3 Data Link Layer (10)

-UBM

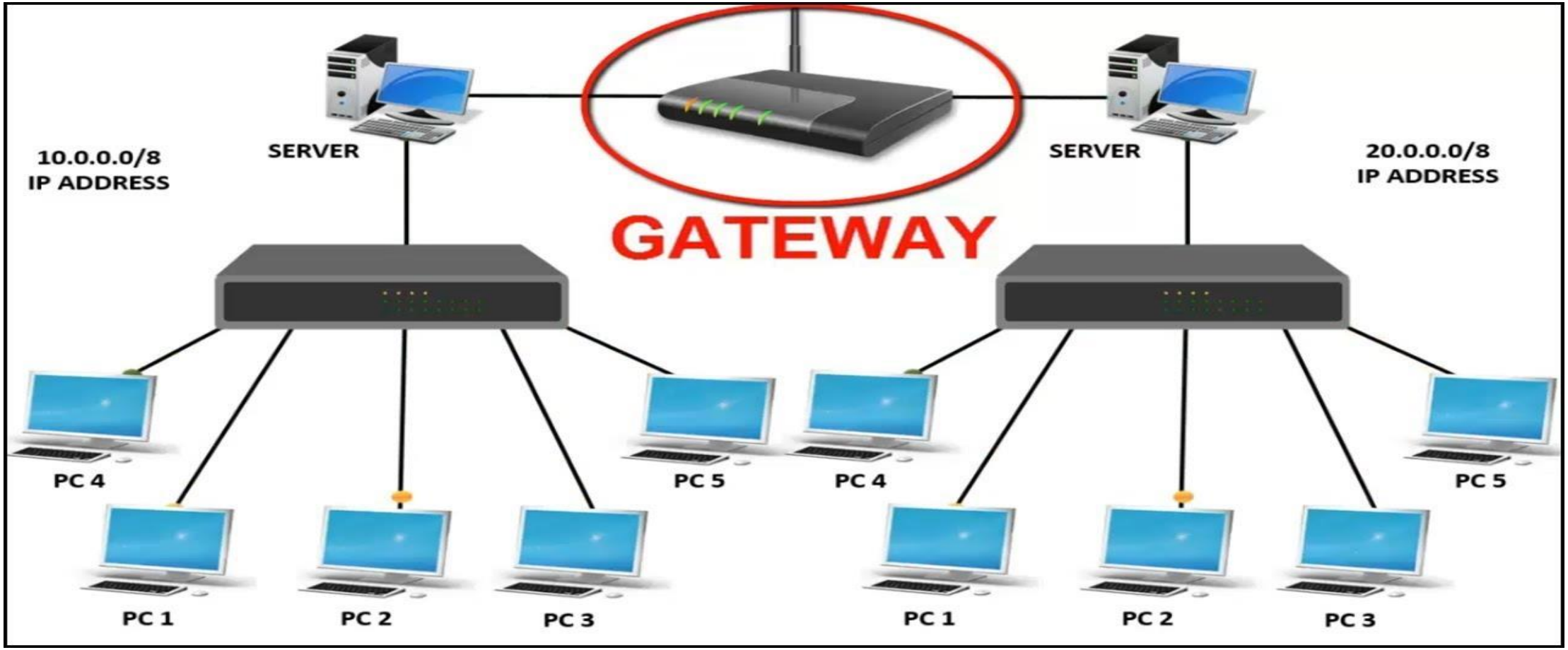
3 Data Link Layer (10)

- **3.1** DLL Design Issues (Services, Framing, Error Control, Flow Control), Error Detection and Correction(Hamming Code, CRC, Checksum) , Elementary Data Link protocols , Stop and Wait, Sliding Window(Go Back N, Selective Repeat), HDLC
- **3.2 Medium Access Control sublayer** Channel Allocation problem, Multiple access Protocol(Aloha, Carrier Sense Multiple Access (CSMA/CD), Local Area Networks - Ethernet (802.3)

Data Link Layer

- In the TCP/IP model, it is the a 4th layer from the top and 2nd layer from the bottom.
- The **main responsibility is to transfer the datagram across an individual link.**
- DLL **protocol does**
  - **defines the format of the packet exchanged across the nodes**
  - **Error detection,**
  - **retransmission,**
  - **flow control,**
  - **and random access.**
- The DLL **protocols are Ethernet, token ring, FDDI and PPP.**

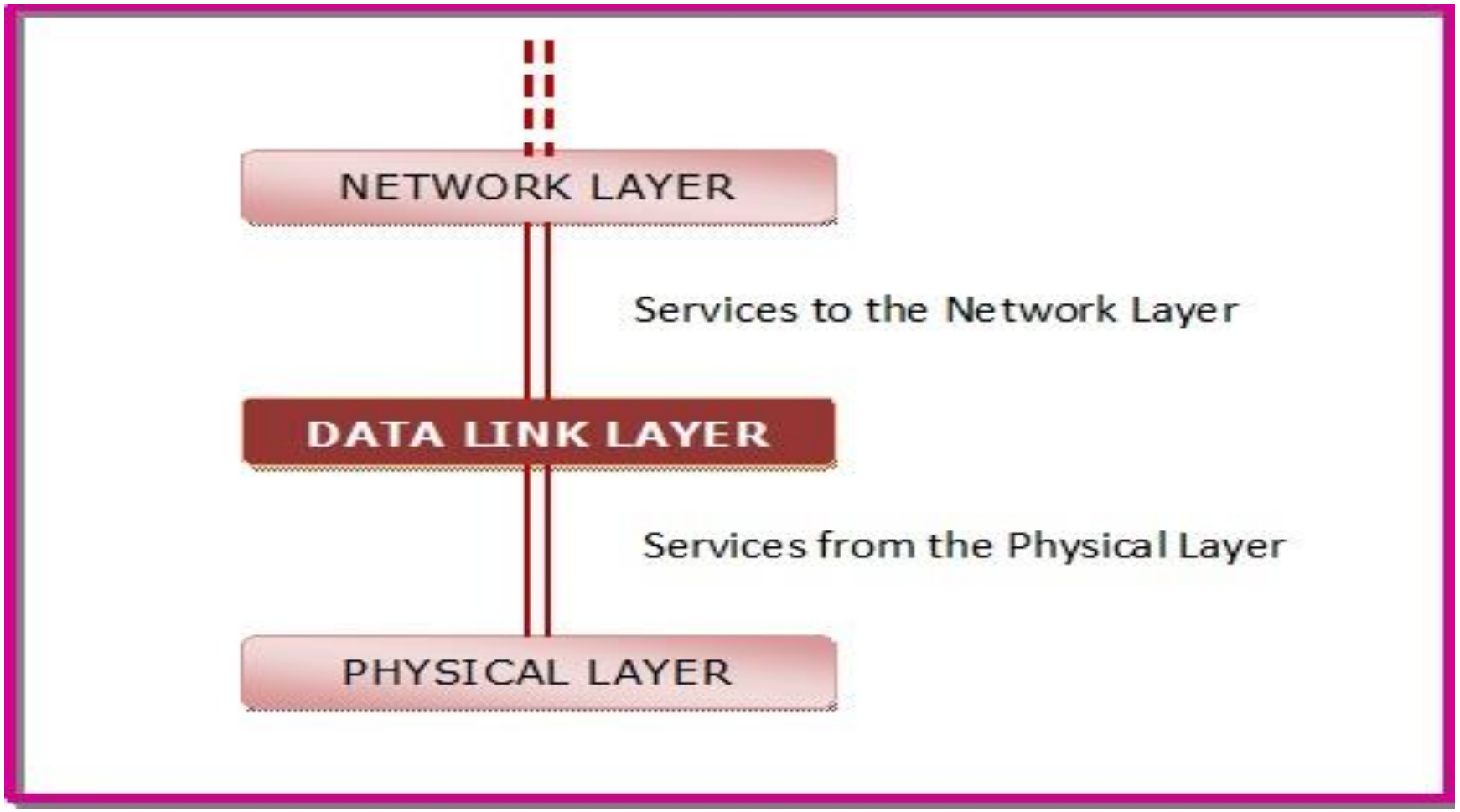
Computer Network



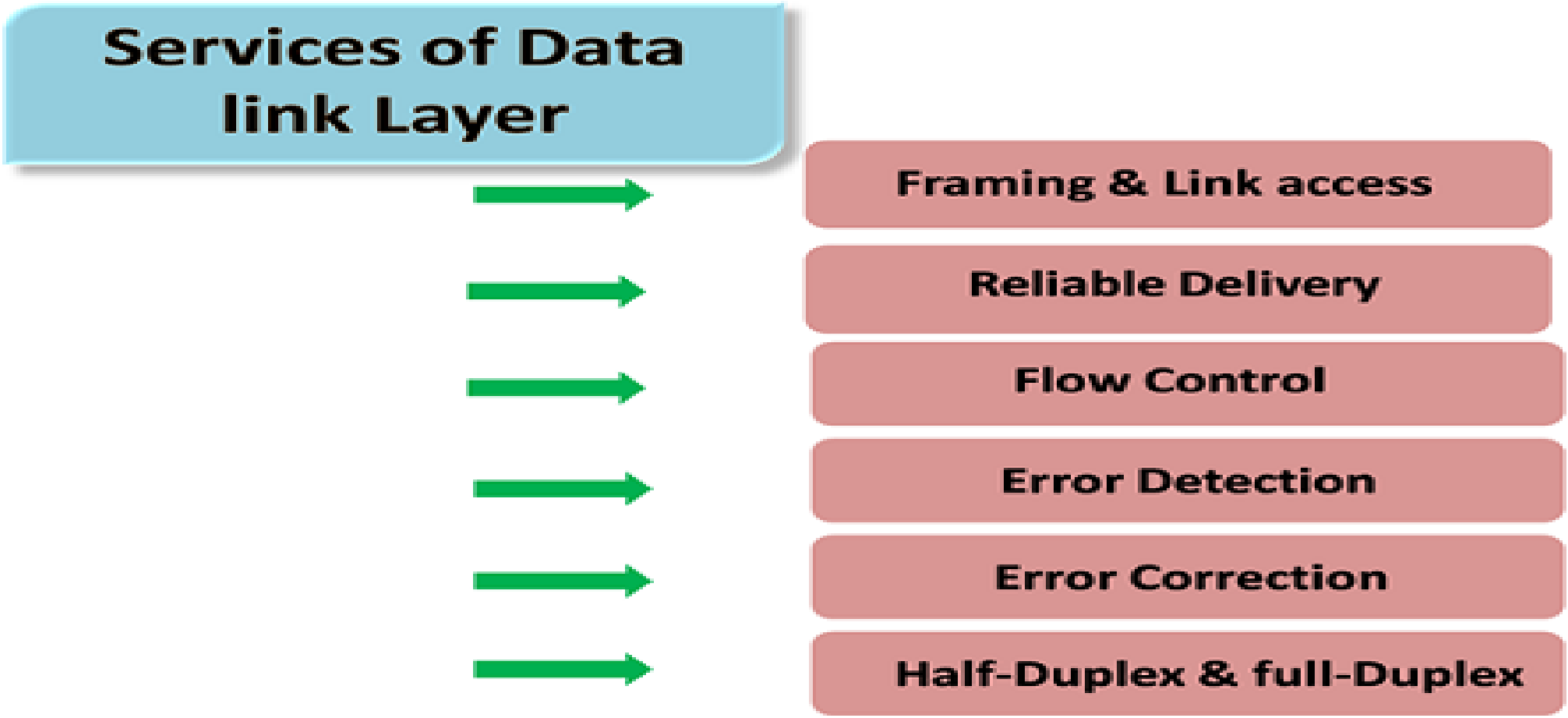
3.1 DLL Design Issues

- The main functions and the design issues of this layer are
  1. **Providing services to the network layer**
  2. **Framing**
  3. **Error Control**
  4. **Flow Control**
  5. **Reliable Delivery**

Services to the Network Layer



Services of Datalink Layer



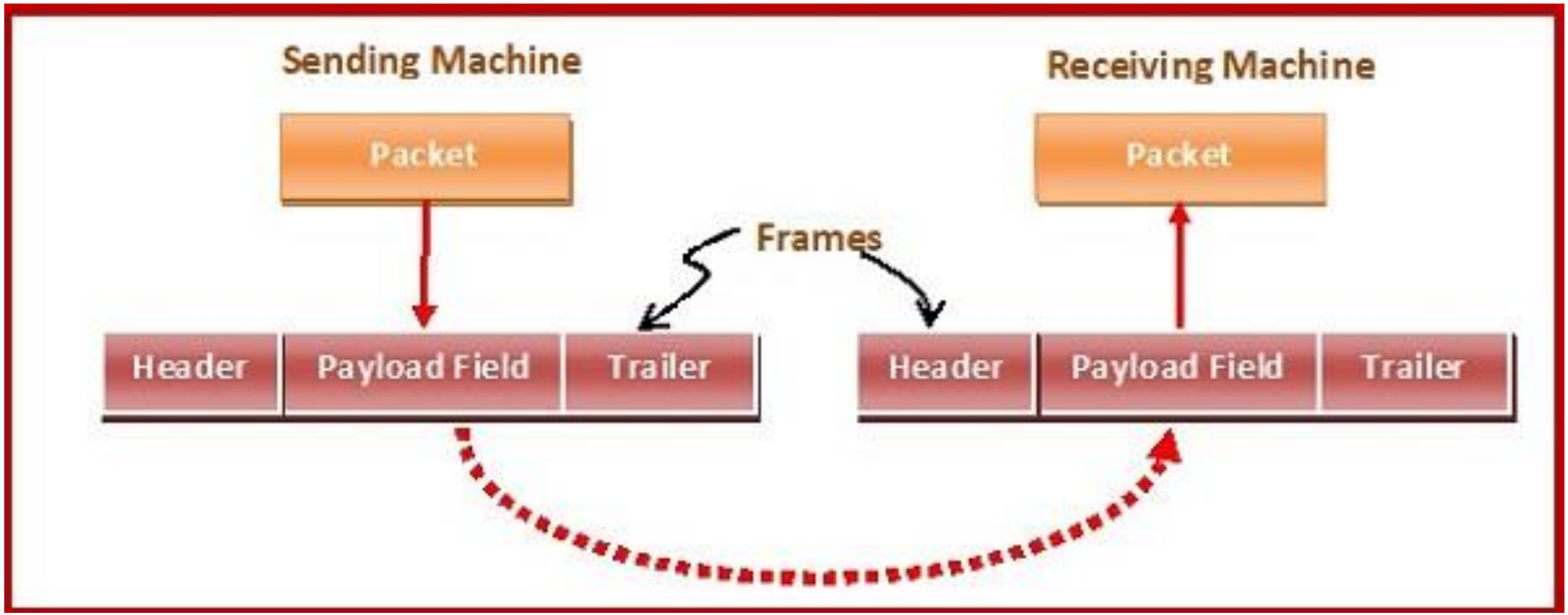
Services of Datalink Layer

- **Framing & Link access: DLL protocols encapsulate each network frame within a Link layer frame** before the transmission across the link.
- **A frame consists of a data field in which network layer datagram is inserted and a number of data fields.**
- **It specifies the structure of the frame as well as a channel access protocol** by which frame is to be transmitted over the link.

Framing

- The DLL **encapsulates each data packet from the network layer into frames** that are then transmitted.
- A frame has three parts, namely –
  - Frame Header
  - **Payload field that contains the data packet** from network layer
  - Trailer

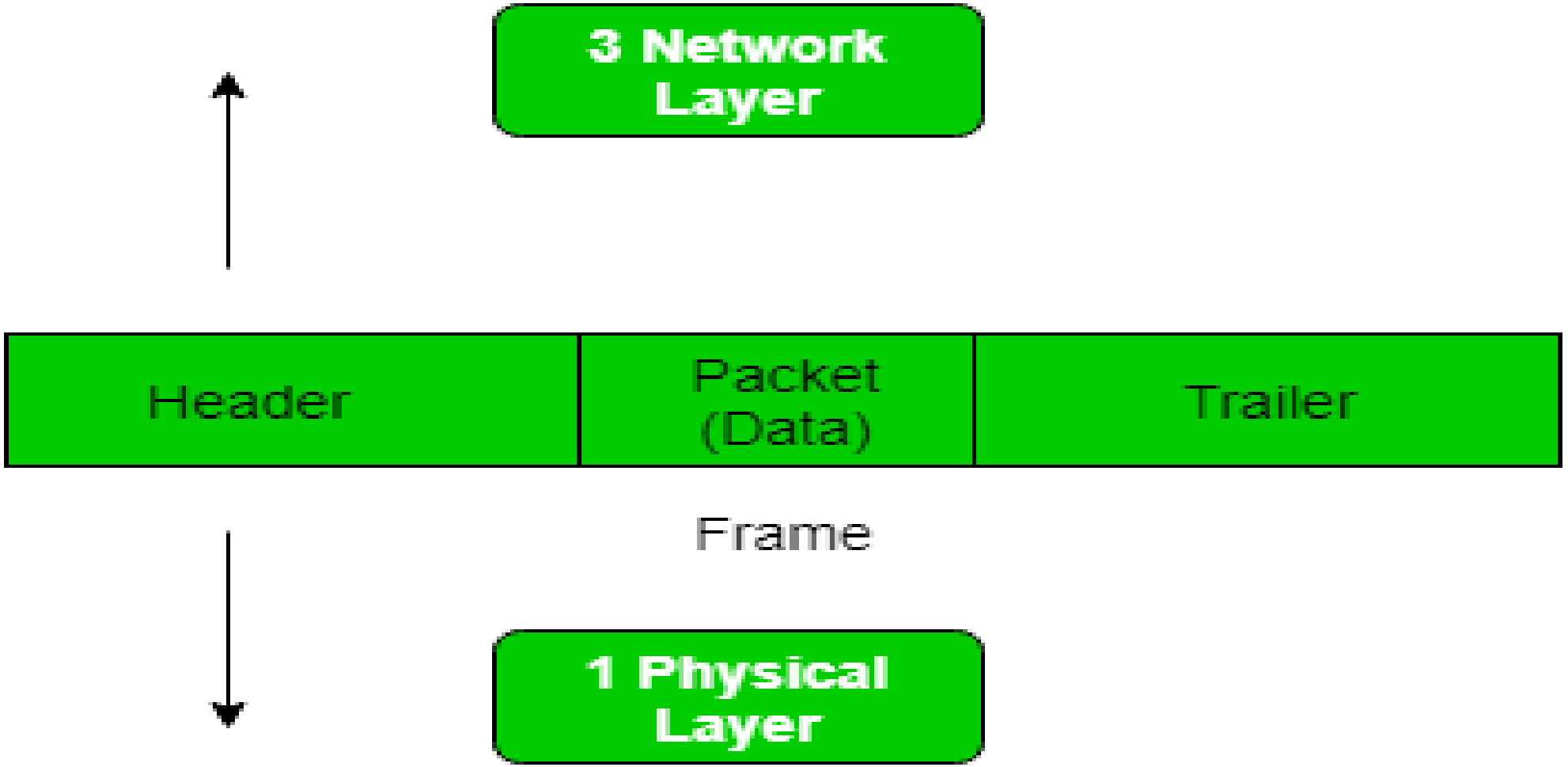
Framing


Sending Machine
Receiving Machine
Packet
Packet
Frames
Header
Payload Field
Trailer
Header
Payload Field
Trailer

Framing

- Framing provides **a way for a sender to transmit a set of bits that are meaningful to the receiver.**
- **Ethernet, token ring, frame relay, and other DLL technologies have their own frame structures**.
- Frames **have headers** that contain information such as error-checking codes

Data Link Layer Services



Problems in Framing

- **Detecting start of the frame: Every** Station detect frames by l**ooking out for special sequence of bits that marks the beginning** of the frame i.e. SFD (Starting Frame Delimiter).
- **How do station detect a frame:** Every station listen to link for SFD pattern through a sequential circuit. If **SFD is detected, sequential circuit alerts station. Station checks destination address to accept or reject frame**.
- **Detecting end of frame:** When to stop reading the frame.
- **Advantages:** of using frames is that data is broken up into recoverable chunks that can easily be checked for corruption.

Figure 3.2 *Ethernet Frame*

Preamble: 56 bits of alternating 1s and 0s.

SFD: Start frame delimiter, flag (10101011)

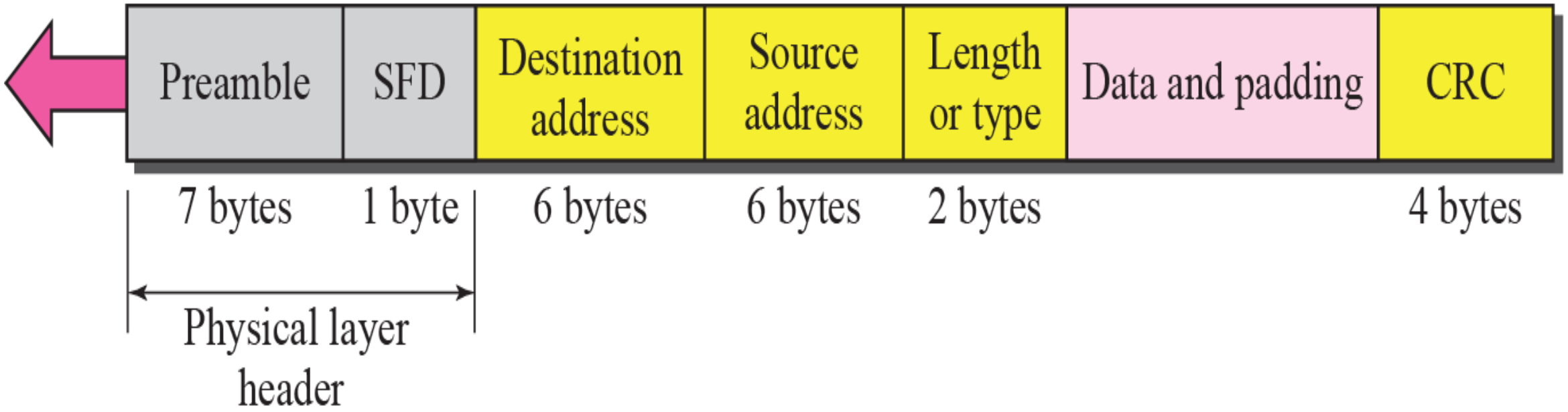
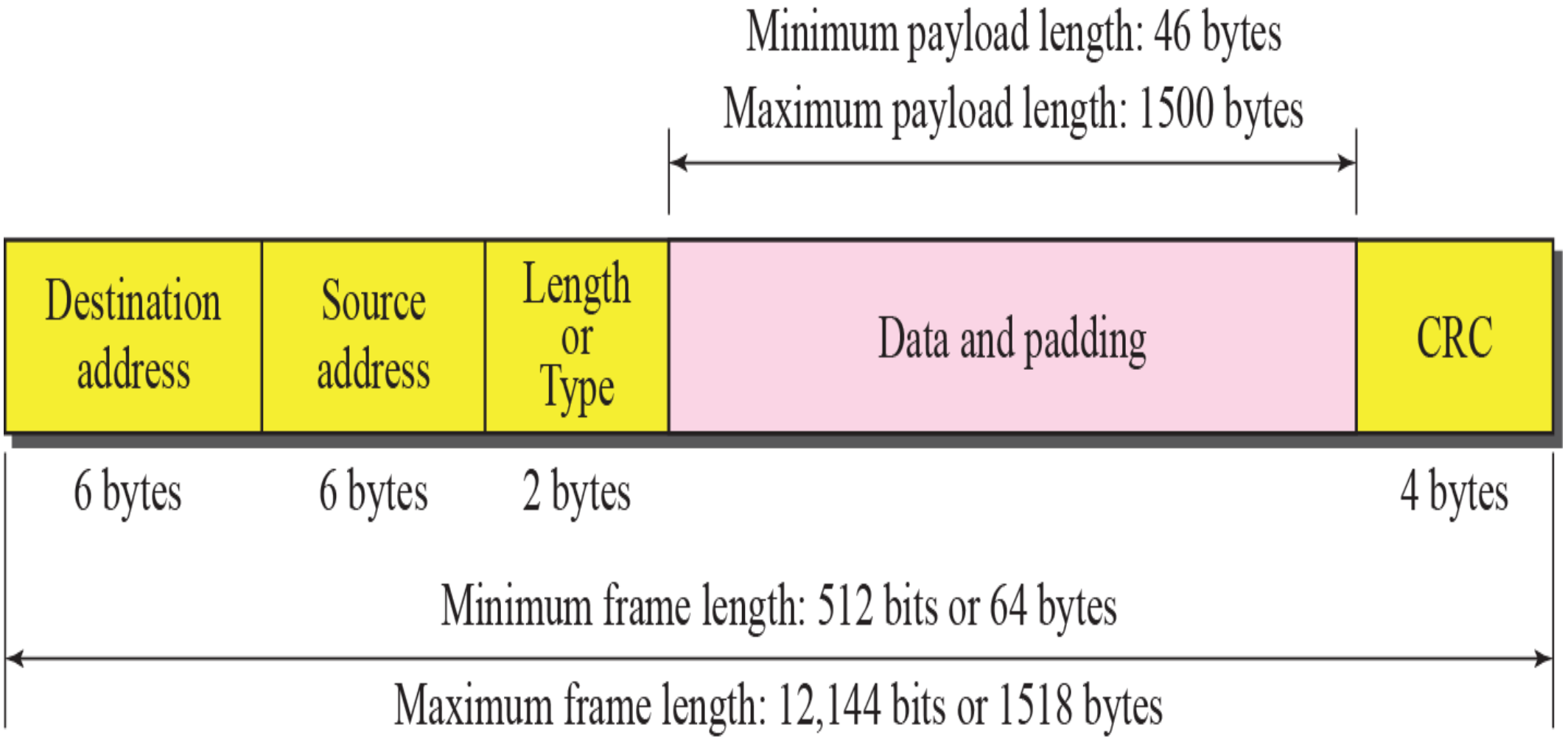


Figure 3.3 *Maximum and minimum lengths*



Types of Framing

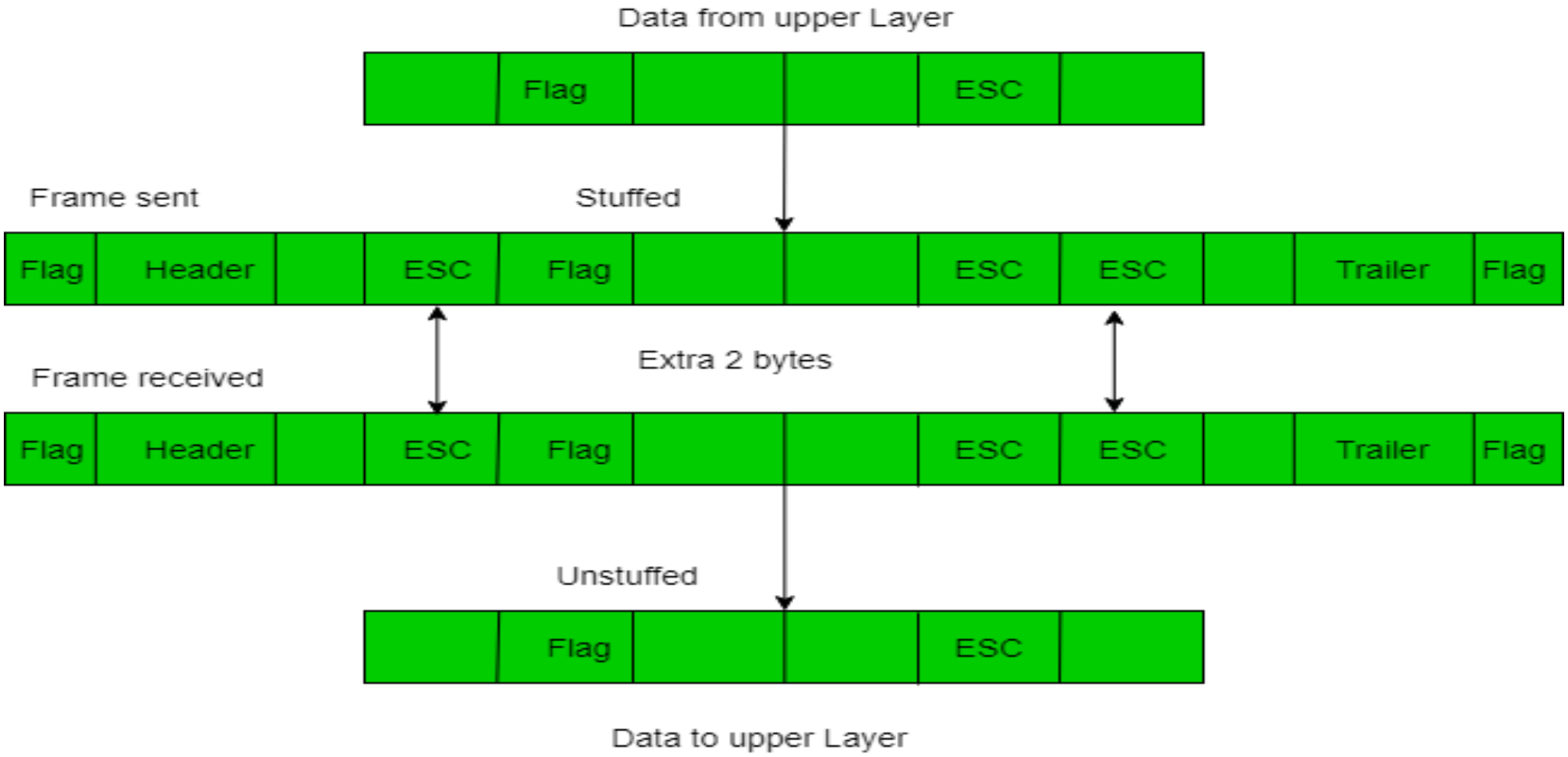
- **1. Fixed size –** In this frame there is **no need to provide boundaries to the frame, length of the frame itself acts as delimiter.**
  - **Drawback: It suffers from internal fragmentation if data size is less than frame size**
  - **Solution: Padding**

Types of Framing

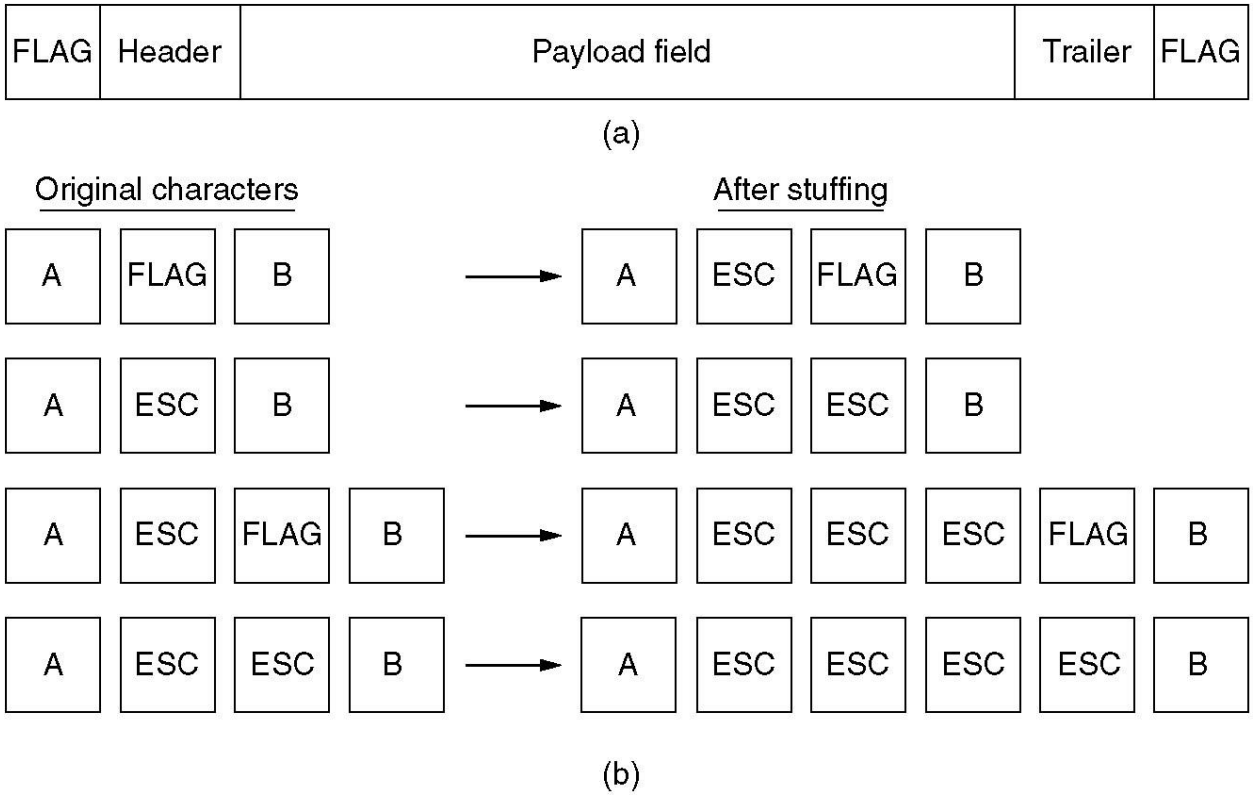
2. Variable size – In this there is **need to define end of frame as well as beginning of next frame to distinguish.** This can be done in two ways:

- **Length field – We can introduce a length field** in the frame. Used in **Ethernet(802.3)**. The **problem with this is that sometimes the length field might get corrupted.**
- **End Delimiter (ED) –** We can introduce an ED(pattern) to indicate the end of the frame. Used in **Token Ring**.

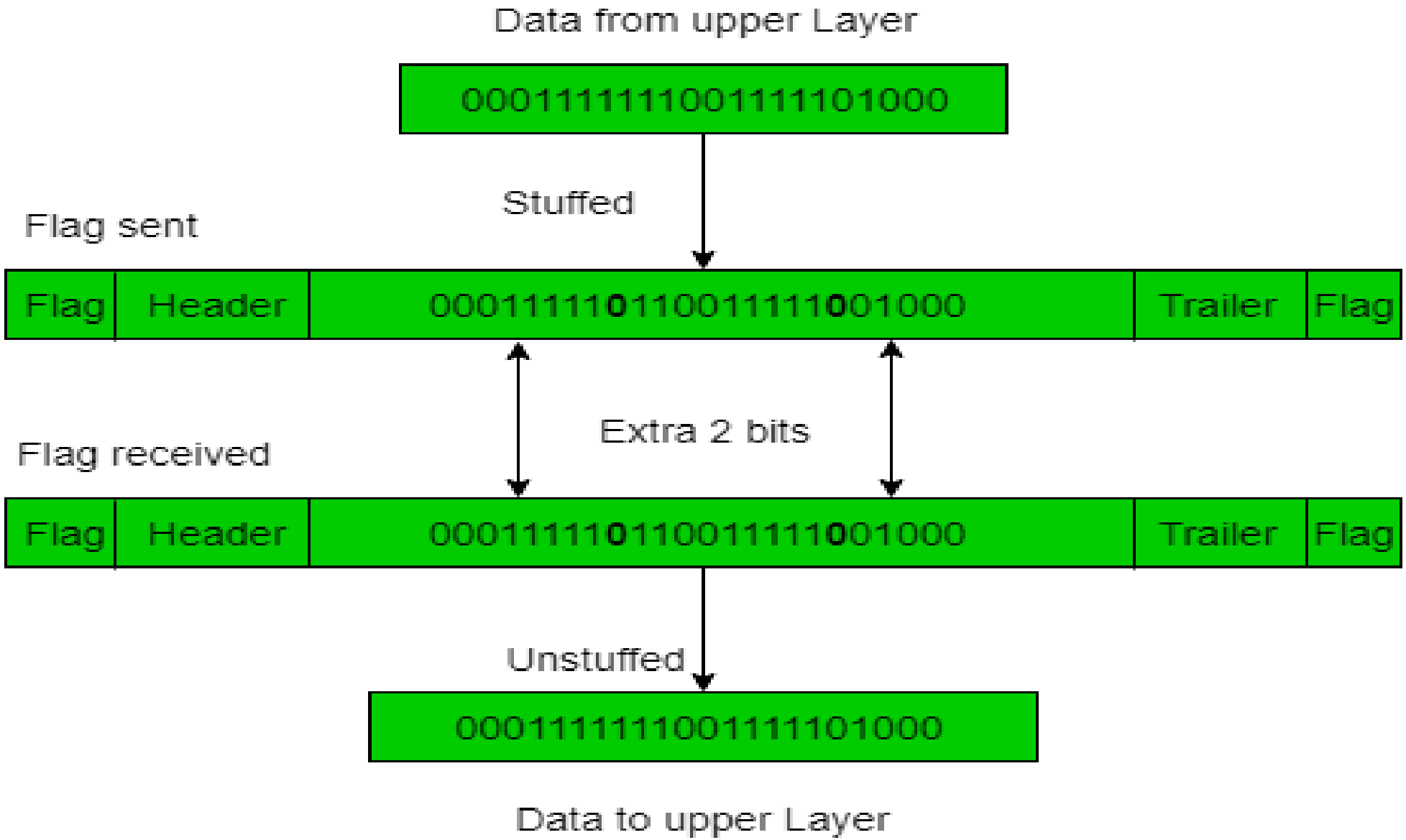
Byte stuffing



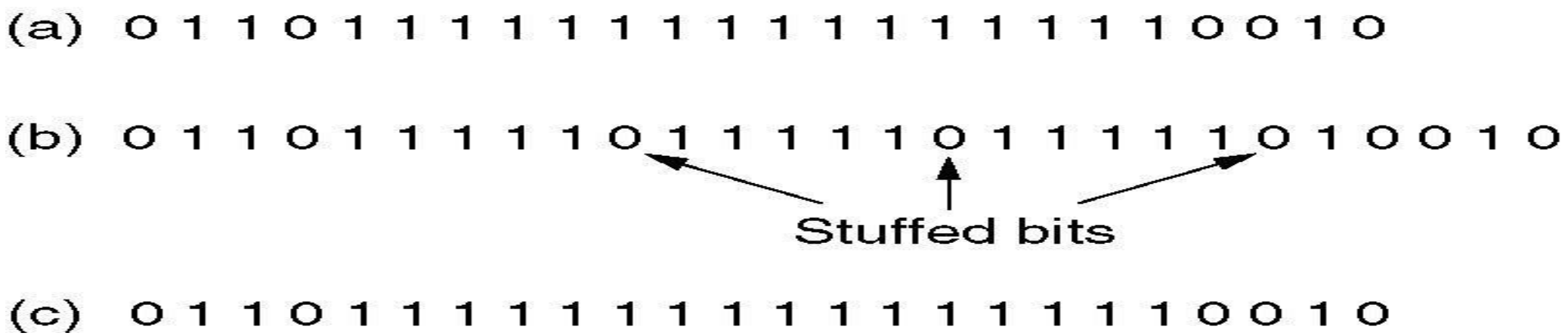
(2.2) Framing



- (a) A frame delimited by flag bytes.
- (b) Four examples of byte sequences before and after stuffing.


Data from upper Layer
00011111110011111101000
Stuffed
Flag sent
Flag
Header
00011111011001111101000
Trailer
Flag
Extra 2 bits
Flag received
Flag
Header
00011111011001111101000
Trailer
Flag
Unstuffed
00011111110011111101000
Data to upper Layer

(2.3) Framing – Bit Stuffing



- Bit stuffing
- (a) The original data.
- (b) The data as they appear on the line.
- (c) The data as they are stored in receiver's memory after destuffing.

Services of Datalink Layer

- **Reliable delivery:** DLL **transmits the network layer datagram without any error.**
- It is **accomplished with transmissions and acknowledgements.**
- A DLL mainly provides this service over the links having higher error rates and **they can be corrected locally, link at which an error occurs rather than forcing to retransmit the data.**

Services of Datalink Layer

- **Flow control: A receiving node can receive the frames at a faster rate than it can process the frame**.
- **Without flow control, the receiver's buffer can overflow, and frames can get lost**.
- To overcome this problem, the DLL uses the **flow control to prevent the sending node on one side of the link from overwhelming the receiving node on another side of the link.**

Note

Data can be corrupted
during transmission.

Some applications require that
errors be detected and corrected.

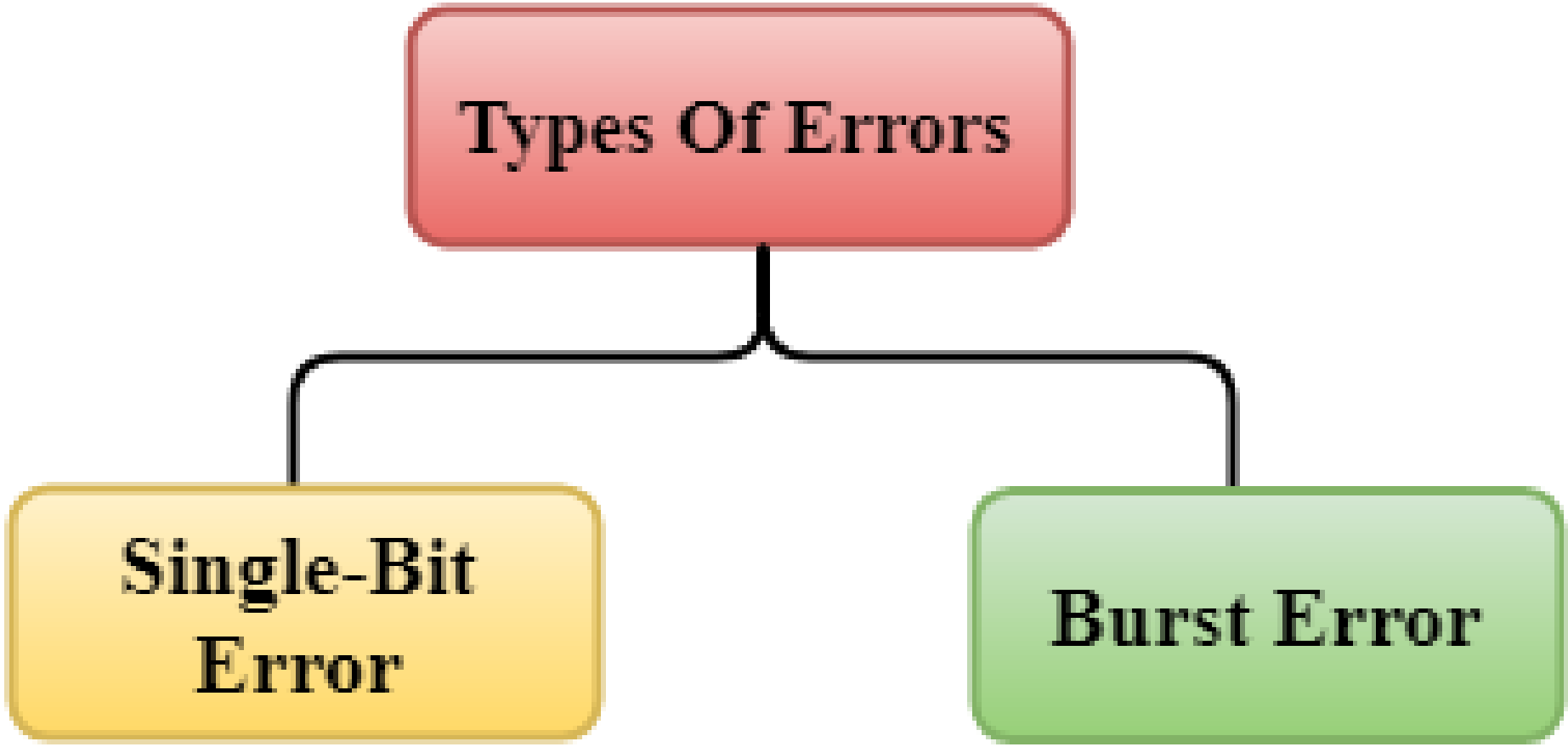
Services of Datalink Layer

- **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: Source message != Receivers Message**
- Errors can be **introduced by signal attenuation and noise.**
- DLL protocol provides a mechanism to detect one or more errors. **This is achieved by adding error detection bits in the frame and then receiving node can perform an error check**.

Services of Datalink Layer

- **Error correction:** Error correction is similar to the Error detection, except that **receiving node not only detect the errors but also determine where the errors have occurred in the frame.**

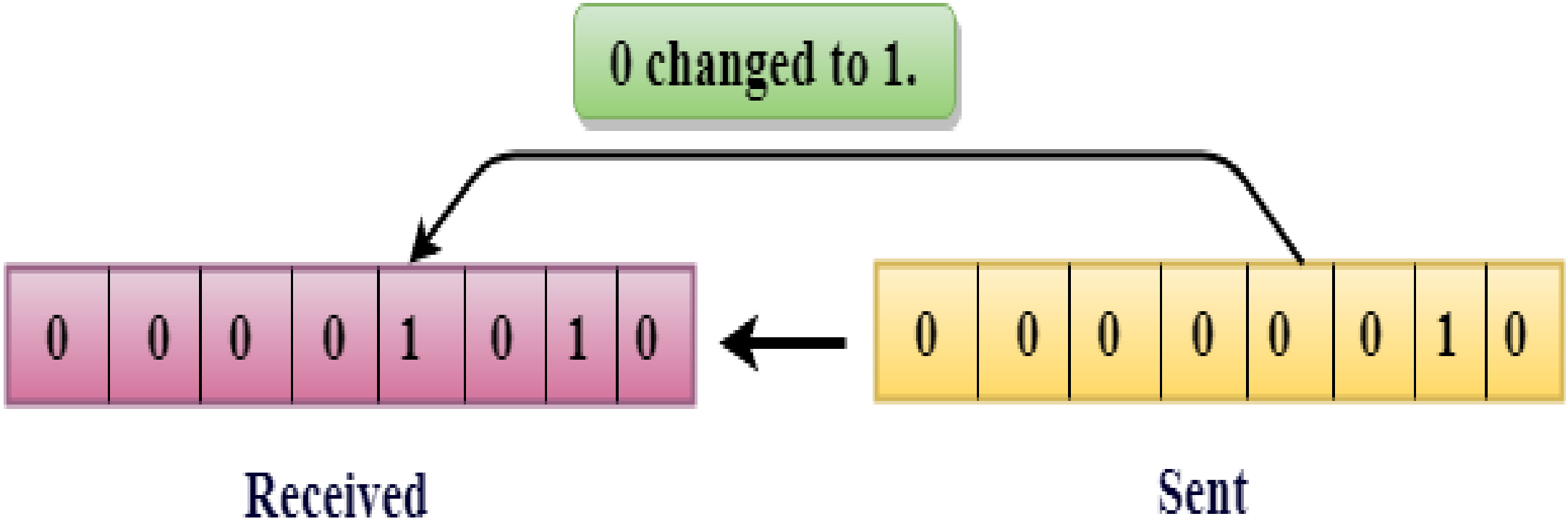
Types of Errors



Note

In a single-bit error, only 1 bit in the data unit has changed.

Single-bit Error



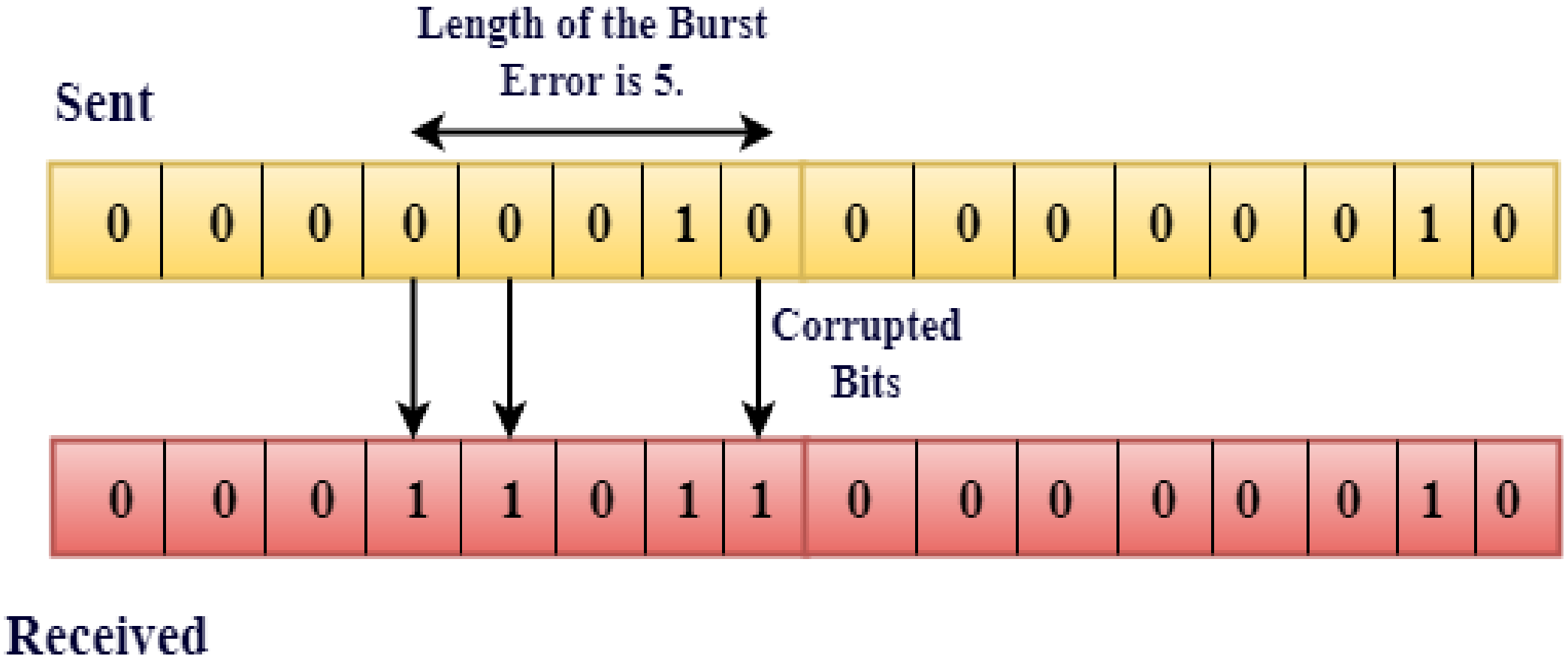
Single-Bit Error:

- **The only one bit of a given data unit is changed**
- In the above figure, the message which is sent is corrupted as single-bit, i.e., 0 bit is changed to 1.
- **Single-Bit Error** does **not appear more likely in Serial Data Transmission.**
- For example, Sender sends the data at **10 Mbps, this means that the bit lasts only for 1sec and for a single-bit error to occurred, a noise must be more than 1sec.**
- Single-**Bit Error mainly occurs in Parallel Data Transmission.** For example, if eight wires are used to send the eight bits of a byte, **if one of the wire is noisy, then single-bit is corrupted per byte.**

Note

A burst error means that 2 or more bits in the data unit have changed.

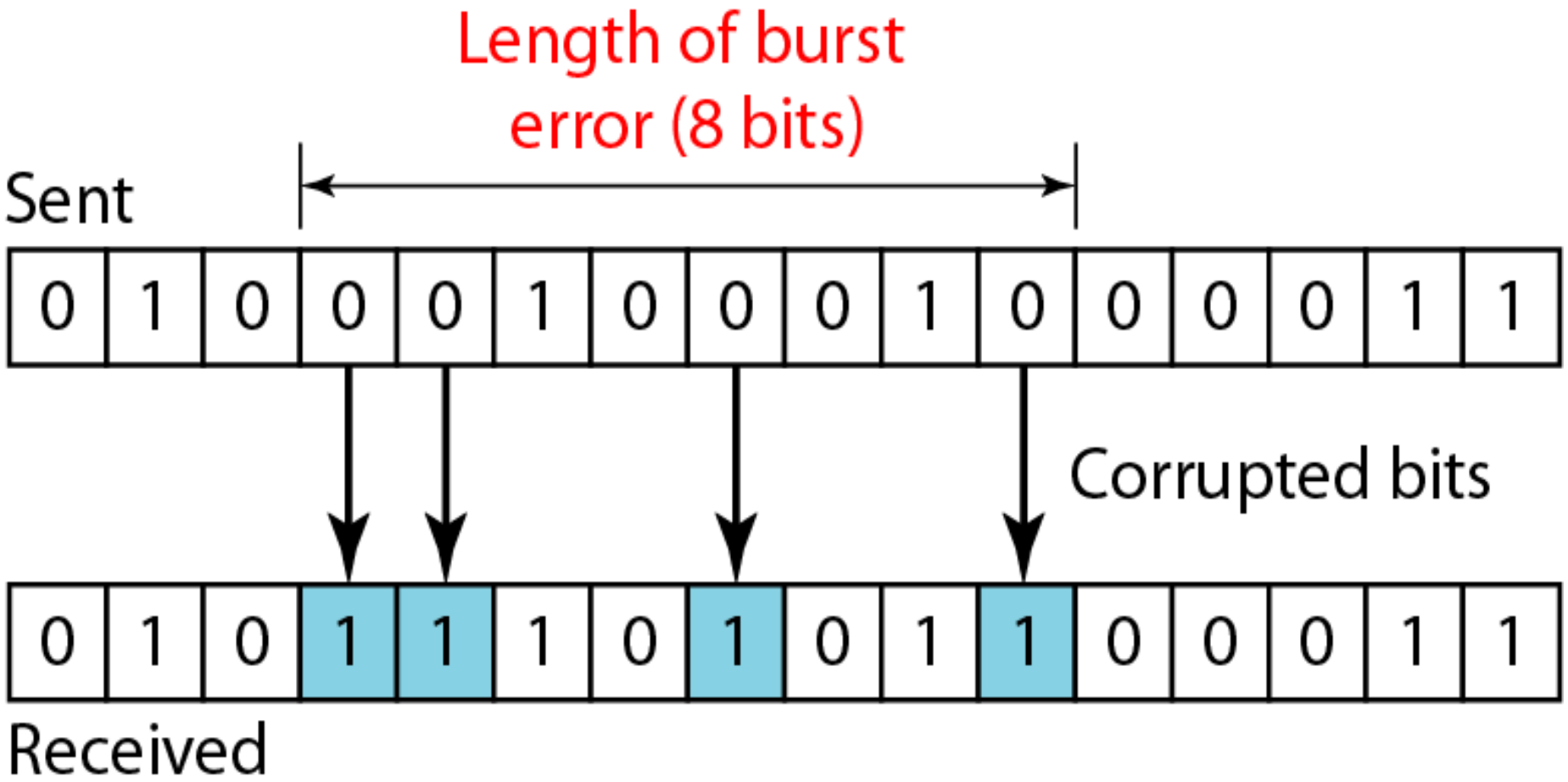
Burst Error:


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

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.**
- The **duration of noise in Burst Error is more than the duration of noise in Single-Bit**.
- **Burst Errors are most likely to occur in Serial Data Transmission.**
- The number of affected bits depends on the duration of the noise and data rate.

Figure 10.2 *Burst error of length 8*

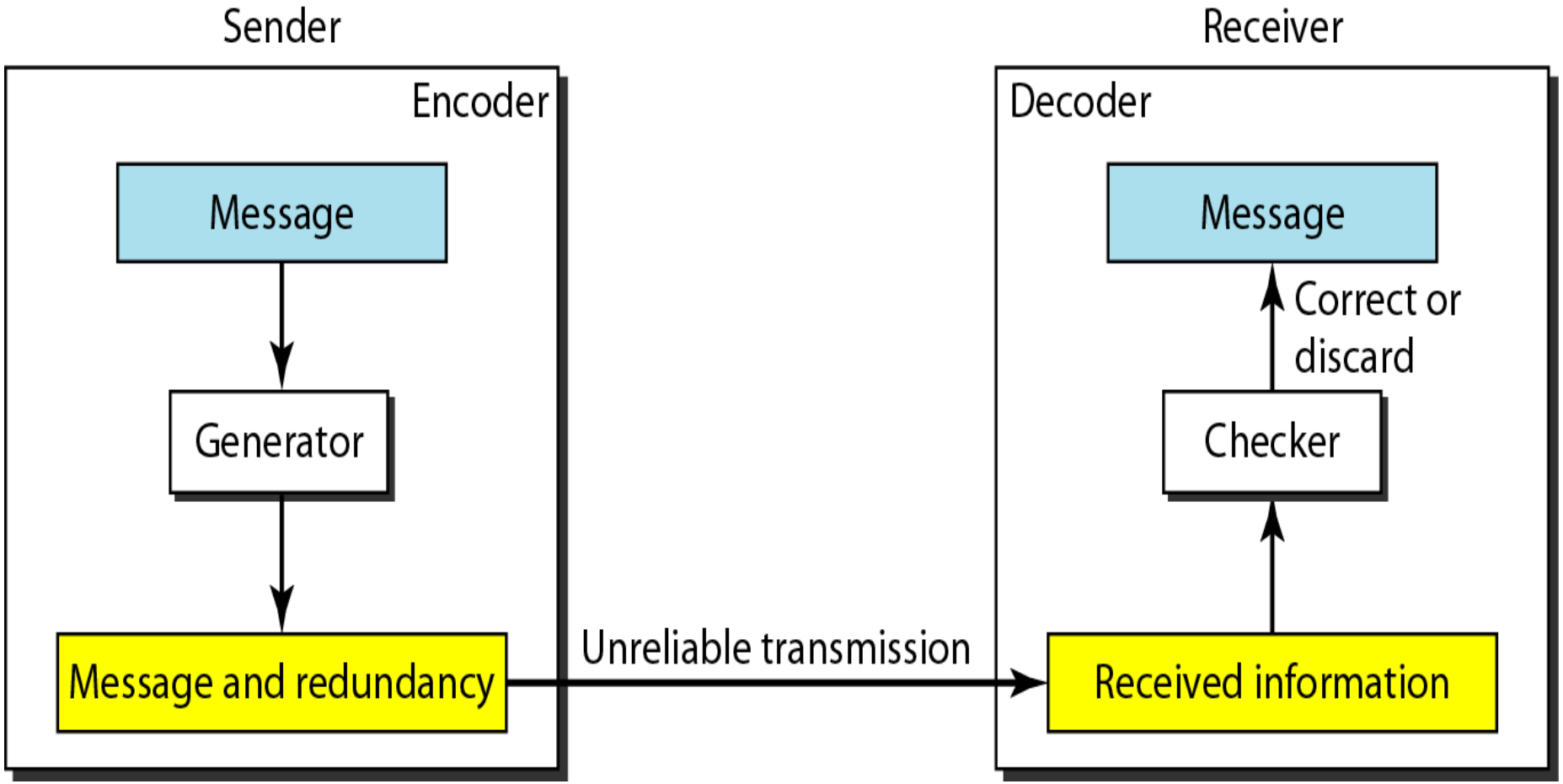


Note

Redundant Bits

To detect or correct errors, we need to send extra (redundant) bits with data.

Figure 10.3 *The structure of encoder and decoder*



10-2 BLOCK CODING

In block coding, we divide our message into blocks, each of k bits, called data words. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called code words.

Note

Block coding is normally referred to as *m*B/*n*B coding; it replaces each *m*-bit group with an *n*-bit group.

Figure 4.14 *Block coding concept*

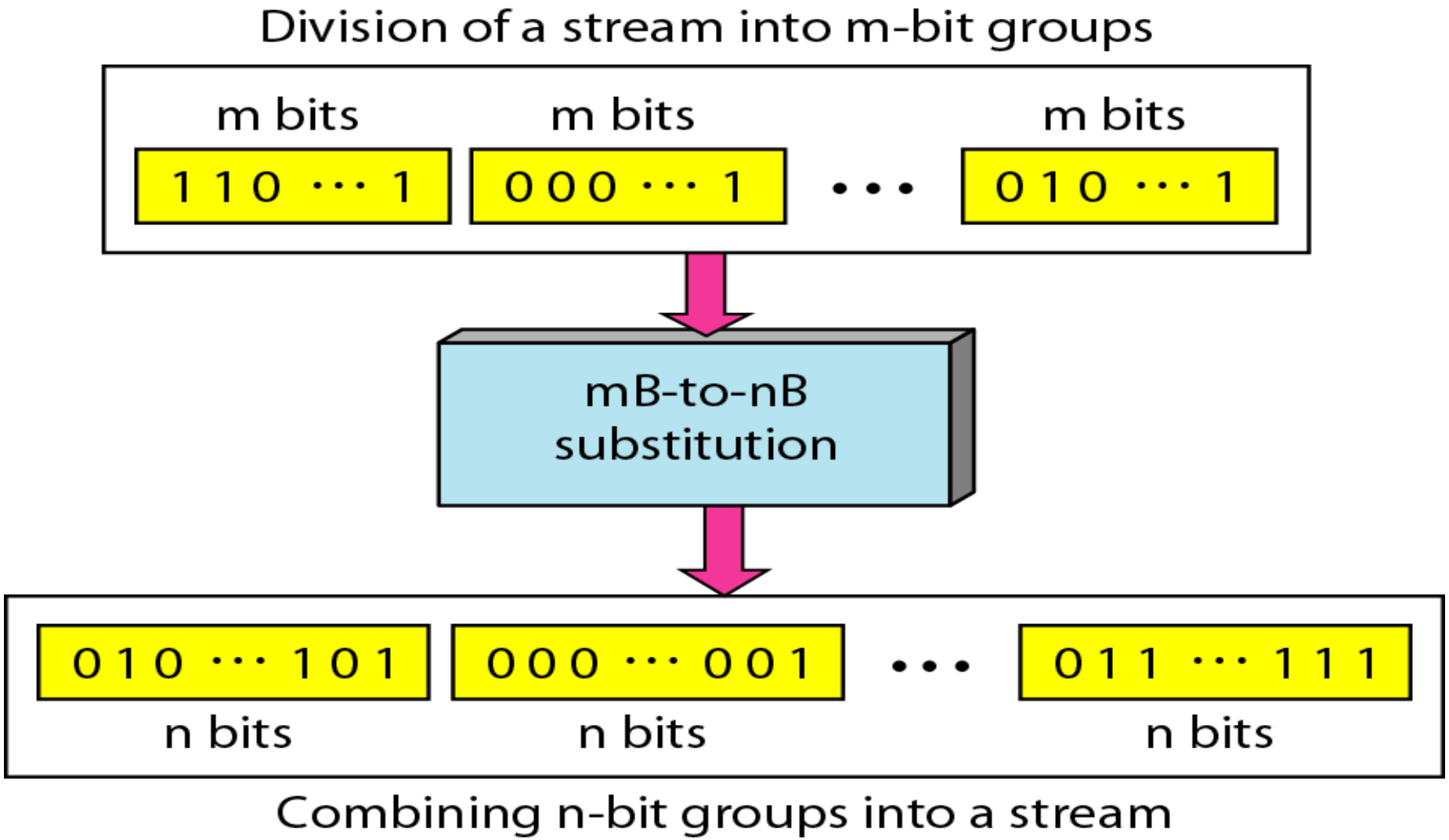
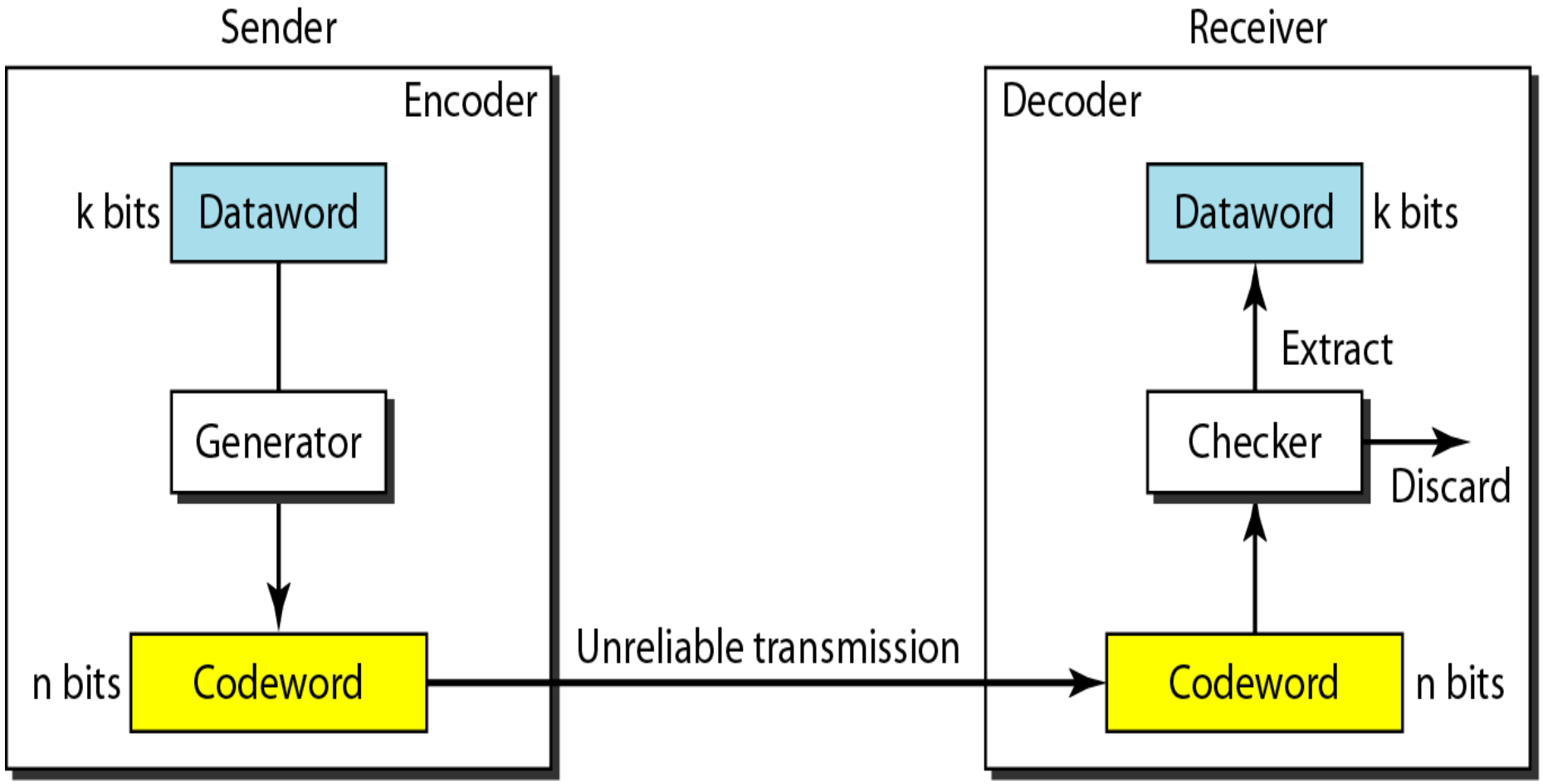


Figure 10.6 *Process of error detection in block coding*



Error Detecting Techniques:

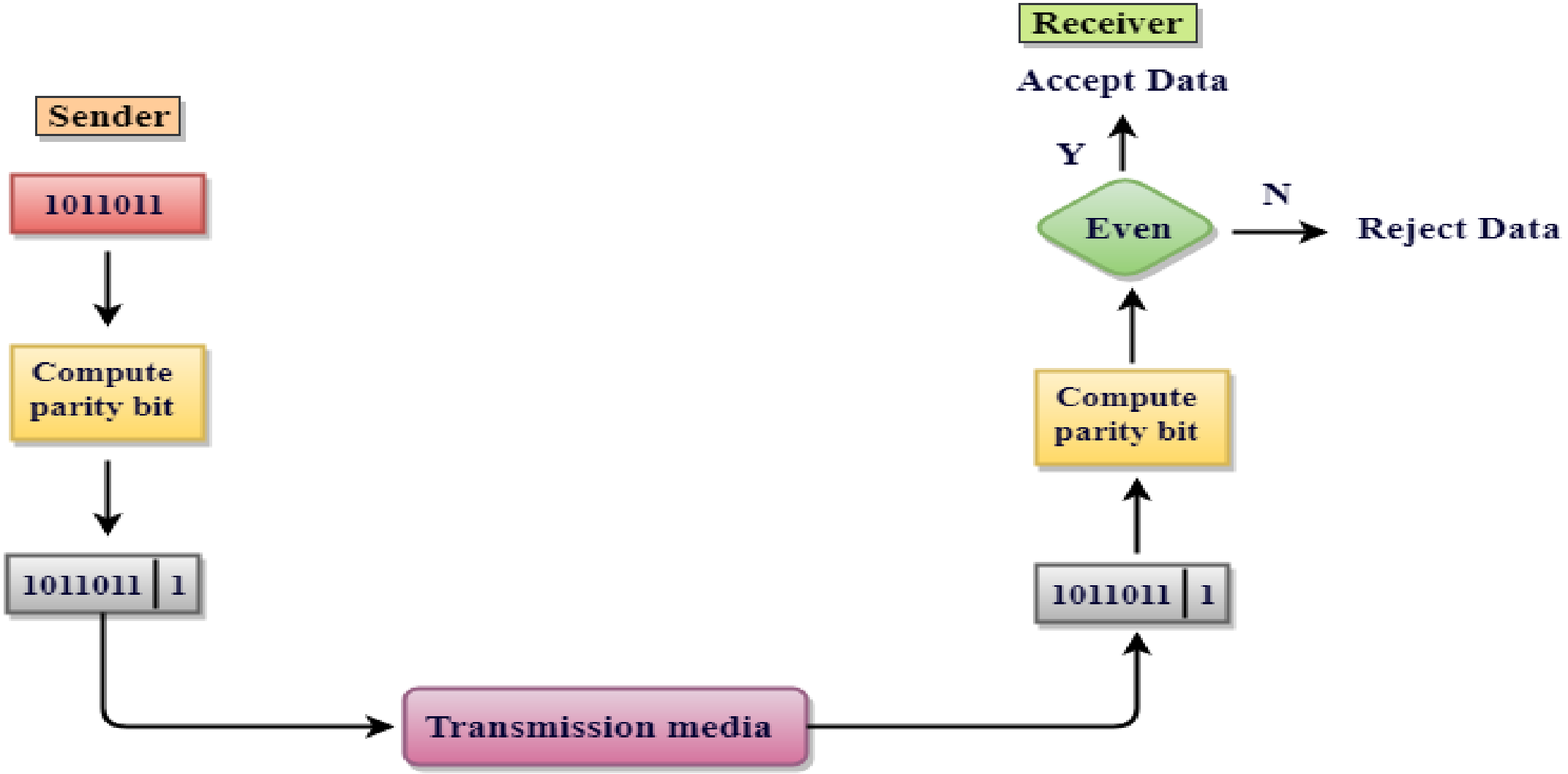
The most popular Error Detecting Techniques are:

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

Single Parity Check

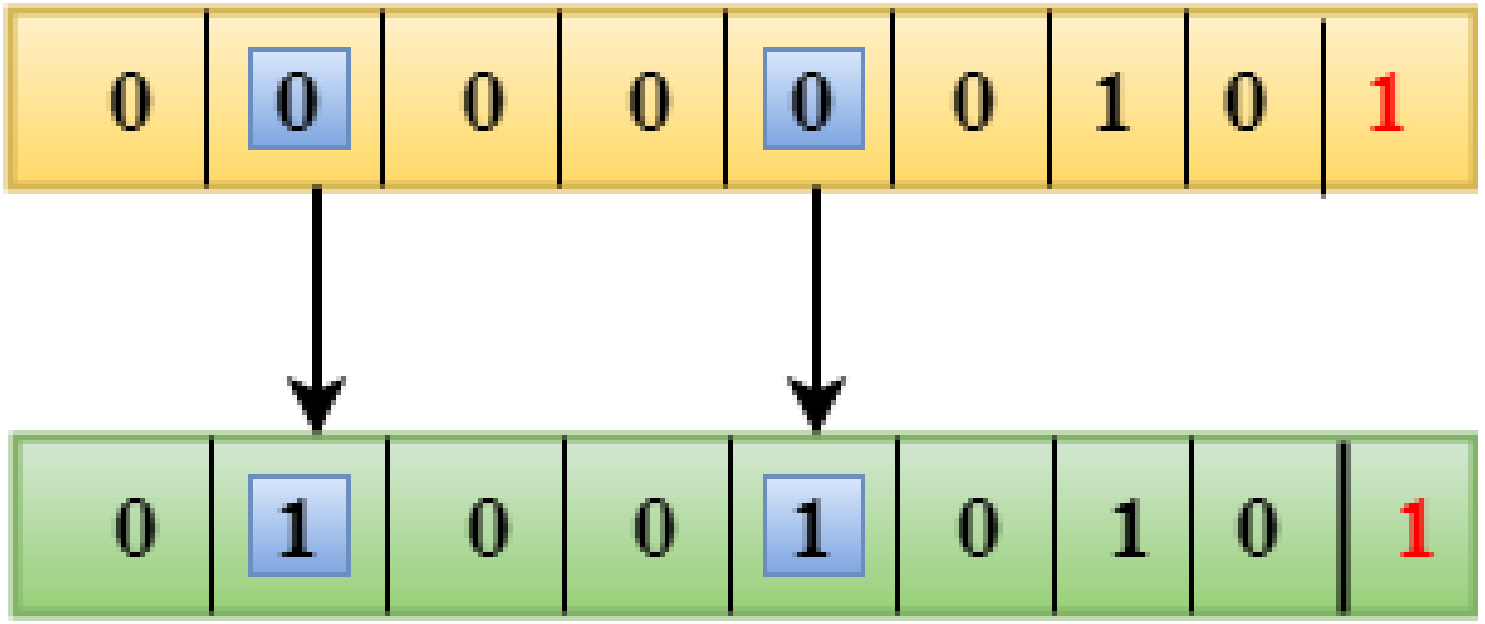
- It is the **simple mechanism and inexpensive.**
- In this a **redundant bit is also known as a parity bit which is appended at the end of the data unit so that the number of 1s becomes even.** Therefore, the total number of transmitted bits would be 9 bits.
- **If the number of 1s bits is odd, then 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.**

Single Parity Check



Drawbacks Of Single Parity Checking

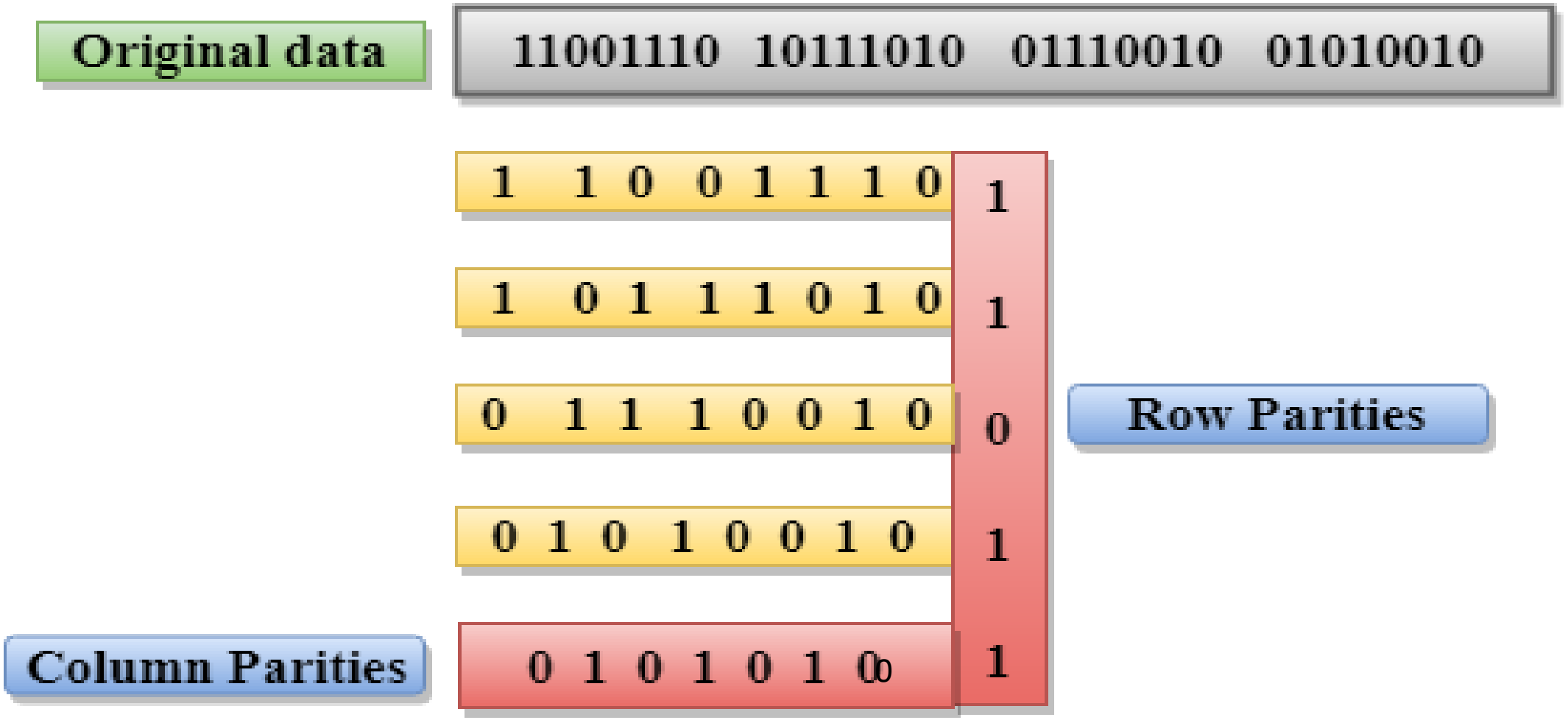
- It can only detect single-bit errors which are very rare.
- If two bits are interchanged, then it cannot detect the errors.



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 this, **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.**

Two-Dimensional Parity Check



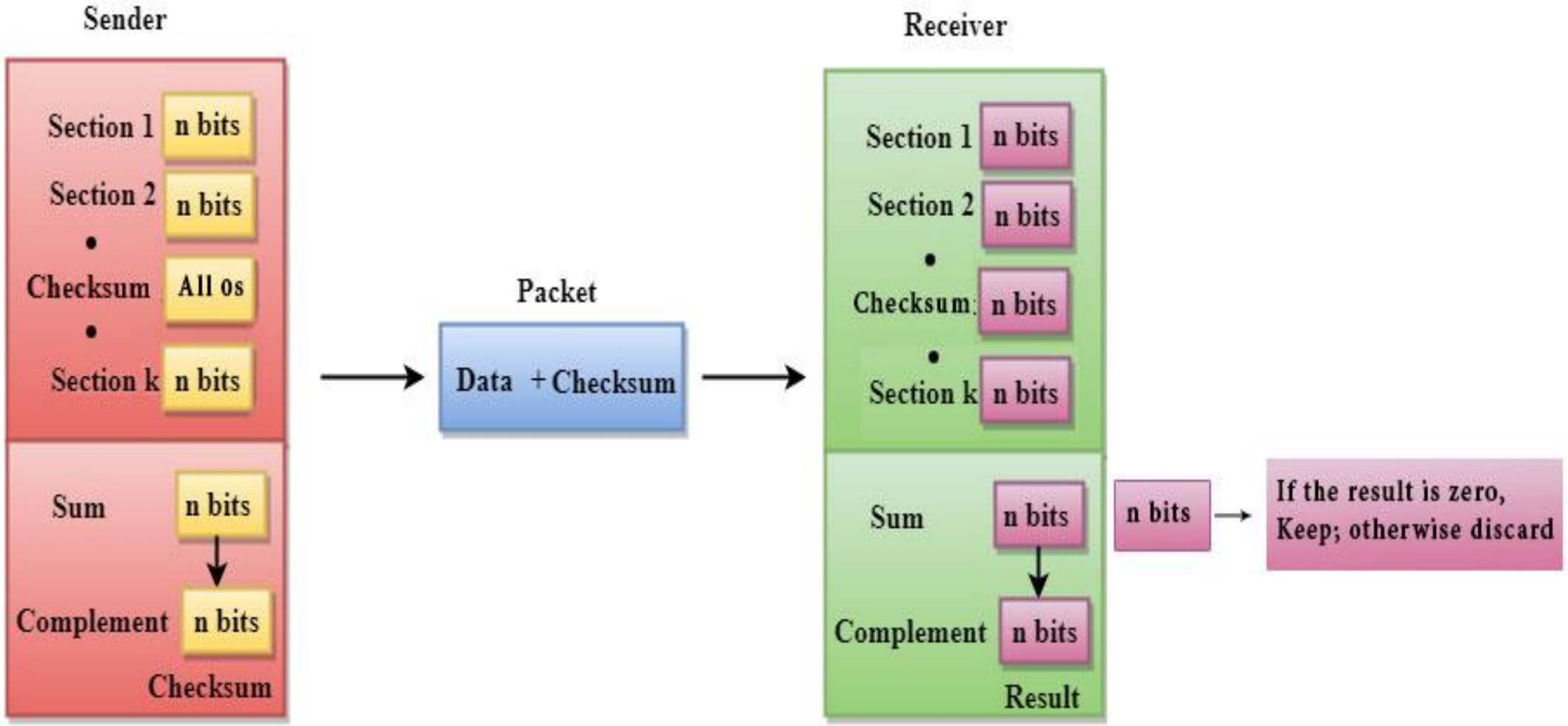
Drawbacks Of 2D Parity Check

- **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.

Checksum

- A Checksum is an error detection technique based on the concept of redundancy.
- **It is divided into two parts:**
  - Checksum Generator
  - Checksum Checker

Checksum


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

Checksum Generator

- A **Checksum is generated at the sending side**.
- It **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**

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 complem ent to get the sum.
- The sum is complemented and it becomes the checksum field
- The original data and checksum field are sent across the netw ork.

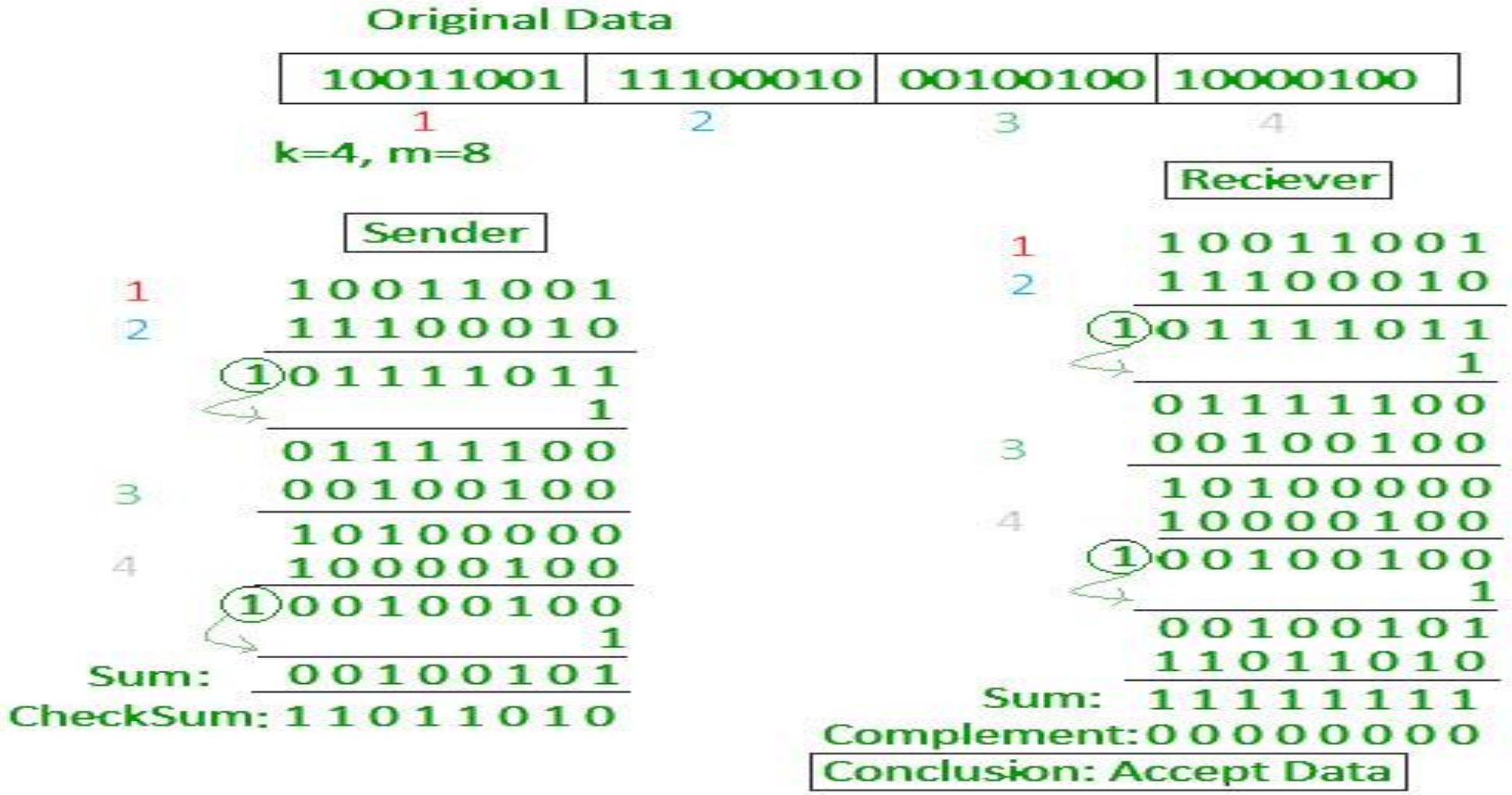
Checksum Checker

- 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 comple ment algorithm to get the sum.
- The sum is complemented.
- If the result of the sum is zero, then the data is accepted ot herwise the data is discarded.

Checksum Example



Cyclic Redundancy Check (CRC)

- A code added to data which is used to detect errors occurring during transmission, storage, or retrieval.
- **CRC is a redundancy error technique used to determine the error**.
- **Following are the steps used in CRC for error detection:**

Cyclic Redundancy Check (CRC)

- **Sender:**
- 1. 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.**
- 2. 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**.
- 3. **Thirdly, the CRC remainder replaces the appended 0s at the end of the original data.**

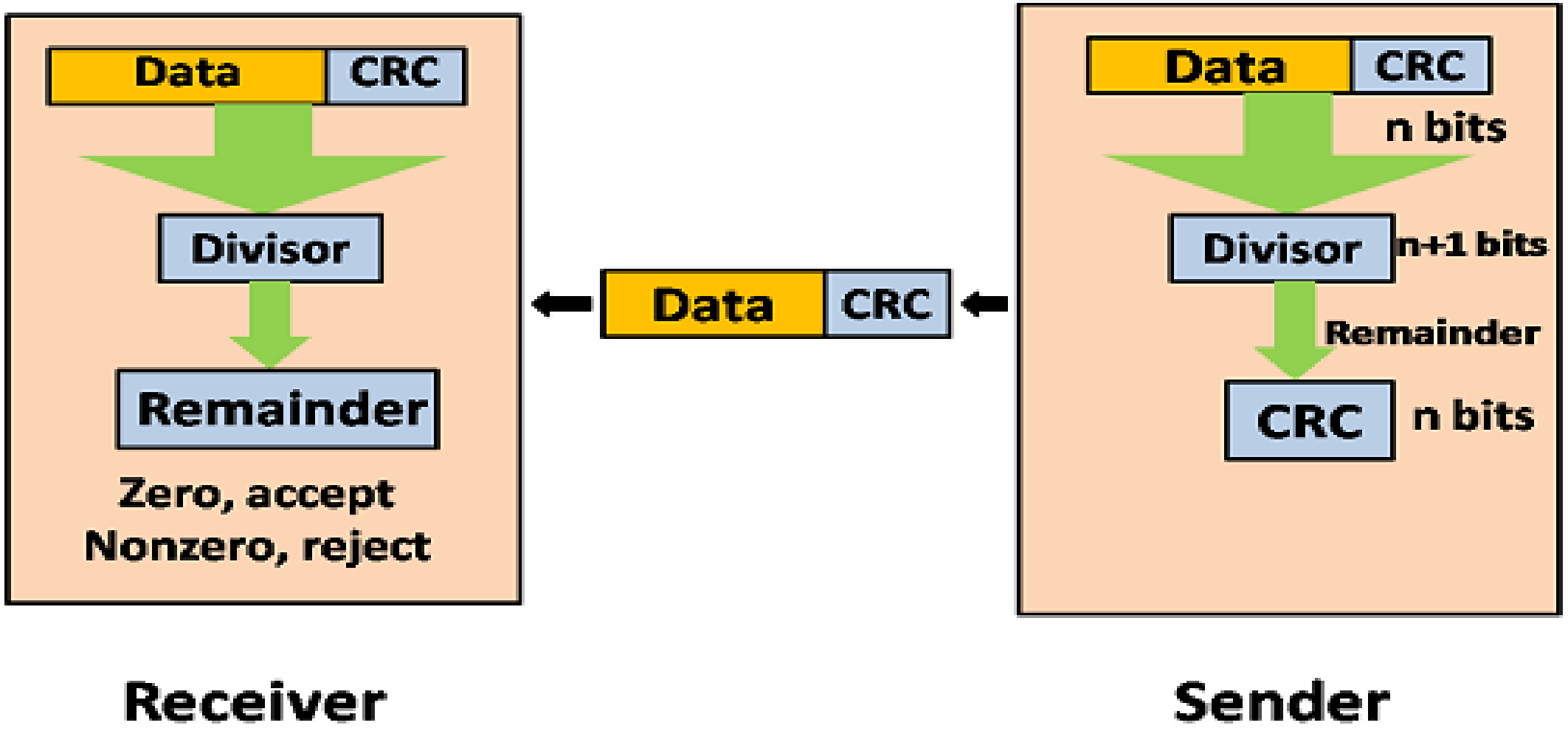
This newly generated unit is sent to the receiver.

Cyclic Redundancy Check (CRC)

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.**

Cyclic Redundancy Check (CRC)



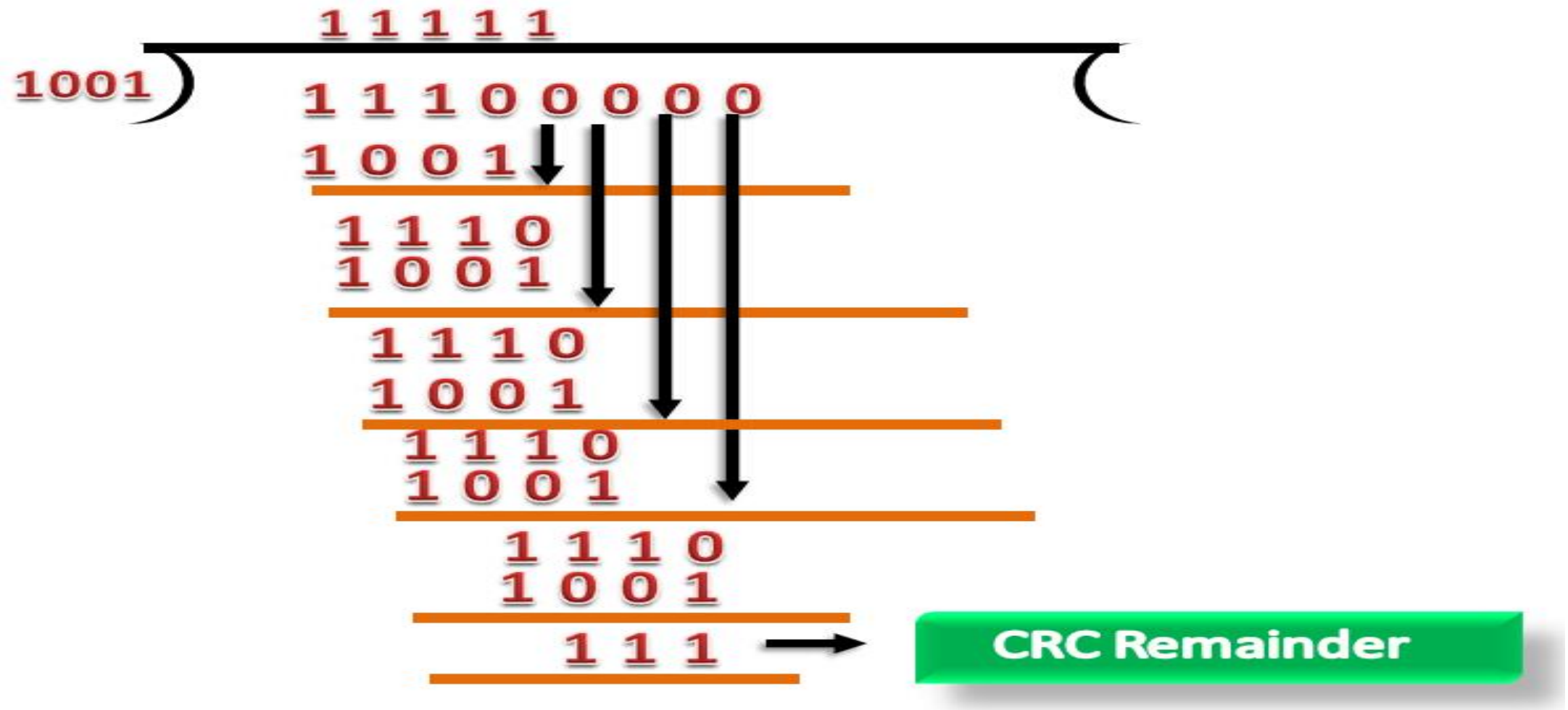
Cyclic Redundancy Check (CRC)

- 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.

Modulo 2 Division:

- **Modulo 2 Division**: The process of **modulo-2** binary **division** is the same as the familiar **division** process we use for decimal numbers. **Just that instead of subtraction, we use XOR here**. In each step, a copy of the divisor (or data) is XORed with the k bits of the dividend (or key).

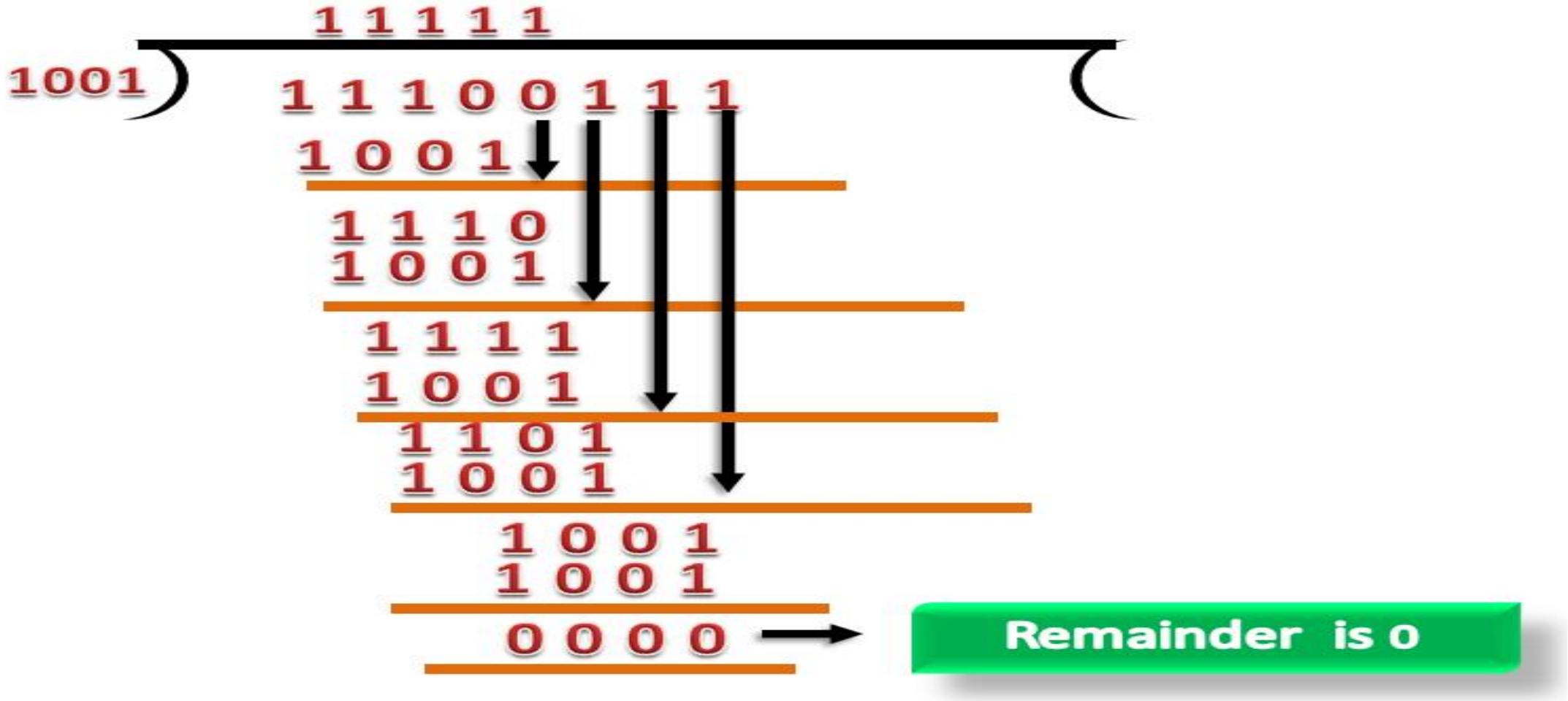
Cyclic Redundancy Check (CRC)



Cyclic Redundancy Check (CRC)

- **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.

Cyclic Redundancy Check (CRC)



Error Correction

- **Error Correction codes are used to detect and correct the errors.**

Error Correction can be handled in two ways:

- **Backward error correction:** Once the error is discovered, **the receiver requests the sender to retransmit** the entire data unit.
- **Forward error correction:** In this case, **the receiver uses the error-correcting code which automatically corrects the errors.**

Error Correction

- A single additional bit can detect the error, but cannot correct it.
- Suppose r is the number of redundant bits and d is the total number of the data bits. The number of redundant bits r can be calculated by using the formula:

$$2^r >= d+r+1$$

Error Correction: Hamming Code

- **Parity bits:** The bit which is appended to the original data of binary bits so that the total number of 1s is even or odd.
- **Even parity:** To check for even parity, if the total number of 1s is even, then the value of the parity bit is 0. If the total number of 1s occurrences is odd, then the value of the parity bit is 1.
- **Odd Parity:** To check for odd parity, if the total number of 1s is even, then the value of parity bit is 1. If the total number of 1s is odd, then the value of parity bit is 0.

Error Correction using Hamming code.

Algorithm of Hamming code:

- An information of **'d' bits are added to the redundant bits 'r' to form d+r.**
- The location of each of the (d+r) digits is assigned a decimal value.
- **The 'r' bits are placed in the positions 1,2,.....2^{k-1}.**
- At the receiving end, the parity bits are recalculated. The decimal value of the parity bits determines the position of an error.

Error Correction using Hamming code.

Total number of data bits 'd' = 4

Number of redundant bits r : $2^r >= d+(r+1)$ $2^r >= 4+(r+1)$

Therefore, the value of r is 3 that satisfies the above relation.

Total number of bits = d+r = 4+3 = 7;

Error Correction using Hamming code.

Relationship between Error position & binary number.

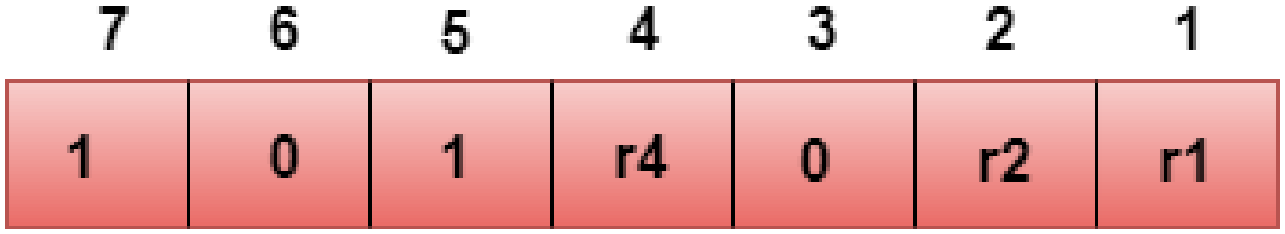
Let's understand the concept of Hamming code through an example:

Suppose the original data is 1010 which is to be sent.

Error Position	Binary Number
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Error Correction using Hamming code.

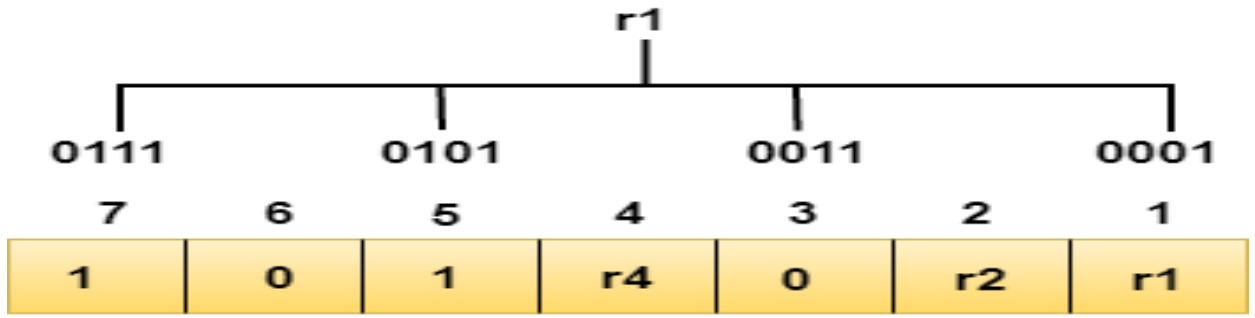
- Determining the position of the redundant bits
- The number of redundant bits is 3. The three bits are represented by r1, r2, r4. The position of the redundant bits is calculated with corresponds to the raised power of 2. Therefore, their corresponding positions are **1, 2^1, 2^2**.
- The position of r1 = 1
- The position of r2 = 2
- The position of r4 = 4



Representation of Data on the addition of parity bits:

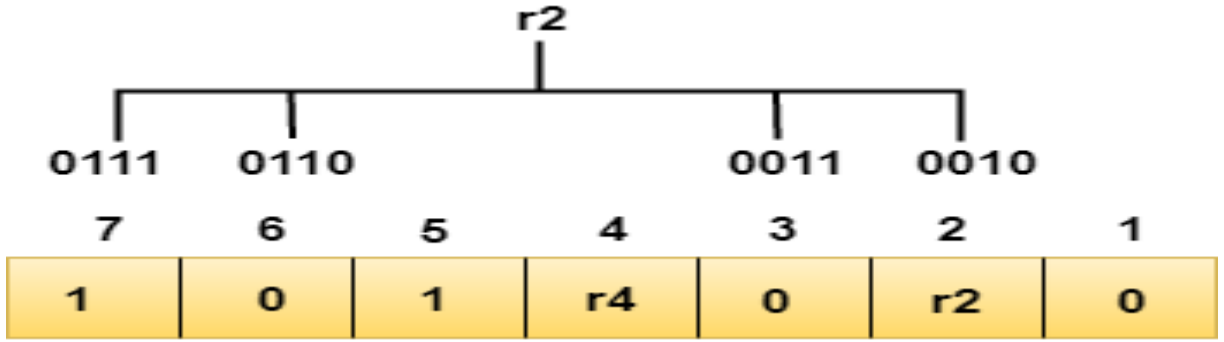
Error Correction using Hamming code.

- Determining the r1 bit
- The r1 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the first position
- We observe from the above figure that the bit positions that includes 1 in the first position are 1, 3, 5, 7. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r1 is **even, therefore, the value of the r1 bit is 0**.



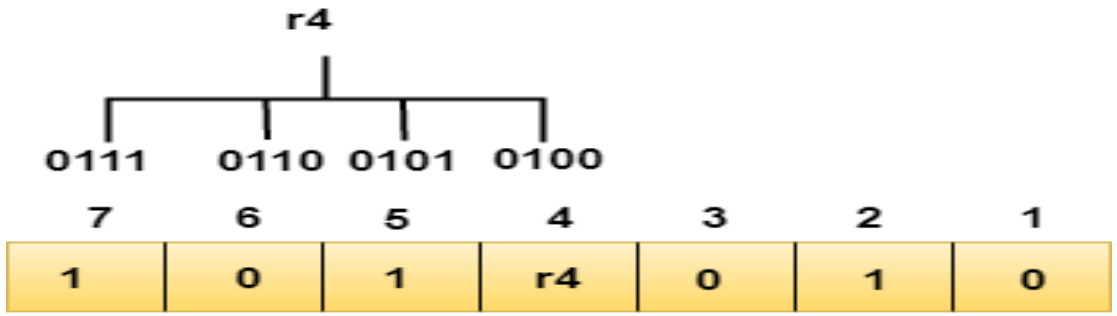
Error Correction using Hamming code.

- Determining r2 bit
- The r2 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the second position.
- We observe from the above figure that the bit positions that includes 1 in the second position are **2, 3, 6, 7**. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r2 is **odd, therefore, the value of the r2 bit is 1**



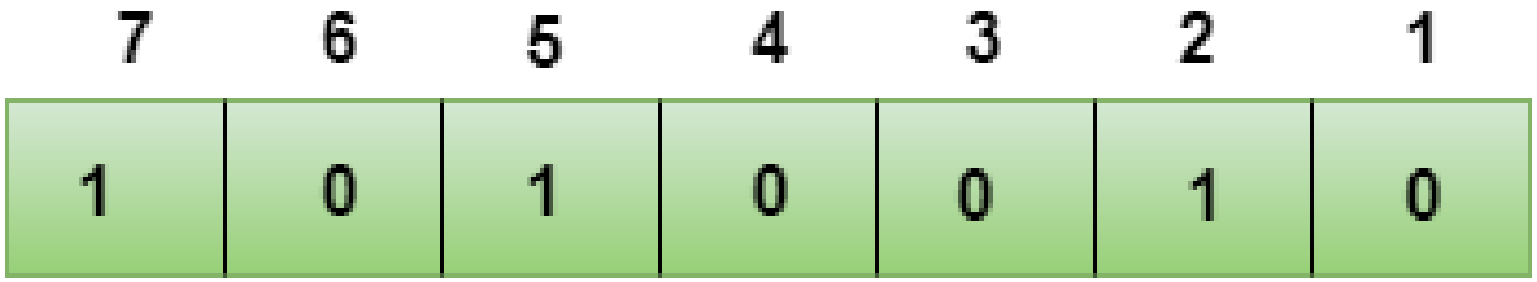
Error Correction using Hamming code.

- Determining r4 bit
- The r4 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the third position.
- We observe from the above figure that the bit positions that includes 1 in the third position are **4, 5, 6, 7**. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r4 is **even, therefore, the value of the r4 bit is 0**.



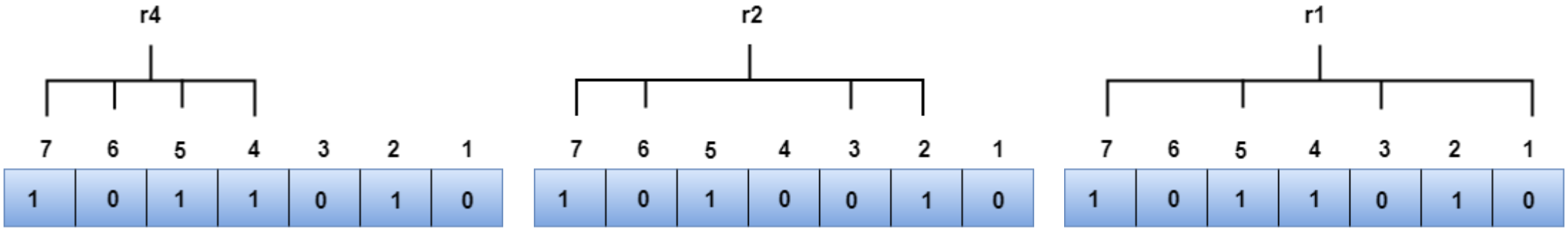
Error Correction using Hamming code.

- **Data transferred is given below:**
- Suppose the 4th bit is changed from 0 to 1 at the receiving end, then parity bits are recalculated.



Error Correction using Hamming code.

- R1 bit: The bit positions of the r1 bit are 1,3,5,7=1100
- R2 bit: The bit positions of r2 bit are 2,3,6,7= 1001
- R4 bit: The bit positions of r4 bit are 4,5,6,7.=1011



Error Correction using Hamming code.

- We observe from the above figure that the binary representation of r4 is 1011. Now, we perform the even-parity check, the total number of 1s appearing in the r4 bit is an odd number. Therefore, the value of r4 is 1.
- *The binary representation of redundant bits, i.e., r4 r2 r1 is 100, and its corresponding decimal value is 4. Therefore, the error occurs in a 4^{th} bit position. The bit value must be changed from 1 to 0 to correct the error.*

Note

Minimum length: 64 bytes (512 bits)

Maximum length: 1518 bytes (12,144 bits)

Figure 3.4 *Ethernet address in hexadecimal Notation*

d: Hexadecimal digit

$$d_1d_2 : d_3d_4 : d_5d_6 : d_7d_8 : d_9d_{10} : d_{11}d_{12}$$

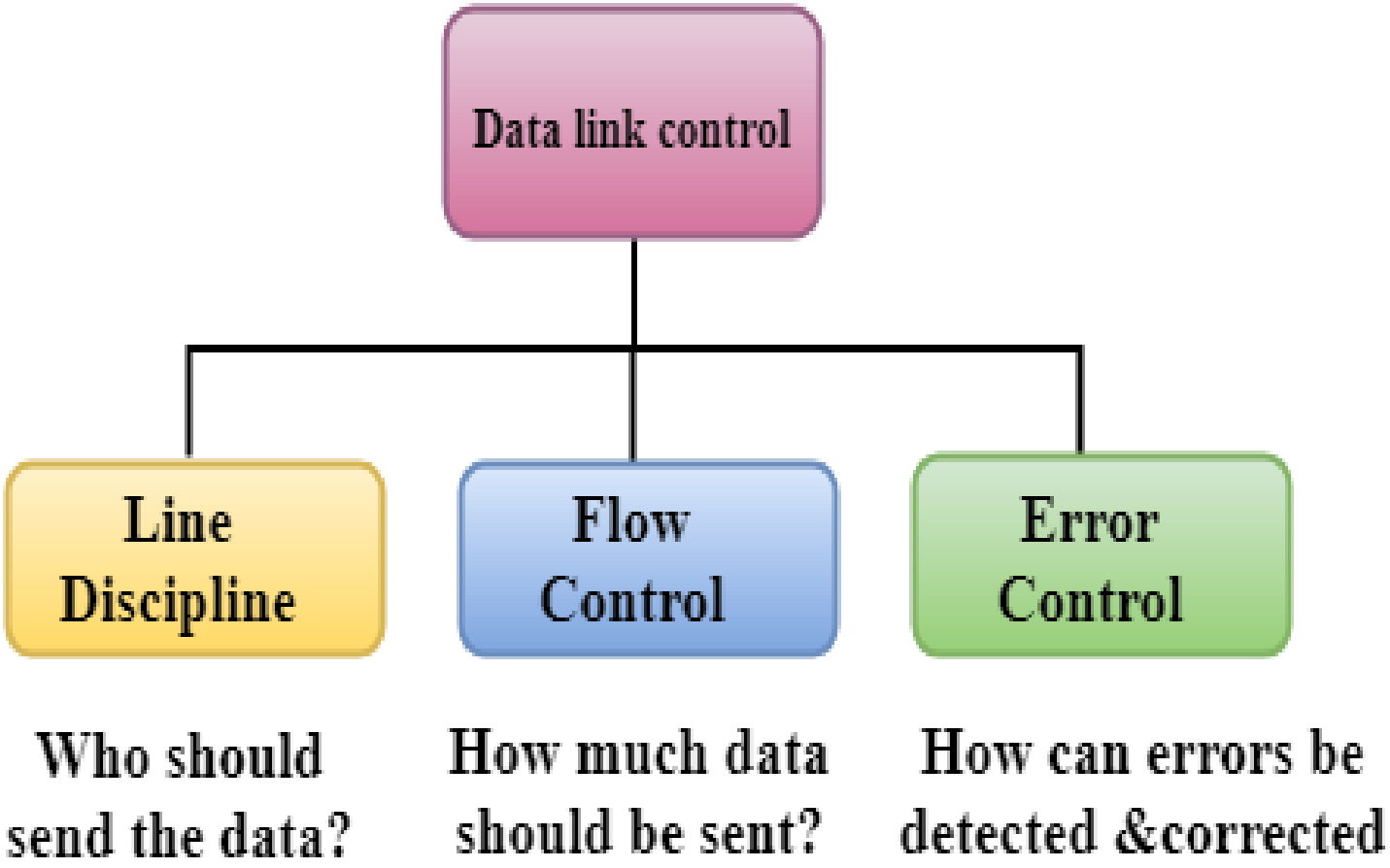
6 bytes = 12 hexadecimal digits = 48 bits

Services to the Network Layer

- The DLL uses the services offered by the physical layer. The primary function of this layer is to provide a well defined service interface to network layer above it.
- **The types of services provided can be of three types –**
  1. **Unacknowledged connectionless service**
  2. **Acknowledged connectionless service**
  3. **Acknowledged connection - oriented service**

The Data link layer provides three functions:

- Line discipline
- Flow Control
- Error Control



Line Discipline

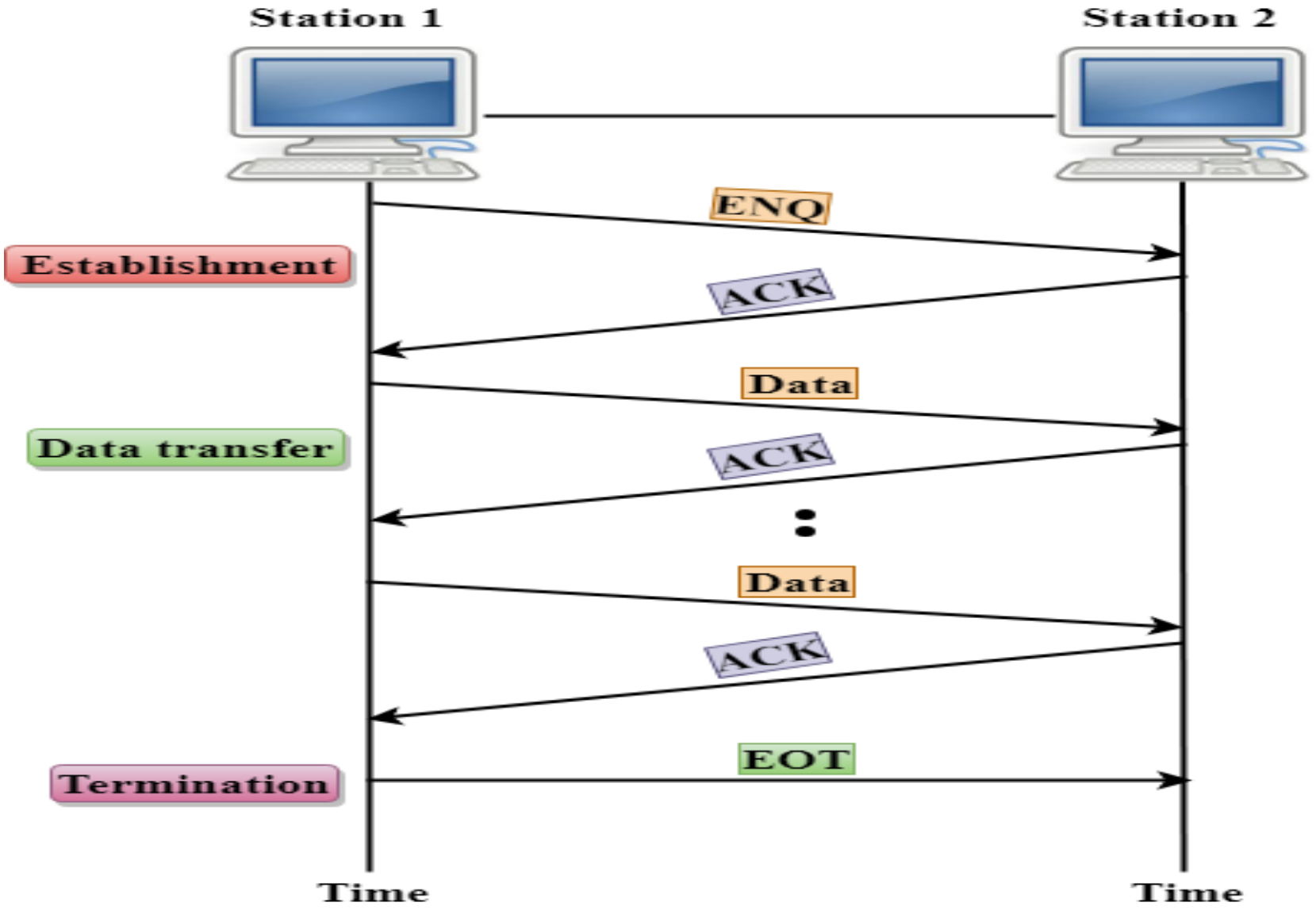
- Line Discipline is a functionality of the Data link layer that provides the coordination among the link systems.
- It determines which device can send, and when it can send the data.
- **Line Discipline can be achieved in two ways:**
  - ENQ/ACK
  - Poll/select
- **ENQ/ACK**
  - ENQ/ACK stands for Enquiry/Acknowledgement is used **when there is no wrong receiver available on the link and having a dedicated path between the two devices** so that the device capable of receiving the transmission is the intended one.
  - ENQ/ACK coordinates which device will start the transmission and whether the recipient is ready or not.

Line Discipline

- **Working of ENQ/ACK**
  - The transmitter transmits the frame called an Enquiry (ENQ) asking whether the receiver is available to receive the data or not.
  - The receiver responses either with the positive acknowledgement(ACK) or with the negative acknowledgement(NACK).
- **Following are the responses of the receiver:**

1. If the response to the ENQ is positive,
   - the sender will transmit its data,
   - and once all of its data has been transmitted,
   - the device finishes its transmission with an EOT (END-of-Transmission) frame.
2. If the response to the ENQ is negative, then the sender disconnects and restarts the transmission at another time.
3. If the response is neither negative nor positive,
   - the sender assumes that the ENQ frame was lost during the transmission and makes three attempts to establish a link before giving up

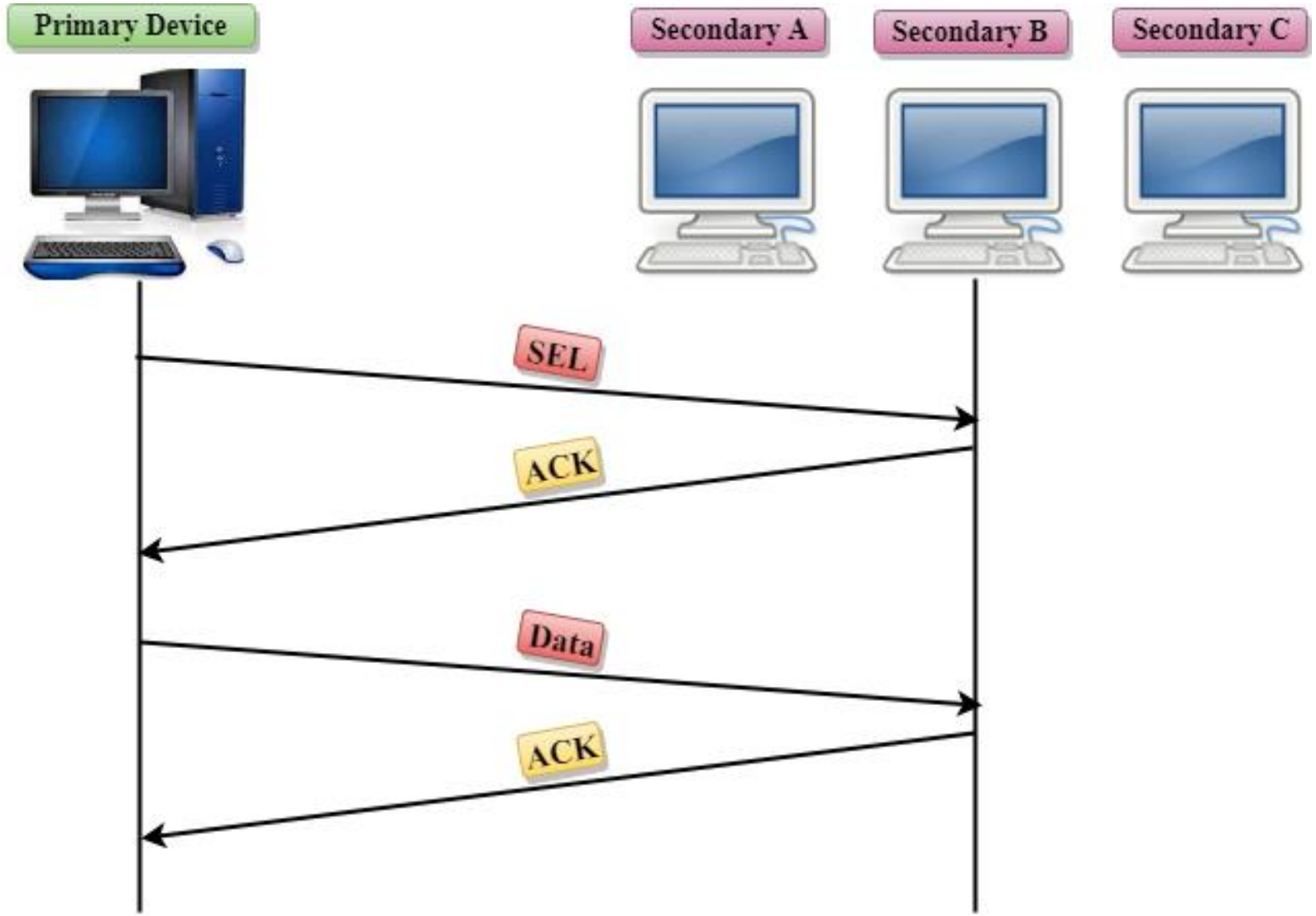
ENQ/ACK


Station 1
Station 2
ENQ
Establishment
ACK
Data
Data transfer
ACK
Data
ACK
EOT
Termination
Time
Time

Poll/Select

- works with those topologies where one device is designated as a primary station, and other devices are secondary stations.
- **Working of Poll/Select**
- In this, the primary device and multiple secondary devices consist of a single transmission line,
- and all the exchanges are made through the primary device even though the destination is a secondary device.
- The primary device has control over the communication link, and the secondary device follows the instructions of the primary device.
- The primary device determines which device is allowed to use the communication channel. Therefore, we can say that it is an initiator of the session.
- If the primary device wants to receive the data from the secondary device, it asks the secondary device that they anything to send, this process is known as polling.
- If the primary device wants to send some data to the secondary device, then it tells the target secondary to get ready to receive the data, this process is known as selecting.

Poll/Select


Primary Device
Secondary A
Secondary B
Secondary C
SEL
ACK
Data
ACK

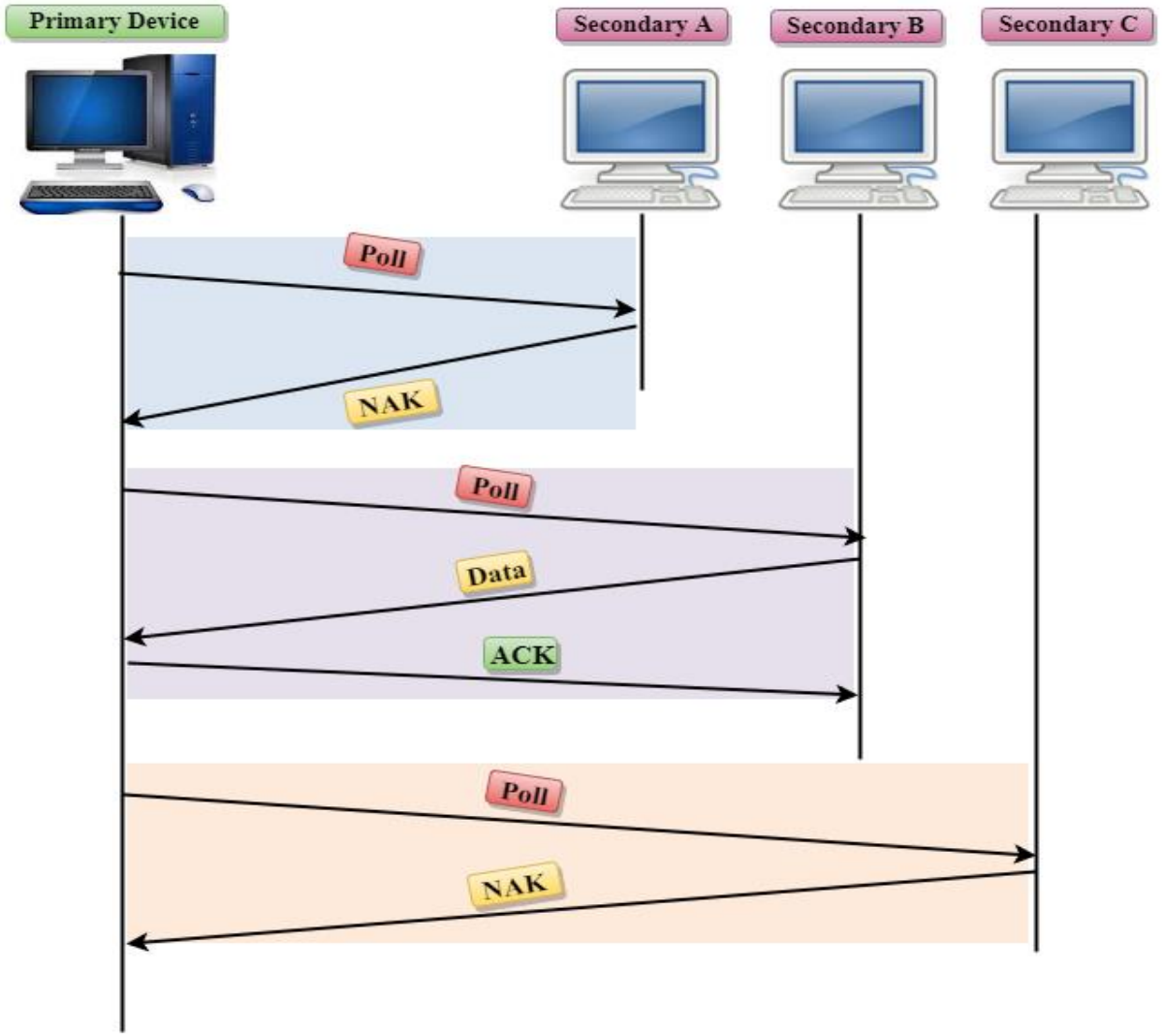
Poll

- The Poll mode is used **when the primary device wants to receive some data from the secondary device.**
- When a primary device wants to receive the data, then it asks each device whether it has anything to send.
- Firstly, the primary asks (poll) the first secondary device, if it responds with the NACK (Negative Acknowledgement) means that it has nothing to send. Now, it approaches the second secondary device, it responds with the ACK means that it has the data to send.
- The secondary device can send more than one frame one after another or sometimes it may be required to send ACK before sending each one, depending on the type of the protocol being used

Select

- The select mode is used when the **primary device has something to send.**
- When the primary device wants to send some data, then **it alerts the secondary device for the upcoming transmission by transmitting a Select (SEL) frame,** one field of the frame includes the address of the intended secondary device.
- When the **secondary device receives the SEL frame, it sends an acknowledgement** that indicates the secondary ready status.
- If the secondary device is ready to accept the data, **then the primary device sends two or more data frames to the intended secondary device.**
- Once the data has been transmitted, **the secondary sends an acknowledgement** specifies that the data has been received.

Poll


Primary Device
Secondary A
Secondary B
Secondary C
Poll
NAK
Poll
Data
ACK
Poll
NAK

Note

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

Flow Control

- It is a set of procedures that **tells the sender how much data it can transmit before the data overwhelms the receiver**.
- The **receiving device has limited speed and limited memory to store the data**. Therefore, the receiving device must be able to inform the sending device to stop the transmission temporarily before the limits are reached.
- It **requires a buffer**, a block of memory for storing the information until they are processed.
- **Two methods have been developed to control the flow of data:**
- **Stop-and-wait**
- **Sliding window**

Stop-and-wait

- In this, **the sender waits for an acknowledgement after every frame it sends.**
- **When acknowledgement is received, then only next frame is sent.** This process continues until the sender transmits the EOT (End of transmission) frame.
- **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.

Sliding Window

- The Sliding Window is a method of flow control in which a **sender can transmit the several frames before getting an acknowledgement.**
- In Sliding Window Control, multiple frames can be sent one after the another due to which capacity of the **communication channel can be utilized efficiently.**
- A **single ACK acknowledge multiple frames**.
- Sliding Window **refers to imaginary boxes at both the sender and receiver end.**
- The **window can hold the frames at either end, and it provides the upper limit on the number of frames** that can be transmitted before the acknowledgement.

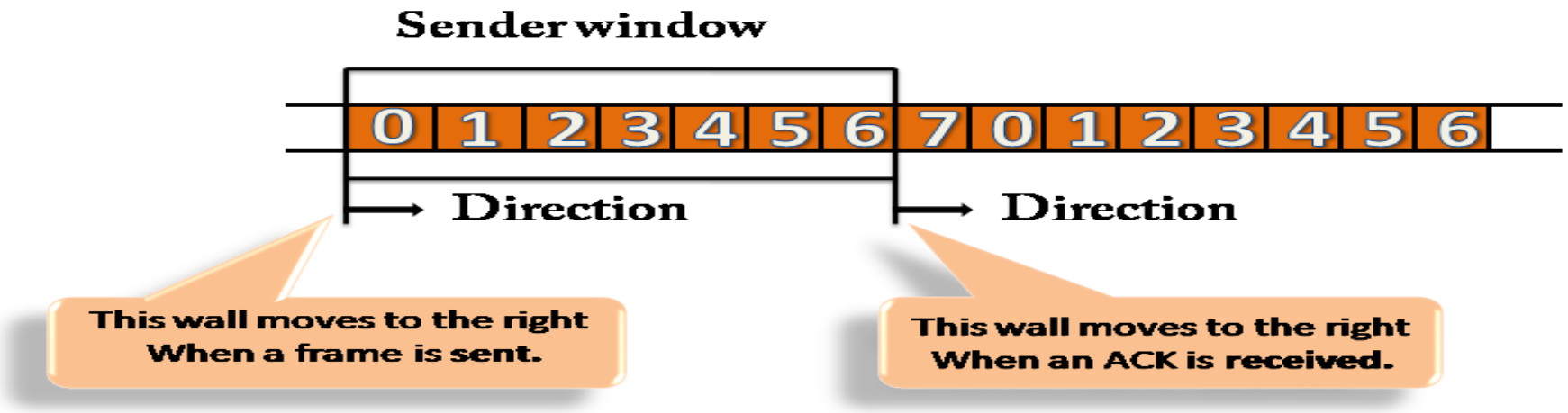
Sliding Window

- The **window has a specific size in which they are numbered as modulo-n means** that they are numbered from 0 to n-1. For example, if n = 8, the frames are numbered from 0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1.....
- **When the receiver sends the ACK, it includes the number of the next frame that it wants to receive.**
- For example, to acknowledge the string of frames ending with frame number 4, the receiver will send the ACK containing the number 5. When the sender sees the ACK with the number 5, it got to know that the frames from 0 through 4 have been received

Sliding Window

- **Sender Window**
- **At the beginning of a transmission, the sender window contains n-1 frames, and when they are sent out, the left boundary moves inward shrinking the size of the window.** For example, if the size of the window is w if three frames are sent out, then the number of frames left out in the sender window is w-3.
- Once the ACK has arrived, then the sender window expands to the number which will be equal to the number of frames acknowledged by ACK.
- For example, the size of the window is 7, and if frames 0 through 4 have been sent out and no acknowledgement has arrived, then the sender window contains only two frames, i.e., 5 and 6. Now, if ACK has arrived with a number 4 which means that 0 through 3 frames have arrived undamaged and the sender window is expanded to include the next four frames. Therefore, the sender window contains six frames (5,6,7,0,1,2)

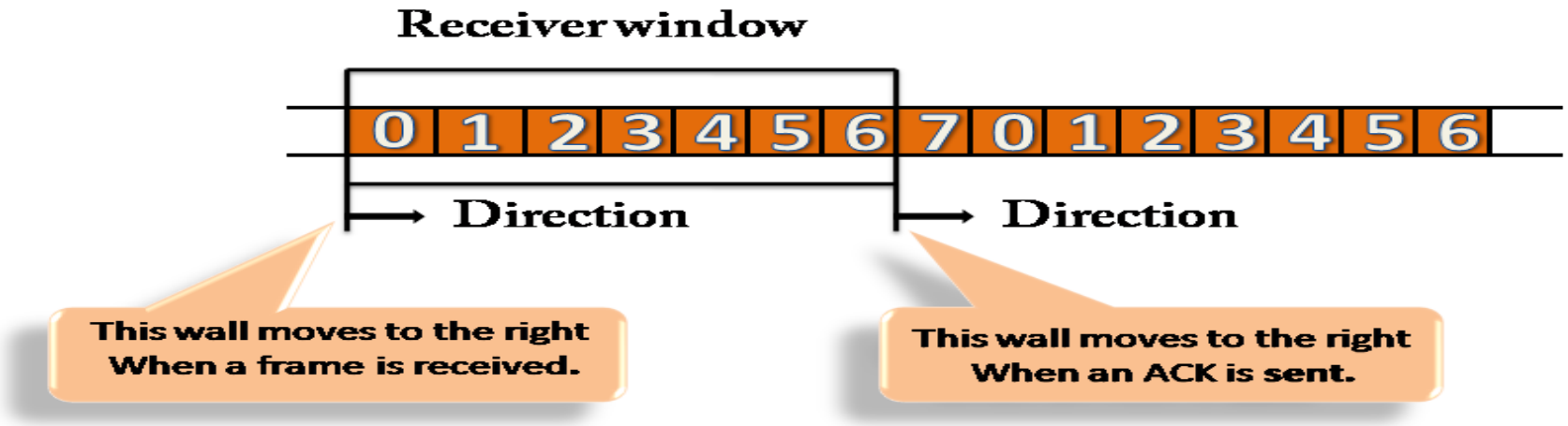
Sliding Window



Sliding Window

- **Receiver Window**
- At the beginning of transmission, **the receiver window does not contain n frames, but it contains n-1 spaces for frames.**
- When the **new frame arrives, the size of the window shrinks.**
- The receiver window represents the number of frames that can be received before an ACK is sent.
- For example, the size of the window is w, if three frames are received then the number of spaces available in the window is (w-3).
- **Once the acknowledgement is sent, the receiver window expands by the number equal to the number of frames acknowledged.**

Sliding Window



Note

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

Error Control

- Error Control is a technique of error detection and retransmission.

Error Control

- The data link layer ensures error free link for data transmission. The issues it caters to with respect to error control are −
  1. Dealing with transmission errors
  2. Sending acknowledgement frames in reliable connections
  3. Retransmitting lost frames
  4. Identifying duplicate frames and deleting them
  5. Controlling access to shared channels in case of broadcasting

Flow Control

- The data link layer regulates flow control so that a fast sender does not drown a slow receiver. When the sender sends frames at very high speeds, a slow receiver may not be able to handle it. There will be frame losses even if the transmission is error-free. The two common approaches for flow control are −
  1. Feedback based flow control
  2. Rate based flow control