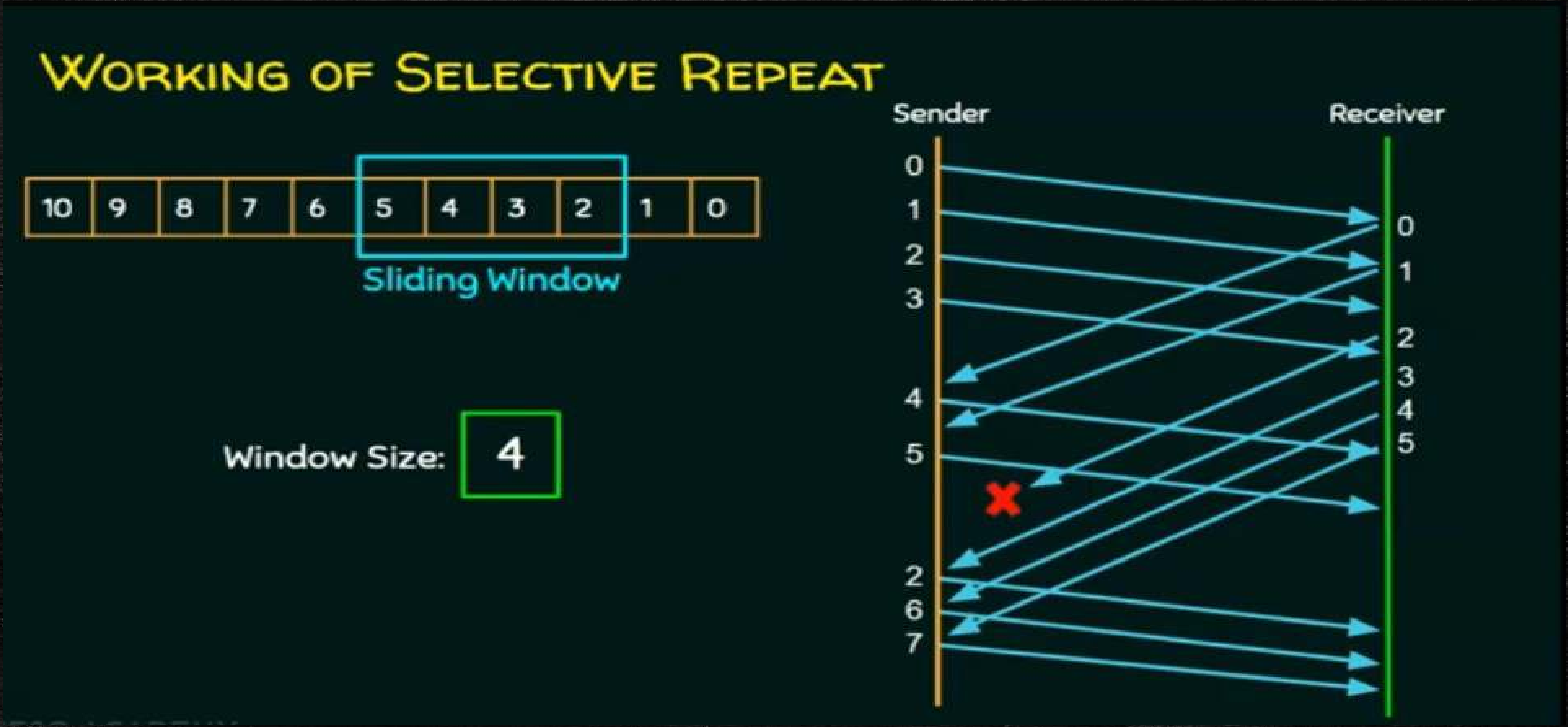
Working of Go-Back-N ARQ



Selective Repeat ARQ

- This is another implementation of Sliding Window protocol.
- In this only erroneous or lost frames are re-transmitted, while correct frames are received and buffered.
- The receiver while keeping track of sequence number, buffers the frames in the memory and sends NACK (negative acknowledgment) or no acknowledgment for only frame which is missing or damaged.
- The sender will send/re-transmit packet for which NACK is received.

Working of Selective Repeat ARQ



WORKING OF SELECTIVE REPEAT
10 9 8 7 6 5 4 3 2 1 0
Sliding Window
Window Size: 4
Sender
Receiver
0
1
2
3
4
5
2
6
7
0
1
2
3
4
5