UNIT-2

The Data Link Layer

Data Link Layer Design Issues

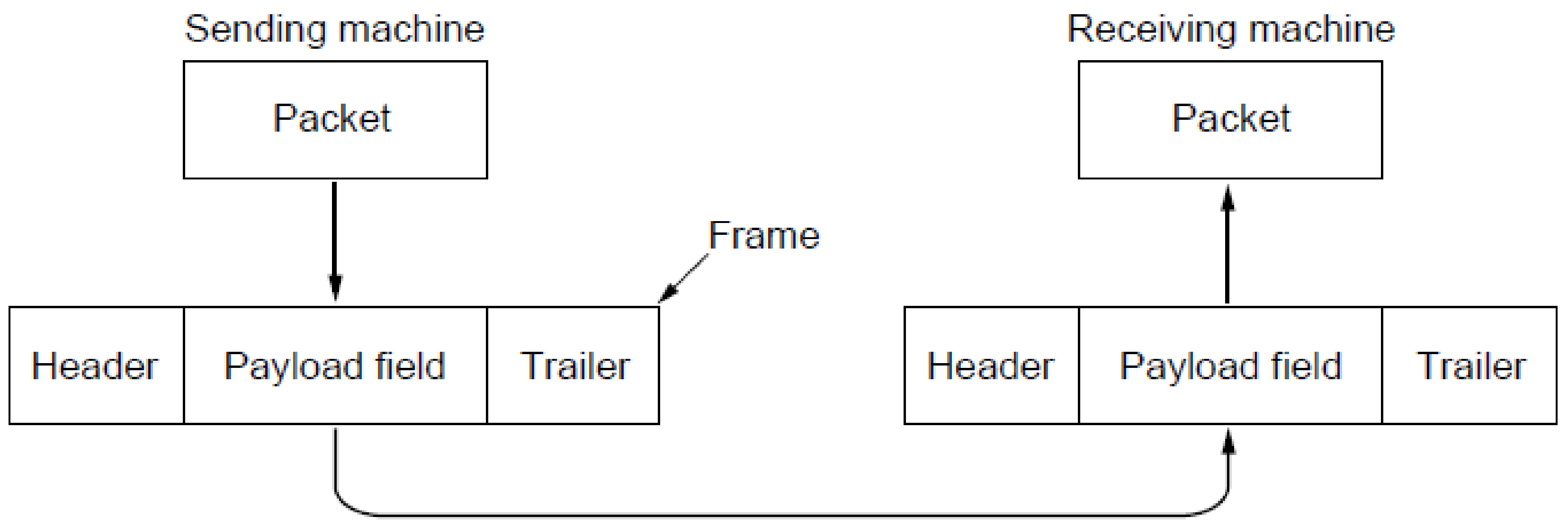
- **Services Provided to the Network Layer:** **Providing a well-defined service interface to the network layer**
- **Framing :** **Framing sequences of bytes**
- **Error Control:** **Detecting and Correction transmission errors**
- **Flow Control :** **Regulating the flow of data so that slow receivers are not swamped by fast senders.**

Data Link Layer Design Issues

- Physical layer delivers bits of information to and from data link layer. The functions of Data Link Layer are:
  - Providing a well-defined service interface to the network layer.
  - Dealing with transmission errors.
  - Regulating the flow of data so that slow receivers are not swamped by fast senders.
- Data Link layer takes the packets from Network layer, and encapsulates them into **frames**

Data Link Layer Design Issues

- Each frame has a
  - frame header – a payload field for holding the packet, and
  - frame trailer.
- Frame Management is what Data Link Layer does.

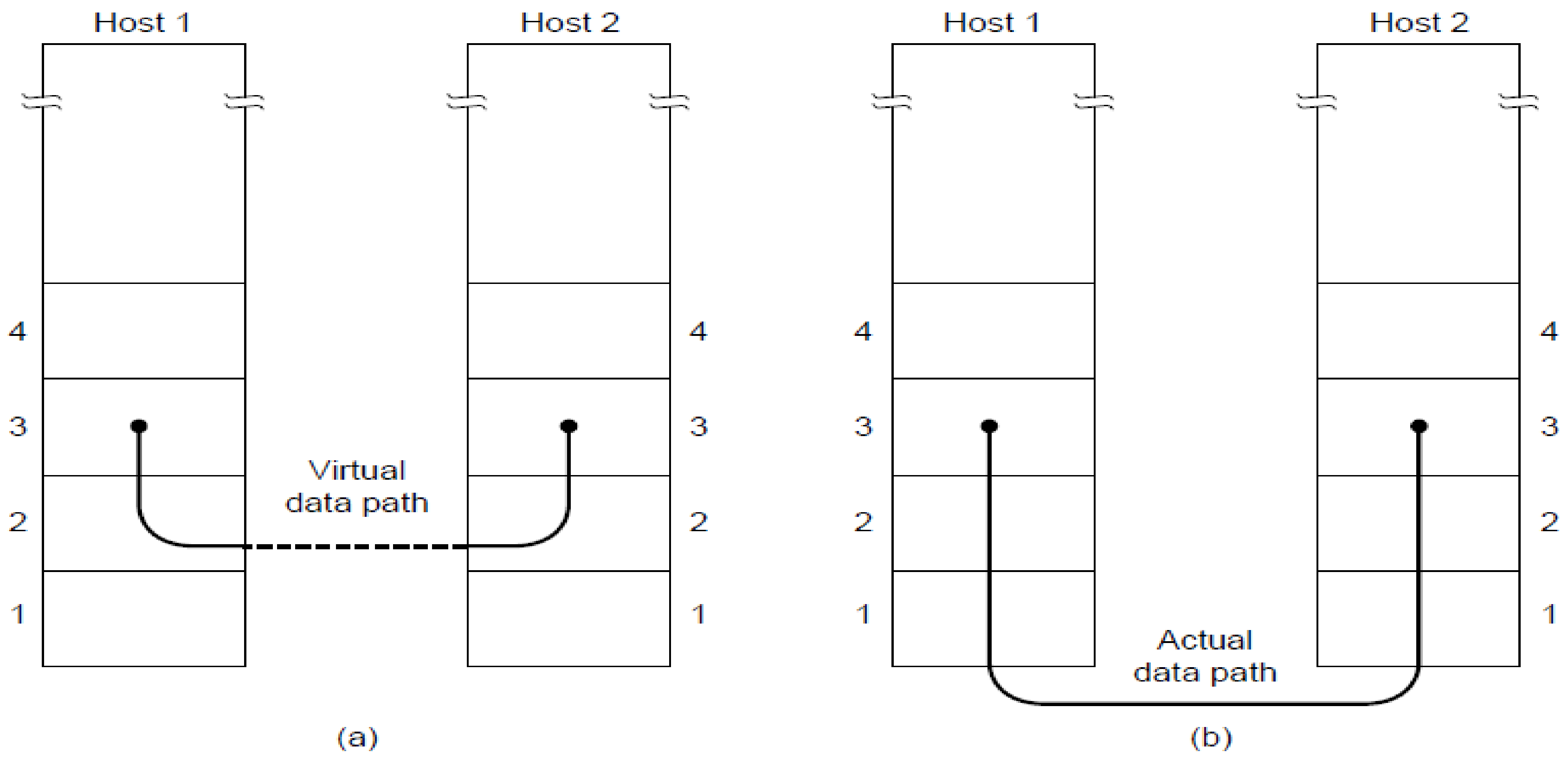


Relationship between packets and frames.

Services Provided to the Network Layer

- Principal Service Function of the data link layer is to transfer the data from the network layer on the source machine to the network layer on the destination machine.
- Process in the network layer that hands some bits to the data link layer for transmission.
- Job of data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there (see figure in the next slide).

Services Provided to the Network Layer



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

Possible Services Offered

1. Unacknowledged connectionless service.
2. Acknowledged connectionless service.
3. Acknowledged connection-oriented service.

Unacknowledged Connectionless Service

- It consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them.

- Ex: Ethernet, Voice over IP, etc. in all the communication channel where real time operation is more important than quality of transmission.

Acknowledged Connectionless Service

- Each frame send by the Data Link layer is acknowledged and the sender knows if a specific frame has been received or lost.

- Typically the protocol uses a specific timer that it has elapsed without getting acknowledgment, it will re-send the frame.

- This service is useful for communication when an unreliable channel is being utilized (e.g., 802.11 WiFi).

- Network layer does not know the frame size of the packets and other restrictions of the data link layer. Hence it becomes necessary for data link layer to have some mechanism to optimize the transmission.
- Example : Satellite channel communication,

Acknowledged Connection Oriented Service

- Source and Destination establish a connection first.
- Each frame sent is numbered
  - Data link layer guarantees that each frame sent is indeed received.
  - It guarantees that each frame is received only once and that all frames are received in the correct order.
- Examples:
  - Long-distance telephone communication, etc.

Acknowledged Connection Oriented Service

- Three distinct phases:

1. Connection is established by having both side initialize variables and counters needed to keep track of which frames have been received and which ones have not.

2. One or more frames are transmitted.

3. Finally, the connection is released – freeing up the variables, buffers, and other resources used to maintain the connection.

Framing

- Framing is the process of **encapsulating a network layer packet into a data link layer frame** so that it can be transmitted over a physical medium. It provides a way to **identify the start and end of a message**.
- Frame Structure

Component	**Description**
Header	Control information like addresses, etc.
Payload	Actual data from the network layer
Trailer	Error-checking data like CRC or checksum

- Frames can be of two types :

  - **Fixed Length Framing - Not Efficient**
  - **Variable Length Framing - Very Efficient**

Framing

Why is Framing Needed?

1. **Synchronization**: Helps receiver know where the frame starts and ends.
2. **Error Control**: Errors can be detected and corrected within a frame.
3. **Flow Control**: Frames regulate data transmission rate between sender and receiver.
4. **Reliable Delivery**: Easier to detect and retransmit only the damaged frame, not the entire stream.

Variable Length Framing Methods

1. Byte count.

2. Flag bytes with byte stuffing.

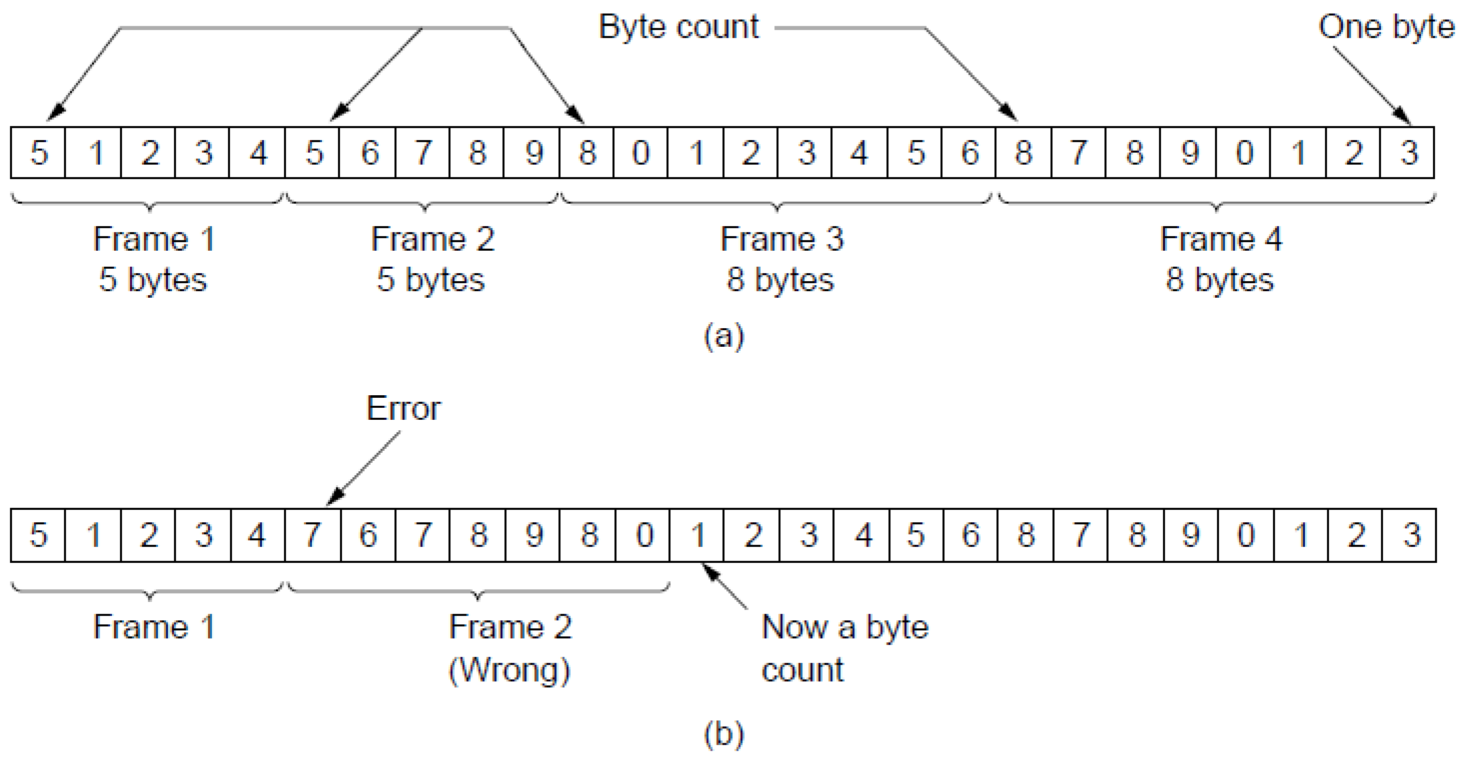
3. Flag bits with bit stuffing.

4. Physical layer coding violations.

Byte Count Framing Method

- It uses a field in the header to specify the number of bytes in the frame.
- Once the header information is being received it will be used to determine end of the frame.
- Trouble with this algorithm is that when the count is incorrectly received, the destination will get out of synchro. with transmission.
- Destination may be able to detect that the frame is in error but it does not have a means (in this algorithm) how to correct it.

Byte Count Framing Method



A byte stream. (a) Without errors. (b) With one error.

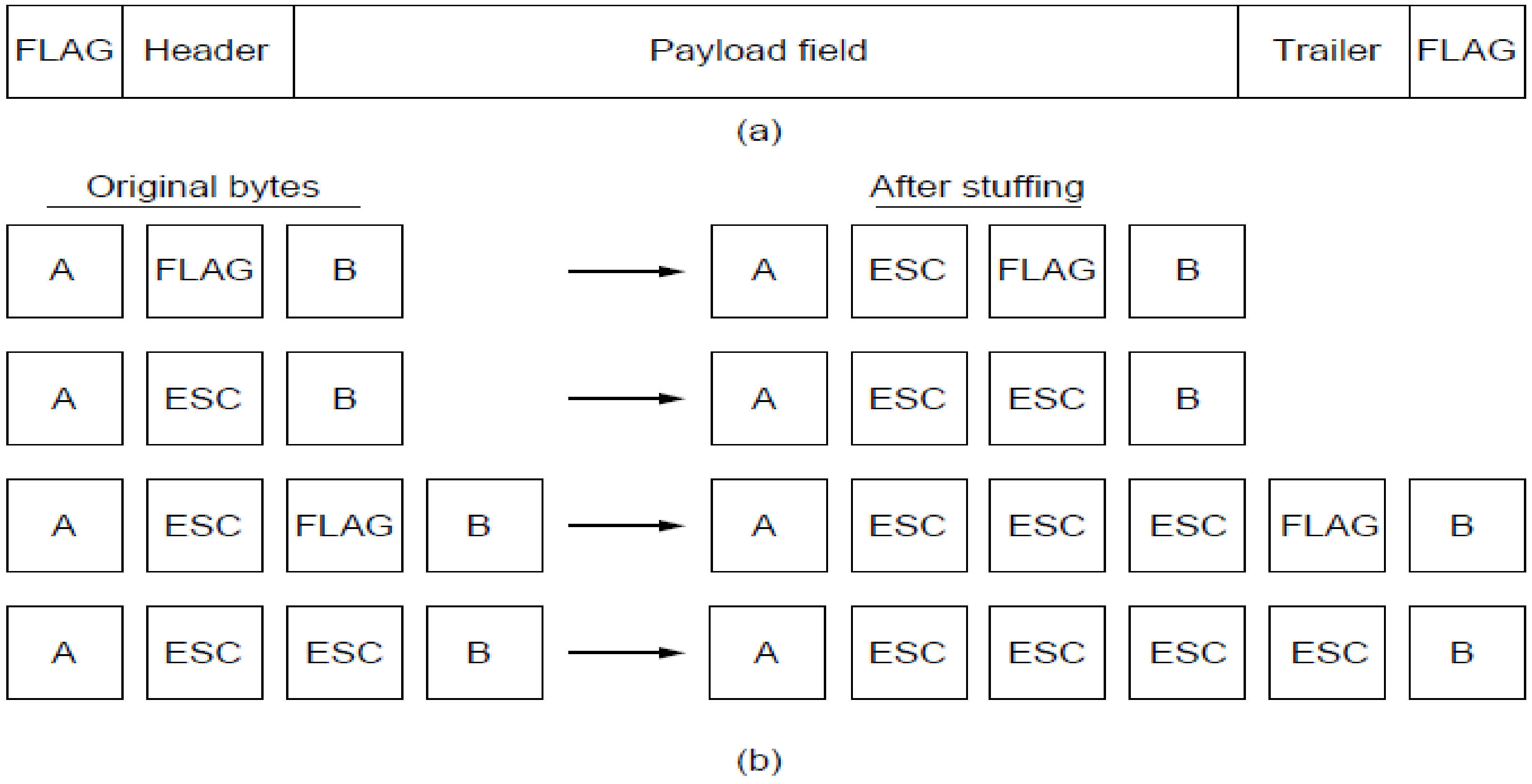
Flag Bytes with Byte Stuffing

- This method gets around the boundary detection of the frame by having each appended by the frame start and frame end special bytes.

- If they are the same (beginning and ending byte in the frame) they are called **flag bytes**.

- If the actual data contains a byte that is identical to the FLAG byte (e.g., picture, data stream, etc.) the convention that can be used is to have escape character inserted just before the "FLAG" character.

Framing



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

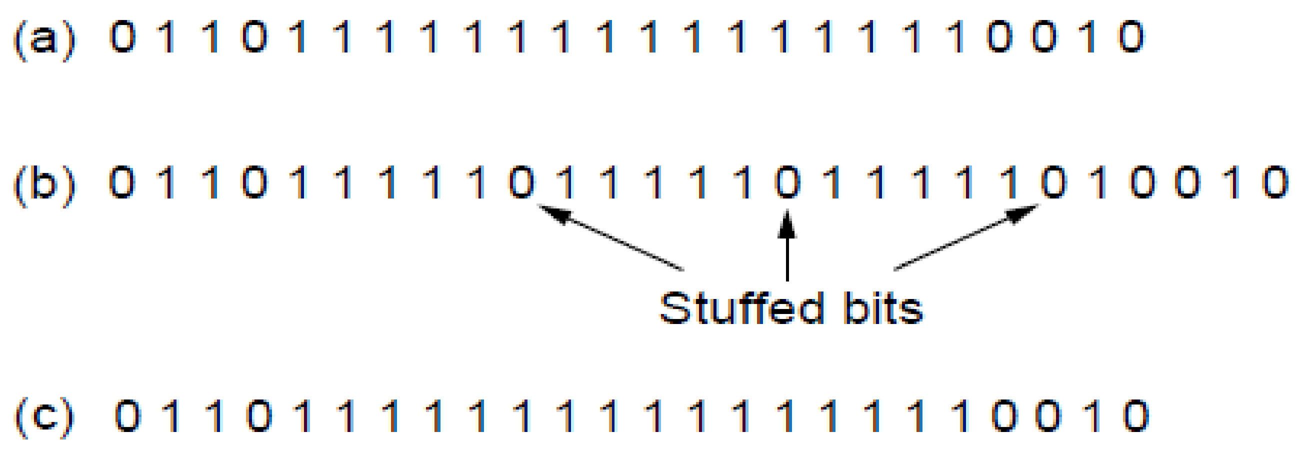
Flag Bits with Bit Stuffing

- This method achieves the same thing as Byte Stuffing method by using Bits (1) instead of Bytes (8 Bits).

- It was developed for High-level Data Link Control (HDLC) protocol.

- Each frame begins and ends with a special bit pattern:
  - 01111110 or 0x7E <- Flag Byte
  - Whenever the sender's data link layer encounters five consecutive 1s in the data it automatically stuffs a 0 bit into the outgoing bit stream.
  - USB communication uses bit stuffing.

Framing (3)


(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0
Stuffed bits
(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Bit stuffing.

(a) The original data from NL.

(b) The data as they appear on the line.

(c) The data as they are stored in the receiver's memory after destuffing.

Physical Layer coding violation

- Using a shortcut from the physical layer.
- When the physical layer uses encoding mechanism with redundant bits, using those redundant bits for representing beginning or ending of a frame

Error Control

- ➢**Error Control** refers to the techniques used in **computer networks** to **detect and correct errors** that occur during data transmission over unreliable communication channels.

Why is Error Control Important?

1. **Noisy Channels**: Physical media like cables or wireless links may introduce errors (bit flips, losses).
2. **Data Integrity**: Ensures the **correctness of transmitted data**.
3. **Reliable Communication**: Especially crucial for TCP/IP-based systems.

Error Control

- ➢Types of Errors

Error Type	**Description**
Single-bit error	Only one bit is changed (e.g., 0 → 1)
Burst error	Multiple bits are changed due to noise
Packet loss	Entire frames or packets are lost
Duplicate frames	Receiver gets the same frame more than once
Out-of-order frames	Frames arrive in wrong order

Error Control

Components of Error Control

1. **Error Detection** – Finding out if an error occurred.
2. **Error Correction** – Fixing the detected errors.
3. **Acknowledgements (ACK)** – Positive feedback: frame received correctly.
4. **Negative Acknowledgements (NAK)** – Feedback: error detected.
5. **Retransmission** – Sender resends data if error is detected or ACK is not received

Error Control

Error Detection Techniques

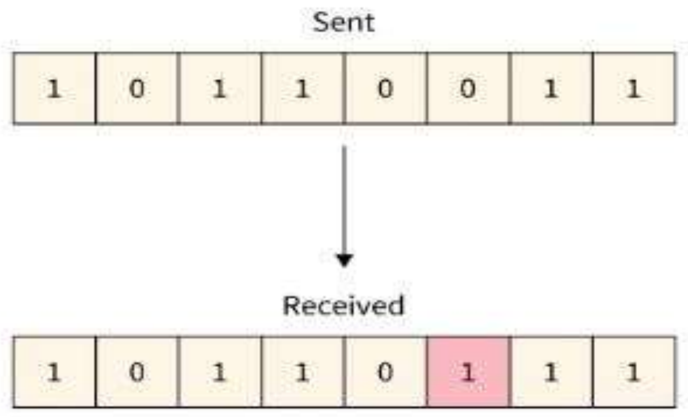
Method	Description
Parity Check	Adds 1 parity bit to data (even/odd parity)
Checksum	Adds all data segments, sends sum
Cyclic Redundancy Check (CRC)	Uses polynomial division to detect errors
Hamming Codes	Uses parity bits to detect and correct errors

Types Of Errors

- **Single-bit Error**
- **Multiple-bit Error**
- **Burst Error**
- **Errors in Communication**
- When the information **received at the receiver** end does not match the sent data. At the time of transmission, errors are introduced into the binary data sent from the sender to the receiver due to noise during transmission. This means that a bit having a **0** value can change to **1** and a bit having a **1** value can change to **0**.

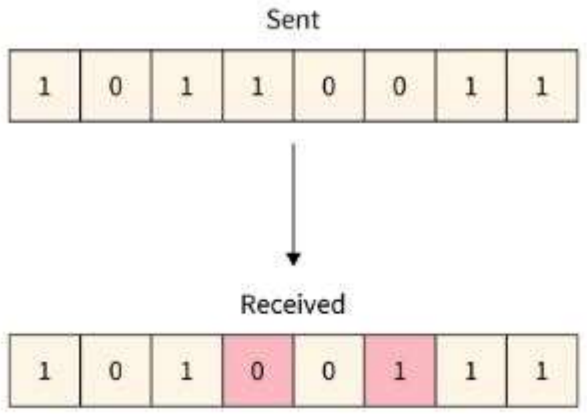
➢ Single-bit Error

- Typically, only **one bit of the frame** received is corrupt, and the **corrupted bit** can be located anywhere in the frame.
- Refer to the below image for the **single-bit error**



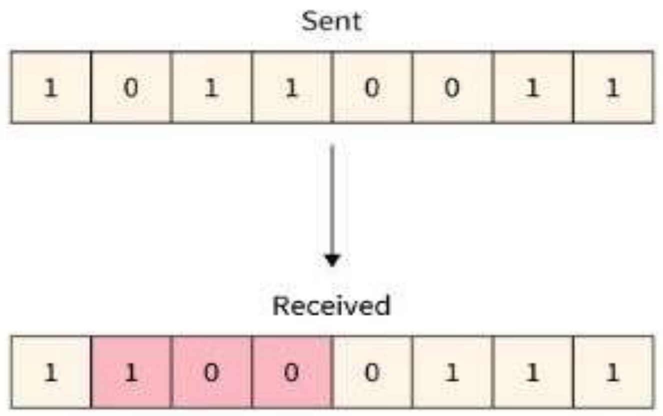
➢Multiple-bit Error

➢More than **one bit received** in the frame is found to be corrupted. Refer to the below image for the **multiple-bit error**



Burst Error

- More than one consecutive bit is corrupted in the received frame.
- Refer to the below image for the **burst-bit error**



Error Control

Error Correction Techniques

Technique	Description
Automatic Repeat reQuest (ARQ)	Sender retransmits frame if error is detected
Forward Error Correction (FEC)	Adds redundancy in data so receiver can correct it
Hamming Code	Detects and corrects single-bit errors

Flow Control

➢ **Flow Control** is a **data link layer and transport layer mechanism** that ensures the **sender does not overwhelm the receiver** by sending data too fast.

Why Flow Control is Needed

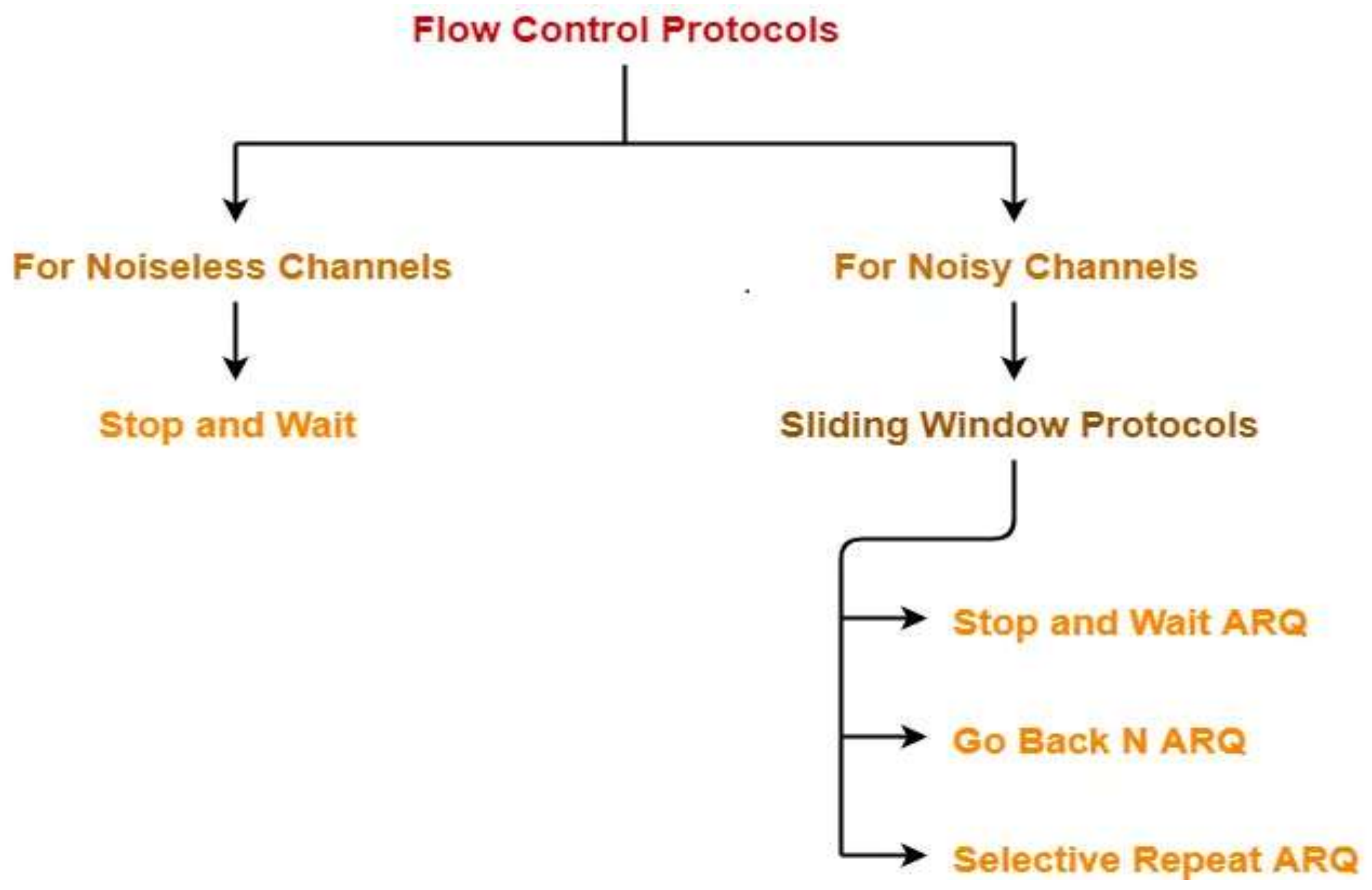
- Imagine the sender is a superfast computer and the receiver is a small device. Without flow control:
- The receiver may **not process data fast enough**.
- **Buffer overflow** can occur.
- Data may be **lost or corrupted**.
- So, flow control helps maintain a **balanced and reliable communication** between sender and receiver.

Flow Control

Goals of Flow Control

1. Prevent **buffer overflow** at receiver.
2. Ensure **efficient utilization** of network resources.
3. Avoid **congestion** and retransmissions.

Flow Control



Stop-and-Wait Protocol

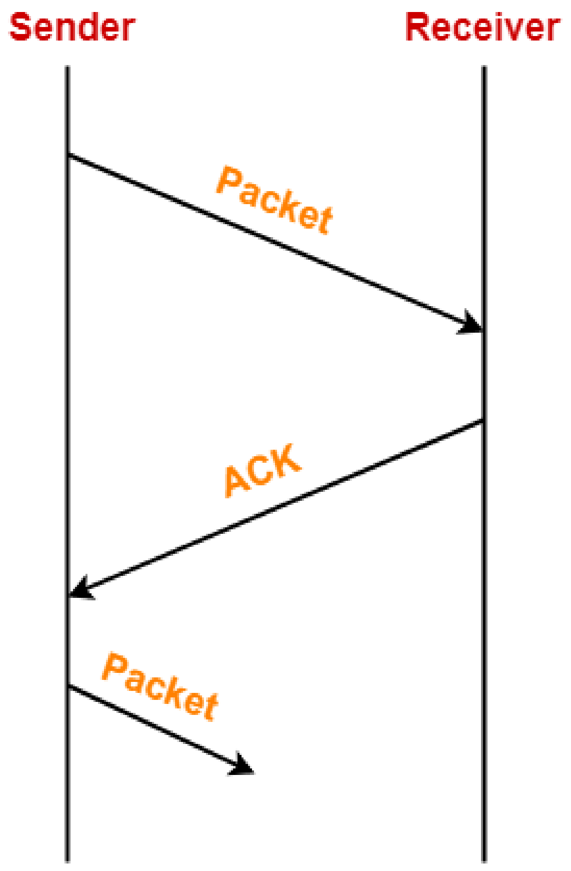
Working:

- Sender sends a data packet to the receiver.
- Sender stops and waits for the acknowledgement for the sent packet from the receiver.
- Receiver receives and processes the data packet.
- Receiver sends an acknowledgement to the sender.
- After receiving the acknowledgement, sender sends the next data packet to the receiver.

Stop-and-Wait protocol

Analysis:

- Sender puts the data packet on the transmission link.
- Data packet propagates towards the receiver's end.
- Data packet reaches the receiver and waits in its buffer.
- Receiver processes the data packet.
- Receiver puts the acknowledgement on the transmission link.
- Acknowledgement propagates towards the sender's end.
- Acknowledgement reaches the sender and waits in its buffer.
- Sender processes the acknowledgement.



Stop and Wait Protocol

Stop-and-Wait

- **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.

Problems of Stop-and-wait protocol

1. Lost Data Frames
2. Lost Acknowledgment Frames
3. Delayed Acknowledgment

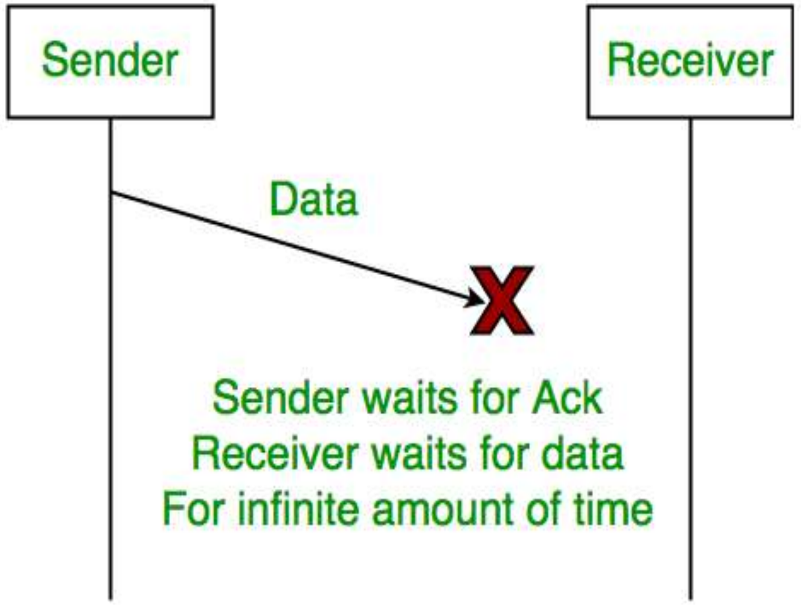
- The above 3 problems can be rectified in stop-and-wait ARQ protocol

Stop-and-Wait ARQ

Lost data frames:

Assume the sender transmits the data packet and it is lost. The receiver has been waiting for the data for a long time. Because the data is not received by the receiver, it does not transmit an acknowledgment. The sender does not receive an acknowledgment, it will not send the next packet. This problem is caused by a loss of data.

- In stop-and-wait ARQ, timer is used to retransmit the same frame. At sender side, timer start wher a packet is transmitted. If the timer expires, the Sender retransmits the same packet.

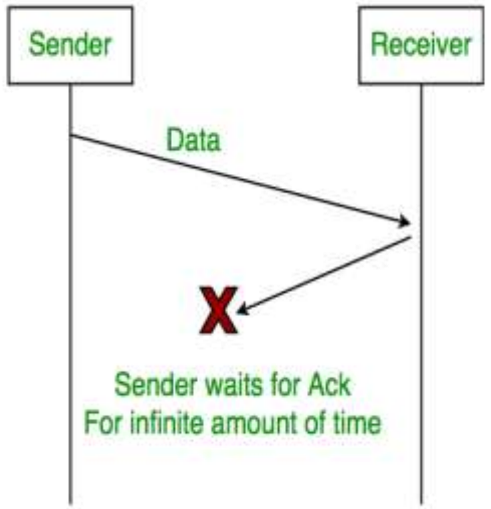


Stop-and-wait ARQ

Lost Acknowledgment (Duplicate Frame problem)

- Assume the sender sends the data, which is also received by the receiver. The receiver sends an acknowledgment after receiving the packet. In this situation, the acknowledgment is lost in the network. The sender does not send the next data packet because it does not receive acknowledgement, under the stop and wait protocol, the next packet cannot be transmitted until the preceding packet's acknowledgment is received.
- In Stop-and-wait ARQ, Sequence Number for frames is used to

  avoid duplication of frames at receiver.

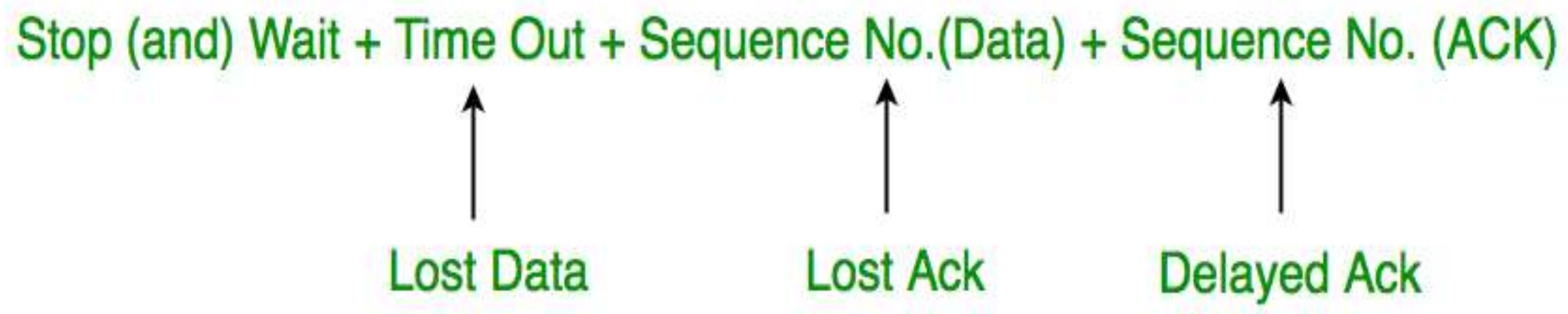


Stop-and-wait ARQ

Delayed Acknowledgement

Assume the sender sends the data, which is also received by the receiver. The receiver then transmits the acknowledgment, which is received after the sender's timeout period. After a timeout on the sender side, a long-delayed acknowledgement might be wrongly considered as acknowledgement of some other recent packet.

- It uses acknowledgement number for received frames by the receiver to the sender.
- The sender can identify for which frame that acknowledgement is by using ack. number.


Stop (and) Wait + Time Out + Sequence No.(Data) + Sequence No. (ACK)
Lost Data
Lost Ack
Delayed Ack

Sliding Window Protocols

Sliding Window

- This protocol improves the efficiency of stop and wait protocol by allowing multiple frames to be transmitted before receiving an acknowledgment.
- Both the sender and the receiver has finite sized buffers called windows. The sender and the receiver agrees upon the number of frames to be sent based upon the buffer size.
- The sender sends multiple frames in a sequence, without waiting for acknowledgment. When its sending window is filled, it waits for acknowledgment. On receiving acknowledgment, it advances the window and transmits the next frames, according to the number of acknowledgments received.

Some of the protocols are :

- Stop-and-Wait ARQ
- Go-Back-N
- Selective-Repeat ARQ

Sliding Window Protocol – Go-Back-N Protocol

- The number of frames that can be sent at a time totally depends on the size of the sender's window. So, we can say that 'N' is the number of frames that can be sent at a time before receiving the acknowledgment from the receiver.
- If the acknowledgment of a frame is not received within an agreed-upon time period, then all the frames available in the current window will be retransmitted.
- The sequence number of the outbound frames depends upon the size of the sender's window. Suppose the sender's window size is 2, and we have ten frames to send, then the sequence numbers will not be 1,2,3,4,5,6,7,8,9,10.
- N is the sender's window size.
- If the size of the sender's window is 4 then the sequence number will be 0,1,2,3,0,1,2,3,0,1,2, and so on.

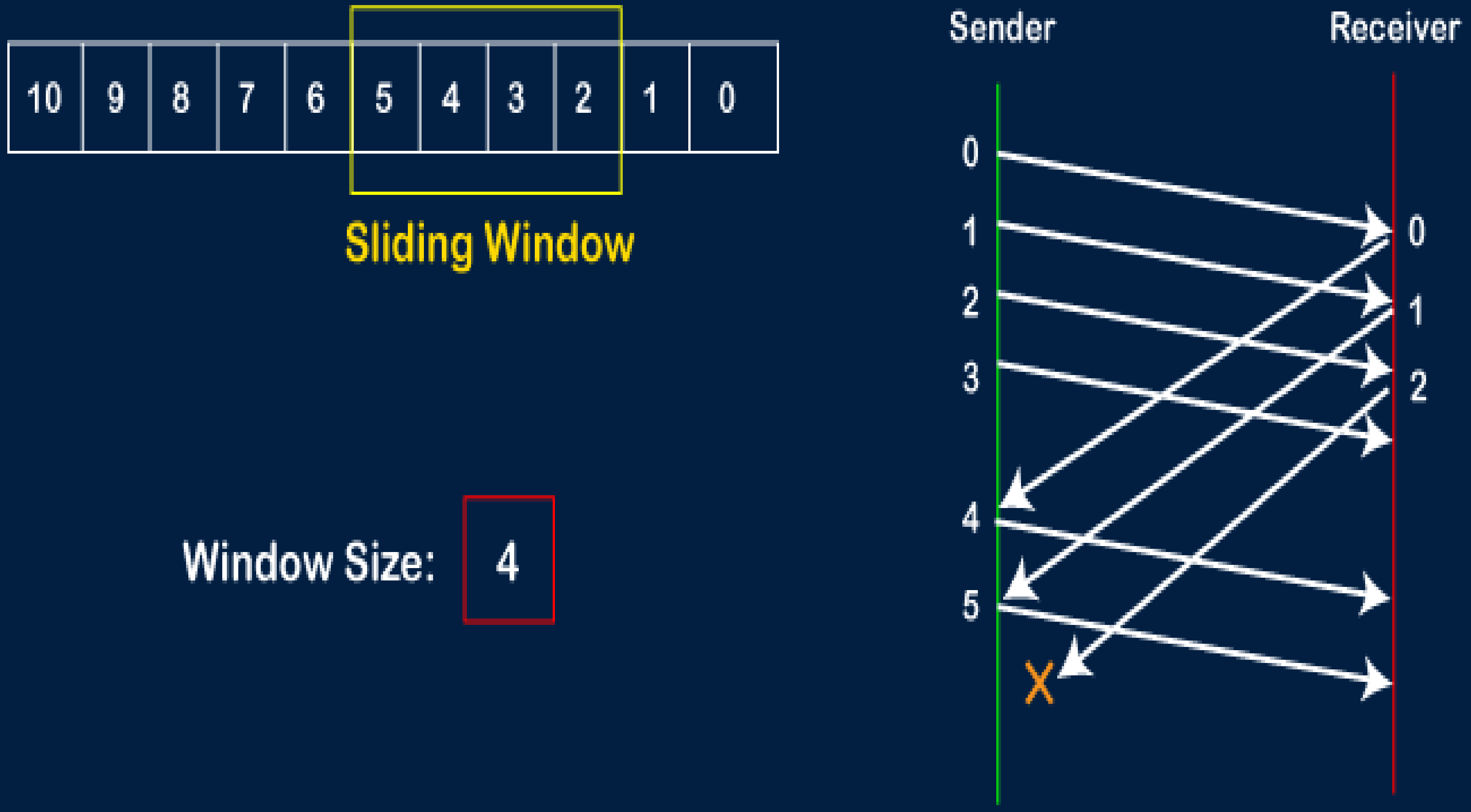
Working of Go-Back-N ARQ

- Suppose there are a sender and a receiver, and let's assume that there are 11 frames to be sent. These frames are represented as 0,1,2,3,4,5,6,7,8,9,10, and these are the sequence numbers of the frames. Mainly, the sequence number is decided by the sender's window size. But, for the better understanding, we took the running sequence numbers, i.e., 0,1,2,3,4,5,6,7,8,9,10. Let's consider the window size as 4, which means that the four frames can be sent at a time before expecting the acknowledgment of the first frame.
- **Step 1:** Firstly, the sender will send the first four frames to the receiver, i.e., 0,1,2,3, and now the sender is expected to receive the acknowledgment of the 0th frame.
- Step 2 : If the receiver has sent the acknowledgment for the 0 frame, and the receiver has successfully received it.
- Step 3 : The sender will then send the next frame, i.e., 4, and the window slides containing four frames (1,2,3,4)

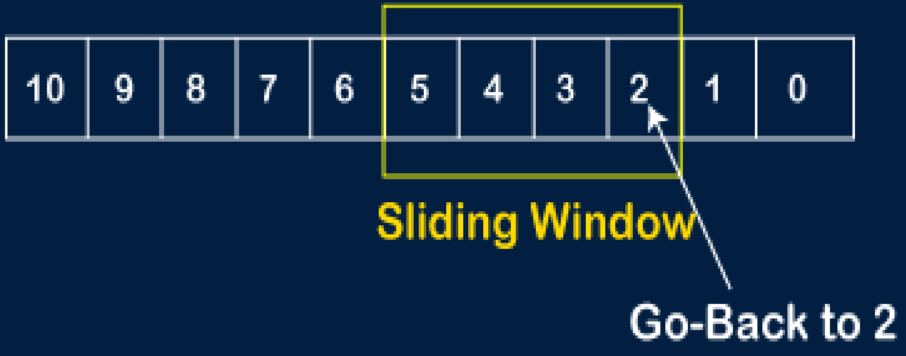
Go-Back-N

- Step 4 : The receiver will then send the acknowledgment for the frame no 1. After receiving the acknowledgment, the sender will send the next frame, i.e., frame no 5, and the window will slide having four frames (2,3,4,5).
- Step 5 : let's assume that the receiver is not acknowledging the frame no 2, either the frame is lost, or the acknowledgment is lost. Instead of sending the frame no 6, the sender Go-Back to 2, which is the first frame of the current window, retransmits all the frames in the current window, i.e., 2,3,4,5.
- **Pipelining** of frames, i.e. sending multiple frames before receiving the acknowledgment for the first frame. The frames are sequentially numbered and a finite number of frames are sent depending upon the size of the sending window.
- **Negative ACK** – Whenever the receiver receives a damaged frame or a duplicate frame, it sends a NACK back to the sender and sender must retransmit the correct frame.

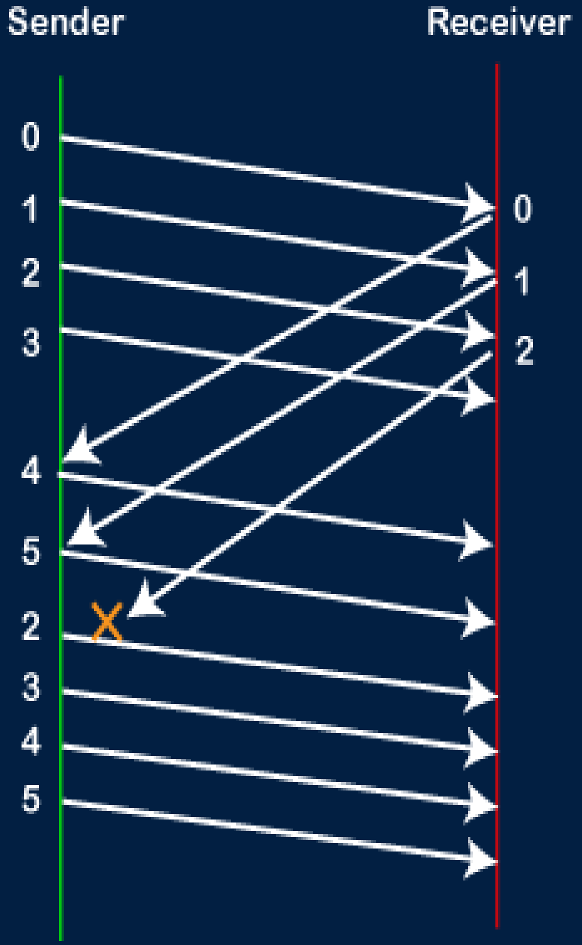
WORKING OF GO-BACK-N ARQ



WORKING OF GO-BACK-N ARQ



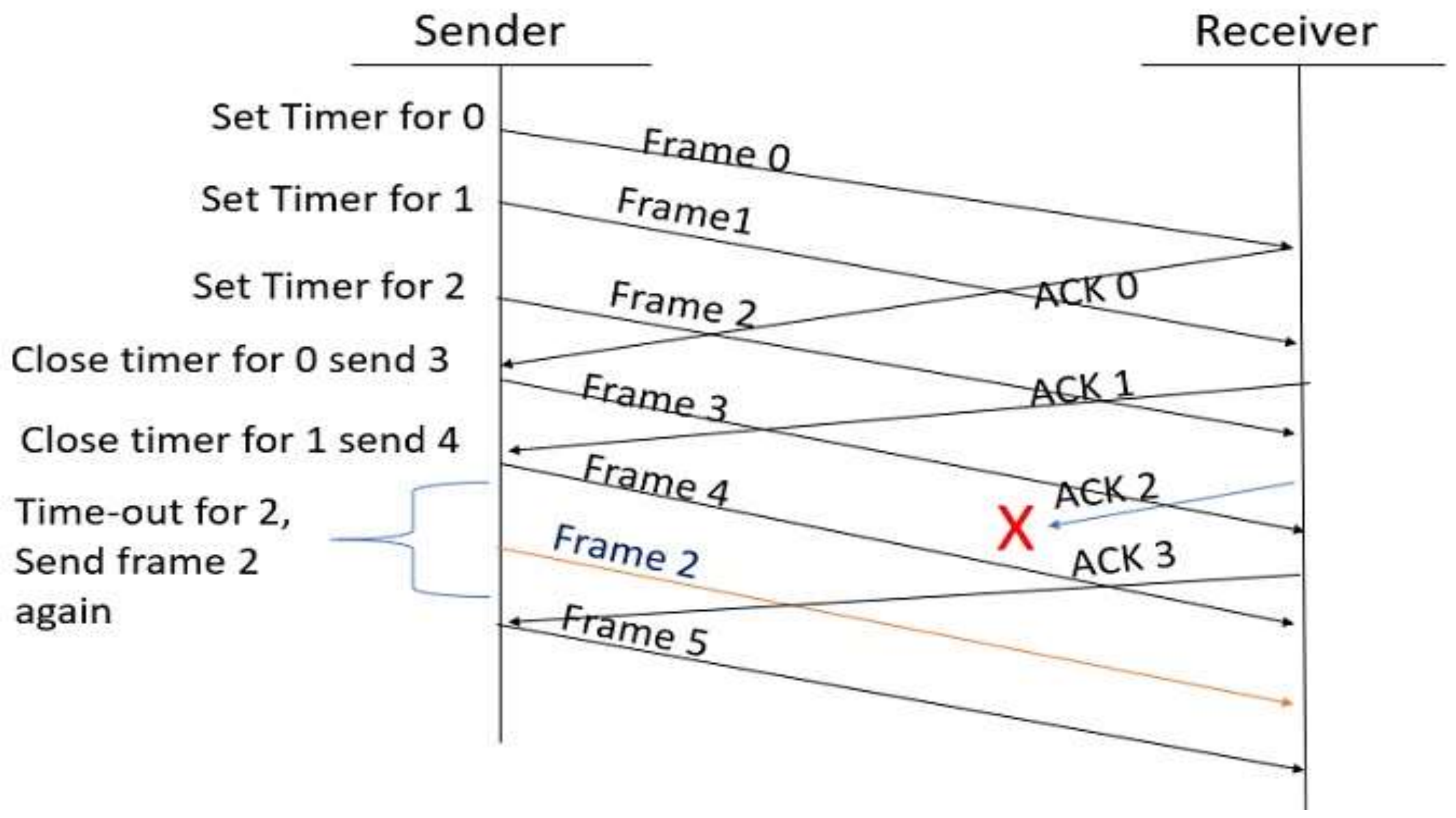
Window Size: 4



Selective Repeat ARQ

- It is also known as Sliding Window Protocol and used for error detection and control in the data link layer.
- In the selective repeat, the sender sends several frames specified by a window size even without the need to wait for individual acknowledgement from the receiver as in Go-Back-N ARQ. In selective repeat protocol, the retransmitted frame is received out of sequence.
- In Selective Repeat ARQ only the lost or error frames are retransmitted, whereas correct frames are received and buffered.
- The receiver while keeping track of sequence numbers buffers the frames in memory and sends NACK for only frames which are missing or damaged. The sender will send/retransmit a packet for which NACK is received.

Selective Repeat ARQ



Selective Repeat ARQ

Explanation

- **Step 1** – Frame 0 sends from sender to receiver and set timer.
- **Step 2** – Without waiting for acknowledgement from the receiver another frame, Frame1 is sent by sender by setting the timer for it.
- **Step 3** – In the same way frame2 is also sent to the receiver by setting the timer without waiting for previous acknowledgement.
- **Step 4** – Whenever sender receives the ACK0 from receiver, within the frame 0 timer then it is closed and sent to the next frame, frame 3.
- **Step 5** – whenever the sender receives the ACK1 from the receiver, within the frame 1 timer then it is closed and sent to the next frame, frame 4.
- **Step 6** – If the sender doesn’t receive the ACK2 from the receiver within the time slot, it declares timeout for frame 2 and resends the frame 2 again, because it thought the frame2 may be lost or damaged.

Error Detection and Correction

- Two basic strategies to deal with errors:
  1. Include enough redundant information to enable the receiver to deduce(infer) what the transmitted data must have been.

     Error correcting codes.

  2. Include only enough redundancy to allow the receiver to deduce that an error has occurred (but not which error).

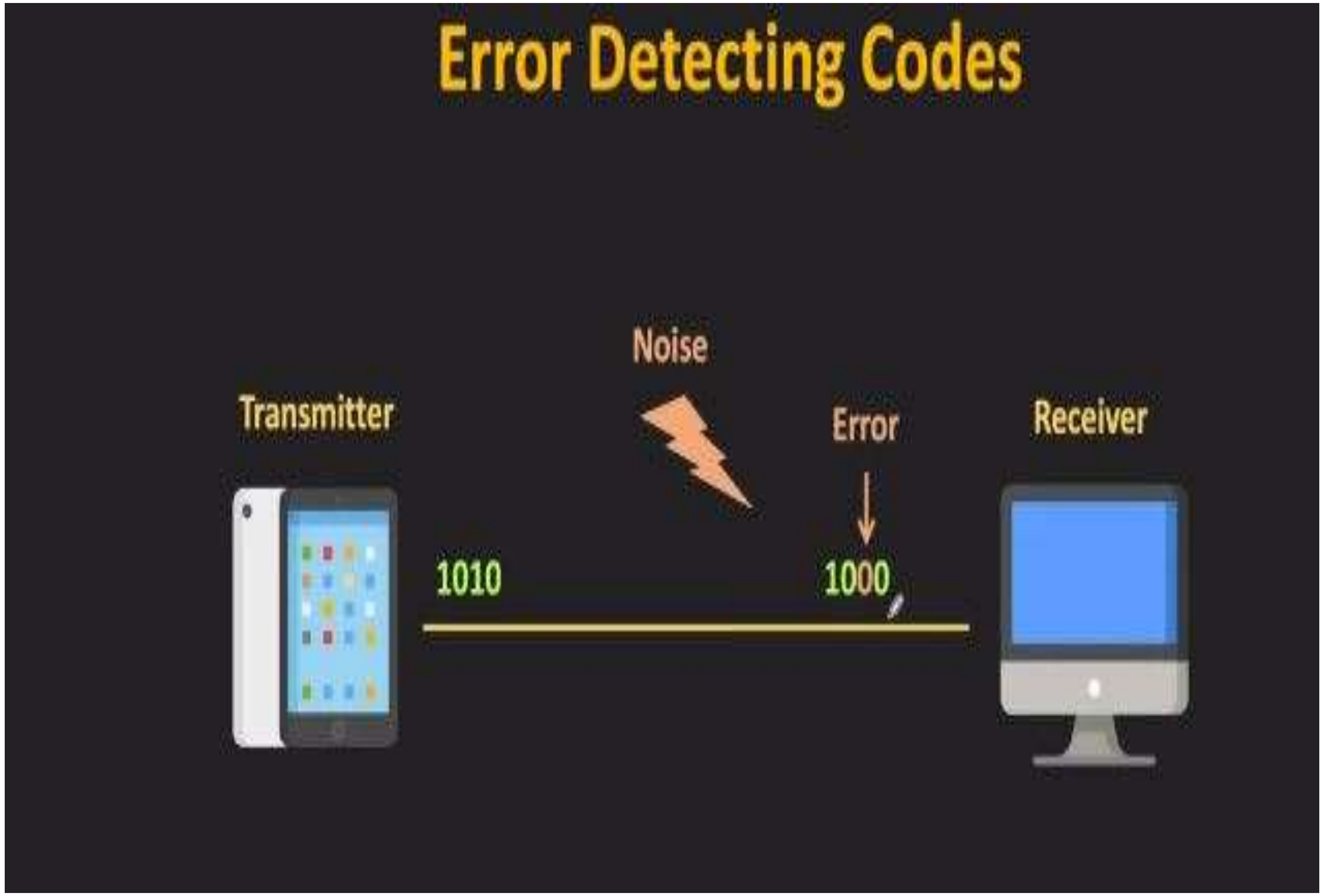
     Error detecting codes.

Error Detection & Correction Code

- All the codes presented in previous slide add redundancy to the information that is being sent.
- A frame consists of
  - ***m*** data bits (message) and
  - ***r*** redundant bits (check).
- ***Block code*** - the ***r*** check bits are computed solely as function of the ***m*** data bits with which they are associated.

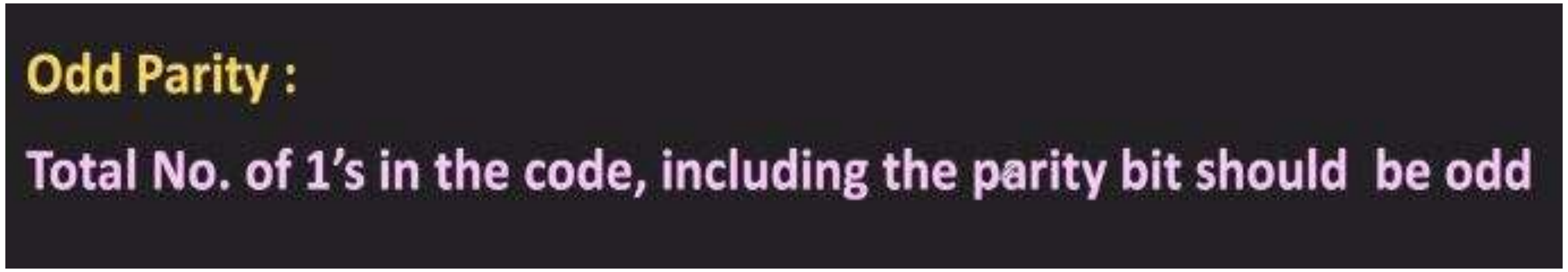
Error Detection Codes or Methods

- **Parity Checking**
- **Checksums**
- **CRC**

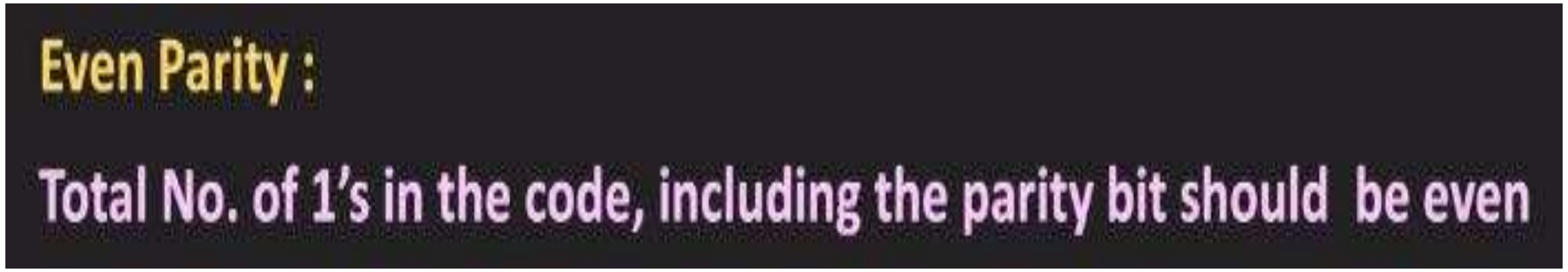

Error Detecting Codes
Noise
Transmitter
Error
Receiver
1010
1000

Parity Checking

- ODD PARITY
- EVEN PARITY


Odd Parity :

Total No. of 1’s in the code, including the parity bit should be odd


Even Parity :

Total No. of 1’s in the code, including the parity bit should be even

EVEN PARITY

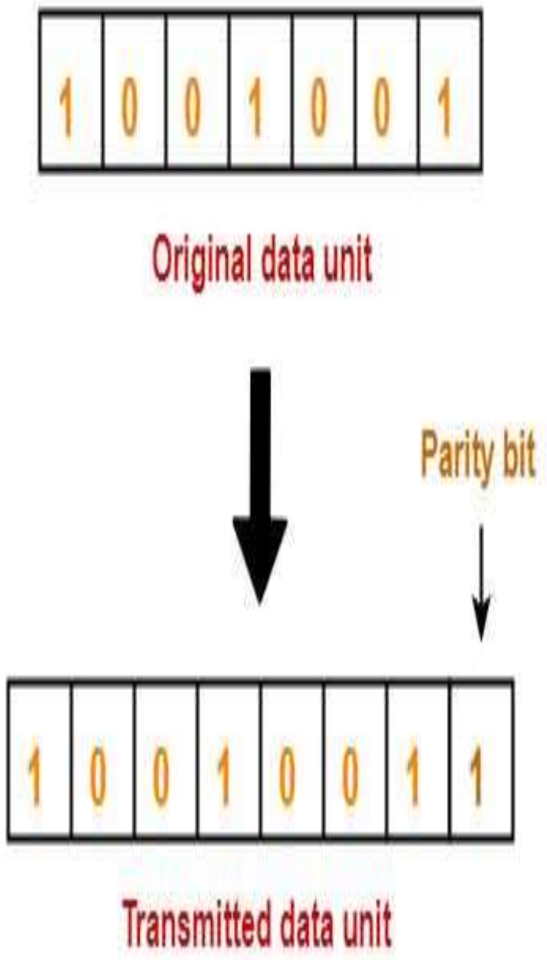
- Consider the data unit to be transmitted is 1001001 and even parity is used.

- **At Sender Side-**

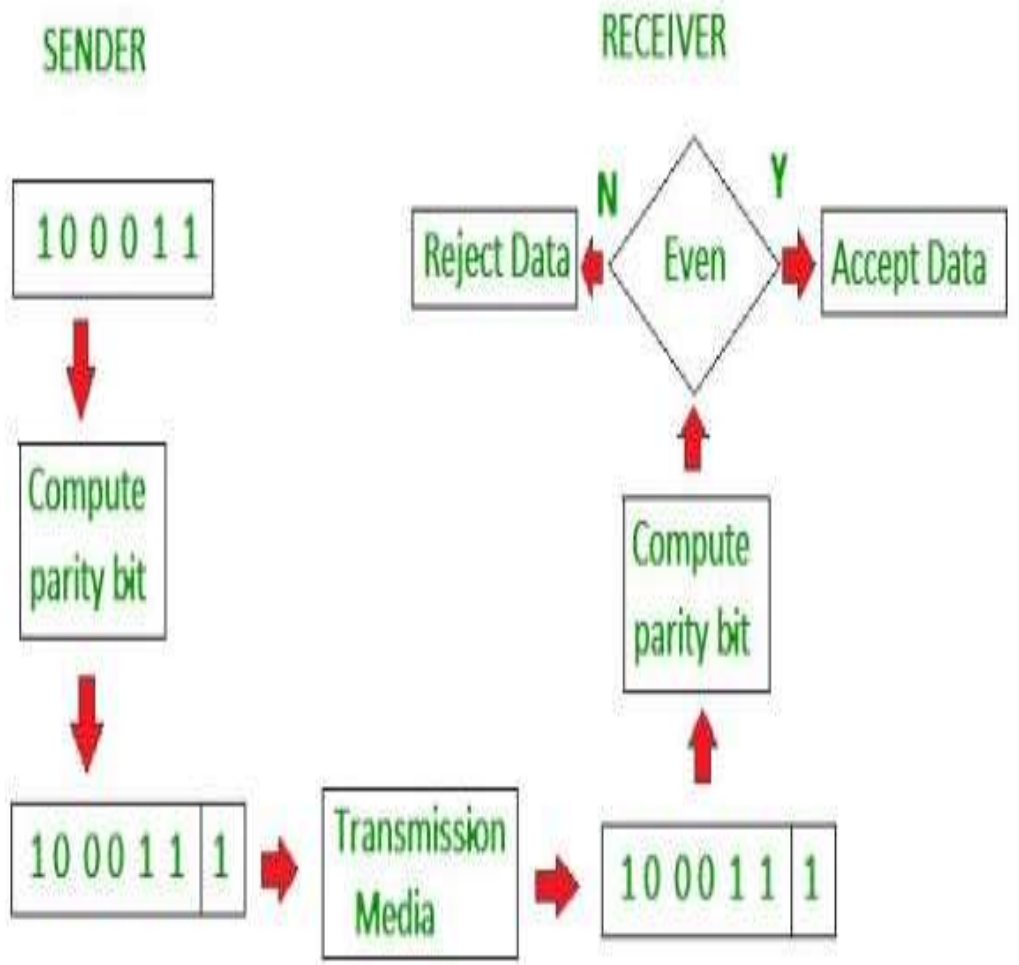
- Total number of 1’s in the data unit is counted.
- Total number of 1’s in the data unit = 3.
- Clearly, even parity is used and total number of 1’s is odd.
- So, parity bit = 1 is added to the data unit to make total number of 1’s even.
- Then, the code word 10010011 is transmitted to the receiver.

- **At Receiver Side-**

- After receiving the code word, total number of 1’s in the code word is counted.
- Consider receiver receives the correct code word = 10010011.
- Even parity is used and total number of 1’s is even.
- So, receiver assumes that no error occurred in the data during the transmission.

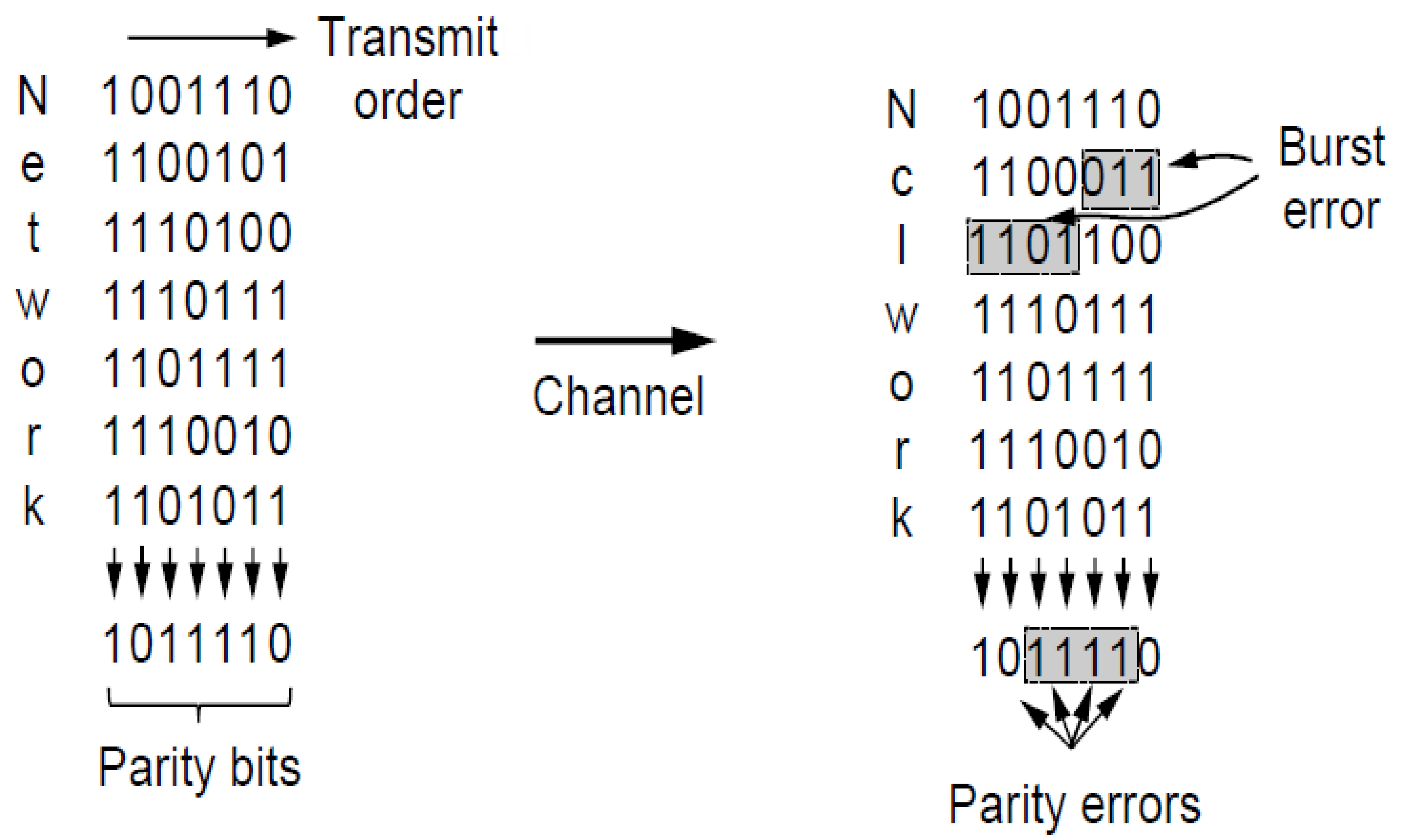


1 0 0 1 0 0 1
Original data unit
Parity bit
1 0 0 1 0 0 1 1
Transmitted data unit



SENDER
RECEIVER
1 0 0 0 1 1
Compute parity bit
10 00 1 1 | 1
Transmission Media
10 00 1 1 | 1
Compute parity bit
Even
N
Y
Reject Data
Accept Data

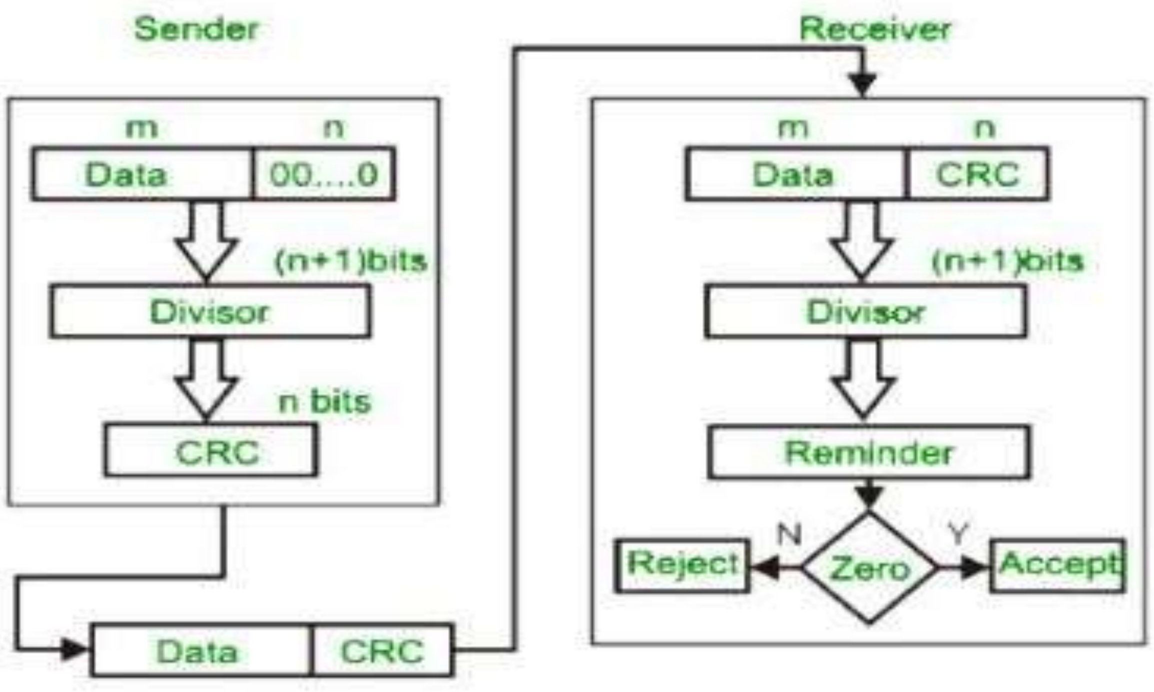
Error-Detecting Codes (2)



Cyclic Redundancy Checks (CRCs).

- Based on treating bit strings as representations of coefficients of polynomial.
- 11001 is representing $1 * x4 + 1 * x3 + 0 * x2 + 0 * x1 + 1 * x0$
- Generator polynomial G(x), Message Polynomial M(x), Transmitted Polynomial T(x)=CRC(G,M)
- Let r be the degree of G(x). Append r zero bits to the low-order end of the message frame M(x) to get $x\ rM(x)$.
- Divide the bit string corresponding to G(x) into the bit string corresponding to $x\ rM(x)$, using Mod 2 division.
- Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to x $rM(x)$ using modulo 2 subtraction.
- The first term at the right-hand side of the equality has a remainder of zero.
- So the syndrome is actually the remainder of the second term on the right-hand side.
- So, syndrome=0 means either $e\ x = 0$, i.e., no error or $e(x)$ is divisible by $g(x)$. So such errors are not caught.

Cyclic Redundancy Checks (CRCs).



Example calculation of the CRC

original message

1 0 1 0 0 0 0

@ means X-OR

Generator polynomial

x^3+1

$1.x^3+0.x^2+0.x^1+1.x^0$

CRC generator

1 0 0 1   4-bit

If CRC generator is of n bit then append (n-1) zeros in the end of original message

Sender

```
1 0 0 1 | 1 0 1 0 0 0 0 0 0 0
         @1 0 0 1
          ---------
          0 0 1 1 0 0 0 0 0 0
            @1 0 0 1
             ---------
              0 1 0 1 0 0 0 0
               @1 0 0 1
                ---------
                0 0 1 1 0 0 0
                  @1 0 0 1
                   ---------
                    0 1 0 1 0
                    @1 0 0 1
                     ---------
                     0 0 1 1
```

Message to be transmitted

```
1 0 1 0 0 0 0 0 0 0
            + 0 1 1
-------------------
1 0 1 0 0 0 0 0 1 1
```

Receiver

```
1 0 0 1 | 1 0 1 0 0 0 0 0 1 1
         @1 0 0 1
          ---------
          0 0 1 1 0 0 0 0 1 1
            @1 0 0 1
             ---------
              0 1 0 1 0 0 1 1
               @1 0 0 1
                ---------
                0 0 1 1 0 1 1
                  @1 0 0 1
                   ---------
                    0 1 0 0 1
                    @1 0 0 1
                     ---------
                      0 0 0 0
```

Zero means data is accepted

Exp1

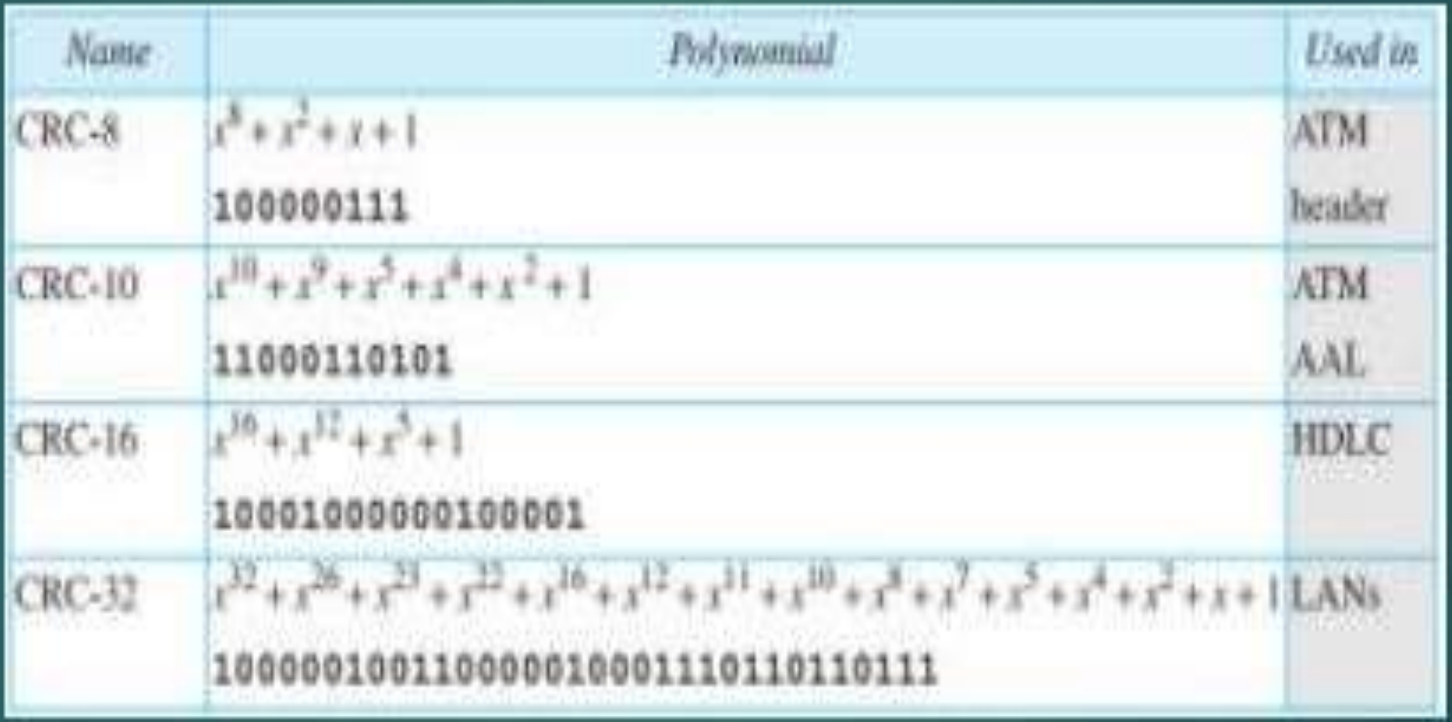
- Calculate CRC for the given polynomial expression where
- $G(x)=x^4+x+1$ and
- $M(x)=x^9+x^8+x^6+x^4+x^3+x+1$.

- $G(x)=10011$
- $M(X)=1101011011$

Exp2

- generator polynomial is of the form like $x^3 + x + 1$
- Data word-100100

Standard CRC Polynomials are:



Name	*Polynomial*	*Used in*
CRC-8	$x^8 + x^2 + x + 1$ **100000111**	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ **11000110101**	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$ **10001000000100001**	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ **100000100110000010001110110110111**	LANs

Checksums

- **Checksum** is the error detection method used by upper layer protocols and is considered to be more reliable than LRC, VRC and CRC. This method makes the use of **Checksum Generator** on Sender side and **Checksum Checker** on Receiver side.

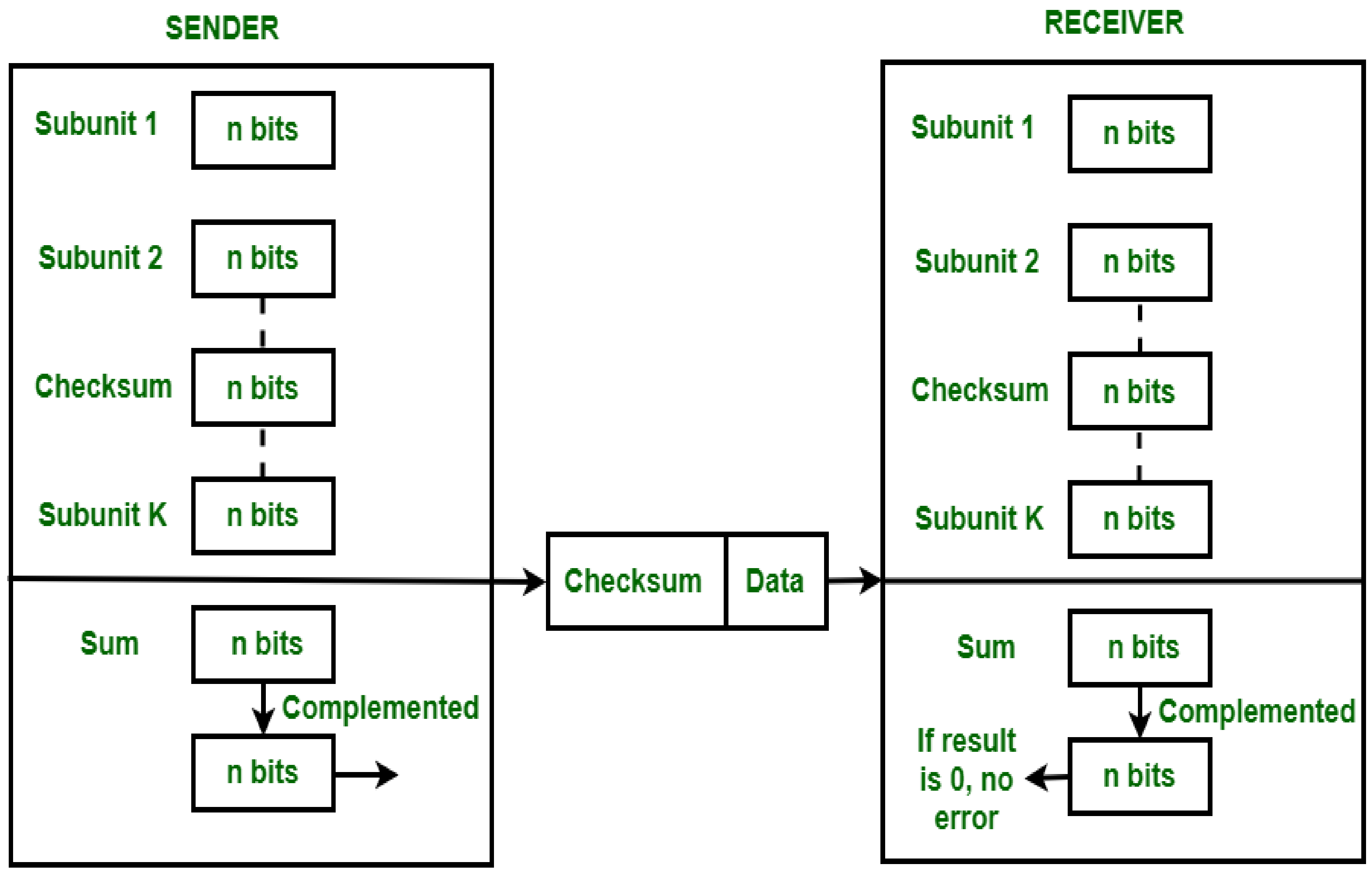
- At the Sender side, the data is divided into equal subunits of n bit length by the checksum generator.

- This bit is generally of 16-bit length. These subunits are then added together using one’s complement method.

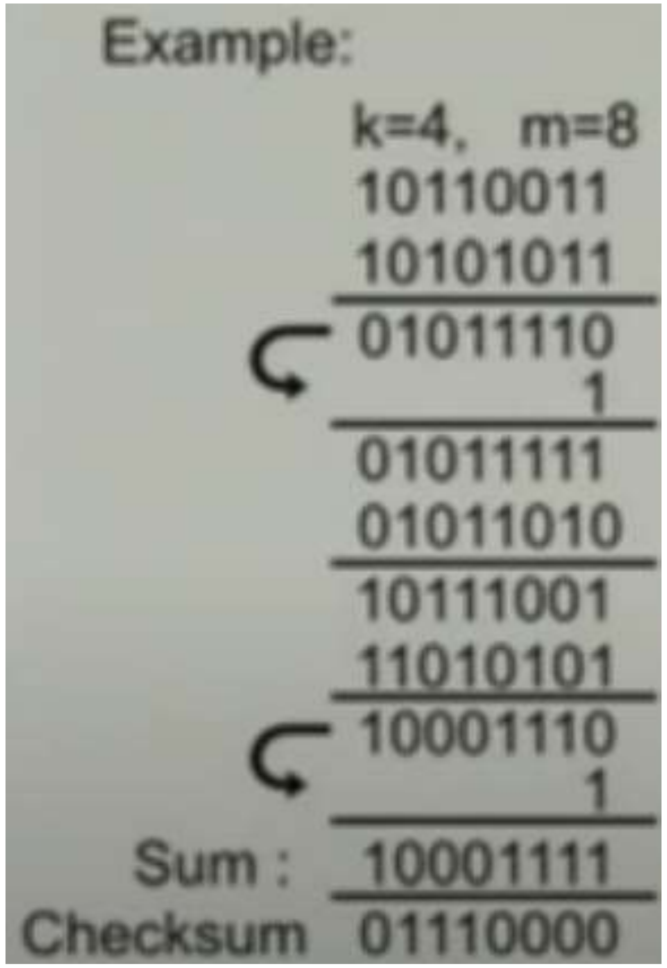
- This sum is of n bits. The resultant bit is then complemented. This complemented sum which is called checksum is appended to the end of original data unit and is then transmitted to Receiver.

-

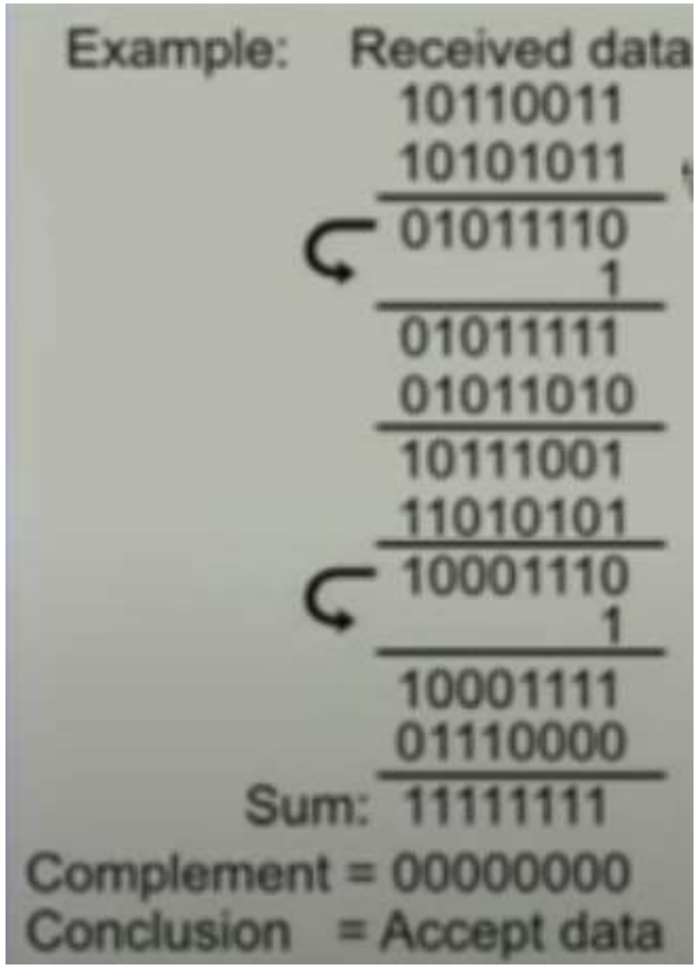
- The Receiver after receiving data + checksum passes it to checksum checker.
- Checksum checker divides this data unit into various subunits of equal length and adds all these subunits.
- These subunits also contain checksum as one of the subunits.
- The resultant bit is then complemented. If the complemented result is zero, it means the data is error-free.
- If the result is non-zero it means the data contains an error and Receiver rejects it.


SENDER
Subunit 1
n bits
Subunit 2
n bits
Checksum
n bits
Subunit K
n bits
Sum
n bits
Complemented
n bits
Checksum
Data
RECEIVER
Subunit 1
n bits
Subunit 2
n bits
Checksum
n bits
Subunit K
n bits
Sum
n bits
Complemented
If result is 0, no error
n bits

- **Example – 10110011   10101011 01011010 11010101**
  Checksum=**01110000**



```
Example:
              k=4,   m=8
              10110011
              10101011
              --------
              01011110
                     1
              --------
              01011111
              01011010
              --------
              10111001
              11010101
              --------
              10001110
                     1
              --------
Sum :         10001111
              --------
Checksum      01110000
```



```
Example:   Received data
              10110011
              10101011
              --------
              01011110
                     1
              --------
              01011111
              01011010
              --------
              10111001
              11010101
              --------
              10001110
                     1
              --------
              10001111
              01110000
              --------
Sum:          11111111
Complement = 00000000
Conclusion   = Accept data
```

Error Correction Codes

- **Hamming codes**
- Hamming code not only provides the detection of a bit error, but also identifies which bit is in error so that it can be corrected.
- Thus hamming code is called error detecting and correcting code.
- The code uses a number of parity bits(dependent on the number of information bits) located at certain position in a group.
- Number of parity bits:
- The number of parity bits depends on the number of information bits
- If the number of bits is designated as x, then the number of parity bits P is determined using the relation 2P ≥x+P+1

- **Location of the parity bits in a code:**
- The parity bits are located in the positions that are numbered corresponding to ascending powers of two(1,2,4,8,....).
- Therefore, for 7-bit code, locations for parity bits and information bits are as follows:
- D4, D3,D2,P3, D,1,P2,P1
- **Assigning values to parity bit:**
- In hamming code, each parity bit provides a check on certain other bits in the total code, therefore we must know the value of these others in order to assign the parity bit value.

Bit Location Table:

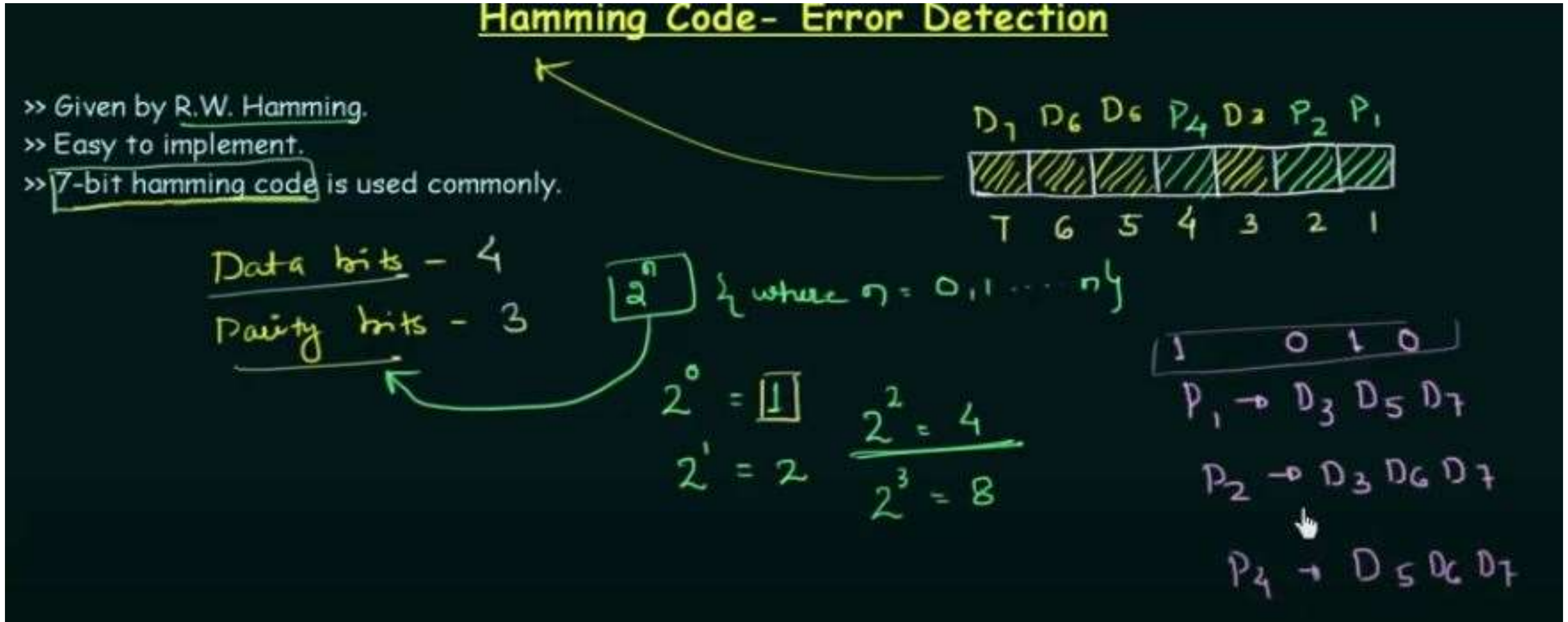
Bit Designation	D_4	D_3	D_2	P_3	D_1	P_2	P_1
Bit Location	7	6	5	4	3	2	1
Binary Location Number	111	110	101	100	011	010	001
Information Bits							
Parity Bits							

- **Assignment of P_1:**
- ✓ This parity bit checks all bit locations, including itself, that have 1s in the same location in the binary location numbers.
- **Assignment of P2:**
- ✓ This parity bit checks all bit locations, including itself, that have 1s in the middle bit.
- **Assignment of P3:**
- ✓ This parity bit checks all bit locations, including itself, that have Is in the left-most bit.

➢SINGLE ERROR CORRECTION AND DOUBLEERROR DETECTION

- ➢With the light modification, it is possible to construct hamming code for single error correction and double error detection.
- ➢✓ A one more parity bit is added in the hamming code to ensure hamming code contains an even number of ones. The resulting hamming code enables single error correction and double error detection.
- ➢✓ When overall parity bit is correct, there is no single error during the transmission of the code.
- ➢✓ If overall parity bit is incorrect, then there is single error and the bit position of the error can be indicated by binary number formed after checking the parity bits.
- ➢ Hence, single error correction can be achieved.

- ➢If overall parity bit is correct and binary number formed after checking the parity bits is other than 0-0-0, there are two errors.
- ➢In this case, double error detection is achieved.
- ➢✓ However, no correction is possible.



Hamming Code- Error Detection

- >> Given by R.W. Hamming.
- >> Easy to implement.
- >> $\boxed{\text{7-bit hamming code}}$ is used commonly.

Data bits - 4

Parity bits - 3

$\boxed{2^n}$ { where $n = 0, 1 \cdots n$ }

$2^0 = \boxed{1}$

$2^1 = 2$

$2^2 = 4$

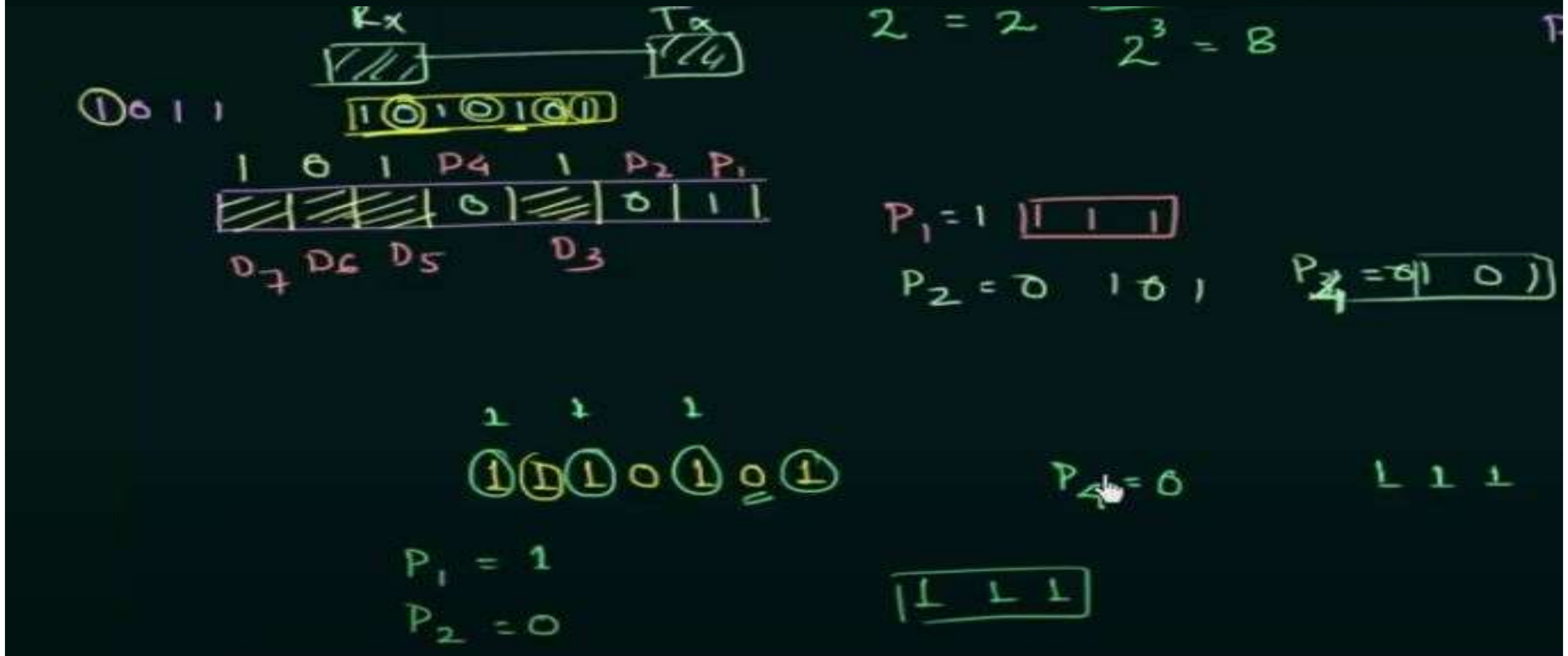
$2^3 = 8$

$\boxed{1 \quad 0 \quad 1 \quad 0}$

$P_1 \to D_3\ D_5\ D_7$

$P_2 \to D_3\ D_6\ D_7$

$P_4 \to D_5\ D_6\ D_7$



$2 = 2$ $\quad 2^3 = 8$

① 0 1 1

$\boxed{1\ ⓪\ 1\ ⓪\ 1\ ⓪\ ①}$

$P_1 = 1$ $\boxed{1 \quad 1 \quad 1}$

$P_2 = 0$ $\quad 1\ 0\ 1$

$P_{\cancel{2}4} = \cancel{0}\ \boxed{1 \quad 0 \quad 1}$

1 1 1

① ① ① 0 ① $\underline{0}$ ①

$P_4 = 0$

1 1 1

$P_1 = 1$

$P_2 = 0$

$\boxed{1 \quad 1 \quad 1}$

Hamming Code-Error Correction

Ex:- If the 7-bit hamming code word received by a receiver is $\boxed{1011011}$. Assuming the $\boxed{\text{even parity}}$ state whether the receivrd code word is correct or wrong. If wrong locate the bit having error.

Sol :-

D7	D6	D5	P4	D3	P2	P1
1	0	1	1	0	1	1

$P_4 \; D_5 \; D_6 \; D_7$

$1 \;\; 1 \;\; 0 \;\; 1 \rightarrow$ odd

$\boxed{P_4 = 1}$

$P_2 \; D_3 \; D_6 \; D_7$

$1 \;\; 0 \;\; 0 \;\; 1 \rightarrow$ even

$\boxed{P_2 = 0}$

$P_1 \; D_3 \; D_5 \; D_7$

$1 \;\; 0 \;\; 1 \;\; 1 \rightarrow$ odd

$\boxed{P_1 = 1}$

Sol :-

$P_4 \; D_5 \; D_6 \; D_7$

$1 \;\; 1 \;\; 0 \;\; 1 \rightarrow$ odd

$\boxed{P_4 = 1}$

$P_2 \; D_3 \; D_6 \; D_7$

$1 \;\; 0 \;\; 0 \;\; 1 \rightarrow$ even

$\boxed{P_2 = 0}$

$P_1 \; D_3 \; D_5 \; D_7$

$1 \;\; 0 \;\; 1 \;\; 1 \rightarrow$ odd

$\boxed{P_1 = 1}$

D7	D6	D5	P4	D3	P2	P1
1	0	$\textcircled{1}$	1	0	1	1

$(1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 1)$ correct code

$4 + 1 = 5$

$$P_4 \, P_2 \, P_1 = (1\,0\,1)_2 = \underbrace{(5)_{10}}_{\text{bit error}}$$

HDLC in Data link layer

High-level Data Link Control (HDLC)

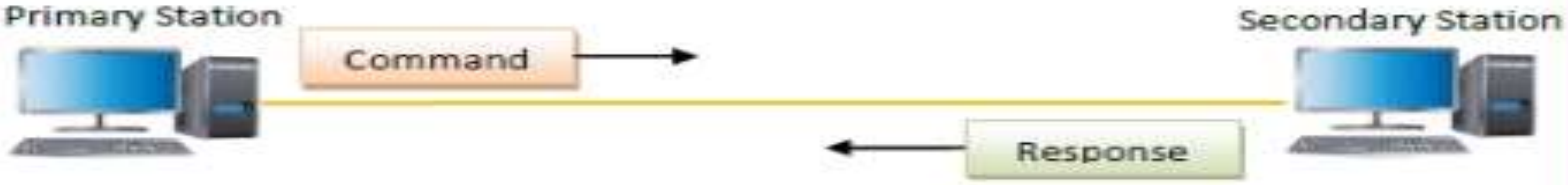
- High-level Data Link Control (HDLC) is a group of communication protocols of the data link layer for transmitting data between network points or nodes.

- It is a bit - oriented protocol that is applicable for both point - to - point and multipoint communications.

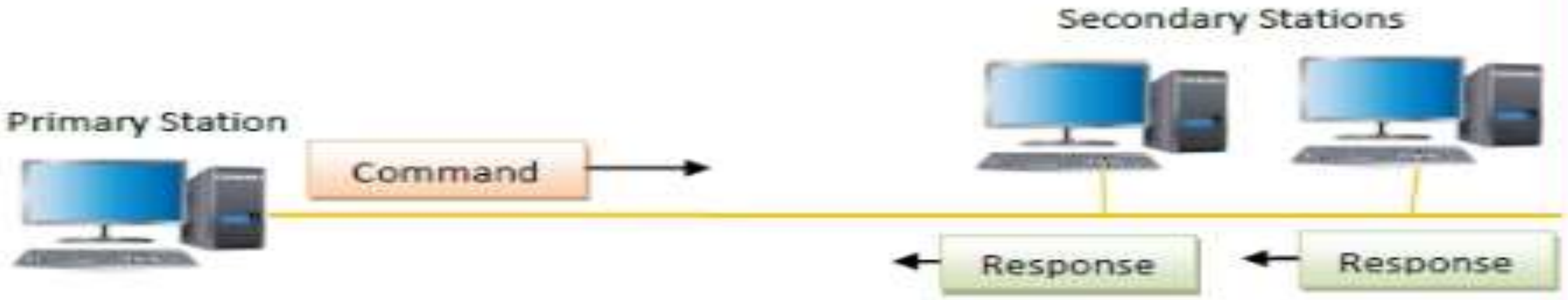
- **Transfer Modes in HDLC:**
- **HDLC supports two types of transfer modes**,
- Normal response mode and
- Asynchronous balanced mode.

- **Normal Response Mode (NRM)** –
- Here, two types of stations are there, a primary station that send commands and secondary station that can respond to received commands.
- It is used for both point - to - point and multipoint communications
- **Asynchronous Balanced Mode (ABM)** –
- Here, the configuration is balanced, i.e. each station can both send commands and respond to commands.
- It is used for only point - to - point communications.

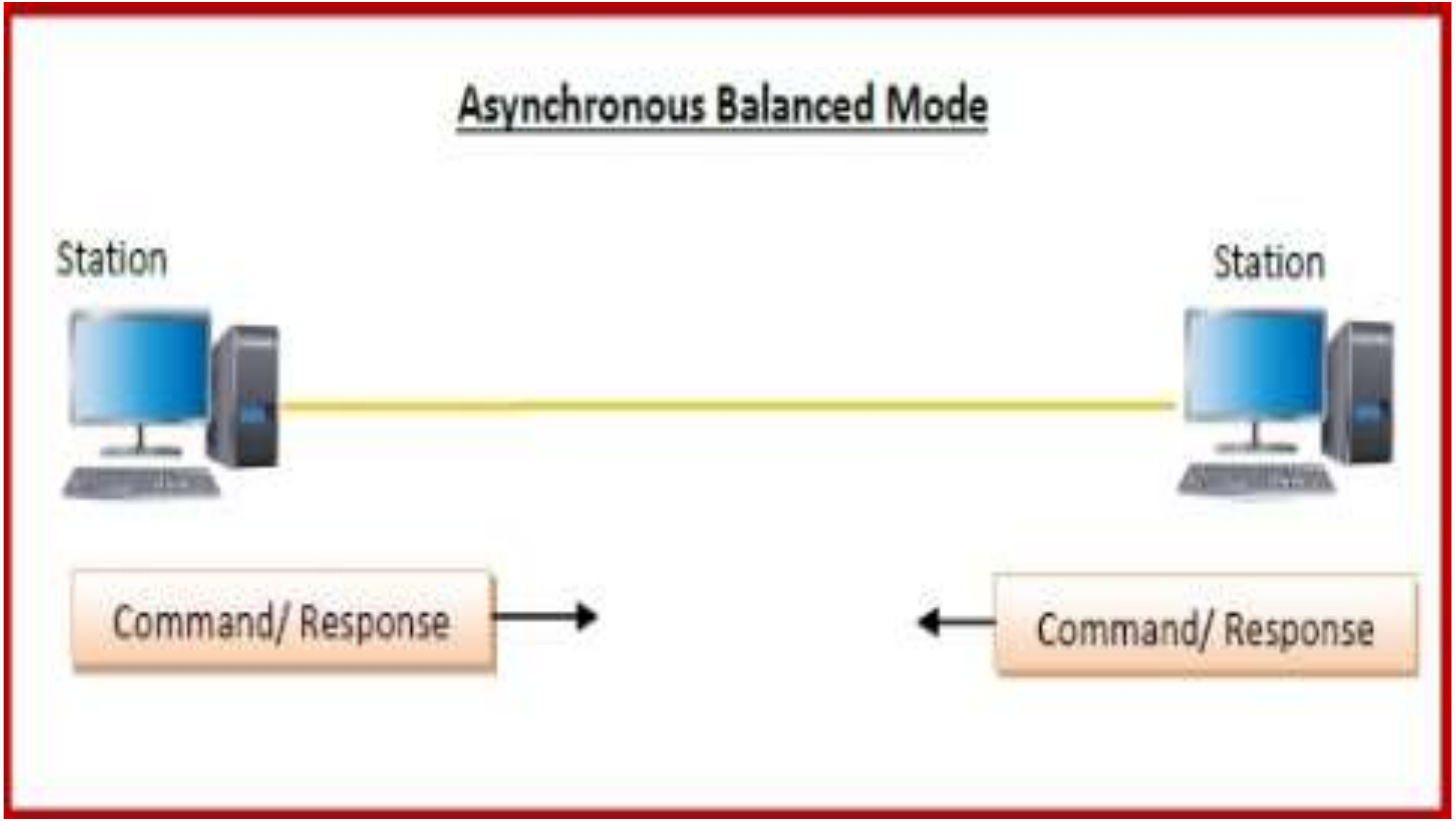
Normal Response Mode



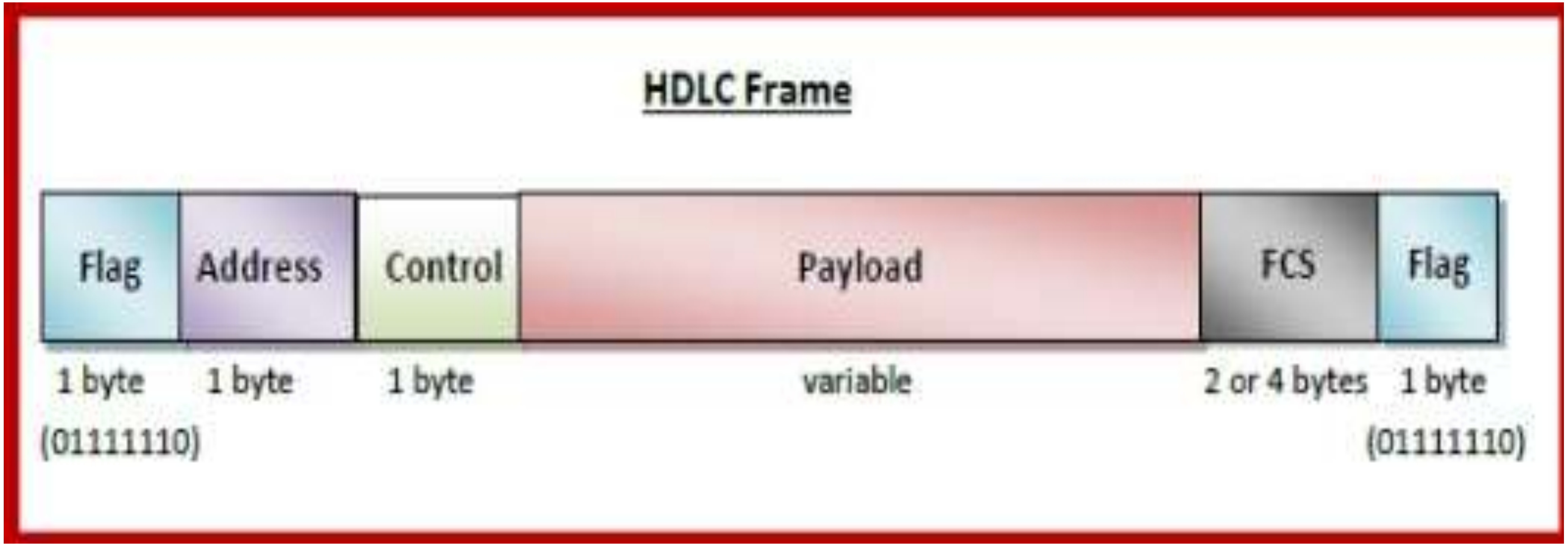
Point – to – point communication



Multipoint communication


Asynchronous Balanced Mode
Station
Station
Command/ Response
Command/ Response

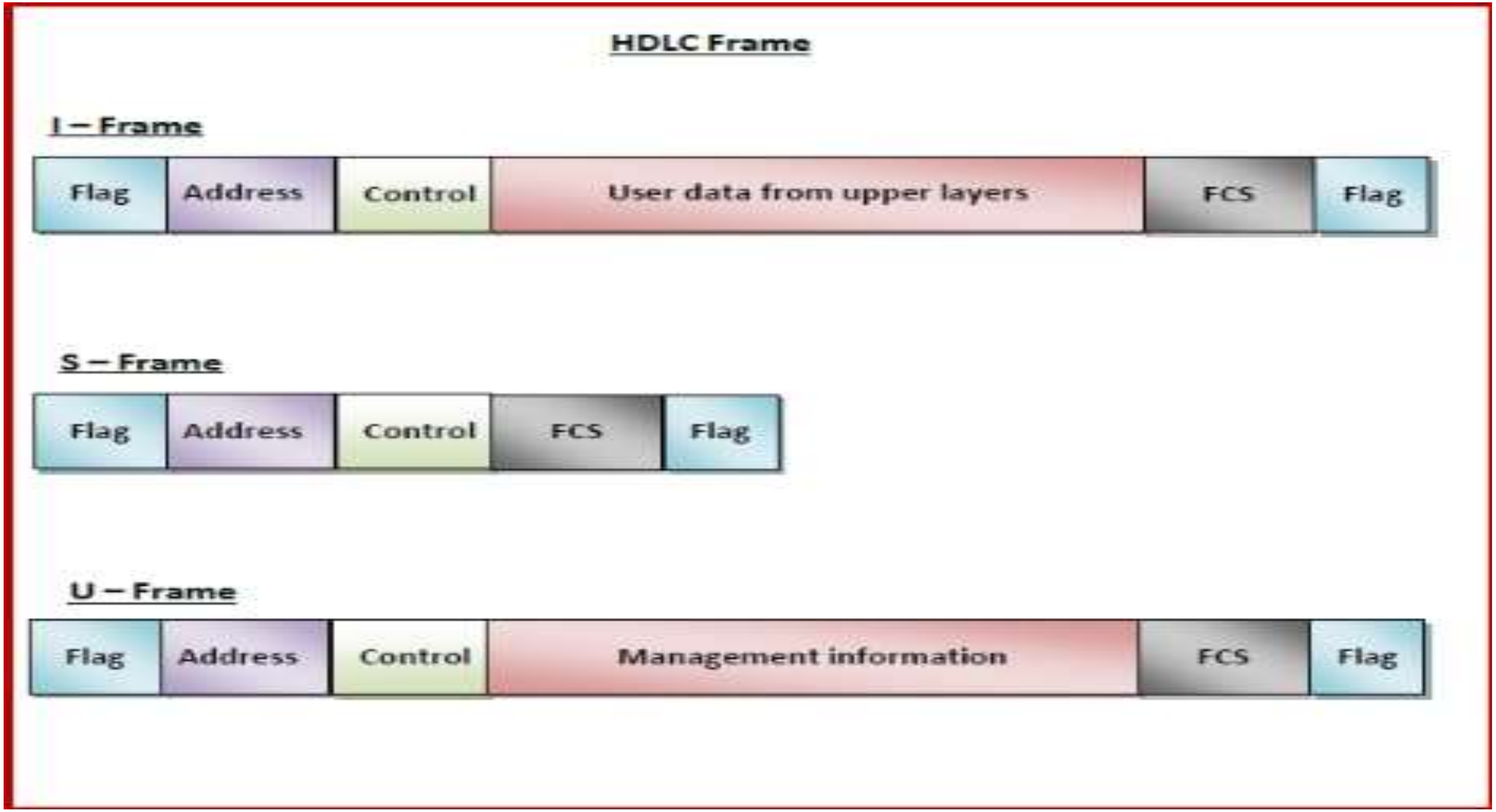
HDLC Frame:


HDLC Frame
Flag
Address
Control
Payload
FCS
Flag
1 byte
1 byte
1 byte
variable
2 or 4 bytes
1 byte
(01111110)
(01111110)

HDLC Frame

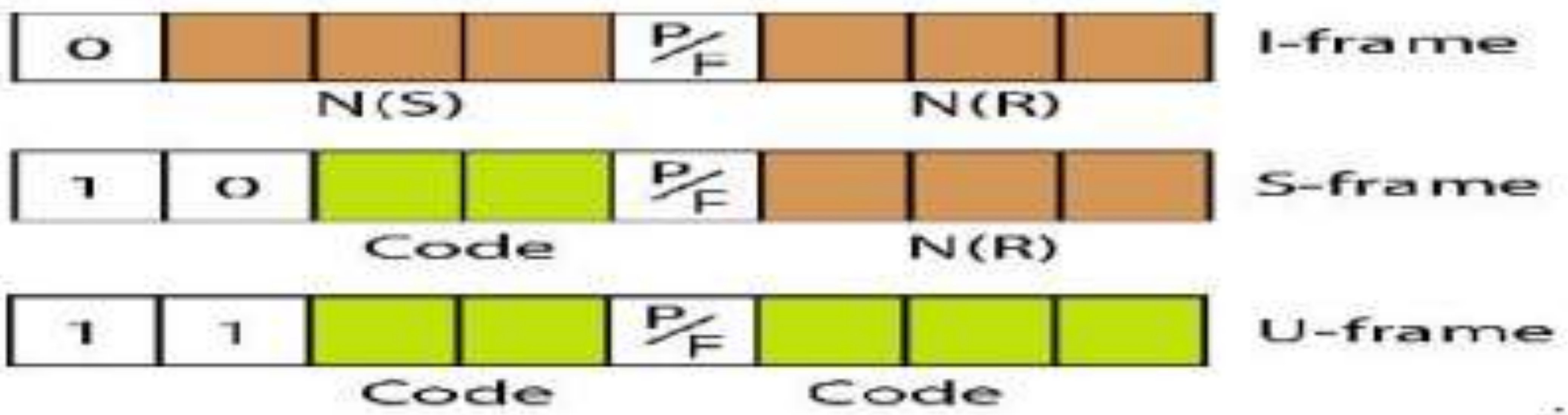
HDLC is a bit - oriented protocol where each frame contains up to six fields. The structure varies according to the type of frame. The fields of a HDLC frame are −

- **Flag** – It is an 8-bit sequence that marks the beginning and the end of the frame. The bit pattern of the flag is 01111110.
- **Address** – It contains the address of the receiver. If the frame is sent by the primary station, it contains the address(es) of the secondary station(s). If it is sent by the secondary station, it contains the address of the primary station. The address field may be from 1 byte to several bytes.
- **Control** – It is 1 or 2 bytes containing flow and error control information.
- **Payload** – This carries the data from the network layer. Its length may vary from one network to another.
- **FCS** – It is a 2 byte or 4 bytes frame check sequence for error detection. The standard code used is CRC (cyclic redundancy code)


HDLC Frame
I – Frame
Flag
Address
Control
User data from upper layers
FCS
Flag
S – Frame
Flag
Address
Control
FCS
Flag
U – Frame
Flag
Address
Control
Management information
FCS
Flag

Types of HDLC Frames

- **I-frame** − I-frames or Information frames carry user data from the network layer. They also include flow and error control information that is piggybacked on user data. The first bit of control field of I-frame is 0.
- **S-frame** − S-frames or Supervisory frames do not contain information field. They are used for flow and error control when piggybacking is not required. The first two bits of control field of S-frame is 10.
- **U-frame** − U-frame stands for Unnumbered frames. These frames are required in various functions like link setup and disconnections. These frames are used to exchange session management and control information between connected devices. U-frame is required for managing link itself. The first two bits of control field of U-frame is 11.



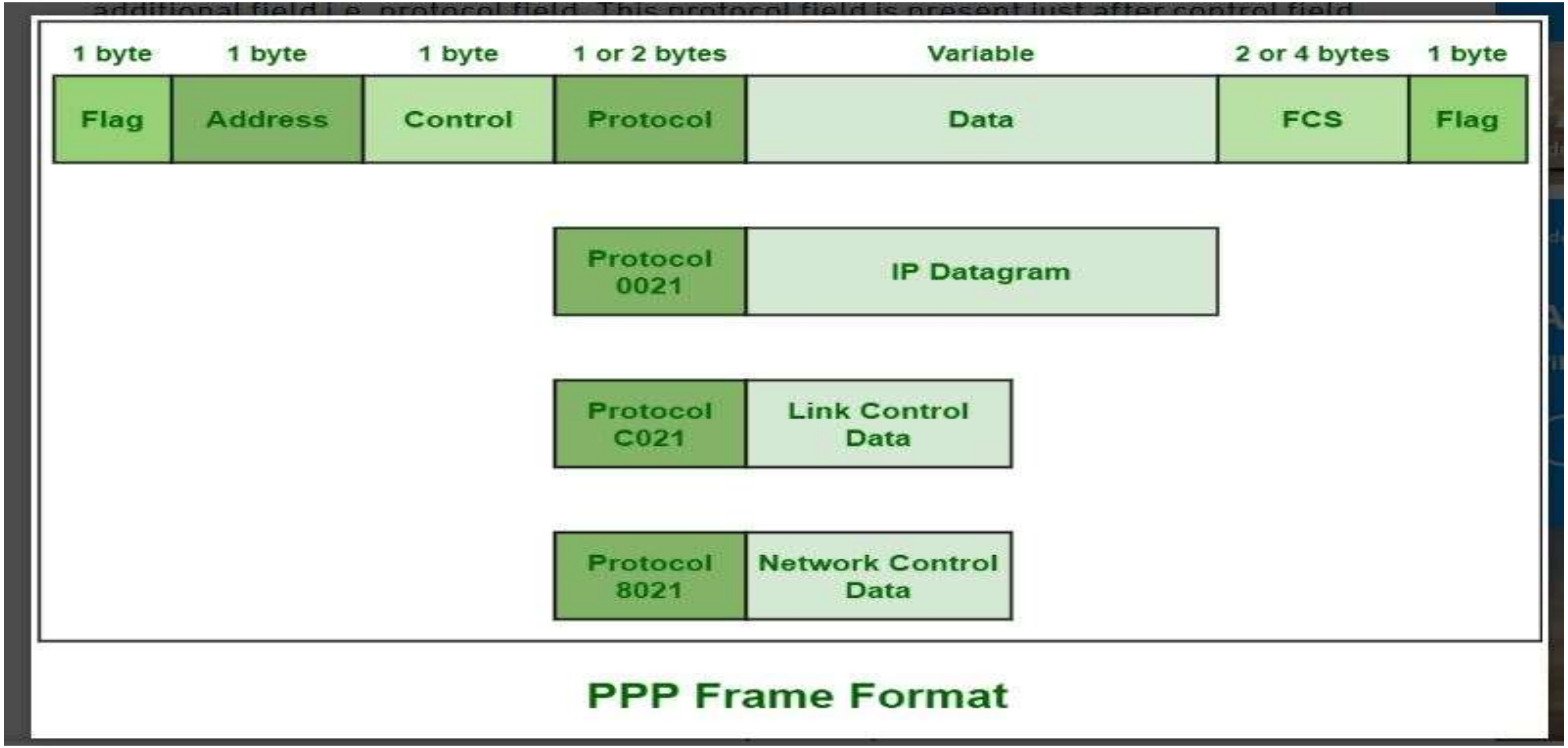
N(S) defines the sequence number of the Frame. **N(R)** defines the acknowledgement(ACK) or negative acknowledgement (NACK). **P/F bit.** Poll/Final is a single bit with two names. It is called Poll when part of a command (set by the primary station to obtain a response from a secondary station), and Final when part of a response (set by the secondary station to indicate a response or the end of transmission).

Point-to-Point Protocol (PPP)

- Today, millions of Internet users who need to connect their home computers to the server of an Internet service provider use PPP.
- Is a byte-oriented protocol
- PPP is a data link layer protocol.
- PPP is a WAN protocol and which is commonly run over Internet links. It means between two routers ,PPP is widely used.
- It is widely used in broadband communications having heavy loads and high speeds.
- It is used to transmit multiprotocol data between two directly connected (point-to-point) computers.

- In PPP, link establishment is controlled and handled mainly by Link Control Protocol (LCP).
- It is also required to connect the Home PC to server of ISP through a modem.
- It was also adopted by ISPs to simply provide dial-up Internet Access.
- **PPP Frame Format :** PPP frame is generally required to encapsulate packets of information or data that simply includes either configuration information or data.
- PPP basically uses the same basic format as that of HDLC.
- PPP usually contains one additional field i.e. protocol field. This protocol field is present just after control field and before information or data field.

PPP-FRAME FORMAT



- **Various fields of Frame are given below** :

- **Flag field –** PPP frame similar to HDLC frame, always begins and ends with standard HDLC flag. It always has a value of 1 byte i.e., 01111110 binary value.
- **Address field –** Address field is basically broadcast address. In this, all 1's simply indicates that all of the stations are ready to accept frame. It has the value of 1 byte i.e., 11111111 binary value. PPP on the other hand, does not provide or assign individual station addresses.
- **Control field –** This field basically uses format of U-frame i.e., Unnumbered frame in HDLC. In HDLC, control field is required for various purposes but in PPP, this field is set to 1 byte i.e., 00000011 binary value. This 1 byte is used for a connection-less data link.
- **Protocol field –** This field basically identifies network protocol of the datagram. It usually identifies the kind of packet in the data field i.e., what exactly is being carried in data field. This field is of 1 or 2 bytes and helps in identifies the PDU (Protocol Data Unit) that is being encapsulated by PPP frame.
- **Data field –** It usually contains the upper layer datagram. Network layer datagram is particularly encapsulated in this field for regular PPP data frames. Length of this field is not constant rather it varies.
- **FCS field –** This field usually contains checksum simply for identification of errors. It can be either 16 bits or 32 bits in size. It is also calculated over address, control, protocol, and even information fields. Characters are added to frame for control and handling of errors.