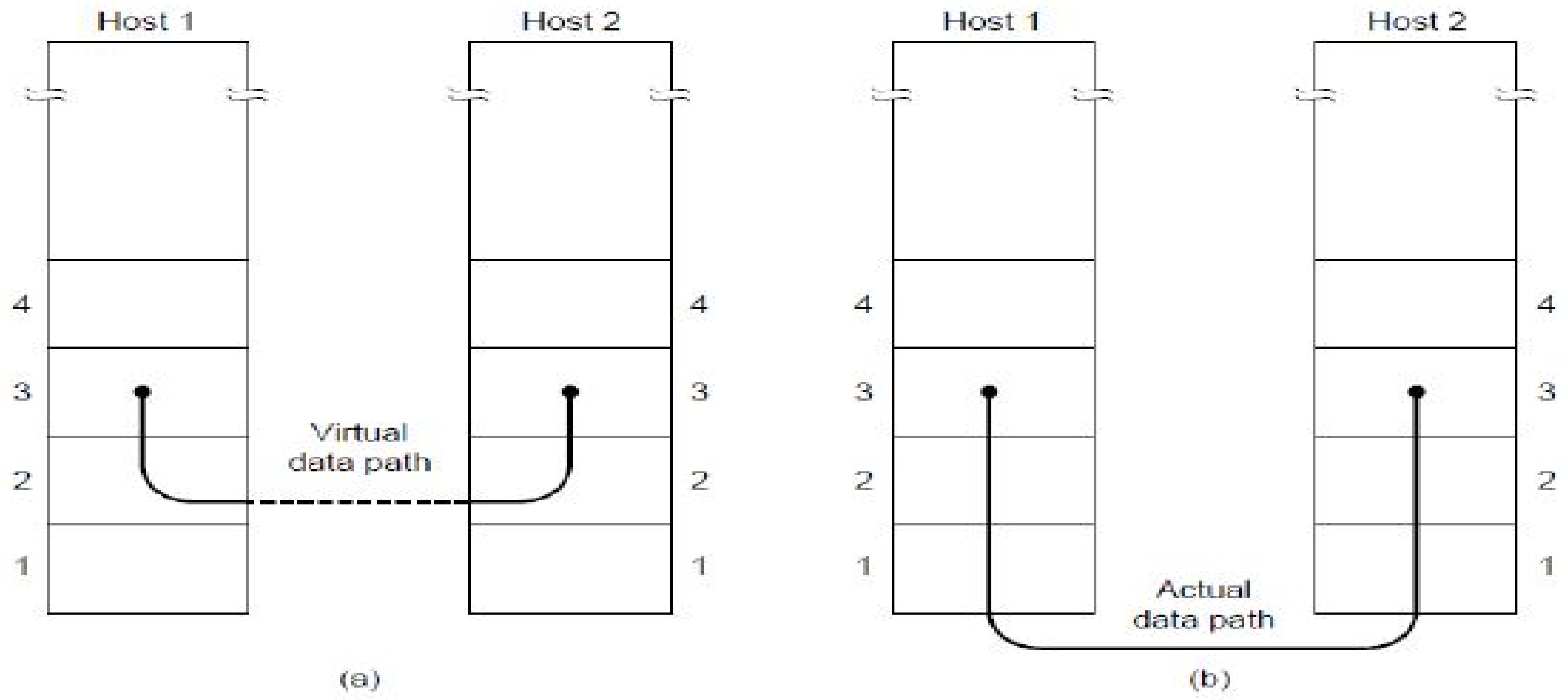
The Data Link Layer

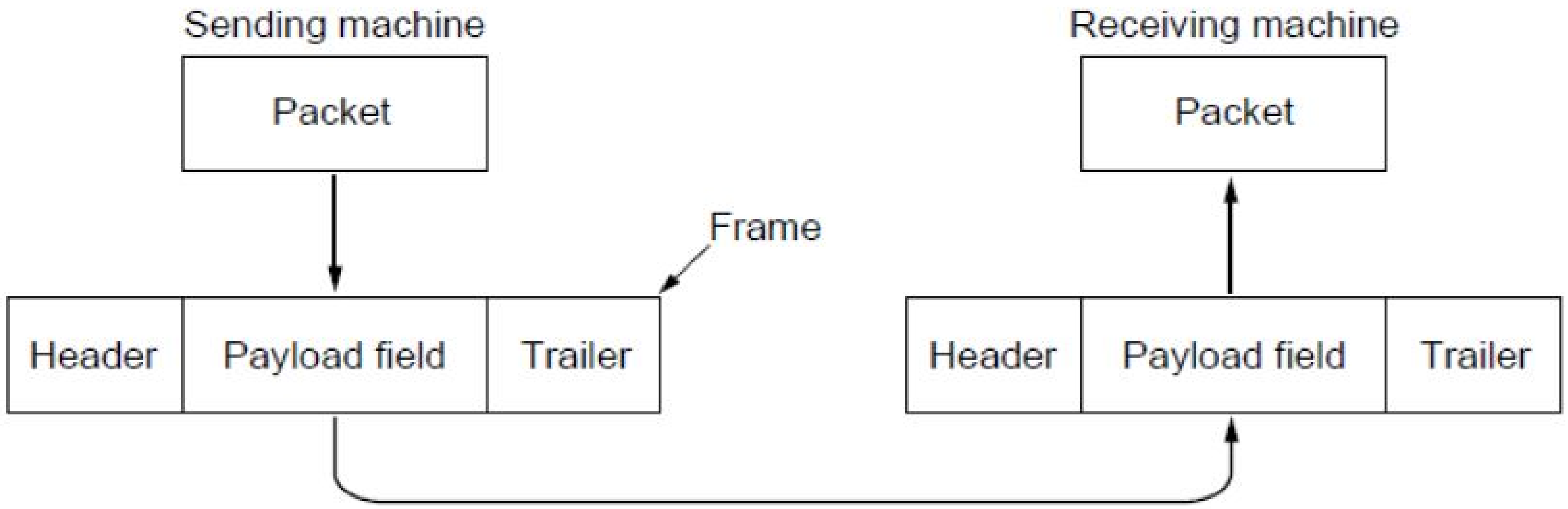
The Data Link Layer and design issue

- Data Link Layer is **second layer** of OSI Layered Model.
- Data link layer works between ***two hosts*** which are ***directly connected*** in some sense. This direct connection could be point to point or broadcast.
- Data link layer is responsible for converting ***data stream to signals*** **bit by bit** and to send that over the underlying hardware. At the receiving end, Data link layer picks up data from hardware which are in the form of electrical signals, assembles them in a recognizable frame format, and hands over to upper layer.

Network Layer Services



(a) Virtual communication. (b) Actual communication.



Relationship between packets and frames.

Data link layer has two sub-layers:

- **Logical Link Control**: It deals with ***protocols, flow-control***, and ***error control***

- **Media Access Control**: It deals with actual ***control of media.***

Functionality of Data-link Layer

- **Framing**: Data-link layer takes packets from Network Layer and encapsulates them into Frames. Then, it sends ***each frame*** bit-by-bit on the hardware. At receiver' end, data link layer picks up signals from hardware and assembles them into frames.

- **Addressing**: Data-link layer provides *layer-2 hardware addressing mechanism.* Hardware address is assumed to be unique on the link. It is ***encoded into hardware at the time of manufacturing*** (i.e. MAC Address - Media Access Control address).

- **Synchronization:** When data frames are sent on the link, *both machines must be synchronized in order to transfer to take place.*

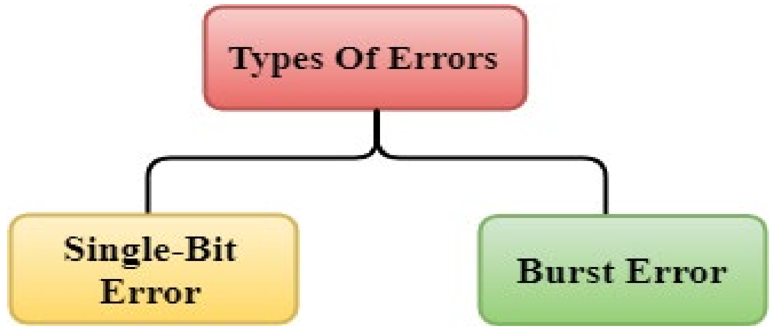
- **Error Control**: Sometimes signals may have encountered problem in transition and the bits are flipped. These *errors are detected* and ***attempted to recover actual data bits***. It also provides *error reporting mechanism to the sender.*

- **Flow Control**: Stations on same link may *have different speed* or *capacity*. Data-link layer ensures flow control that ***enables both machine to exchange data on same speed.***

- **Multi-Access**: When host on the shared link tries to transfer the data, it has a high probability of collision. *Data-link layer provides mechanism such as CSMA/CD (*Carrier-sense multiple access with collision detection*) to equip capability of accessing a shared media among multiple Systems.*

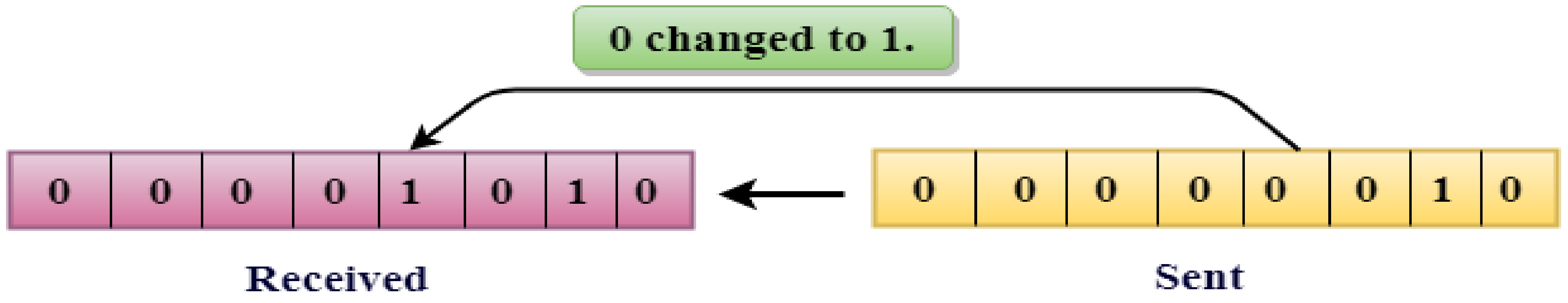
Error Detection and Correction

- 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.
- Errors can be classified into two categories:
  - Single-Bit Error
  - Burst Error



Single-Bit Error:

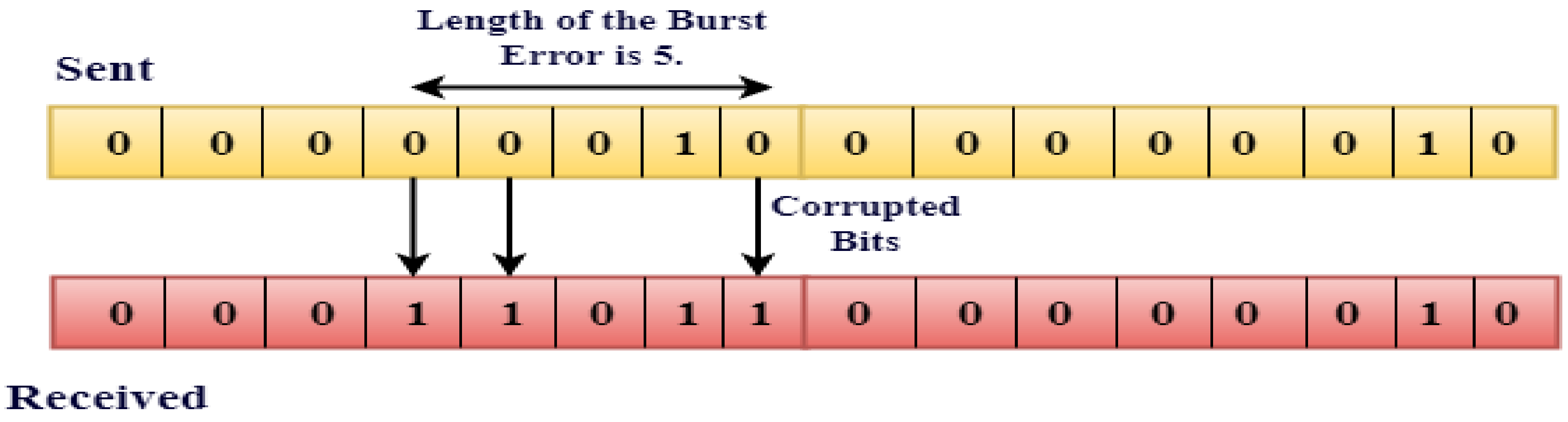
- The only one bit of a given data unit is changed from 1 to 0 or from 0 to 1.



In the above figure, the message which is sent is corrupted as single-bit, i.e., 0 bit is changed to 1.

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.*



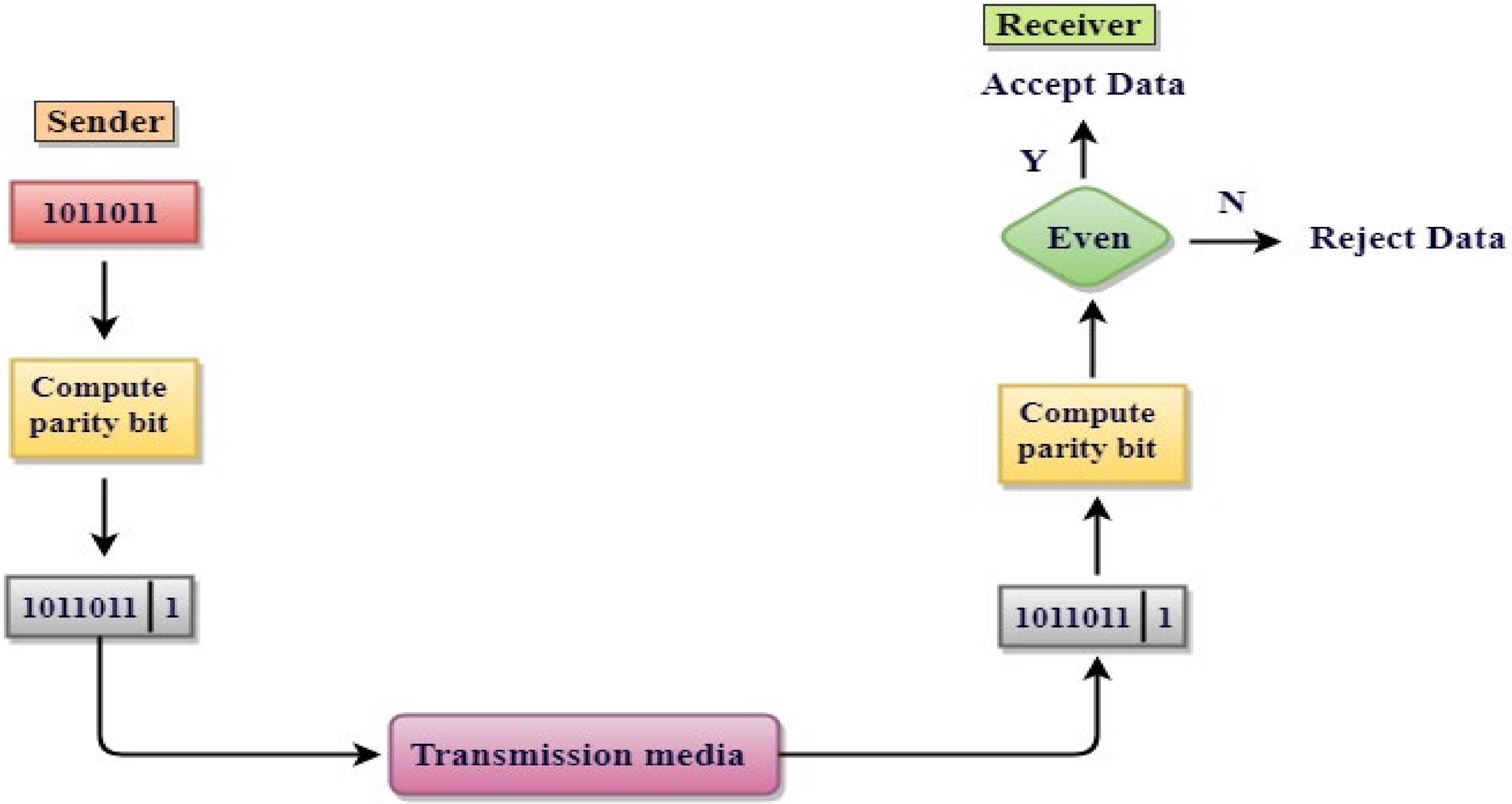
Error Detecting Techniques:

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 one bit more.
- 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***.


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

Sender Side

1's even parity

Data				Appended Bit (Parity)	Sent Data
1	0	1	0	0	10100
1	1	1	0	1	11101

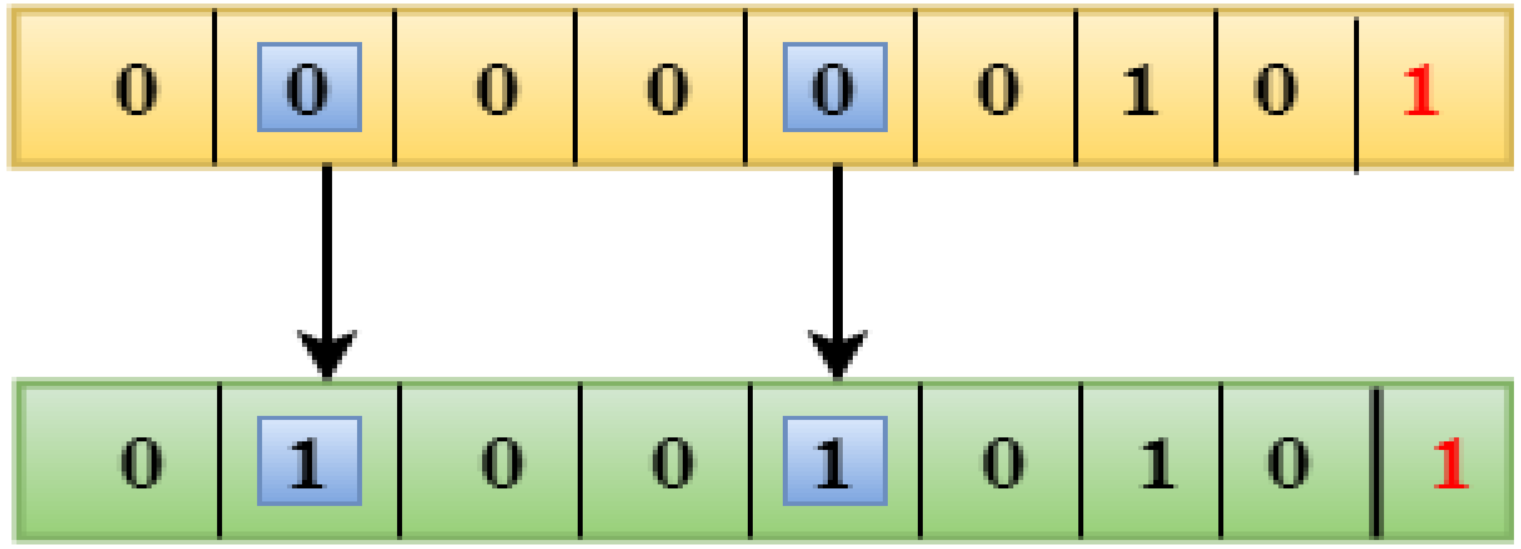
Receiver Side

1's even parity

	Data	1's number				Final data
	10100	2	even	accept the data	Drop Parity Bit/ last bit	1010
	11101	4	even	accept the data		1110
Changed data	11001	3	odd	discard the data		
	10001	2	even	accept		

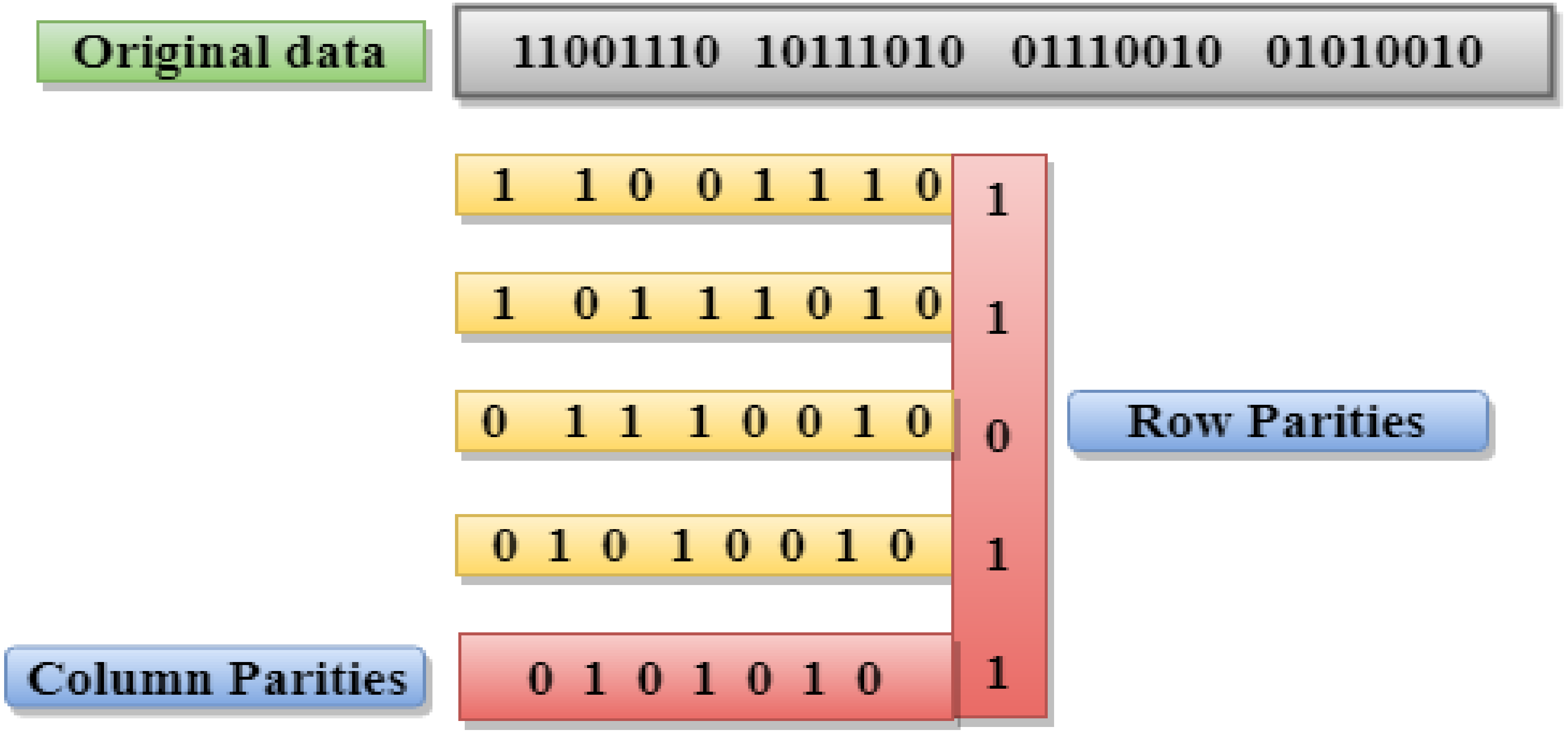
Drawbacks Of Single Parity Checking

- 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 0 1 1 1 0 1 0
0 1 1 1 0 0 1 0
0 1 0 1 0 0 1 0
1
1
0
1
1
Row Parities
Column Parities
0 1 0 1 0 1 0

2-D Parity Check

Sender Side

Data 111 101 001 9 length

111 101 001

Matrix 3 * 3

1's even apprity application

1	1	1	1	even
1	0	1	0	even
0	0	1	1	even
0	1	1	0	even
even	even	even	even	

sent data 1111 1010 OO11 O110 16 -length

Receiver

2_D Parity Check

1's even apprity application

Received data 1111 1010 0011 0110

1	1	1	1	even
1	0	1	0	even
0	0	1	1	even
0	1	1	0	even
even	even	even	even	even

data changed

1	**0**	1	1	odd
1	0	**0**	0	odd
0	0	1	1	even
0	1	1	0	even
even	odd	odd	even	even

Actual Data **111 101 001**

Drawback

1	**0**	**0**	1	even
1	0	1	0	even
0	**1**	**0**	1	even
0	1	1	0	even
even	even	even	even	even

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.

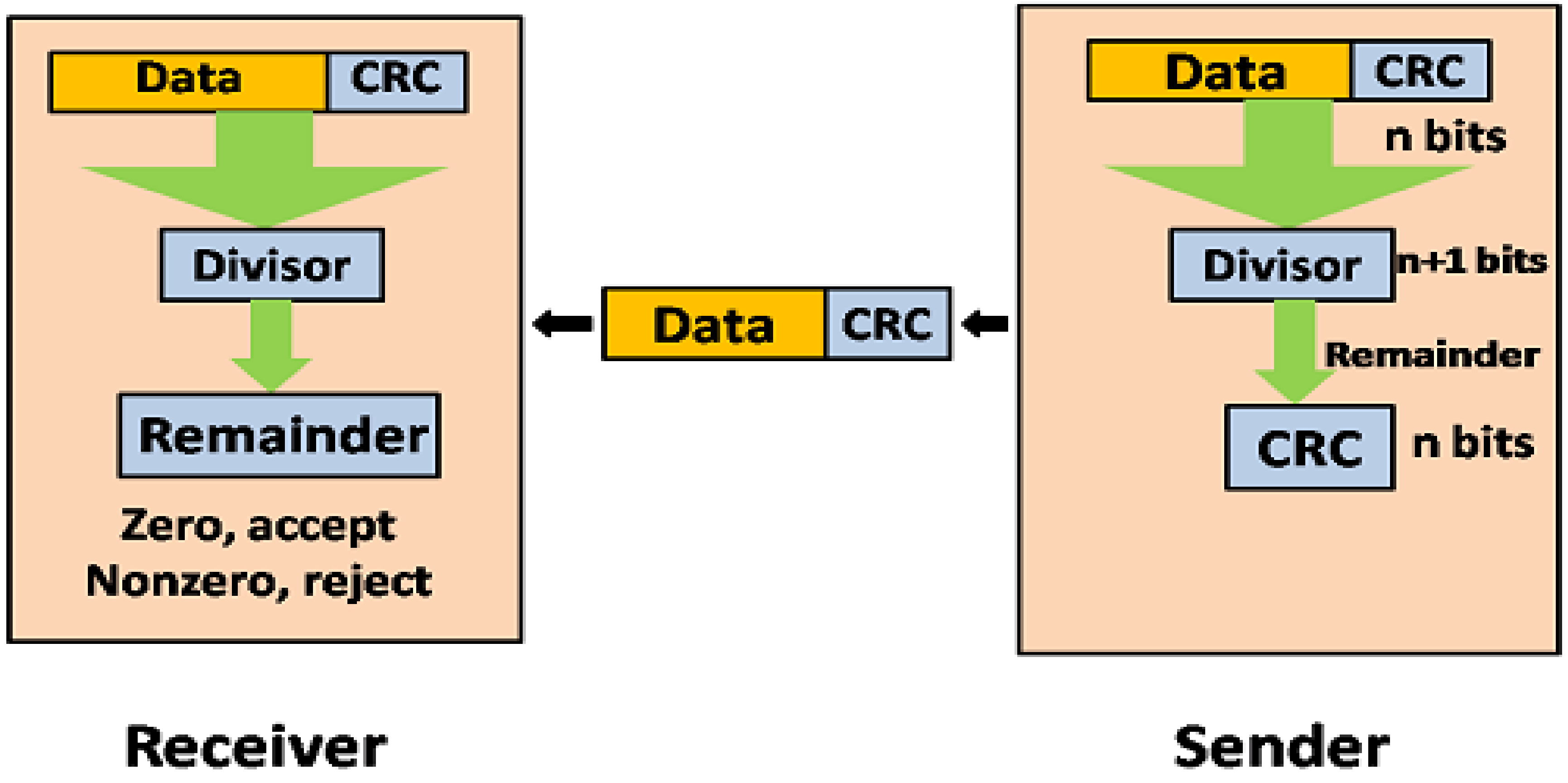
Cyclic Redundancy Check (CRC)

- 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.

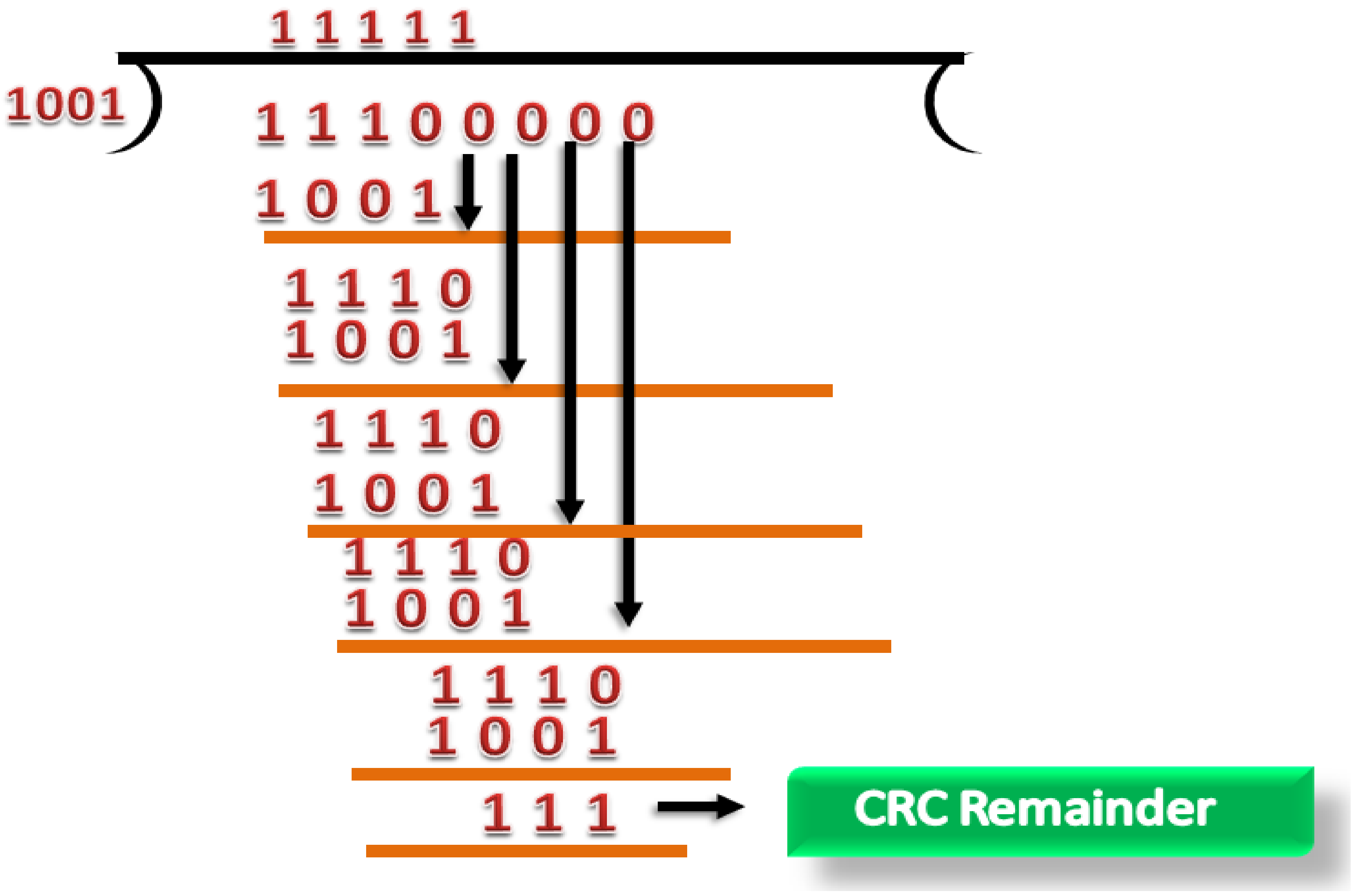
- If the ***resultant of this division is zero*** which means that ***it has no error, and the data is accepted.***
- If the resultant of this division is not zero which means that the data consists of an error. Therefore, the data is discarded.



Suppose the original data is 11100 and divisor is 1001.

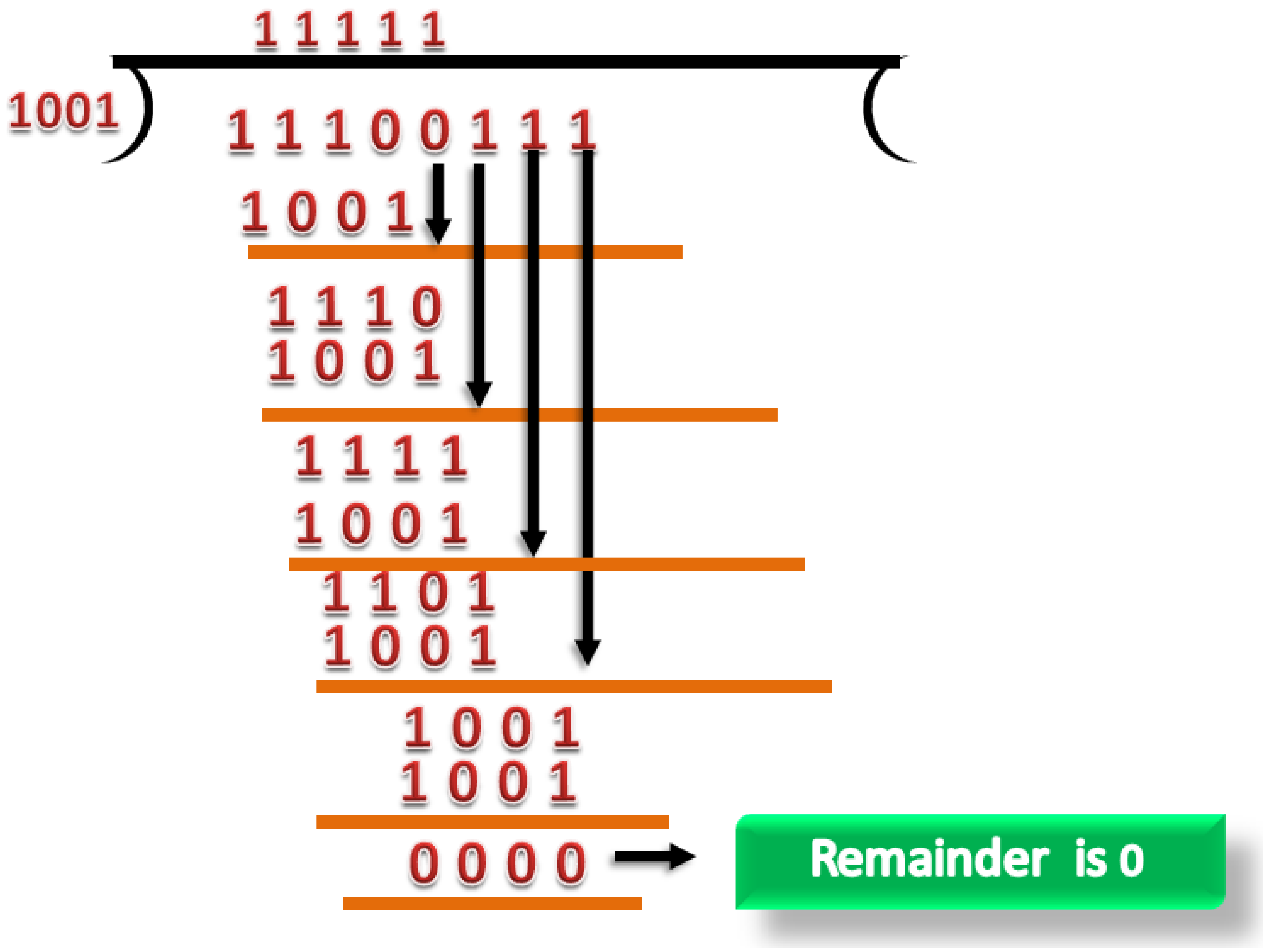
CRC Generator

- 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 **11100 000**, 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**.

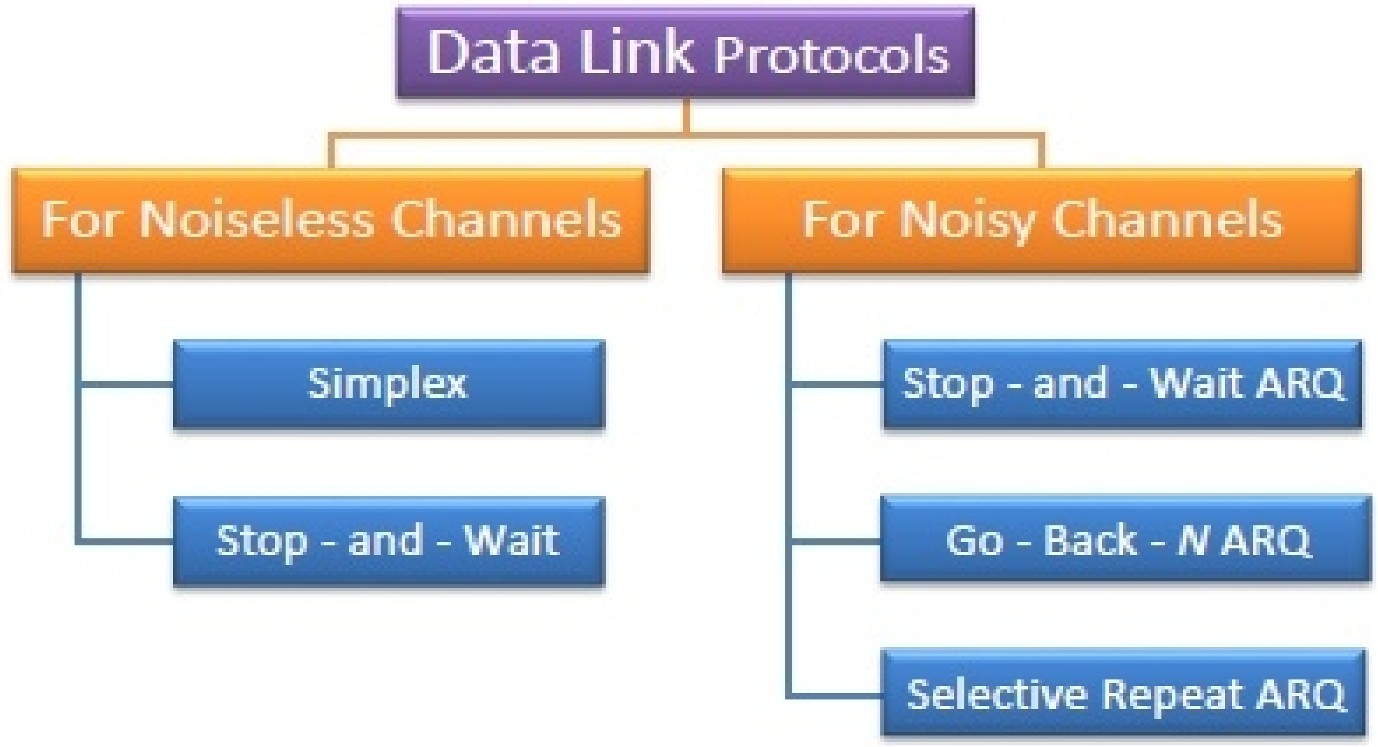

1 1 1 1 1
1001
1 1 1 0 0 1 1 1
1 0 0 1
1 1 1 0
1 0 0 1
1 1 1 1
1 0 0 1
1 1 0 1
1 0 0 1
1 0 0 1
1 0 0 1
0 0 0 0
Remainder is 0

Datalink Protocols

- Protocols in the data link layer are designed so that this layer can perform its basic functions: **framing, error control and flow control**.
- **Framing** is the process of *dividing bit* - streams from physical layer into data frames whose size ranges from a few hundred to a few thousand bytes.
- **Error Control** mechanisms deals with *transmission errors* and *retransmission of corrupted and lost frames*.
- **Flow control** regulates *speed of delivery* and so that a fast sender does not drown a slow receiver.

Types of Data Link Protocols

Data link protocols can be broadly divided into two categories, depending on whether the transmission channel is noiseless or noisy.



Simplex Protocol

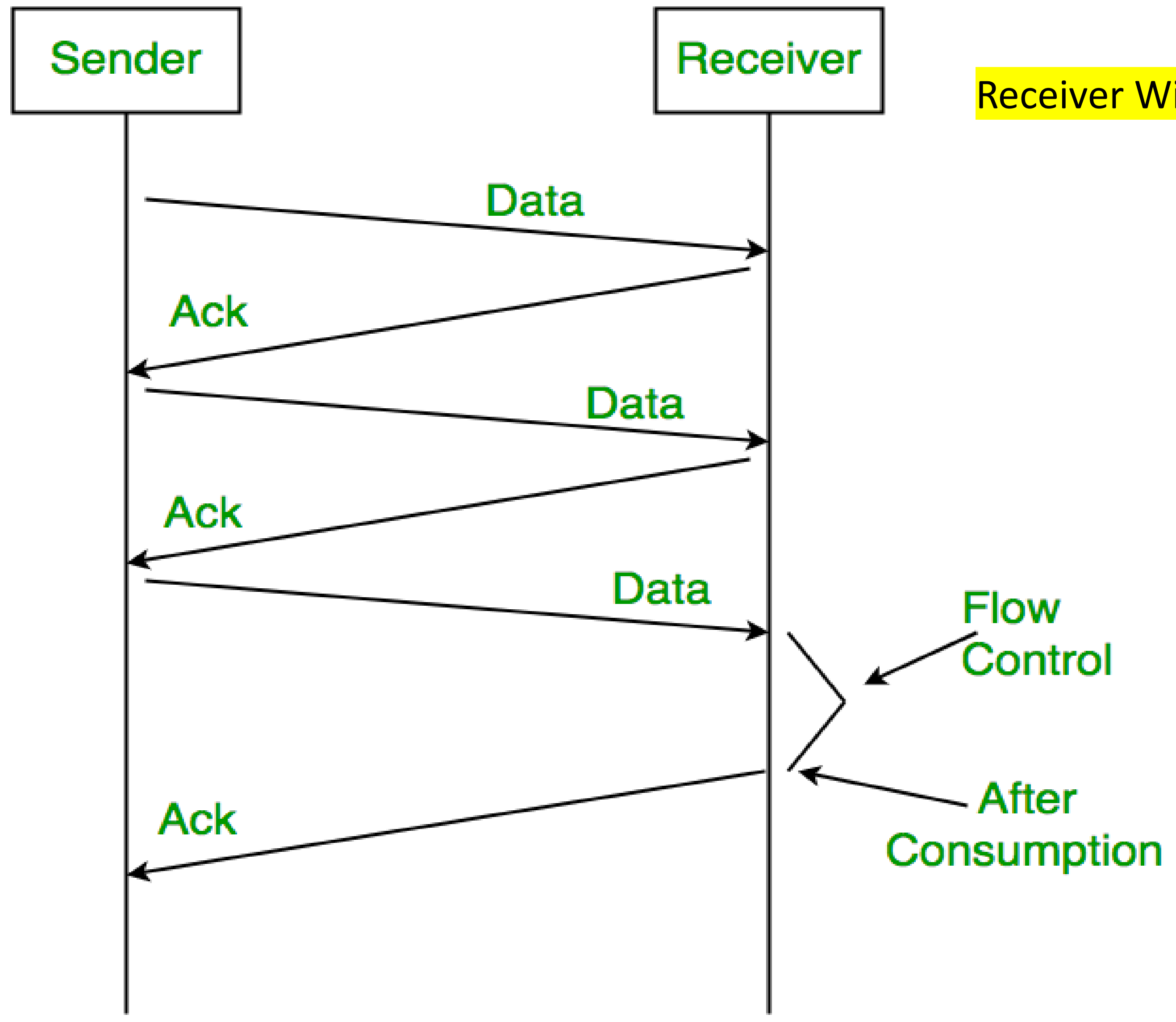
- The Simplex protocol is **hypothetical protocol** designed for **unidirectional data transmission** over an ideal channel, i.e. a channel through which transmission can never go wrong.
- It has distinct procedures for sender and receiver.
- The sender simply sends all its data available onto the channel as soon as they are available its buffer.
- The receiver is assumed to process all incoming data instantly.
- It is *hypothetical since it does not handle flow control or error control*.
- **Sender Windows Size = Receiver Windows Size = Unlimited Size**

Stop – and – Wait Protocol

- Stop – and – Wait protocol is for ***noiseless*** channel too.
- It provides unidirectional data transmission *without any error control facilities.*
- However, ***it provides for flow control*** so that a fast sender does not drown a slow receiver.
- The receiver has a *finite buffer size* with *finite processing speed*.
- The sender can send a frame only when it has received indication from the receiver that it is available for further data processing.
- **Sender Windows Size = Receiver Windows Size = 1**

Sender Windows (N) = 1

Receiver Windows = 1



Advantage of Stop-and-wait

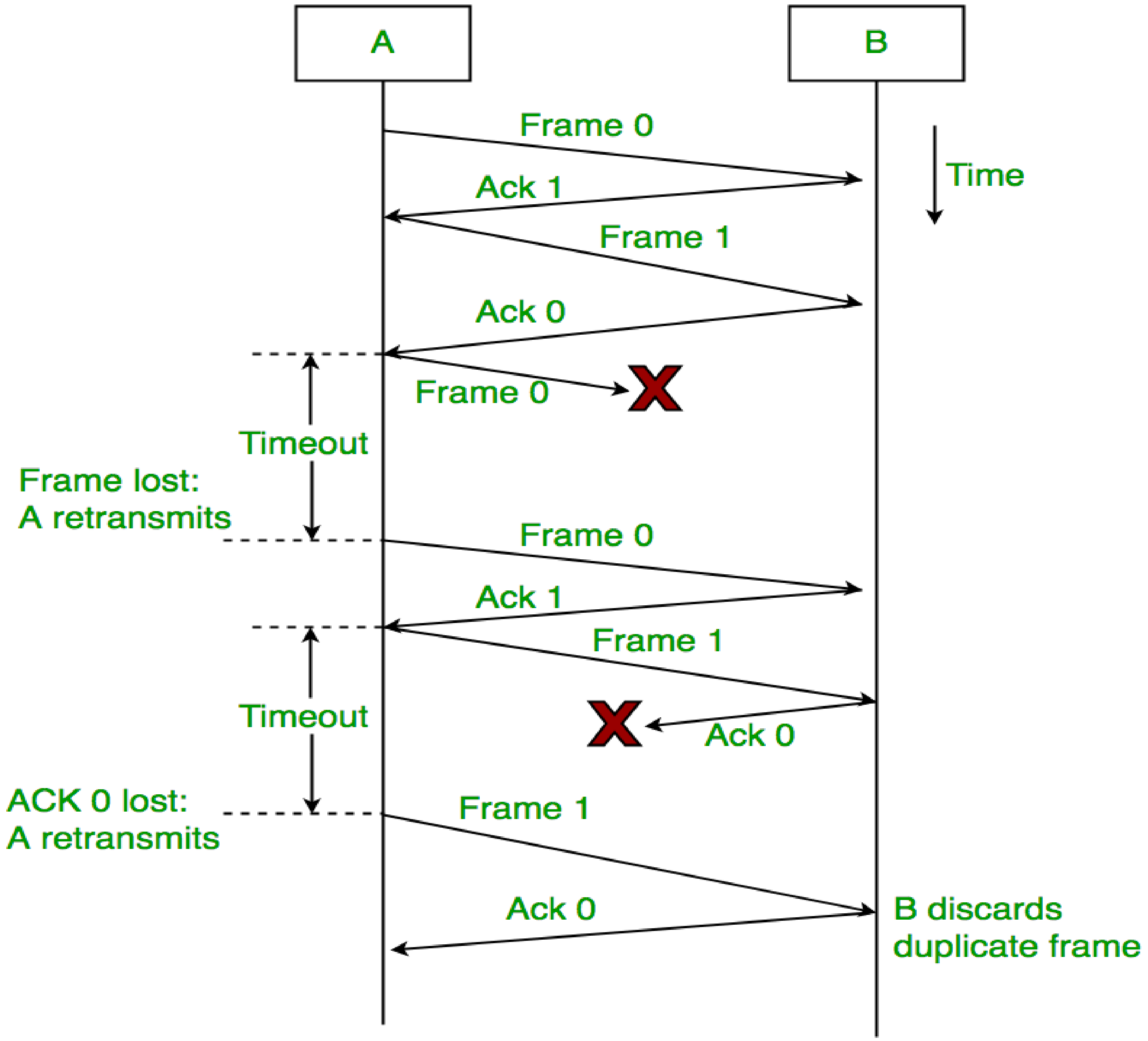
- The Stop-and-wait method is simple as *each frame is checked* and *acknowledged before the next frame is sent.*

Disadvantage of Stop-and-wait

- Stop-and-wait technique is *inefficient* to use as *each frame must travel across all the way to the receiver, and an acknowledgement travels all the way before the next frame is sent.*
- Each frame sent and received *uses the entire time needed to traverse the link.*

Stop – and – Wait ARQ

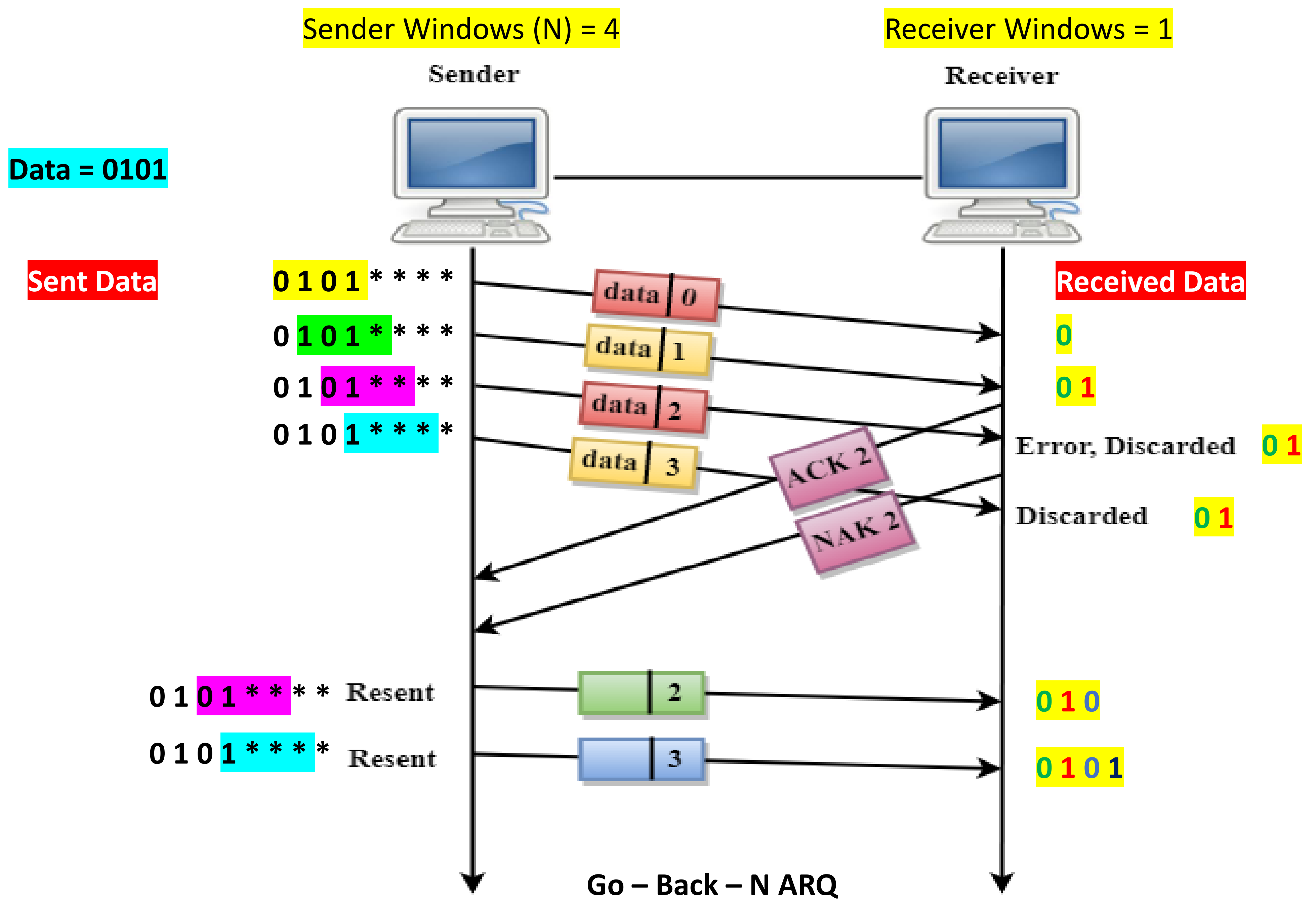
- Stop – and – wait Automatic Repeat Request (Stop – and – Wait ARQ) is a variation of the above protocol with ***added error control mechanisms,*** appropriate for noisy channels.
- The sender *keeps a copy of the sent frame*.
- It then *waits for a finite time* to *receive a positive acknowledgement from receiver.*
- If the *timer expires* or a *negative acknowledgement is received*, the ***frame is retransmitted***.
- If a *positive acknowledgement is received* then the ***next frame is sent***.
- There is only **one bit sequence number** that *implies that both sender and receiver* have buffer for one frame or packet only.


A
B
Frame 0
Time
Ack 1
Frame 1
Ack 0
Frame 0
Timeout
Frame lost:
A retransmits
Frame 0
Ack 1
Frame 1
Timeout
Ack 0
ACK 0 lost:
A retransmits
Frame 1
Ack 0
B discards
duplicate frame

Go – Back – N ARQ

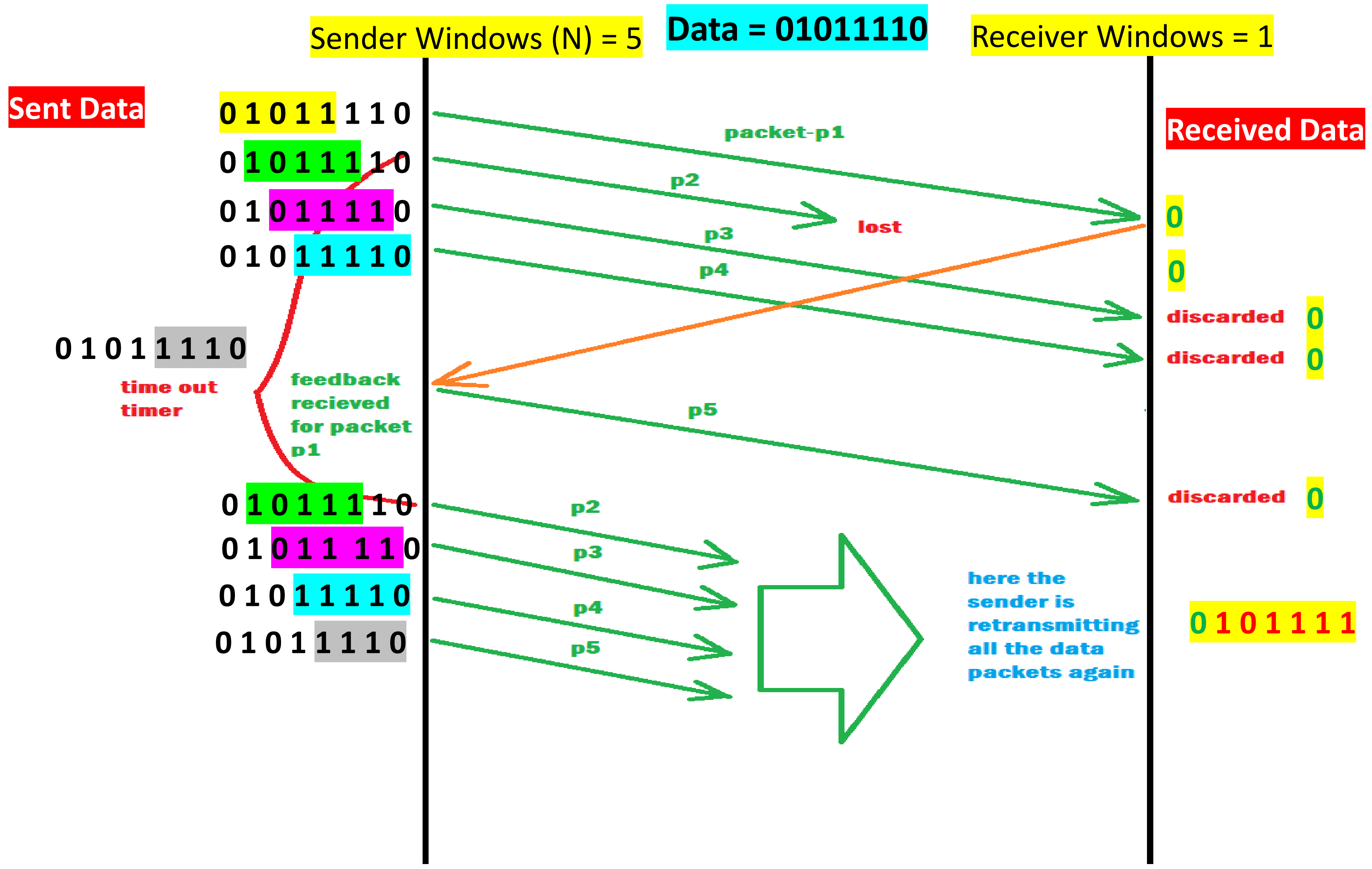
- Go – Back – N ARQ provides for ***sending multiple frames*** *before receiving the acknowledgement for the first frame.*
- It uses the concept of sliding window, and so is also called **sliding window protocol.**
- The *frames are sequentially numbered* and a *finite number of frames are sent.*
- If the acknowledgement of a frame is not received within the time period, *all frames* ***starting from that frame*** *are retransmitted*.
- **Sender Windows Size = N**
- **Receiver Windows Size = 1**

Demo:
https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/go-back-n-protocol/index.html


Sender Windows (N) = 4
Receiver Windows = 1
Sender
Receiver
Data = 0101
Sent Data
0 1 0 1 * * * *
0 1 0 1 * * * *
0 1 0 1 * * * *
0 1 0 1 * * * *
data 0
data 1
data 2
data 3
ACK 2
NAK 2
Received Data
0
0 1
Error, Discarded 0 1
Discarded 0 1
0 1 0 1 * * * * Resent
0 1 0 1 * * * * Resent
2
3
0 1 0
0 1 0 1

Go – Back – N ARQ

- In the above figure, **two frames** have been transmitted before an error discovered in the third frame.
- In this case, ***ACK 2** has been returned telling that the **frames 0,1** have been received successfully* without any error.
- *The receiver discovers the error in **data 2 frame***, so it returns the **NAK 2 frame.**
- The ***frame 3** is also discarded* as it is *transmitted after the damaged frame.*
- Therefore, the *sender retransmits the* **frames 2,3, etc.** *till the size of windows*.


Sender Windows (N) = 5
Data = 01011110
Receiver Windows = 1
Sent Data
Received Data
0 1 0 1 1 1 1 0
0 1 0 1 1 1 1 0
0 1 0 1 1 1 1 0
0 1 0 1 1 1 1 0
0 1 0 1 1 1 1 0
packet-p1
p2
lost
p3
p4
0
0
discarded 0
discarded 0
time out timer
feedback recieved for packet p1
p5
discarded 0
0 1 0 1 1 1 1 0
0 1 0 1 1 1 0
0 1 0 1 1 1 1 0
0 1 0 1 1 1 1 0
p2
p3
p4
p5
here the sender is retransmitting all the data packets again
0 1 0 1 1 1 1

Go – Back – N ARQ

- In the above figure ,consider a sender has to send data packets **indexing from p1 to p5.**
- It sends all the data packets in order (from p1 to p5), but the receiver has **only received p1** and the **data packet p2 is lost** somewhere in the network.
- Then the receiver declines all the data packets after p2 (i.e. p3, p4, p5) ***because the receiver is waiting for packet p2 and will not accept any other data packet than that.***
- So, now as the **time out time index of p2 expires**, the sender goes back 3 packets and **starts sending all the data packets from p2 to p5 again.**

Selective Repeat ARQ

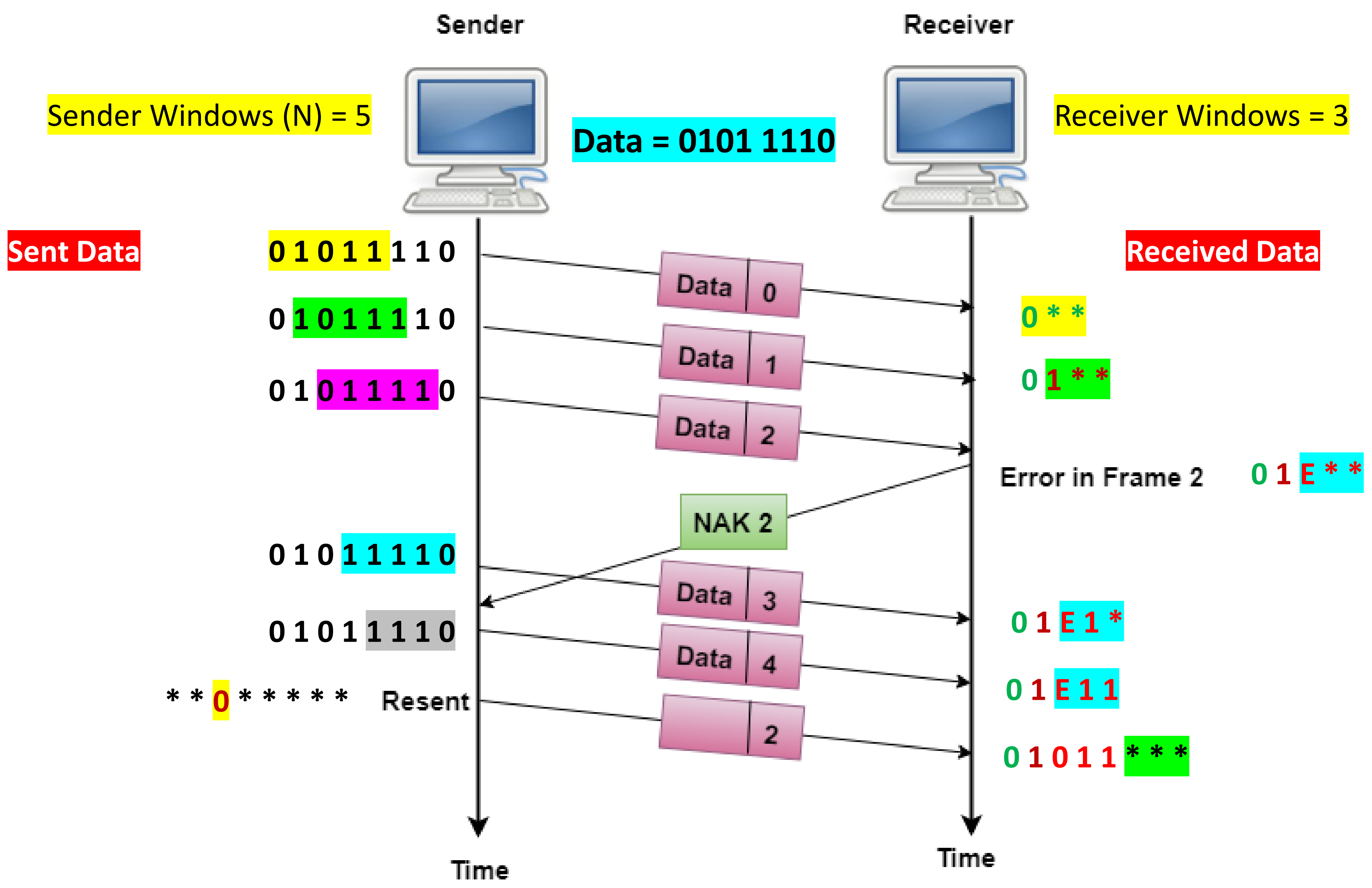
- This protocol also provides for **sending multiple frames** before receiving the acknowledgement for the first frame.
- However, here only the ***erroneous or lost frames are retransmitted***, *while the good frames are received and buffered*.

- **Sender Windows Size = N**
- **Receiver Windows Size = More than 1 (Finite Number)**

Demo:

https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/selective-repeat-protocol/index.html

https://www.ccs-labs.org/teaching/rn/animations/gbn_sr/


Sender
Receiver
Sender Windows (N) = 5
Data = 0101 1110
Receiver Windows = 3
Sent Data
Received Data
0 1 0 1 1 1 1 0
Data 0
0 * *
0 1 0 1 1 1 1 0
Data 1
0 1 * *
0 1 0 1 1 1 1 0
Data 2
Error in Frame 2
0 1 E * *
NAK 2
0 1 0 1 1 1 1 0
Data 3
0 1 E 1 *
0 1 0 1 1 1 1 0
Data 4
0 1 E 1 1
* * 0 * * * * *
Resent
2
0 1 0 1 1 * * *
Time
Time