UNIT - II

- **Data Link Layer:** Fundamentals of Error Detection and Error Correction, Block coding, Hamming Distance, CRC;
- Flow Control and Error control protocols - Stop and Wait, Goback-N ARQ, Selective Repeat ARQ, Sliding Window, Piggybacking,
- Random Access,Multiple access protocols - Pure ALOHA, Slotted ALOHA, CSMA/CD, CDMA/CA

Services

- Framing
- Flow Control
- Error Control
- Congestion Control

Basic concepts

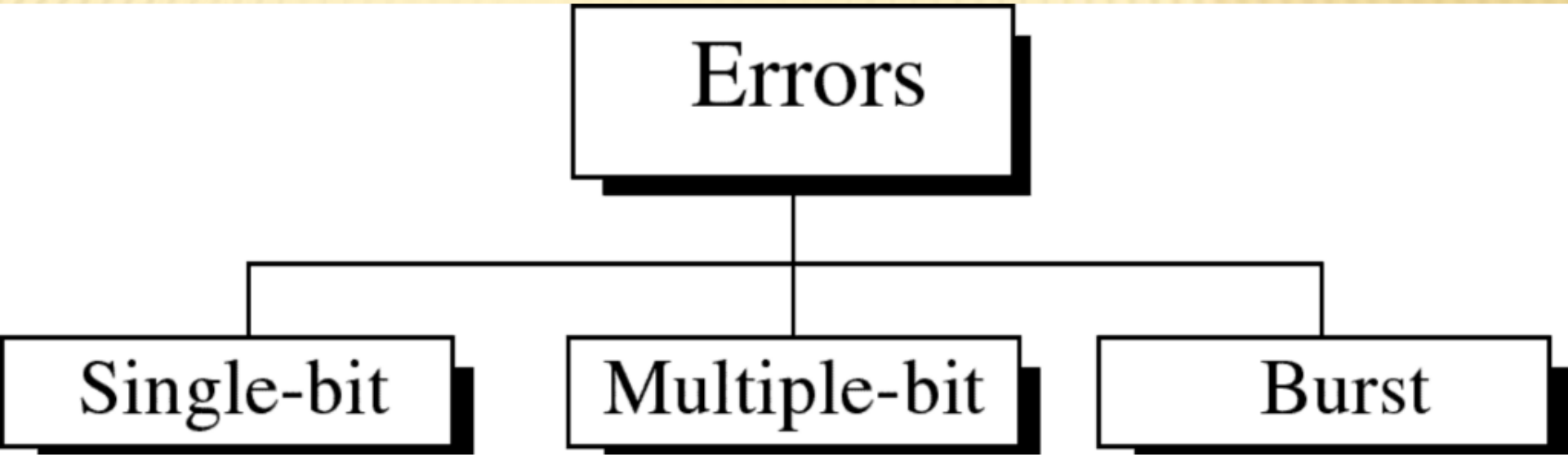
- Networks must be able to transfer data from one device to another with complete accuracy.
- Data can be corrupted during transmission.
- For reliable communication, errors must be detected and corrected.
- Error detection and correction are implemented either at the data link layer or the transport layer of the OSI model

Note

Data can be corrupted during transmission.

Some applications require that errors be detected and corrected.

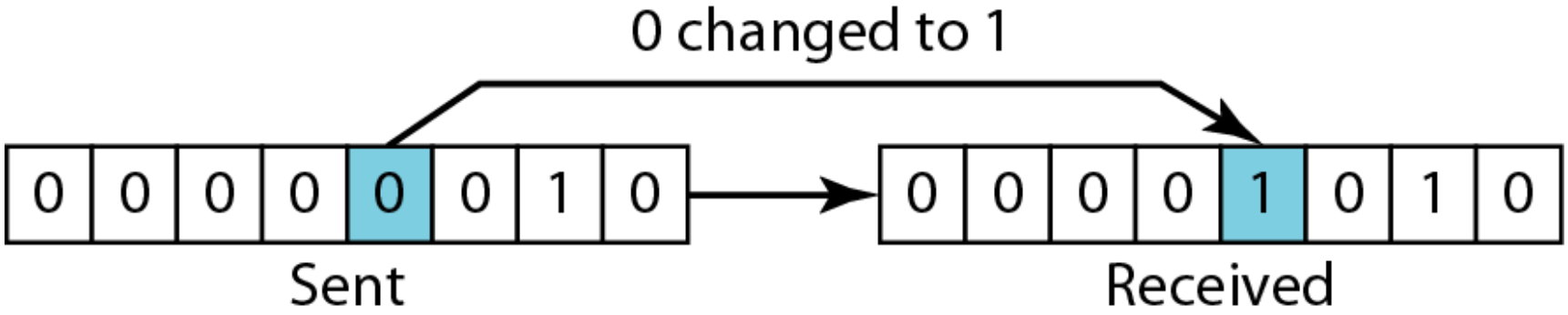
Types of Errors



Note

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

Single-bit error

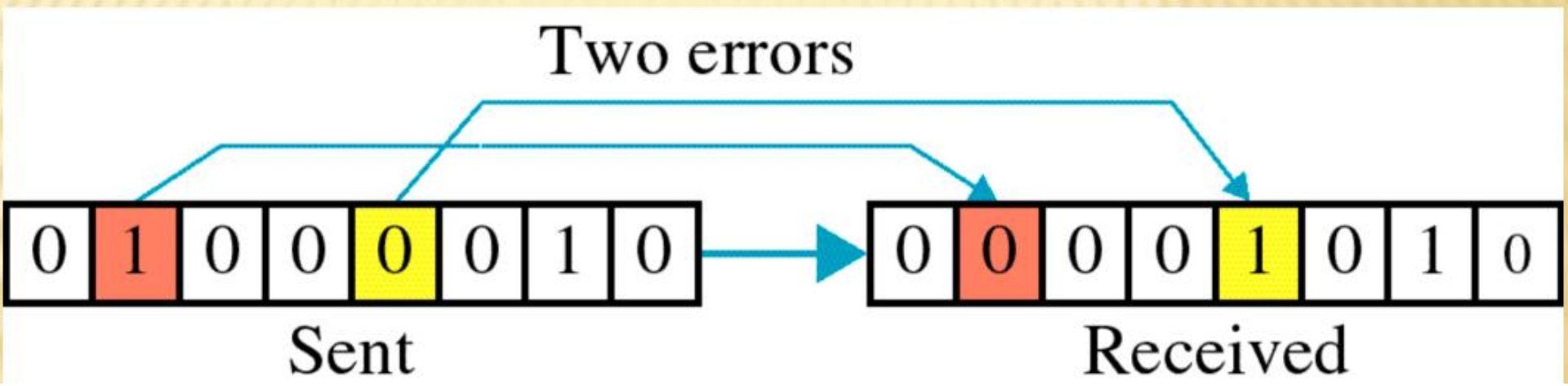


Note

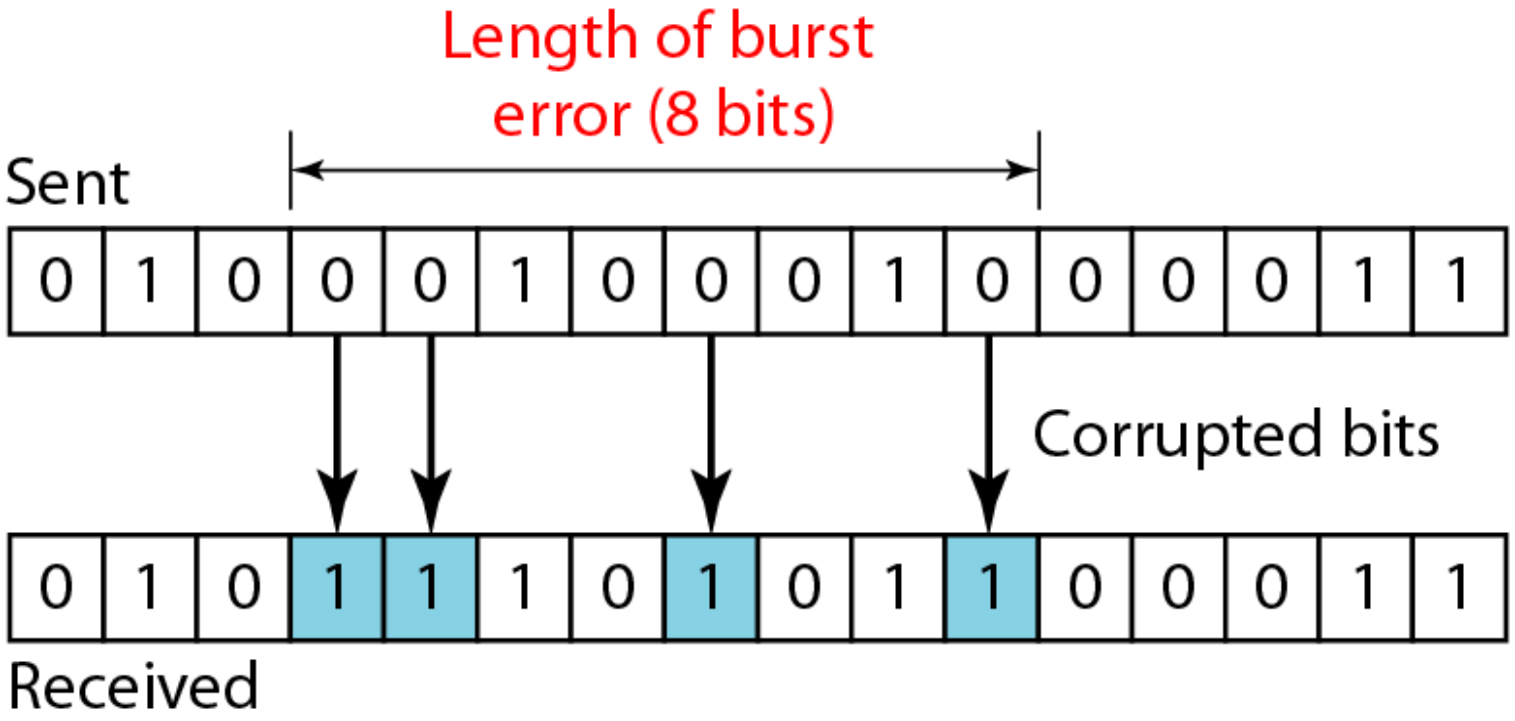
- Single bit errors are the least likely type of errors in serial data transmission because the noise must have a very short duration which is very rare.
- However this kind of errors can happen in parallel transmission.

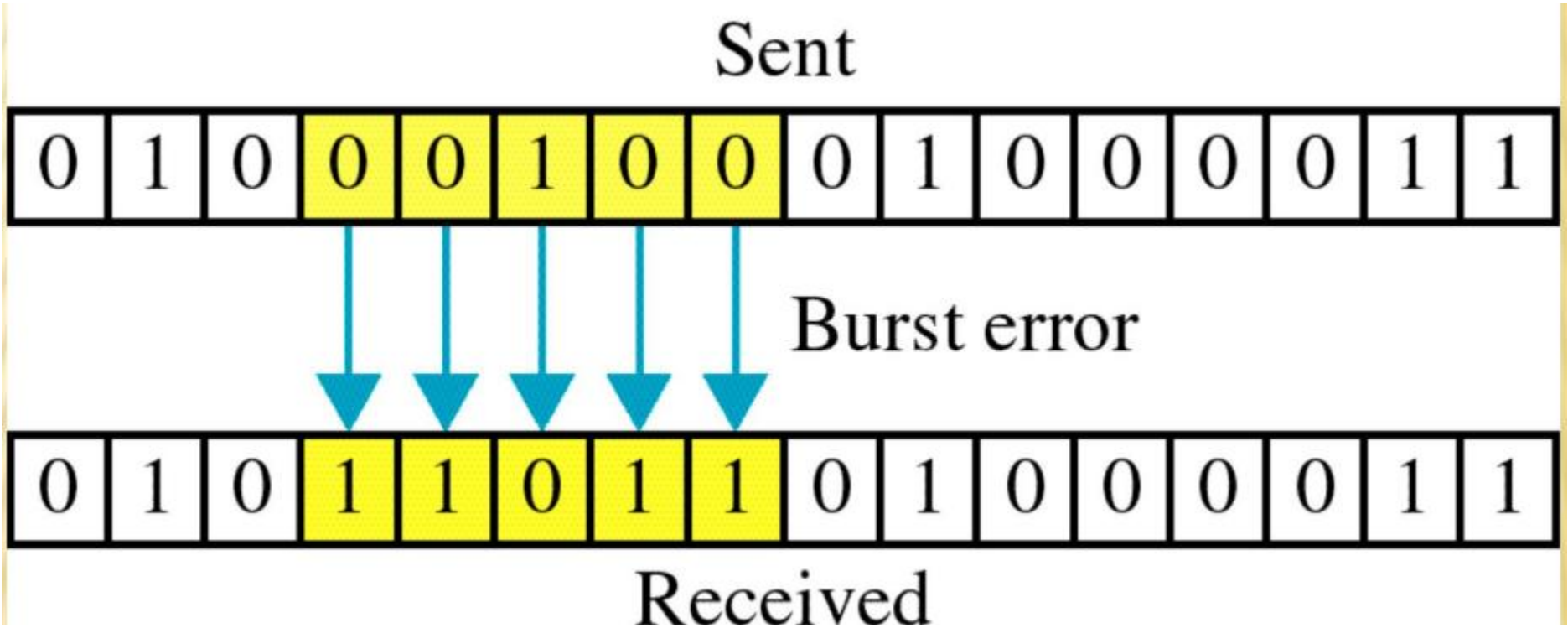
Note

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


Two errors
0 1 0 0 0 0 1 0
Sent
0 0 0 0 1 0 1 0
Received

Burst error of length 8




Sent
0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
Burst error
0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 1
Received

Note

- Burst error is most likely to happen in serial transmission since the duration of noise is normally longer than the duration of a bit.
- The number of bits affected depends on the data rate and duration of noise

ERROR DETECTION

- Error detection means to decide whether the received data is correct or not without having a copy of the original message.
- Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination

- There are 3 ways in which we can detect errors in the received message :
- **1. Parity Bit**
- **2. CheckSum**
- **3. Cyclic Redundancy Check (CRC)**

What is parity bit?

- A parity bit is an extra bit that is added to the message bits or data-word bits on the sender side.
- Data-word bits along with parity bits is called a codeword.
- The parity bit is added to the message bits on the sender side, to help in error detection at the receiver side.

Parity check

- The parity check is done by adding an extra bit, called parity bit, to the data to make the number of 1s either even or odd depending upon the type of parity.
- The parity check is suitable for single bit error detection only.

- **Even Parity** – Here the total number of bits in the message is made even.
- **Odd Parity** – Here the total number of bits in the message is made odd.

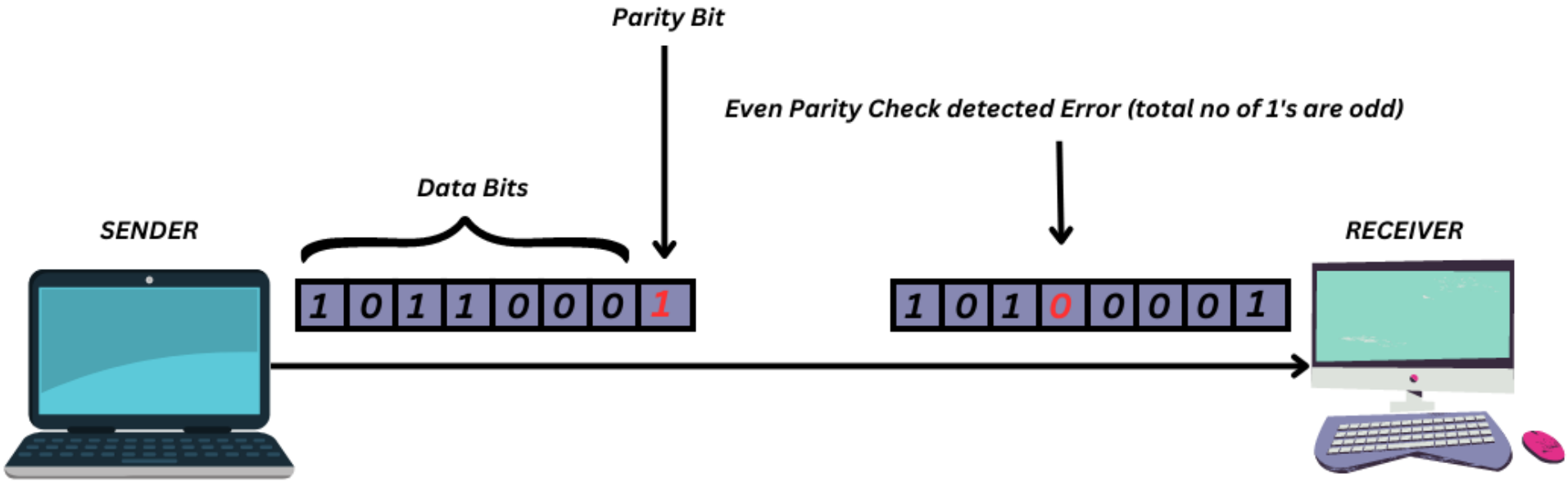
Note

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

Note

Redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

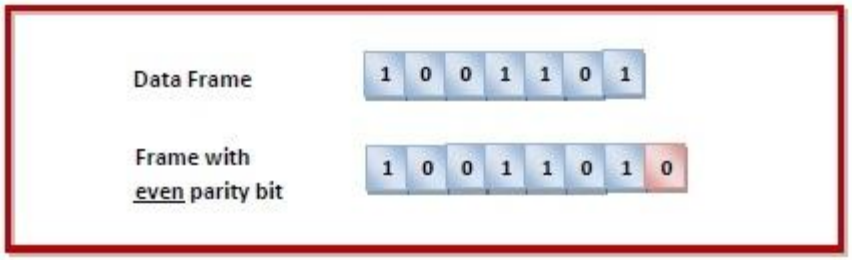
Even Parity Check



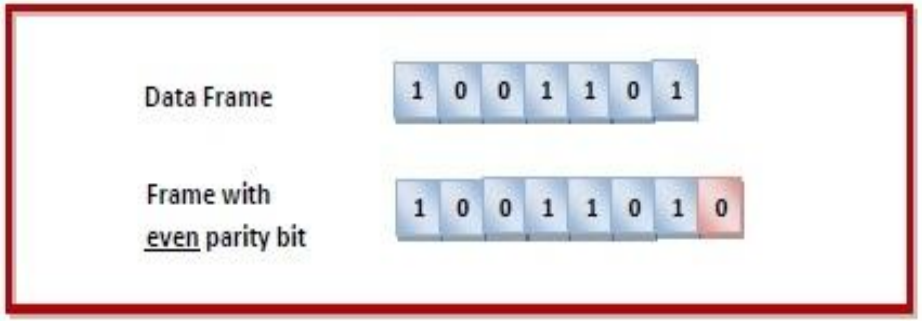
Total number of 1's in the given data bit should be even. So if the total number of 1's in the data bit is odd then a single 1 will be appended to make total number of 1's even else 0 will be appended(if total number of 1's are already even). Hence, if any error occurs, the parity check circuit will detect it at the receiver's end.

Example

- Suppose that a sender wants to send the data 1001101 using even parity check method. It will add the parity bit as shown below.



- The receiver will decide whether an error has occurred by counting whether the total number of 1s is even.



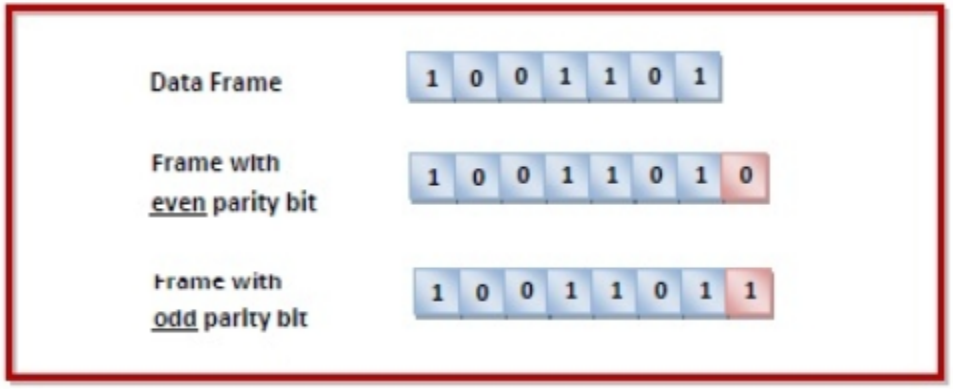
Odd Parity

- In odd parity system, if the total number of 1's in the given binary string (or data bits) are even then 1 is appended to make the total count of 1's as odd else 0 is appended.
- The receiver knows that whether sender is an odd parity generator or even parity generator. Suppose if sender is an odd parity generator then there must be an odd number of 1's in received binary string.
- If an error occurs to a single bit that is either bit is changed to 1 to 0 or 0 to 1, received binary bit will have an even number of 1's which will indicate an error.

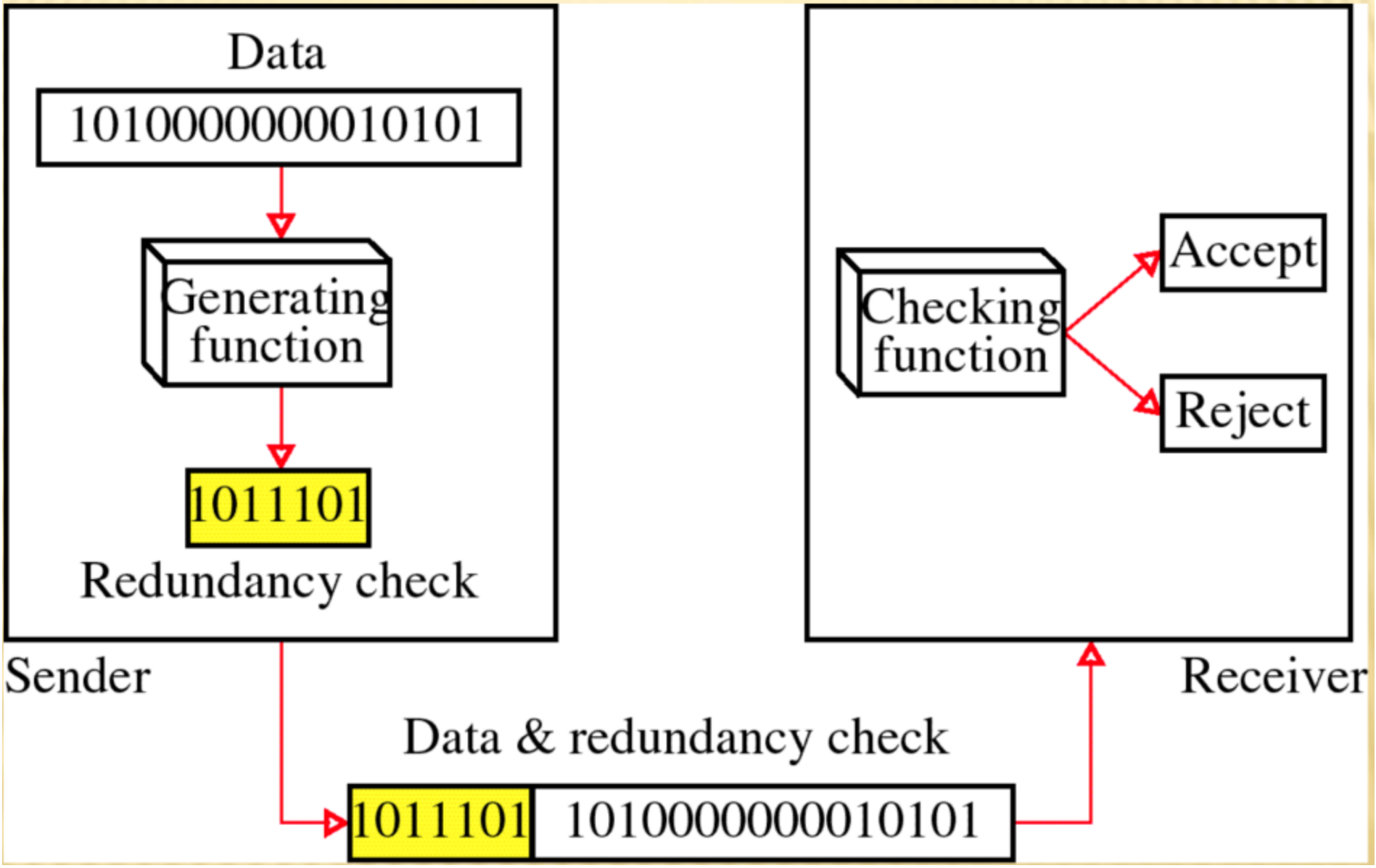
Error Detection by Parity Check

Sender's End – While creating a frame, the sender counts the number of 1s in it and adds the parity bit the value of which is determined as follows -

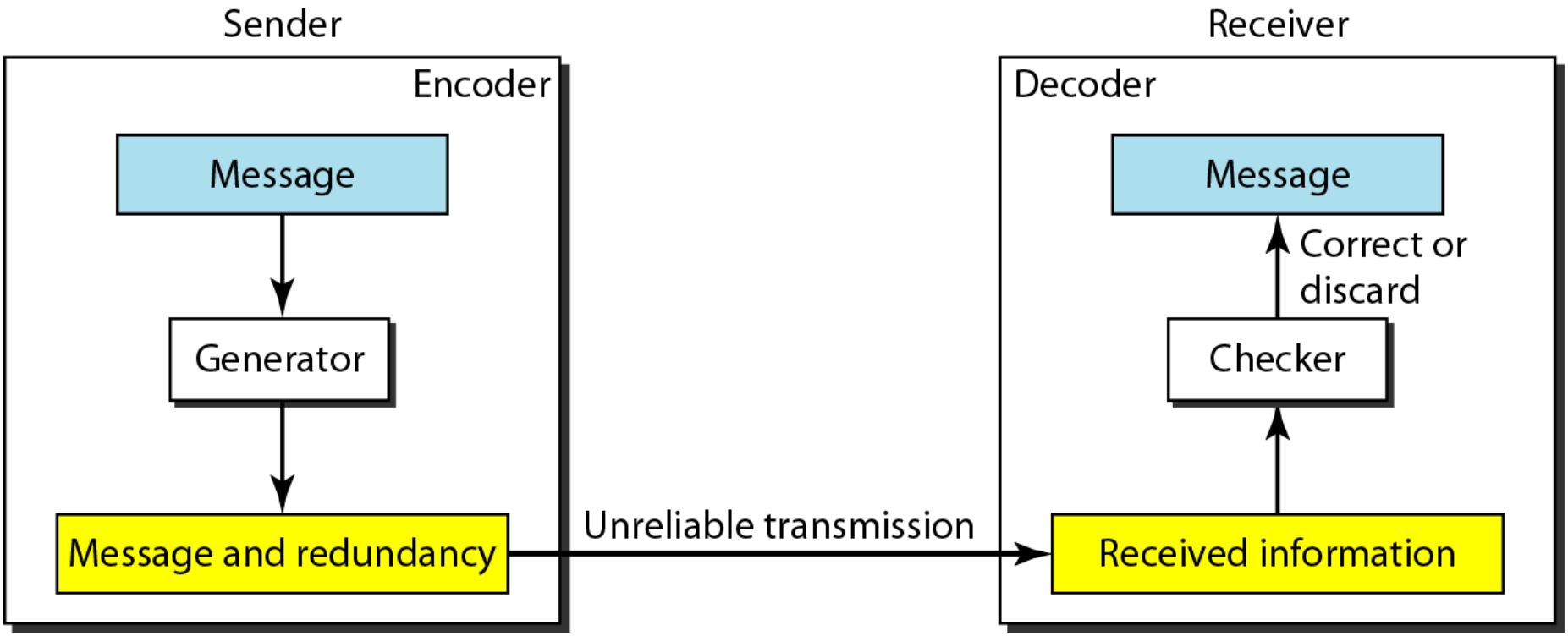
- In the case of even parity: If a number of 1s is even, the parity bit value is 0. If a number of 1s is odd, the parity bit value is 1.
- In case of odd parity: If a number of 1s is odd, the parity bit value is 0. If a number of 1s is even, the parity bit value is 1.



Receiver's End – On receiving a frame, the receiver counts the number of 1s in it. In case of even parity check, if the count of 1s is even, the frame is accepted, otherwise, it is rejected. In case of odd parity check, if the count of 1s is odd, the frame is accepted, otherwise, it is rejected.


Data
1010000000010101
Generating function
1011101
Redundancy check
Sender
Checking function
Accept
Reject
Receiver
Data & redundancy check
1011101
1010000000010101

The structure of encoder and decoder



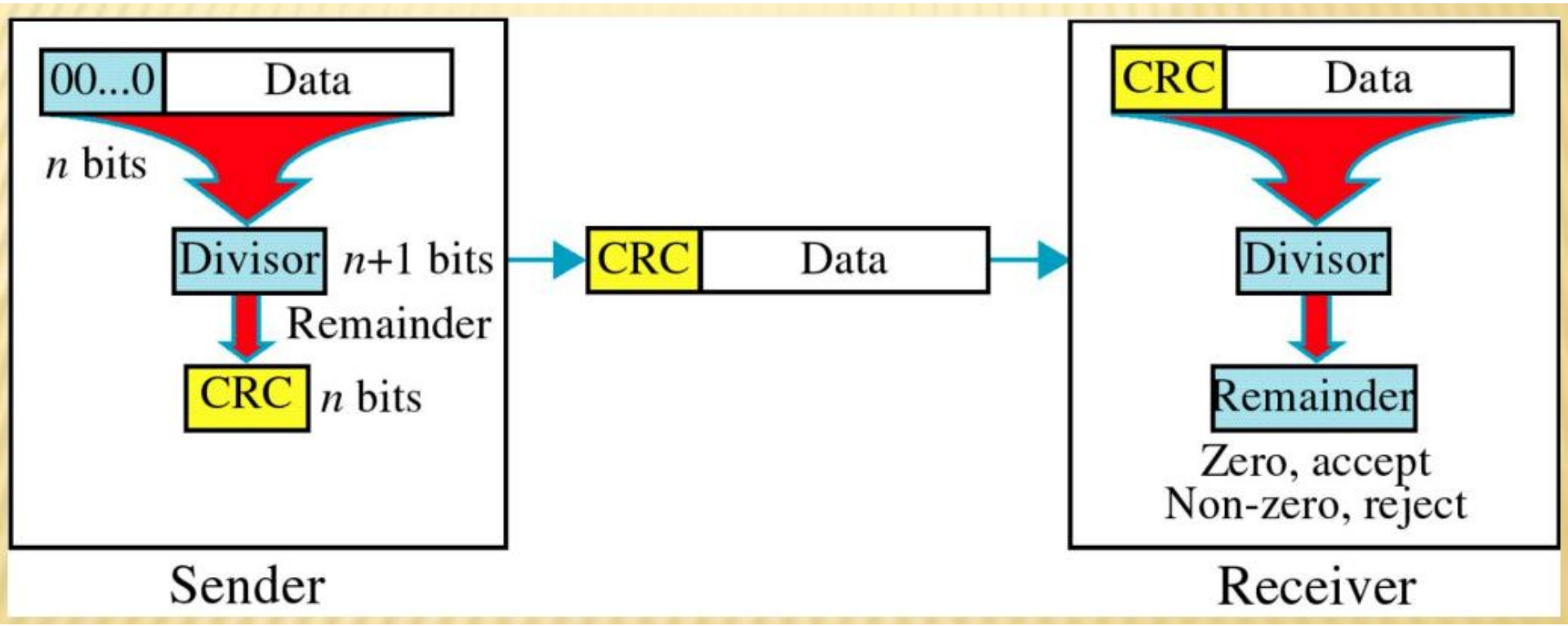
BLOCK CODING

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

CYCLIC CODES

***Cyclic codes** are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.*

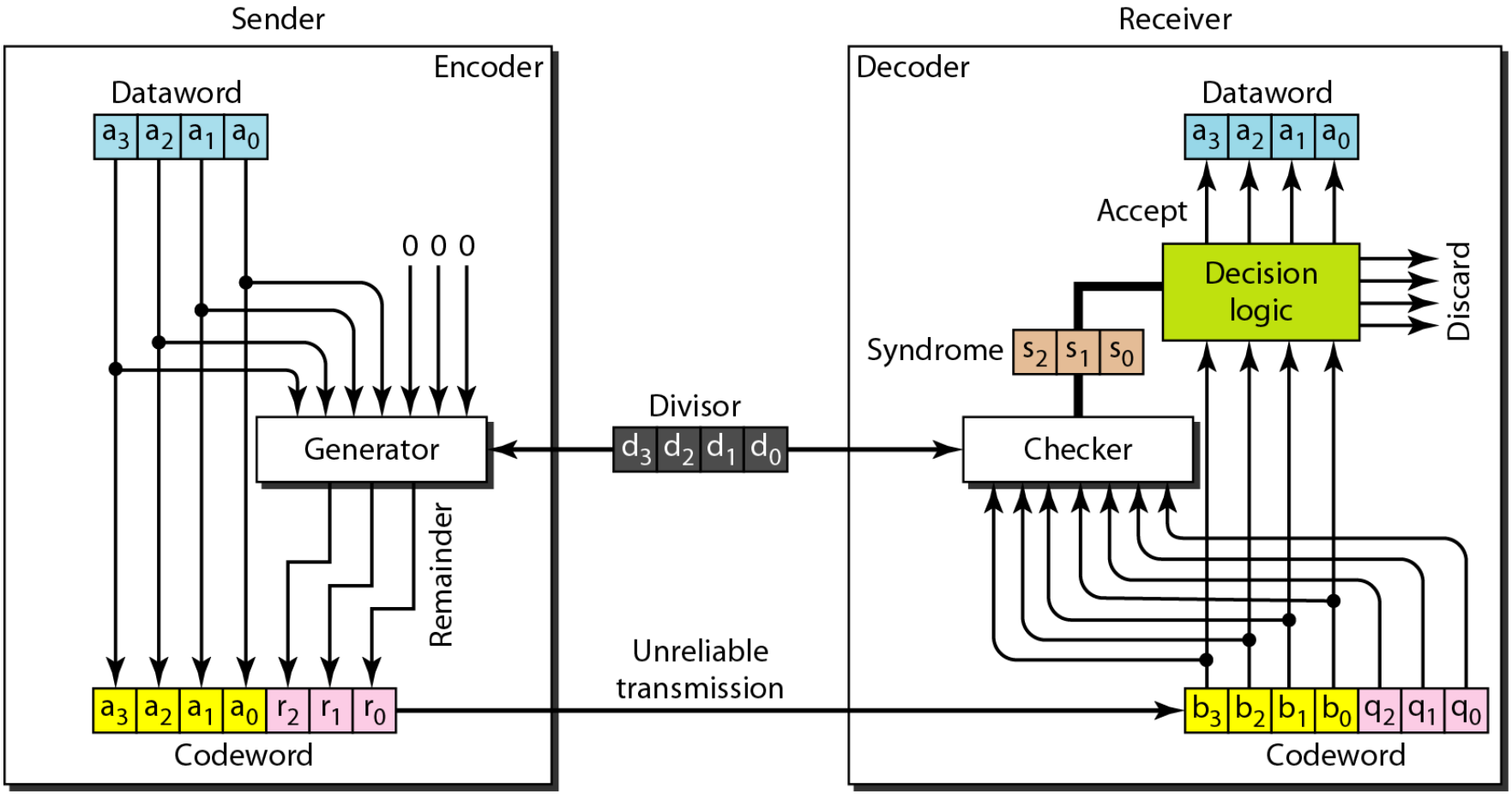
Cyclic Redundancy Check CRC



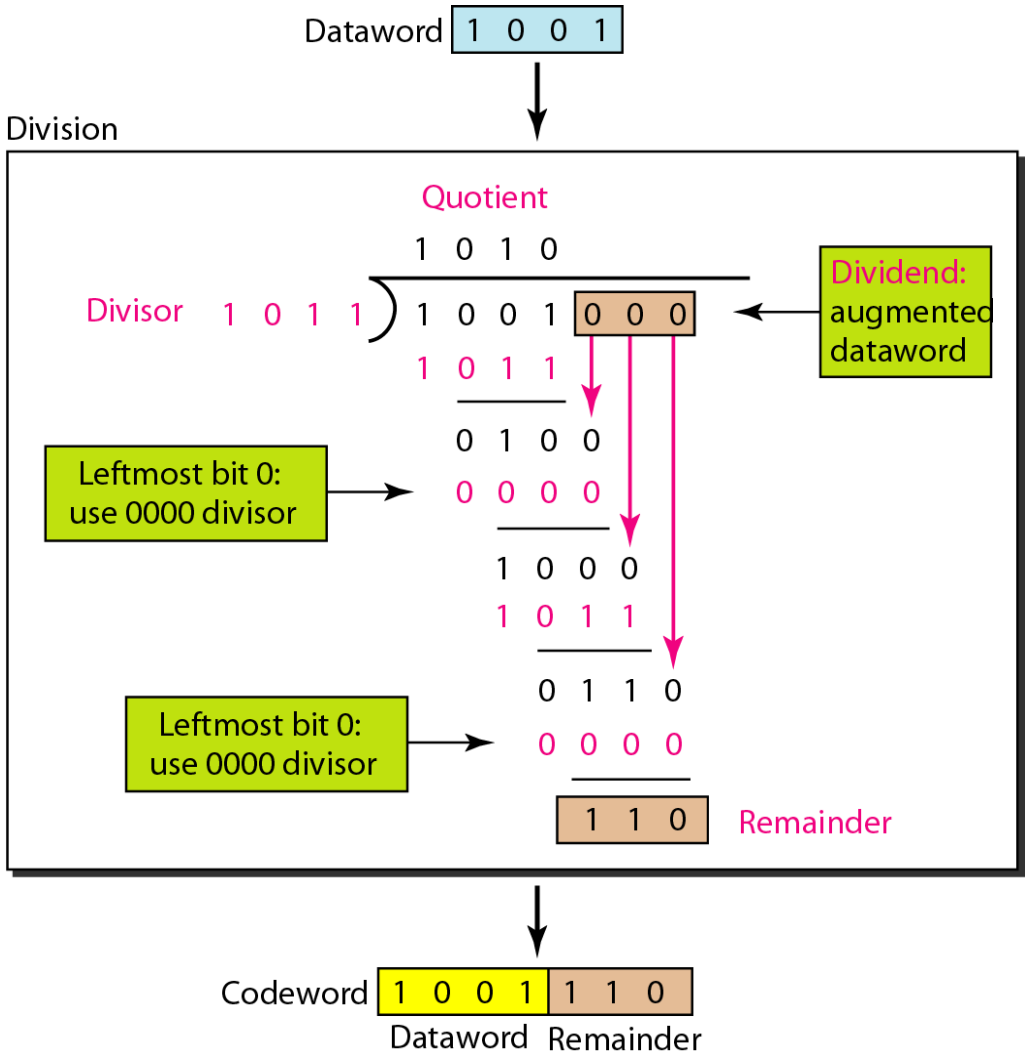
CYCLIC REDUNDANCY CHECK

- Given a k-bit frame or message, the transmitter generates an n-bit sequence, known as a frame check sequence (FCS), so that the resulting frame, consisting of (k+n) bits, is exactly divisible by some predetermined number.
- The receiver then divides the incoming frame by the same number and, if there is no remainder, assumes that there was no error.

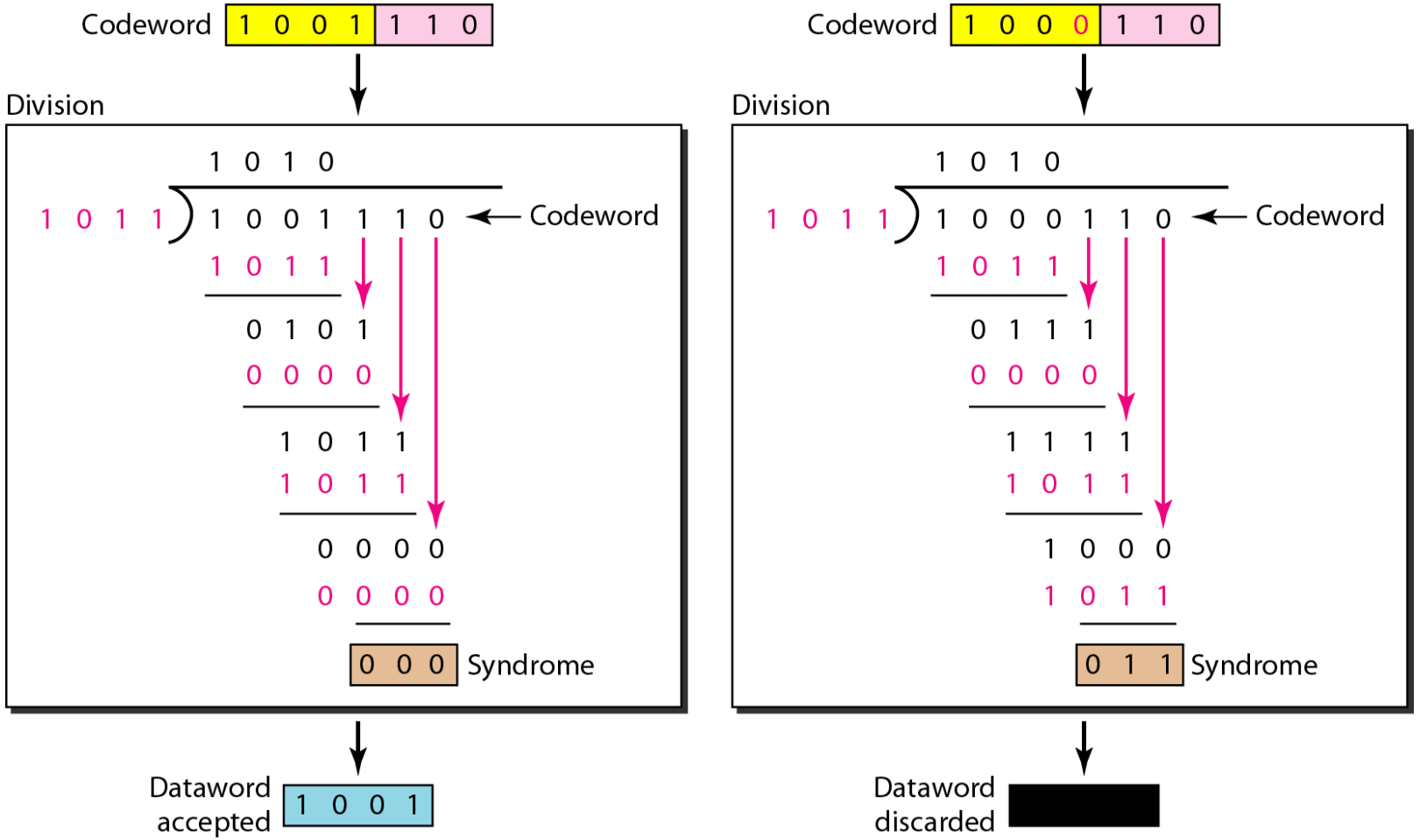
CRC encoder and decoder



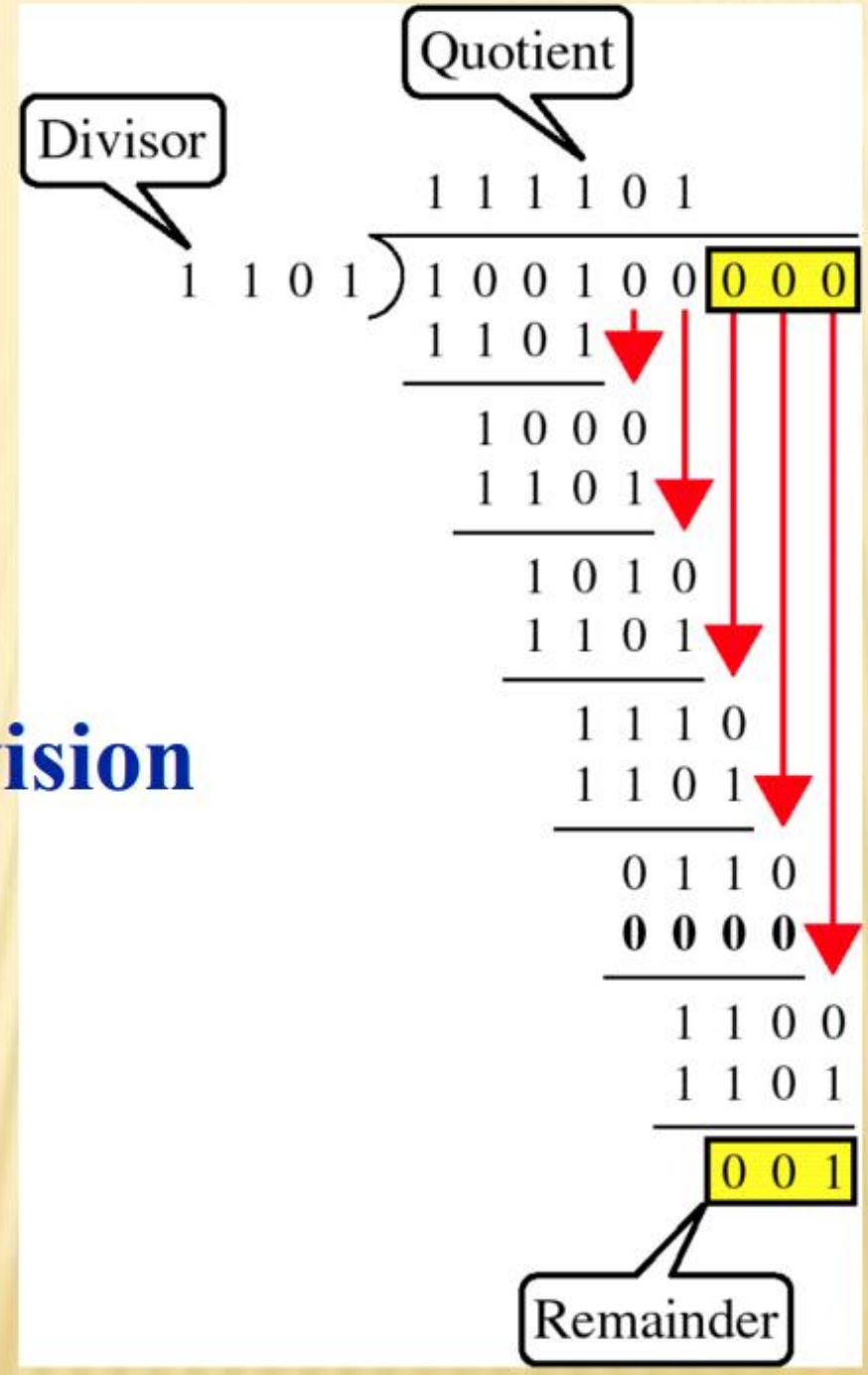
Division in CRC encoder



Division in the CRC decoder for two cases



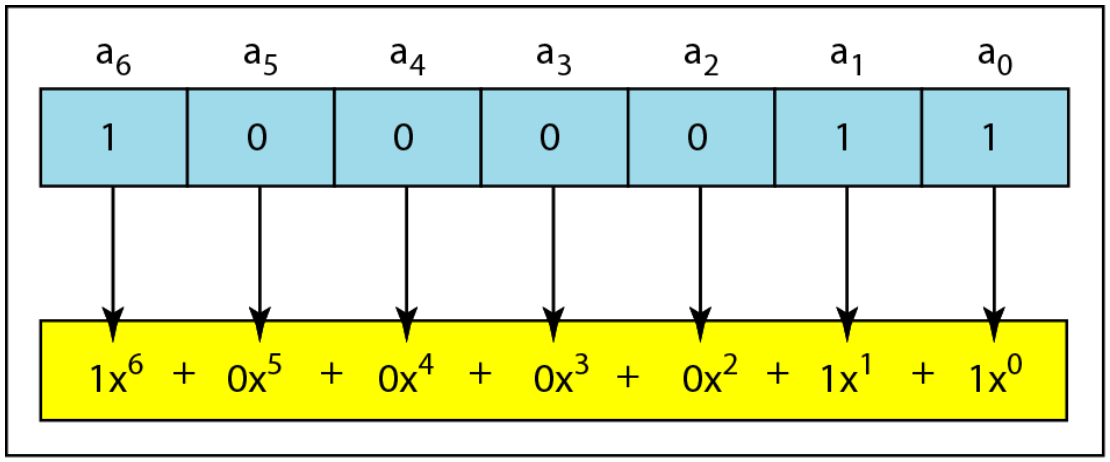
Binary Division



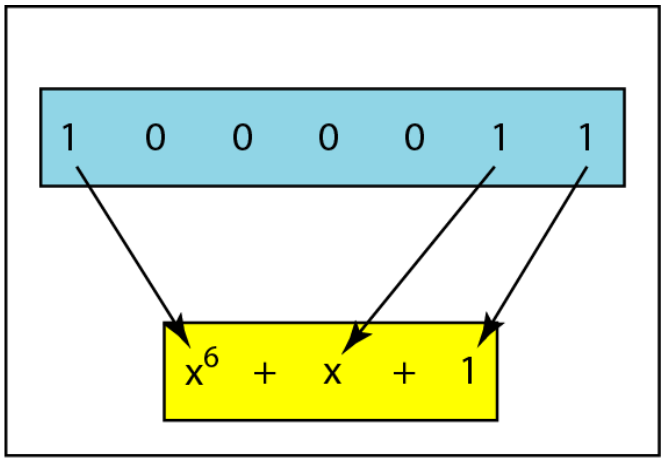
Note

The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

A polynomial to represent a binary word



a. Binary pattern and polynomial



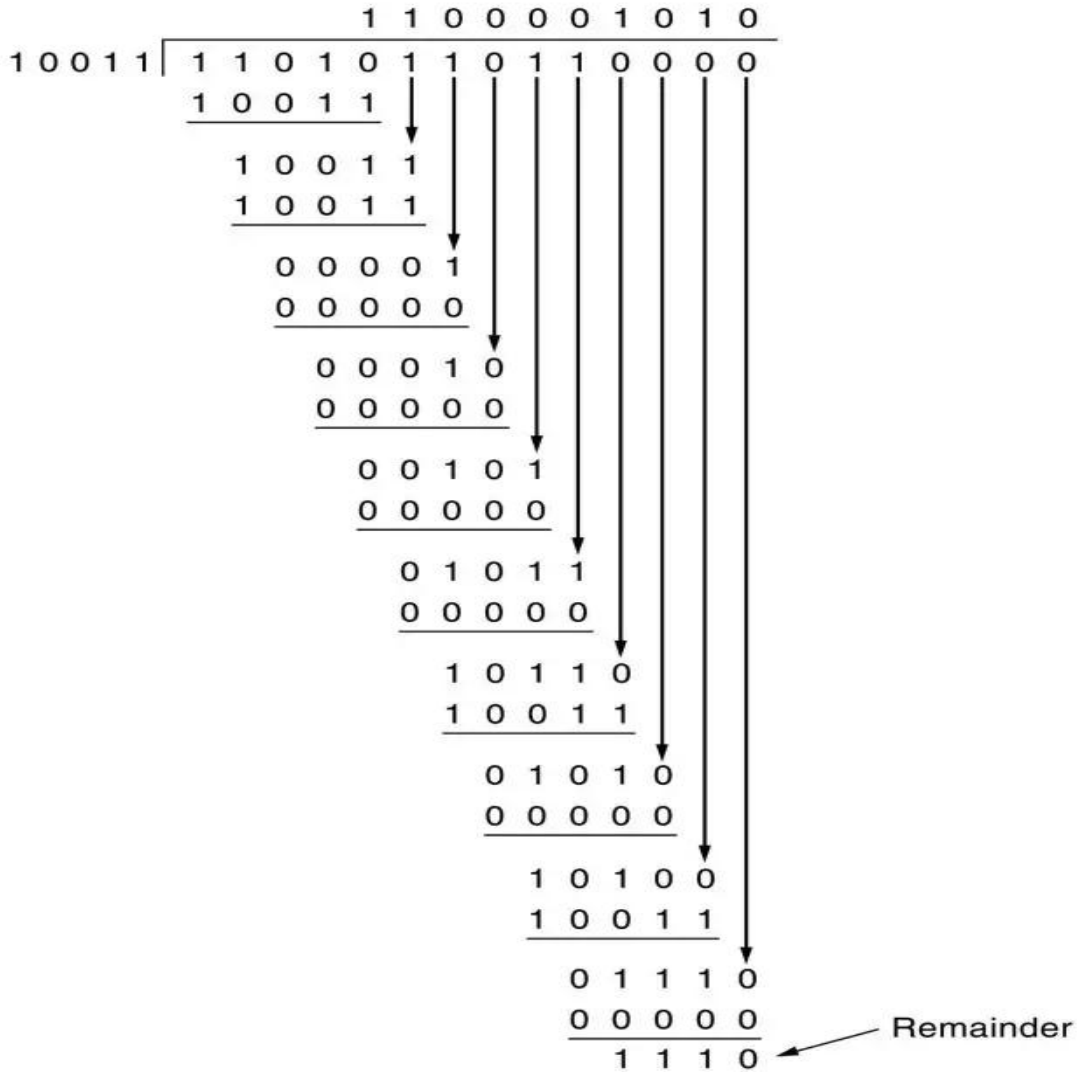
b. Short form

Standard polynomials

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

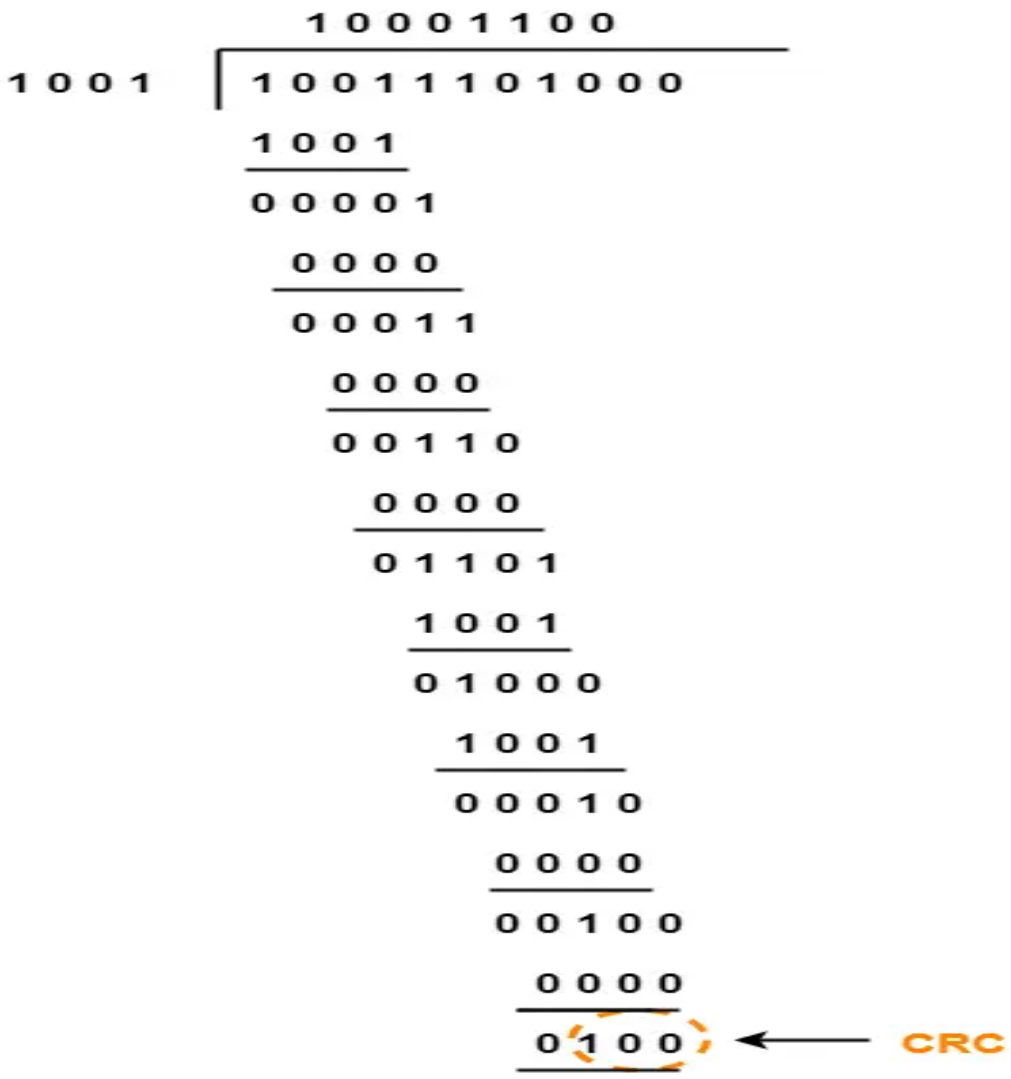
Q1

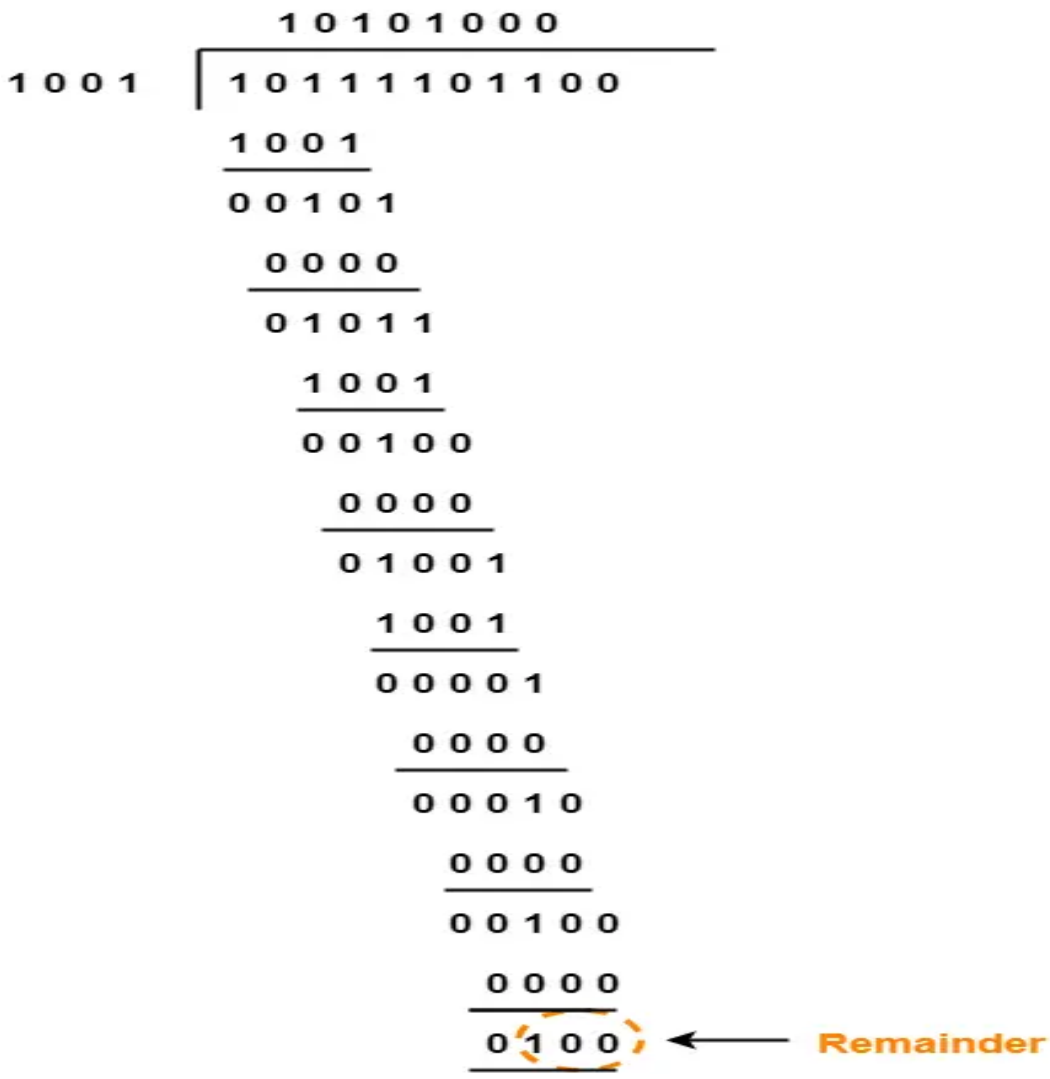
- A bit stream 1101011011 is transmitted using the standard CRC method. The generator polynomial is x^4+x+1. What is the actual bit string transmitted?


1 1 0 0 0 0 1 0 1 0
10011 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
1 0 0 1 1
1 0 0 1 1
1 0 0 1 1
0 0 0 0 1
0 0 0 0 0
0 0 0 1 0
0 0 0 0 0
0 0 1 0 1
0 0 0 0 0
0 1 0 1 1
0 0 0 0 0
1 0 1 1 0
1 0 0 1 1
0 1 0 1 0
0 0 0 0 0
1 0 1 0 0
1 0 0 1 1
0 1 1 1 0
0 0 0 0 0
1 1 1 0
Remainder

Q2

- A bit stream 10011101 is transmitted using the standard CRC method. The generator polynomial is x^3+1.What is the actual bit string transmitted? Suppose the third bit from the left is inverted during transmission. How will receiver detect this error?


1 0 0 0 1 1 0 0
1 0 0 1
1 0 0 1 1 1 0 1 0 0 0
1 0 0 1
0 0 0 0 1
0 0 0 0
0 0 0 1 1
0 0 0 0
0 0 1 1 0
0 0 0 0
0 1 1 0 1
1 0 0 1
0 1 0 0 0
1 0 0 1
0 0 0 1 0
0 0 0 0
0 0 1 0 0
0 0 0 0
0 1 0 0
CRC


1 0 1 0 1 0 0 0
1 0 0 1
1 0 1 1 1 1 0 1 1 0 0
1 0 0 1
0 0 1 0 1
0 0 0 0
0 1 0 1 1
1 0 0 1
0 0 1 0 0
0 0 0 0
0 1 0 0 1
1 0 0 1
0 0 0 0 1
0 0 0 0
0 0 0 1 0
0 0 0 0
0 0 1 0 0
0 0 0 0
0 1 0 0
Remainder

Q3

Consider the cyclic redundancy check (CRC) based error detecting scheme having the generator polynomial $X^3 + X + 1$. Suppose the message $m_4m_3m_2m_1m_0 = 11000$ is to be transmitted. Check bits $c_2c_1c_0$ are appended at the end of the message by the transmitter using the above CRC scheme. The transmitted bit string is denoted by $m_4m_3m_2m_1m_0c_2c_1c_0$. The value of the check bit sequence $c_2c_1c_0$ is

Q4

Consider the message sender wants to send is 1010001101, and the generator polynomial is $x^5 + x^4 + x^2 + 1$. Find the message transmitted by the sender. If the receiver receives the message, check if the receiver receives the correct message or not.

CHECKSUM

The checksum is used in the Internet by several protocols for error checking

Topics discussed in this section:

Idea
One's Complement
Internet Checksum

Example

Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

Example

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 11, 12, 0, 6, −36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

Example

How can we represent the number 21 in one's complement arithmetic using only four bits?

Solution

The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

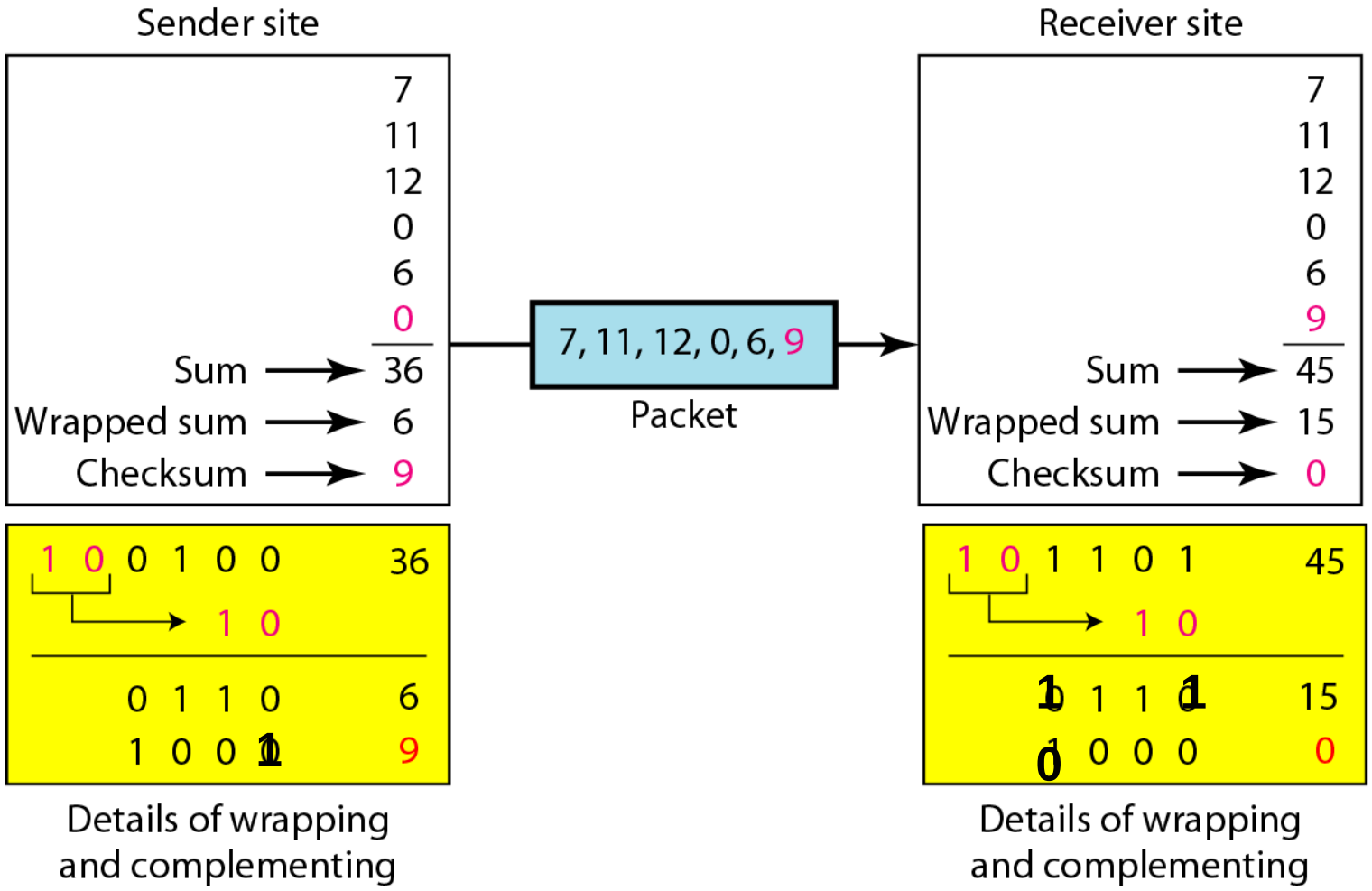
Example

How can we represent the number −6 in one's complement arithmetic using only four bits?

Solution

In one's complement arithmetic, the negative or complement of a number is found by inverting all bits. Positive 6 is 0110; negative 6 is 1001. If we consider only unsigned numbers, this is 9. In other words, the complement of 6 is 9.

Example


Sender site
7
11
12
0
6
0
Sum
36
Wrapped sum
6
Checksum
9
7, 11, 12, 0, 6, 9
Packet
Receiver site
7
11
12
0
6
9
Sum
45
Wrapped sum
15
Checksum
0
1 0 0 1 0 0 36
1 0
0 1 1 0 6
1 0 0 1 9
Details of wrapping and complementing
1 0 1 1 0 1 45
1 0
0 1 1 1 15
1 0 0 0 0
Details of wrapping and complementing

Note

Sender site:

1. **The message is divided into 16-bit words.**
2. **The value of the checksum word is set to 0.**
3. **All words including the checksum are added using one's complement addition.**
4. **The sum is complemented and becomes the checksum.**
5. **The checksum is sent with the data.**

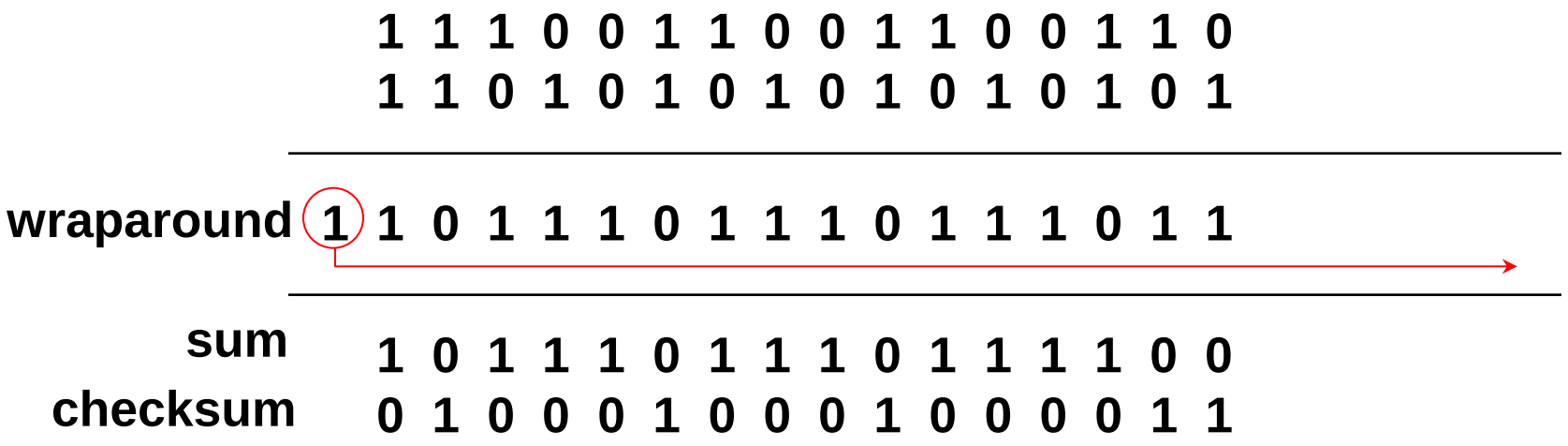
Note

Receiver site:

1. **The message (including checksum) is divided into 16-bit words.**
2. **All words are added using one's complement addition.**
3. **The sum is complemented and becomes the new checksum.**
4. **If the value of checksum is 0, the message is accepted; otherwise, it is rejected.**

Internet Checksum Example

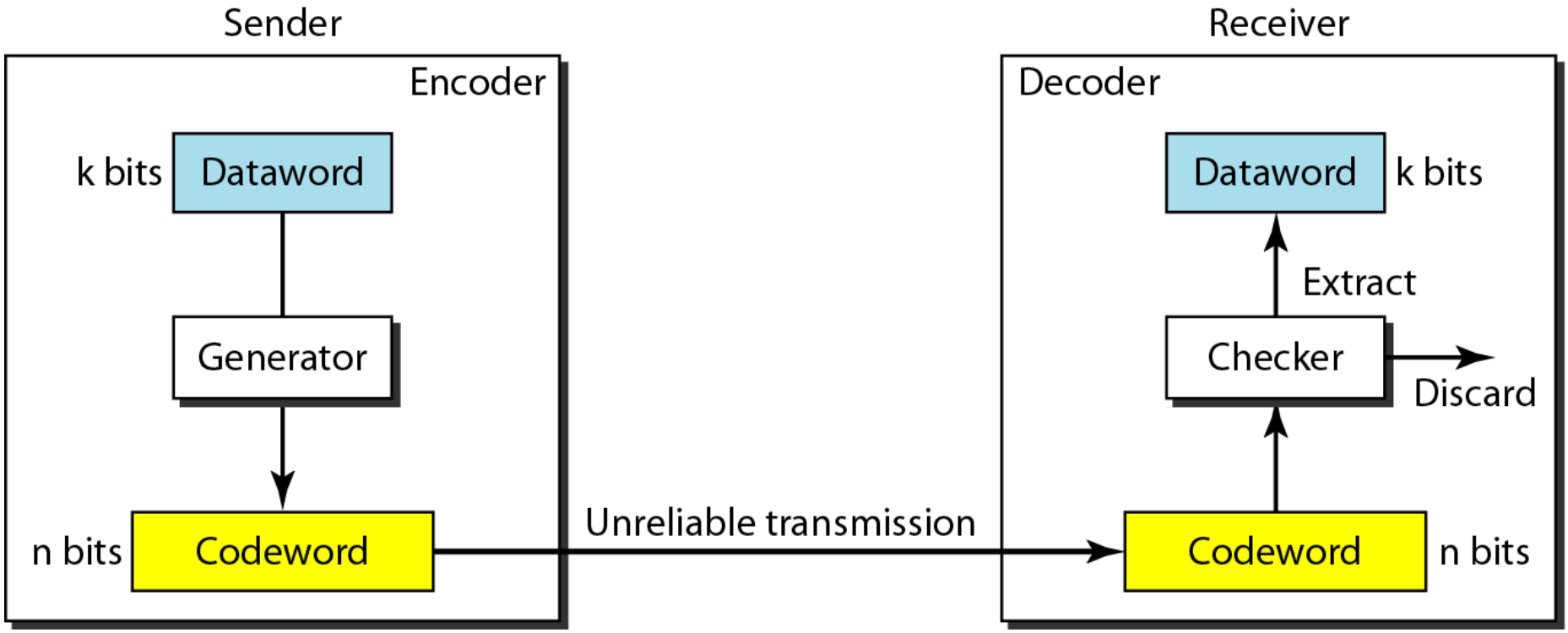
- Note
  - When adding numbers, a carryout from the most significant bit needs to be added to the result
- Example: add two 16-bit integers



10.53

10.54

Process of error detection in block coding



Example

Let us assume that k = 2 and n = 3. Table shows the list of datawords and codewords (even parity). It is only good for detecting one bit error.

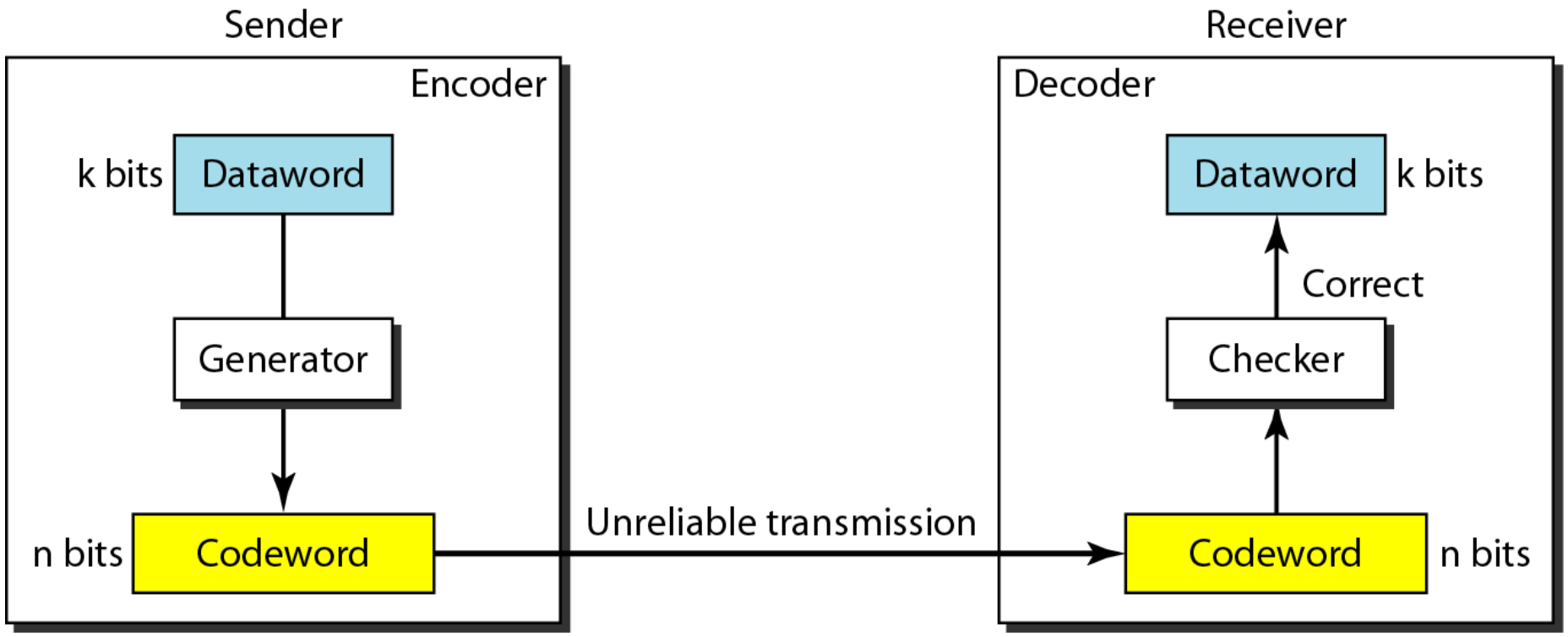
Table *A code for error detection*

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Note

An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

Structure of encoder and decoder in error correction



Note

The Hamming distance between two words is the number of differences between corresponding bits.

Example

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance d(000, 011) is 2 because

$000 \oplus 011$ is 011 (two 1s)

2. The Hamming distance d(10101, 11110) is 3 because

$10101 \oplus 11110$ is 01011 (three 1s)

Note

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Example

Find the minimum Hamming distance of the coding scheme

Solution

We first find all Hamming distances.

$d(000, 011) = 2$	$d(000, 101) = 2$	$d(000, 110) = 2$	$d(011, 101) = 2$
$d(011, 110) = 2$	$d(101, 110) = 2$		

The d_{min} in this case is 2.

Note

To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

Why?

More than s-bit error is possible to detect, but not guaranteed.

Figure 10.8 *Geometric concept for finding d_{min} in error detection*

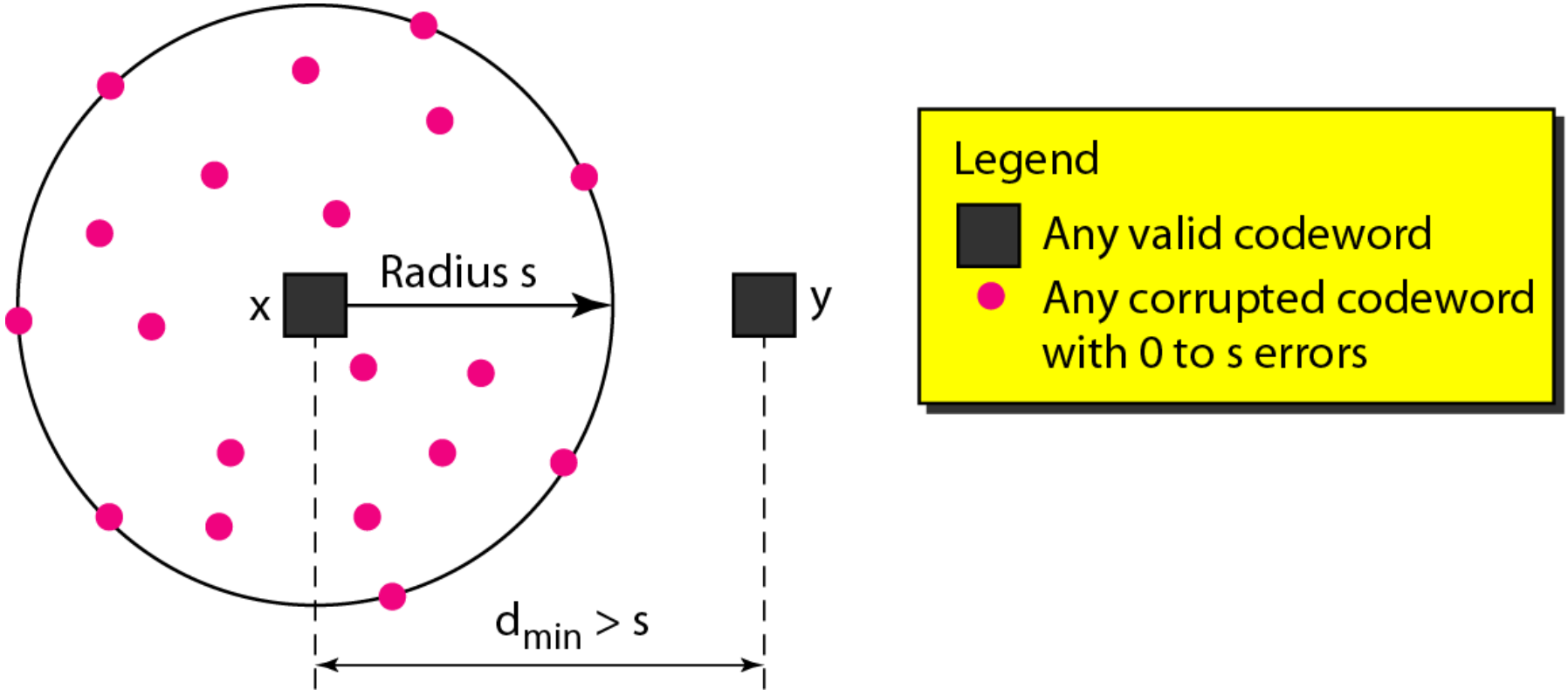
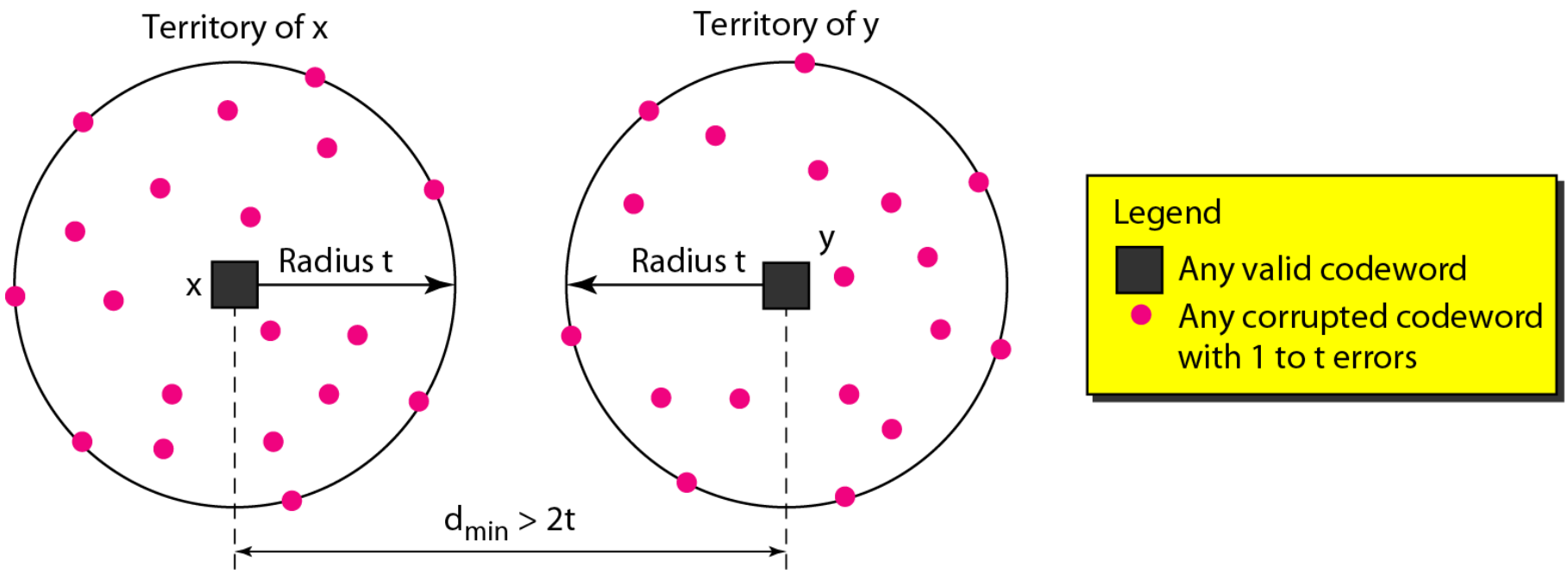


Figure 10.9 *Geometric concept for finding d_{min} in error correction*



Note

To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.

Example 10.9

A code scheme has a Hamming distance $d_{min} = 4$. What is the error detection and correction capability of this scheme?

Solution

This code guarantees the detection of up to three errors ($s = 3$), but it can correct up to one error. In other words, if this code is used for error correction, part of its capability is wasted. Error correction codes should have an odd minimum distance (3, 5, 7, . . .).

LINEAR BLOCK CODES

Almost all block codes used today belong to a subset called linear block codes. A linear block code is a code in which the XOR (addition modulo-2) of two valid codewords creates another valid codeword.

Topics discussed in this section:

Minimum Distance for Linear Block Codes
Some Linear Block Codes

Note

In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.

Example 10.10

Let us see if the two codes we defined in Table 10.1 belong to the class of linear block codes.

The scheme in Table 10.1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Example 10.11

Note

In a linear block code, the minimum Hamming distance is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

In our first code (Table 10.1), the numbers of 1s in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$.

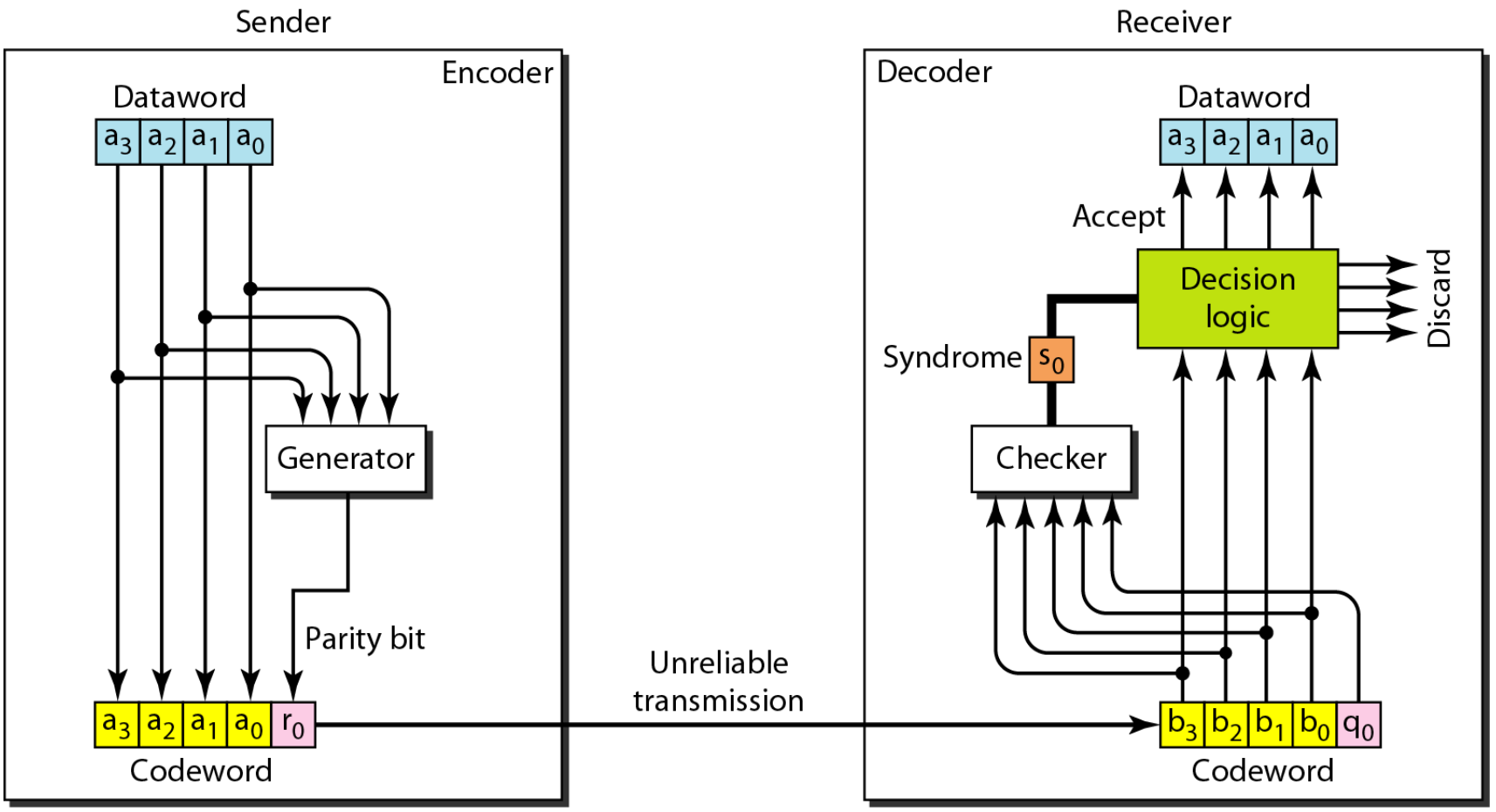
Note

A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$.

Table 10.3 ***Simple parity-check code C(5, 4)***

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Figure 10.10 *Encoder and decoder for simple parity-check code*



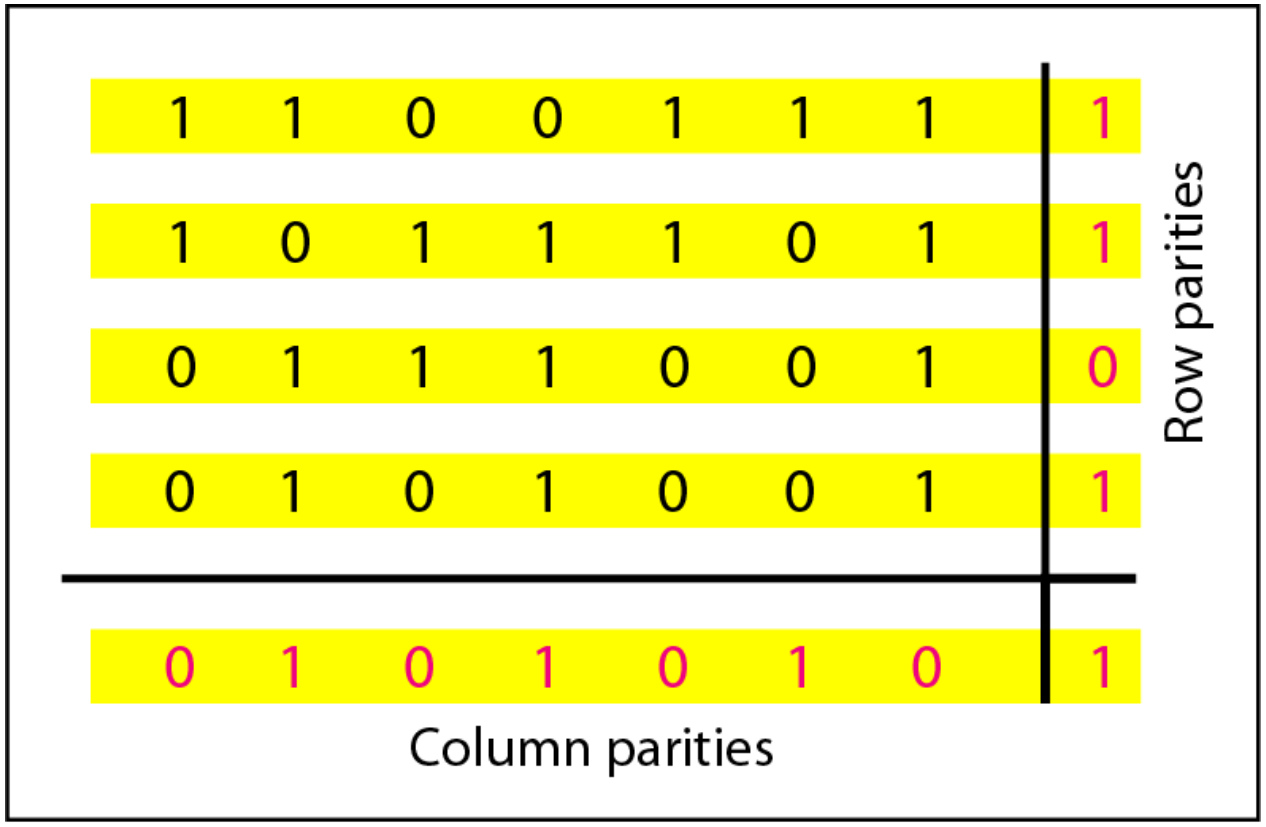
Parity-check code

- $r0 = a3 + a2 + a1 + a1$ (modulo-2)
- Syndrome (calculated by the receiver)
- $s0 = b3 + b2 + b1 + b0 + q0$ (modulo-2)

Note

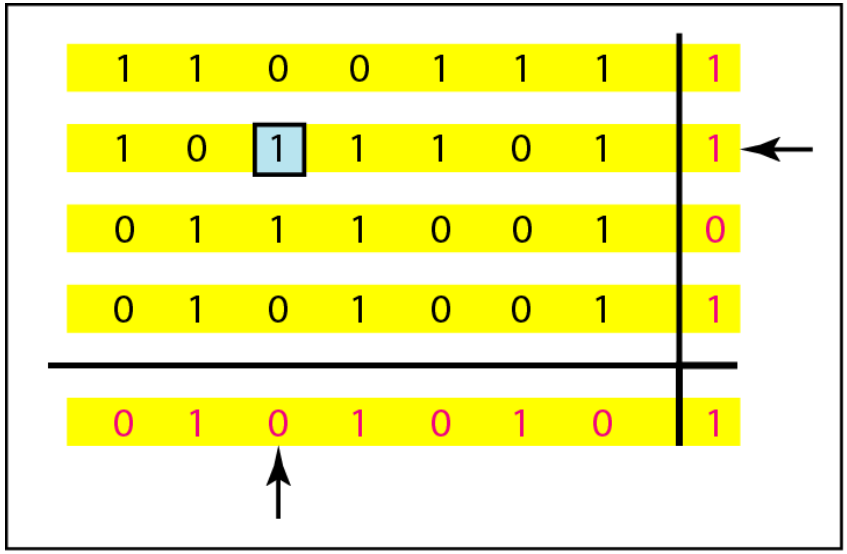
A simple parity-check code can detect an odd number of errors.

Two-dimensional parity-check code

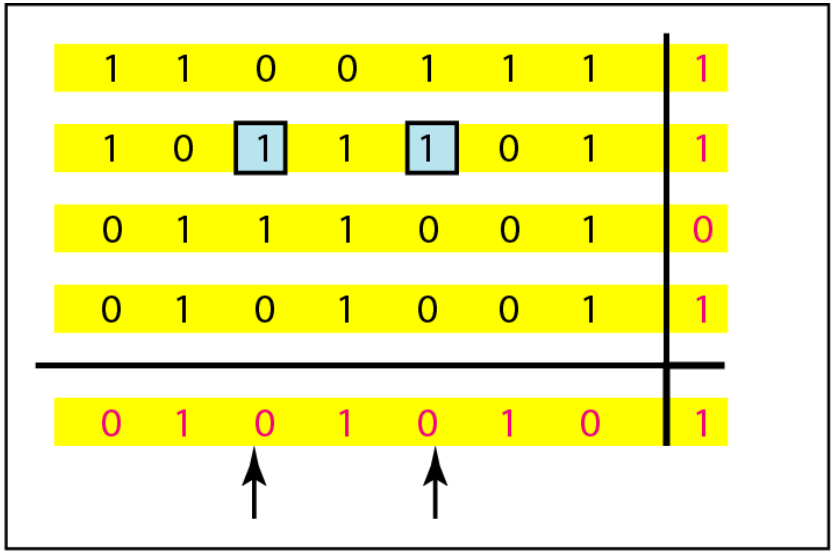


a. Design of row and column parities

Two-dimensional parity-check code

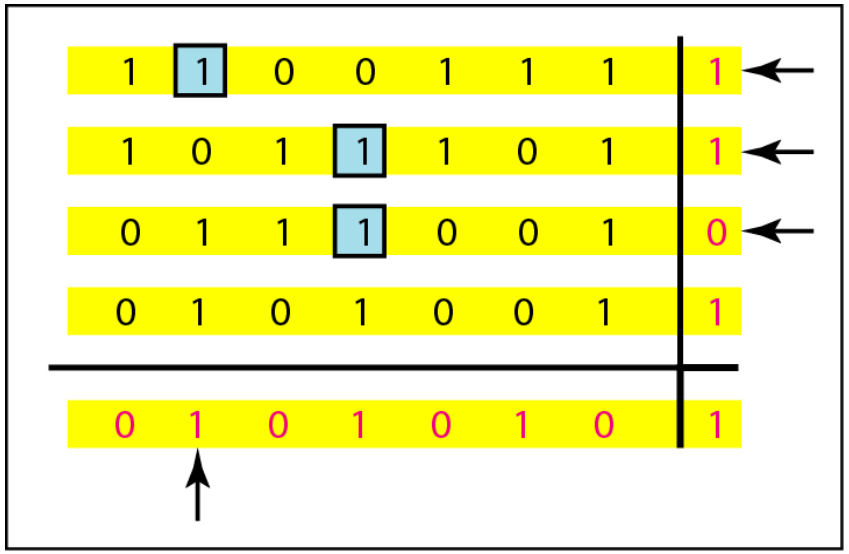


b. One error affects two parities

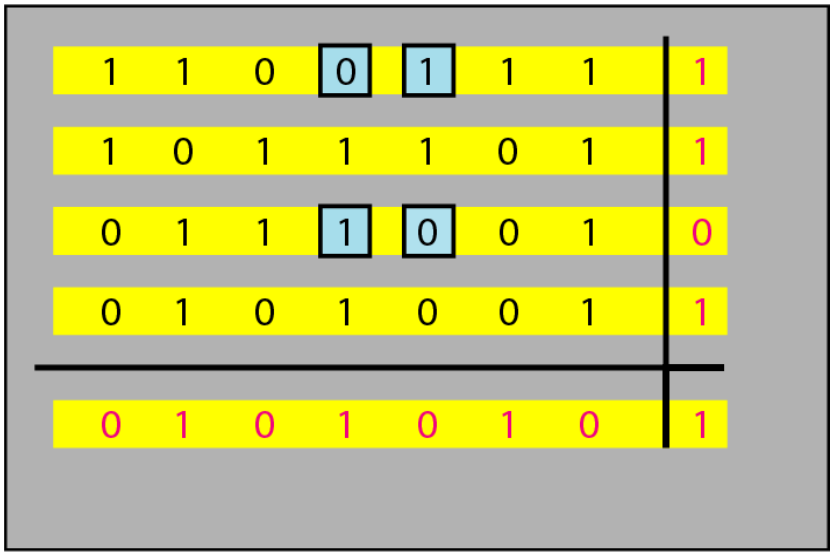


c. Two errors affect two parities

Two-dimensional parity-check code



d. Three errors affect four parities



e. Four errors cannot be detected

Two-dimensional parity-check can detect up to 3-bit errors

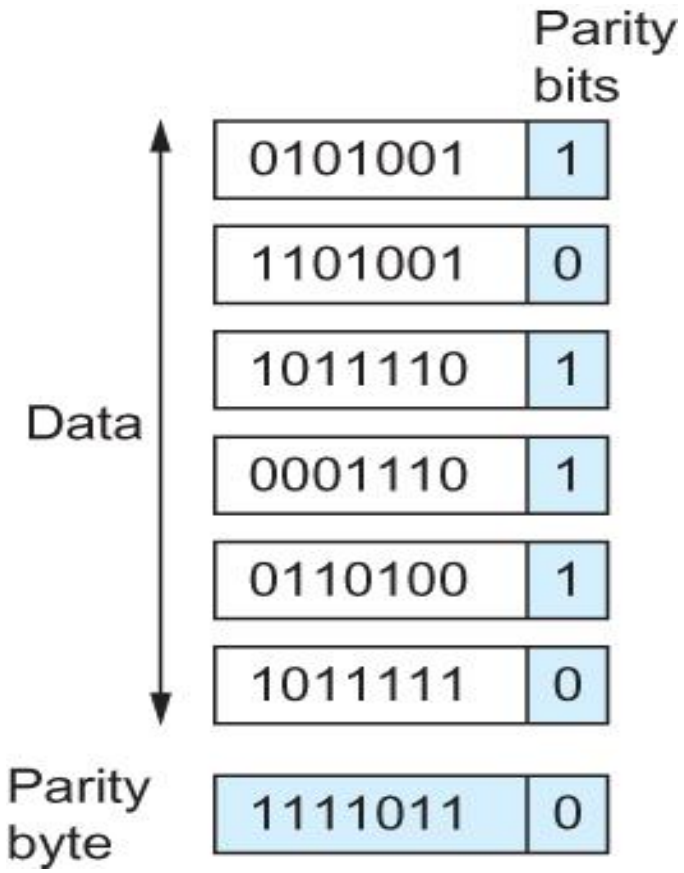
Two-dimensional parity

- Two-dimensional parity is exactly what the name suggests
- It is based on “simple” (one-dimensional) parity, which usually involves adding one extra bit to a 7-bit code to balance the number of 1s in the byte.

Two-dimensional parity

- Two-dimensional parity does a similar calculation for each bit position across each of the bytes contained in the frame
- This results in an extra parity byte for the entire frame, in addition to a parity bit for each byte
- Two-dimensional parity catches all 1-, 2-, and 3-bit errors and most 4-bit errors
- Two-dimensional Parity check bits are calculated for each row, which is equivalent to a simple parity check bit.
- Parity check bits are also calculated for all columns, then both are sent along with the data.
- At the receiving end, these are compared with the parity bits calculated on the received data.

Two-dimensional parity



Two Dimensional Parity

Example

Original Data

10011001	11100010	00100100	10000100

	Row parities
10011001	0
11100010	0
00100100	0
10000100	0
11011011	0

Column parities

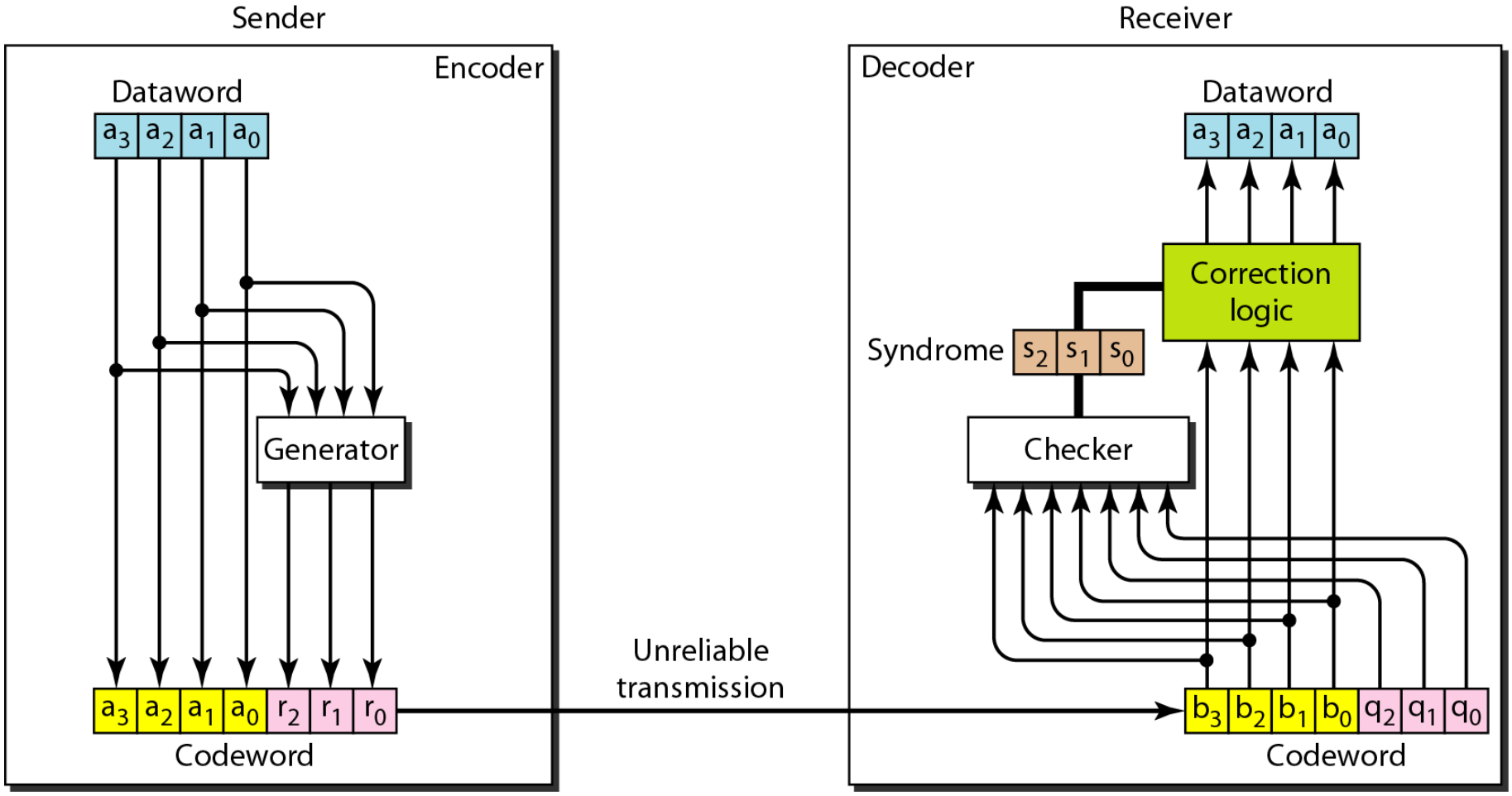
100110010	111000100	001001000	100001000	110110110

Data to be sent

Table 10.4 ***Hamming code C(7, 4)*** ***C(n,k)*** $d_{min}=3$

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

Figure 10.12 *The structure of the encoder and decoder for a Hamming code*



Hamming Code

- Parity checks are created as follow (using modulo-2)
  - *r0 = a2 + a1 + a0*
  - *r1 = a3 + a2 + a1*
  - *r2 = a1 + a0 + a3*

Hamming Code

- The checker in the decoder creates a 3-bit syndrome (s2s1s0).
- In which each bit is the parity check for 4 out of the 7 bits in the received codeword:
- $s0 = b2 + b1 + b0 + q0$
- $s1 = b3 + b2 + b1 + q1$
- $s2 = b1 + b0 + b3 + q2$
- The equations used by the checker are the same as those used by the generator with the parity-check bits added to the right-hand side of the equation.

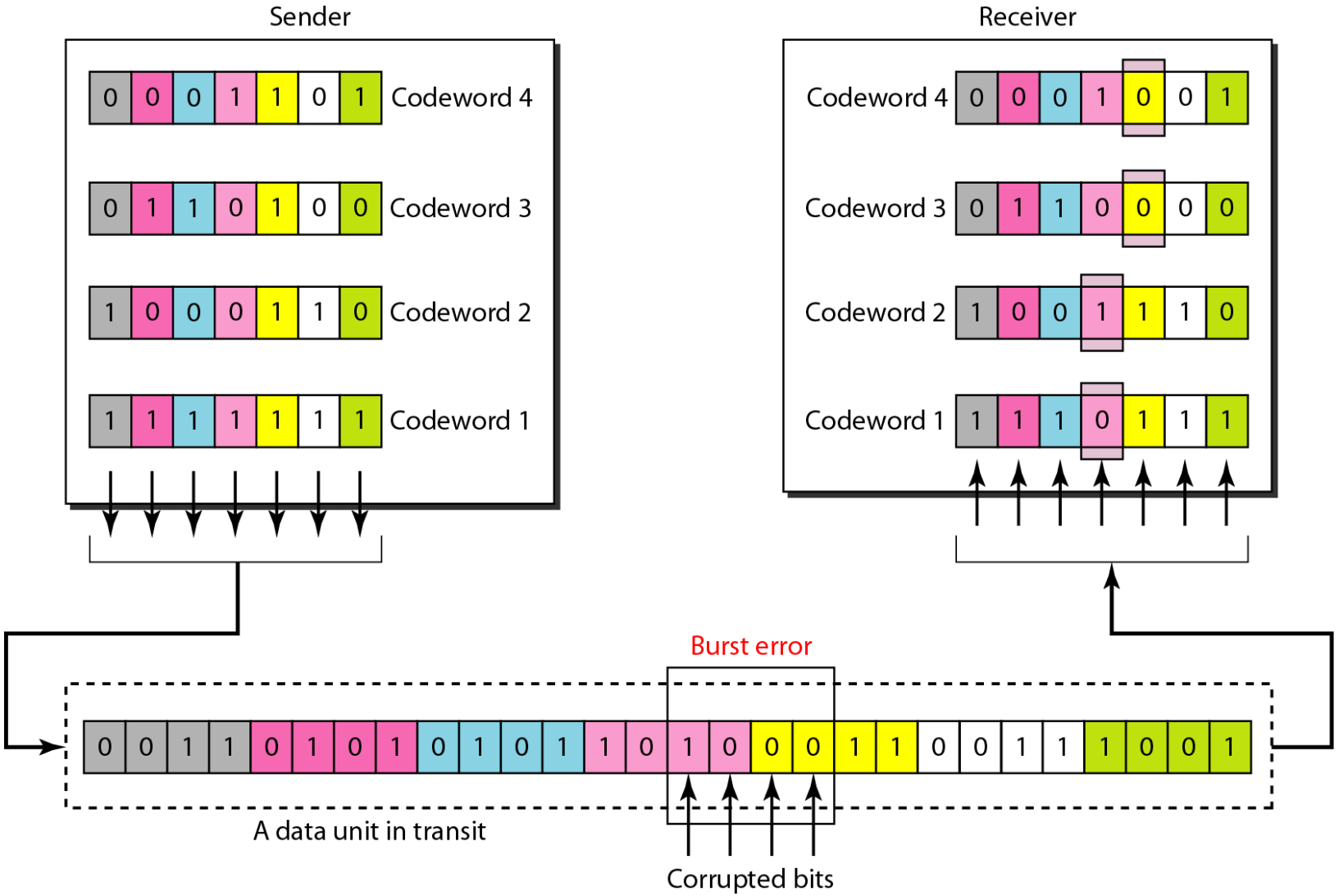
Table 10.5 ***Logical decision made by the correction logic analyzer***

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	q_1	b_2	q_2	b_0	b_3	b_1

Hamming code C(7, 4) can :

- ***detect up to 2-bit error*** $(d_{min} - 1)$
- ***can correct up to 1 bit error*** $(d_{min} - 1)/2$

Burst error correction using Hamming code



Split burst error between multiple codewords

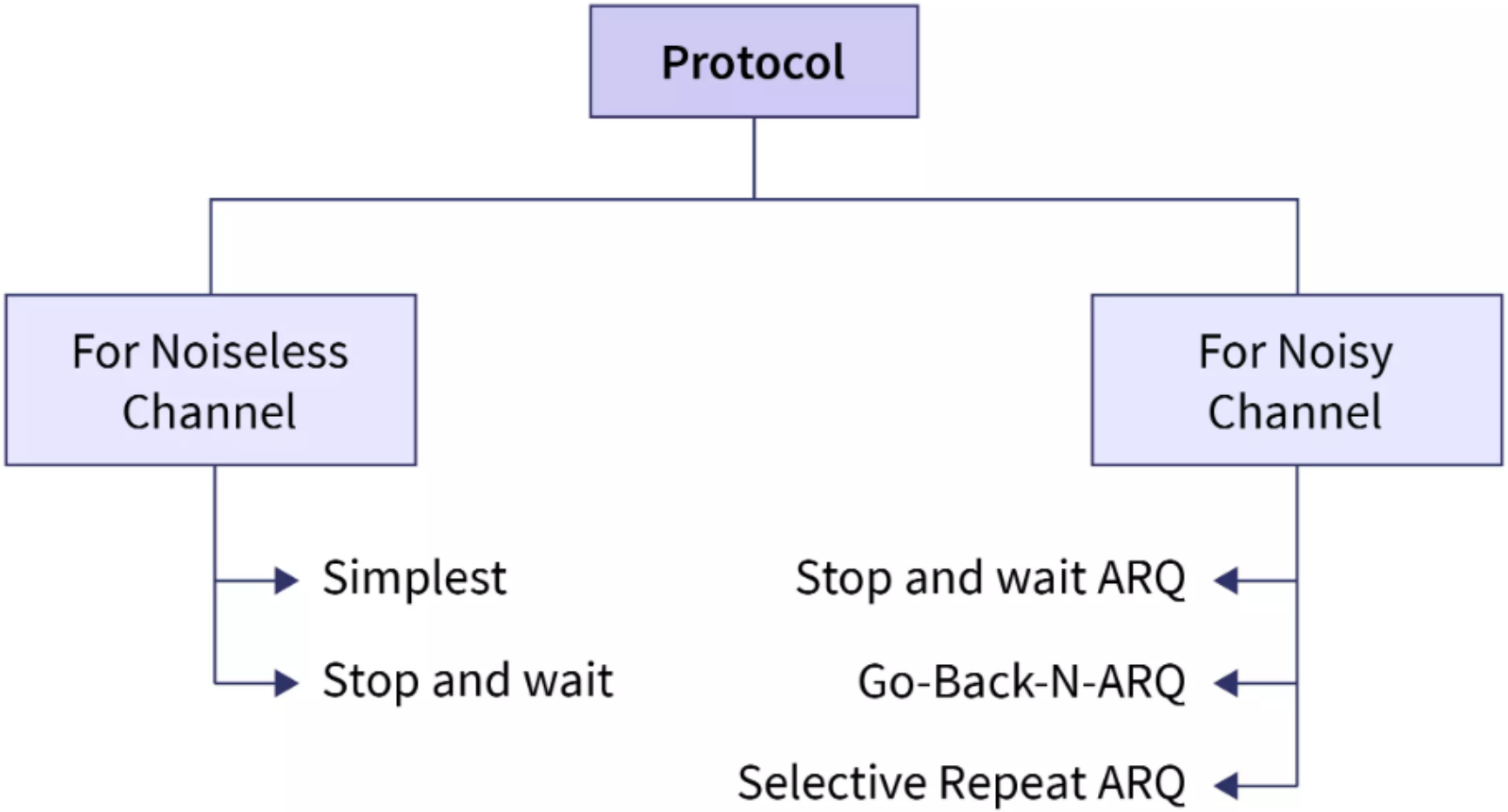
A CRC code with C(7, 4)

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

UNIT - II

- **Data Link Layer:** Fundamentals of Error Detection and Error Correction, Block coding, Hamming Distance, CRC;
- Flow Control and Error control protocols - Stop and Wait, Goback-N ARQ, Selective Repeat ARQ, Sliding Window, Piggybacking,
- Random Access,Multiple access protocols - Pure ALOHA, Slotted ALOHA, CSMA/CD, CDMA/CA

Flow Control



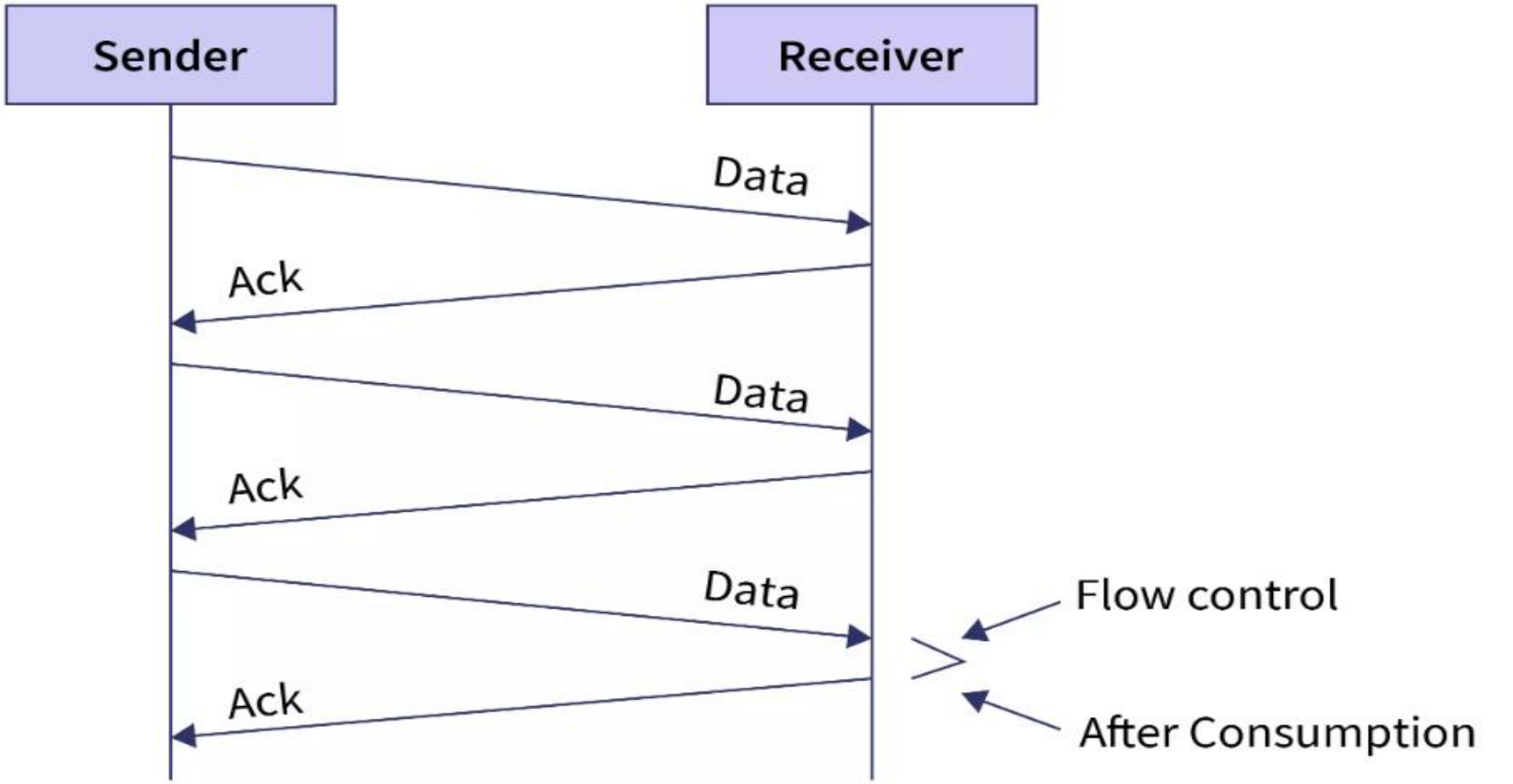
Stop and Wait Protocol

- The Stop-and-Wait Protocol is a fundamental flow control mechanism used in data communication where the sender transmits a single data packet and waits for an acknowledgment (ACK) from the receiver before sending the next packet.
- This ensures reliable data transmission but can lead to inefficiency in high-latency networks.

Working Mechanism

- The sender transmits a data packet to the receiver.
- The receiver processes the packet and sends an ACK if the packet is received correctly.
- The sender waits for the ACK before sending the next packet.
- If an ACK is not received within a timeout period, the sender retransmits the packet.
- This cycle continues until all packets are successfully delivered.

Data transmission in the stop-and-wait protocol



Note

- Suppose there is a situation where the sender is sending the data at a rate higher than the rate of the receiver to process and receive the data. Flow Control protocol ensures that the data does not get lost.
- The flow control protocols ensure that the sender sends the data only at a rate that the receiver can receive and process it.
- This is a flow control protocol that works in a noiseless channel.
- Noiseless channel is an idealistic channel in which no data frames are lost, corrupted, or duplicated.
- After sending the data, the sender will stop and wait until he/she receives an acknowledgment from the receiver.
- Flow control service is provided by the data link layer.

Advantages of Stop and Wait Protocol

✓ Simple Implementation – Easy to design and manage.

✓ Reliable Transmission – Ensures that each packet is correctly received before proceeding.

Disadvantages

❌ **Low Efficiency** – Due to waiting for an ACK after every packet, transmission is slow.

❌ **High Latency** – Delays increase in networks with long propagation times.

Example Scenario

- Consider a sender transmitting a 1 KB packet over a network.
- The round-trip time (RTT) is 100 ms.
- The sender must wait for 100 ms before sending the next packet.
- This leads to **poor utilization of the link**.

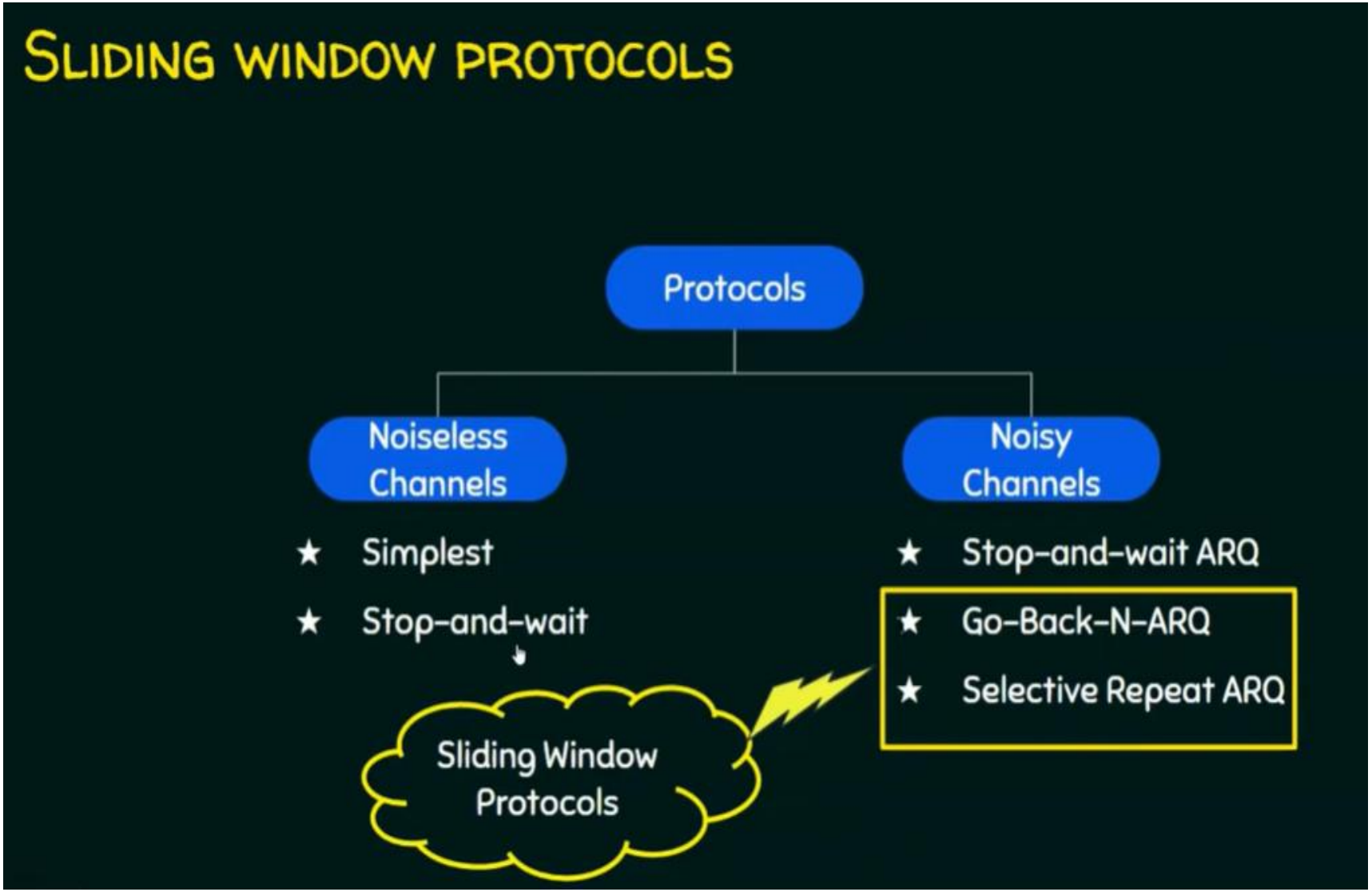
Stop-and-Wait ARQ (Automatic Repeat reQuest)

- Stop-and-Wait ARQ is a reliable data transmission protocol that ensures error-free communication by incorporating error detection and retransmission in the basic Stop-and-Wait protocol.
- It is commonly used in Data Link Layer and Transport Layer communication.

Working Mechanism

- Sender sends a data frame (packet).
- Receiver checks for errors using mechanisms like CRC (Cyclic Redundancy Check).
- If no error is detected, the receiver sends an ACK (Acknowledgment).
- If an error is detected, the receiver discards the frame and does not send an ACK.
- If the sender does not receive the ACK within a timeout period, it retransmits the frame.
- This process continues until all frames are successfully received.

Sliding window protocols



Go-Back-N ARQ (Automatic Repeat reQuest)

- Used in data link layer and transport layer for error control in data transmission.
- It is a sliding window protocol that allows the sender to transmit multiple frames before receiving an acknowledgment (ACK) for the first frame.

Key Concepts:

1. **Sliding Window**:
   - The sender maintains a window of frames that can be sent without waiting for an ACK.
   - The window size is determined by the sequence number range and the buffer capacity.
2. **Sequence Numbers**:
   - Each frame is assigned a unique sequence number.
   - The sequence numbers are used to identify frames and ensure they are processed in the correct order.
3. **Acknowledgments (ACKs)**:
   - The receiver sends an ACK for the last correctly received in-order frame.
   - If a frame is received out of order, the receiver discards it and does not send an ACK.
4. **Timeout and Retransmission**:
   - If the sender does not receive an ACK within a specified timeout period, it retransmits all frames starting from the unacknowledged frame.

How Go-Back-N ARQ Works:

1. **Sender Side**:
   - The sender transmits frames within the window size.
   - It waits for an ACK for the oldest unacknowledged frame.
   - If an ACK is received, the window slides forward, allowing new frames to be sent.
   - If a timeout occurs, the sender retransmits all frames starting from the unacknowledged frame.
2. **Receiver Side**:
   - The receiver accepts frames in order and sends an ACK for the last correctly received in-order frame.
   - If a frame is received out of order, it is discarded, and no ACK is sent for it.
   - The receiver continues to wait for the missing frame.

- In the **Go-Back-N ARQ** protocol, **N** refers to the **window size**, which is the maximum number of frames (or packets) the sender can transmit without receiving an acknowledgment (ACK) from the receiver.
- **Key Points About N:**

1. **Window Size**:
   - N determines how many frames the sender can send before it must stop and wait for an ACK.
   - It is also called the **sender's window size** or the **sliding window size**.
2. **Sequence Numbers**:
   - The sequence numbers of the frames range from **0 to N-1** (if N is the window size).
   - For example, if N = 4, the sequence numbers are 0, 1, 2, and 3. After 3, the sequence numbers wrap around to 0.
3. **Sliding Window Mechanism**:
   - The sender can send up to N frames without waiting for an ACK.
   - Once an ACK is received for the oldest unacknowledged frame, the window "slides" forward, allowing the sender to transmit new frames.
4. **Relationship to Buffering**:
   - The sender must buffer all unacknowledged frames in case they need to be retransmitted.
   - The receiver, however, does not need to buffer out-of-order frames in Go-Back-N ARQ; it simply discards them.

Limitations:

- If N is too large, the sender may need to retransmit many frames unnecessarily in case of an error.
- If N is too small, the sender may not fully utilize the available bandwidth, leading to lower throughput.

Advantages:

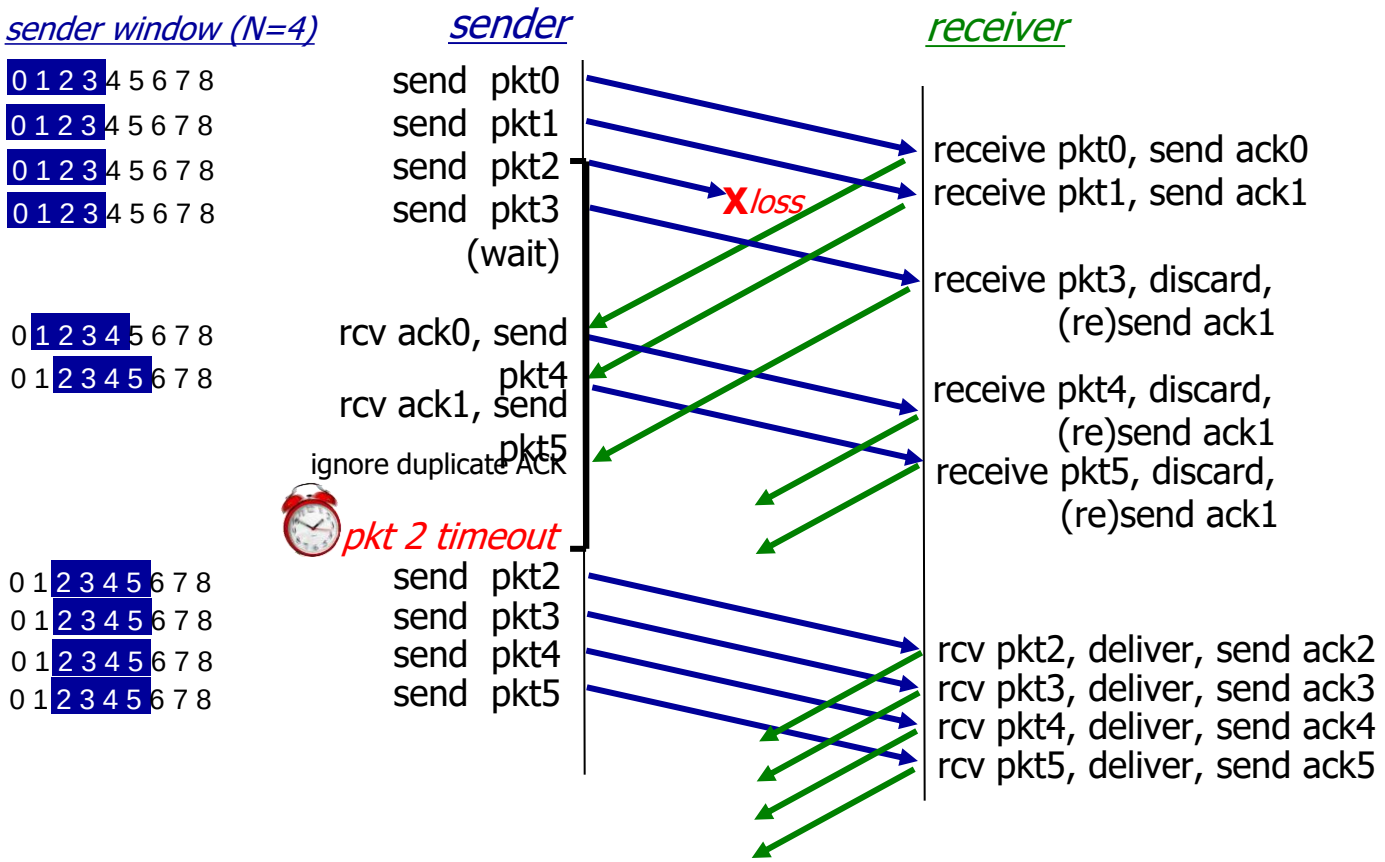
- **Efficiency**: Allows multiple frames to be in transit, improving throughput.
- **Simplicity**: Easier to implement compared to Selective Repeat ARQ.

Disadvantages:

- **Inefficiency**: If a single frame is lost, all subsequent frames in the window are retransmitted, even if they were received correctly.
- **Bandwidth Usage**: Can lead to unnecessary retransmissions, wasting bandwidth.

Go-Back-N ARQ is suitable for environments with low error rates and where simplicity is more critical than efficiency.

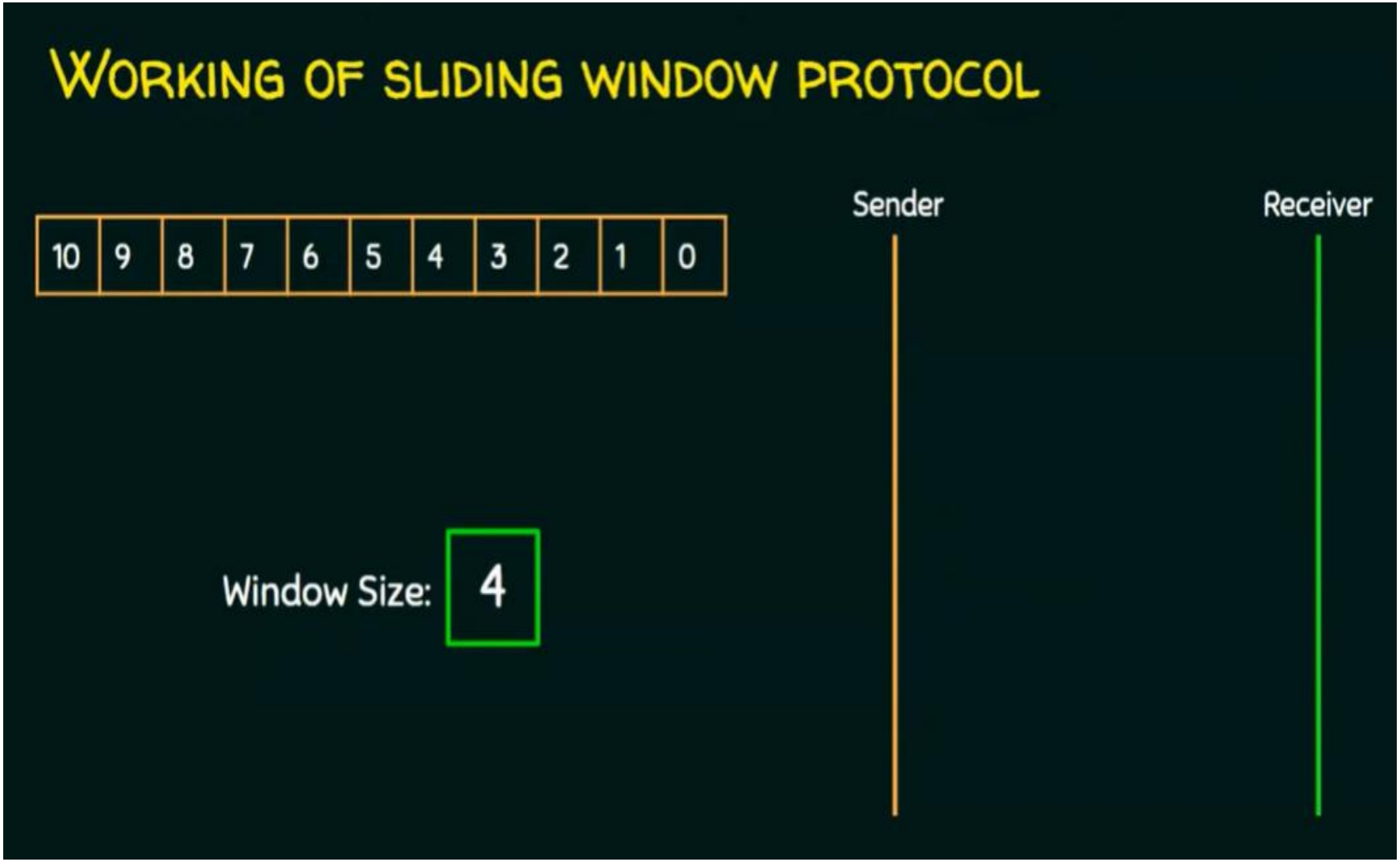
Go-Back-N in action

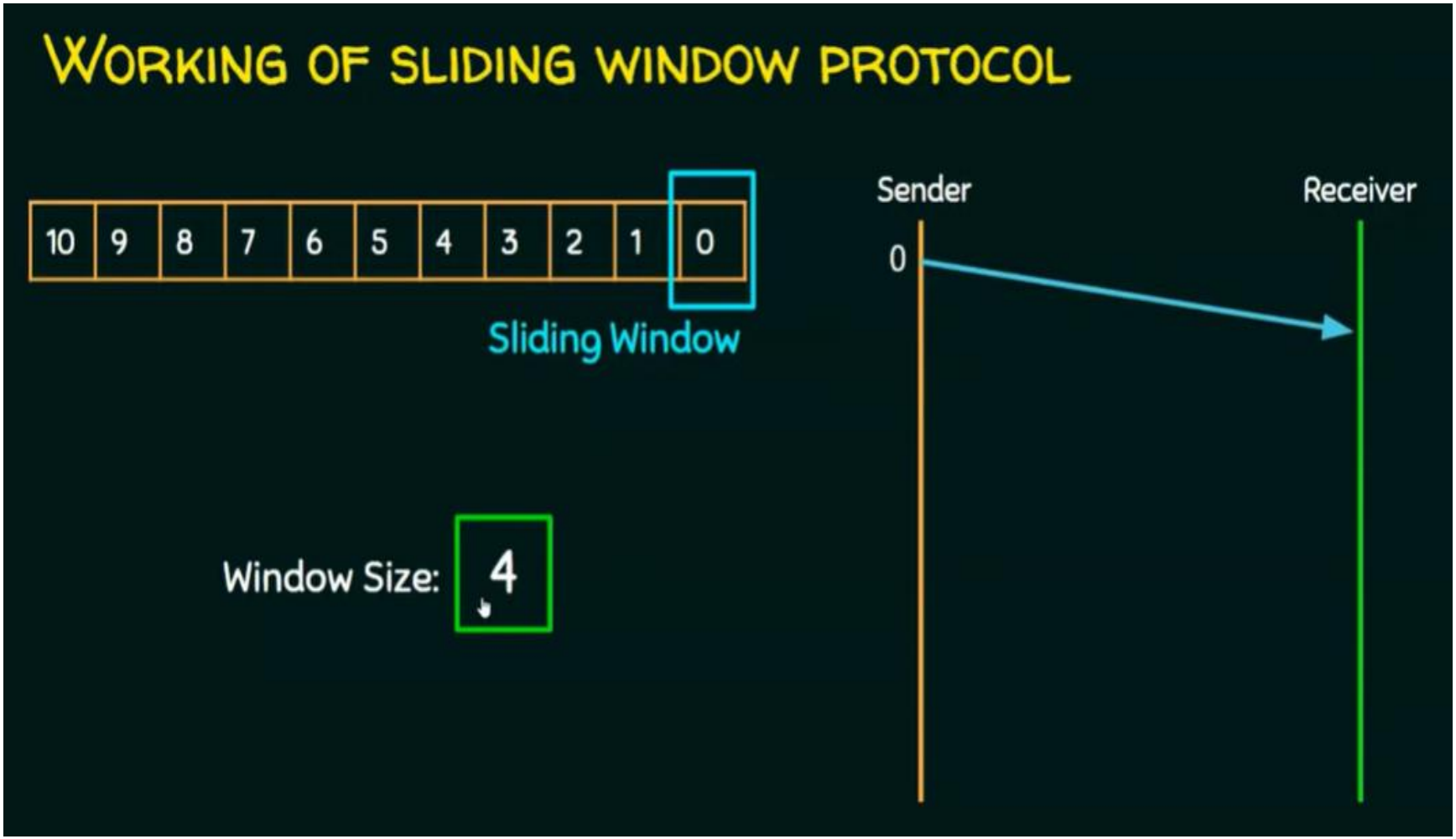


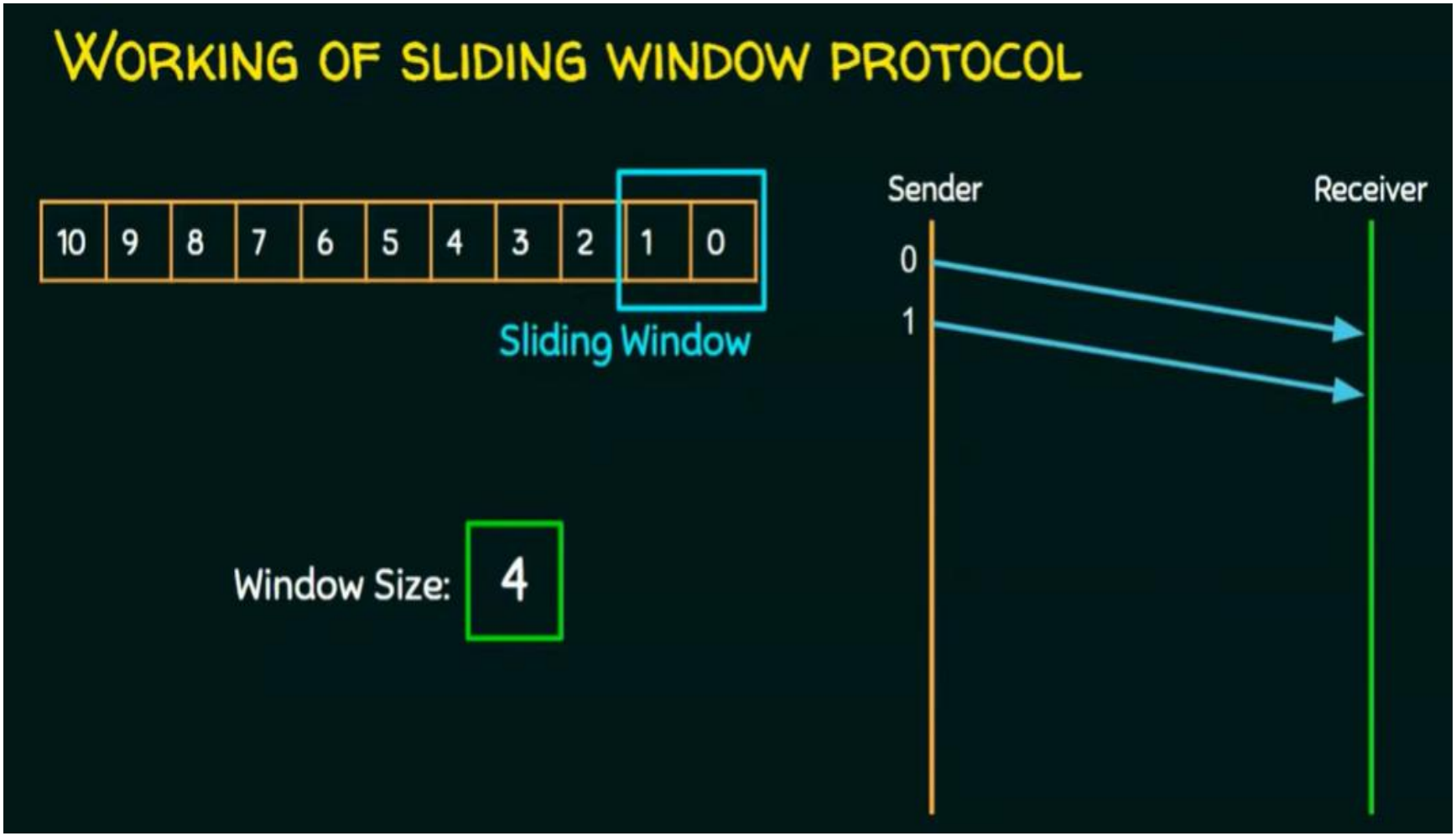
WORKING OF SLIDING WINDOW PROTOCOL

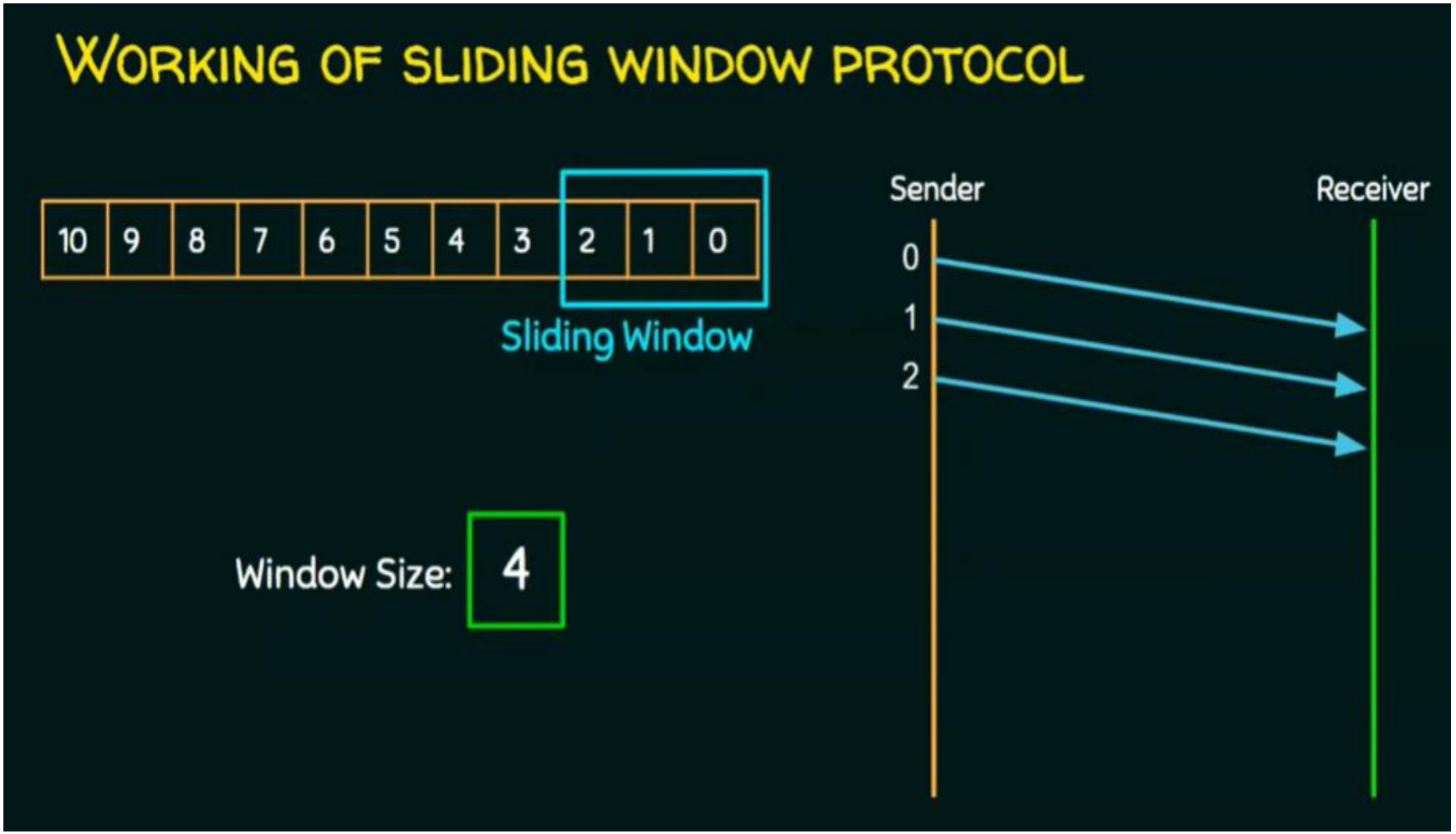
Sender

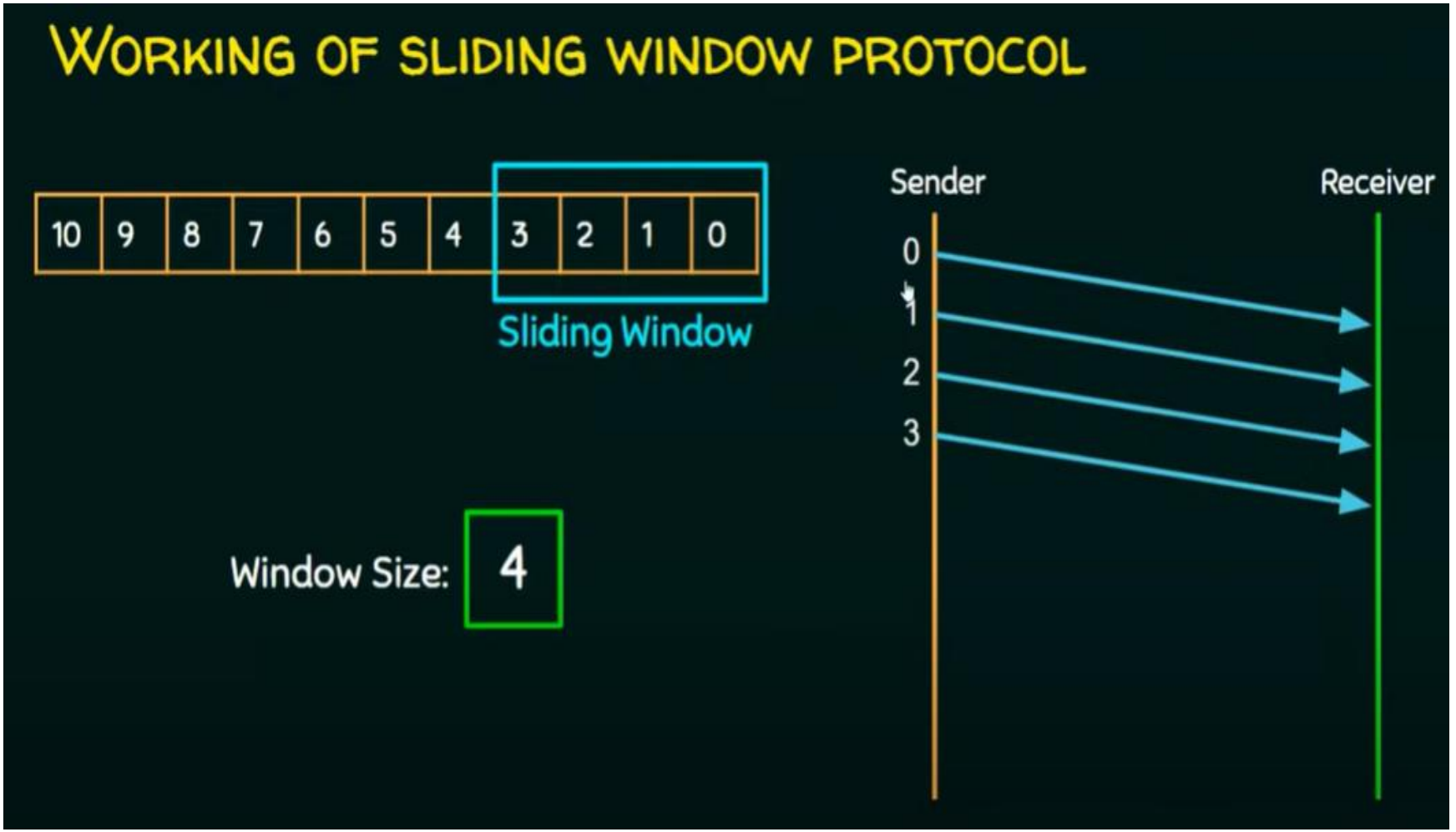
Receiver


WORKING OF SLIDING WINDOW PROTOCOL
10
9
8
7
6
5
4
3
2
1
0
Window Size:
4
Sender
Receiver


WORKING OF SLIDING WINDOW PROTOCOL
10 9 8 7 6 5 4 3 2 1 0
Sliding Window
Window Size: 4
Sender
Receiver
0

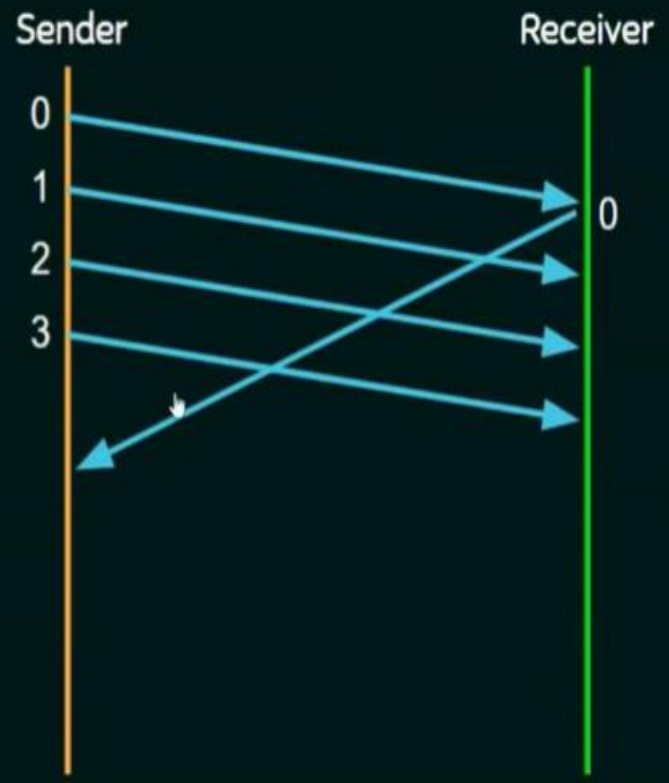

WORKING OF SLIDING WINDOW PROTOCOL
10
9
8
7
6
5
4
3
2
1
0
Sliding Window
Window Size:
4
Sender
Receiver
0
1


WORKING OF SLIDING WINDOW PROTOCOL
10
9
8
7
6
5
4
3
2
1
0
Sliding Window
Window Size:
4
Sender
Receiver
0
1
2


WORKING OF SLIDING WINDOW PROTOCOL
10 9 8 7 6 5 4 3 2 1 0
Sliding Window
Window Size: 4
Sender
Receiver
0
1
2
3

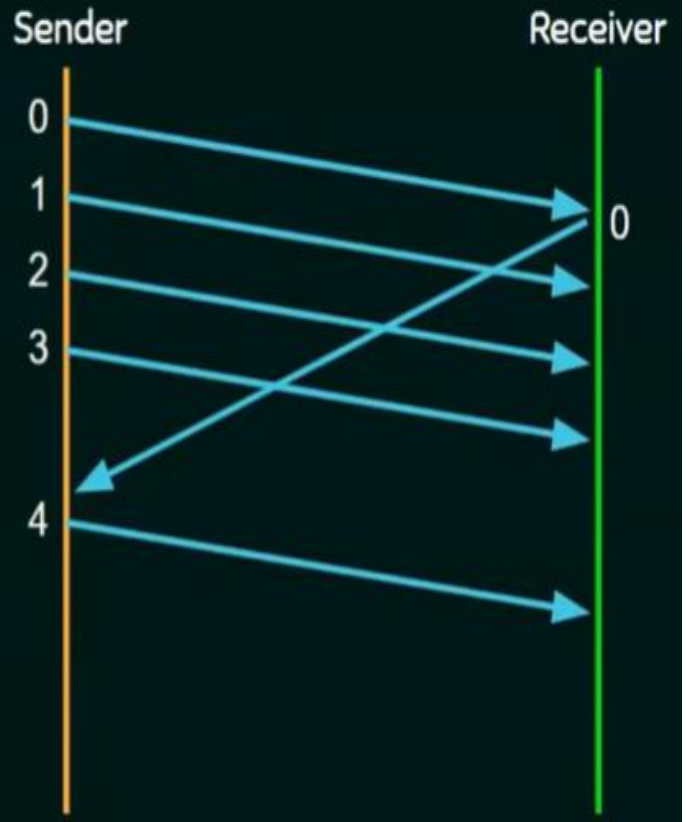
Working of Sliding Window Protocol

Window Size: 4



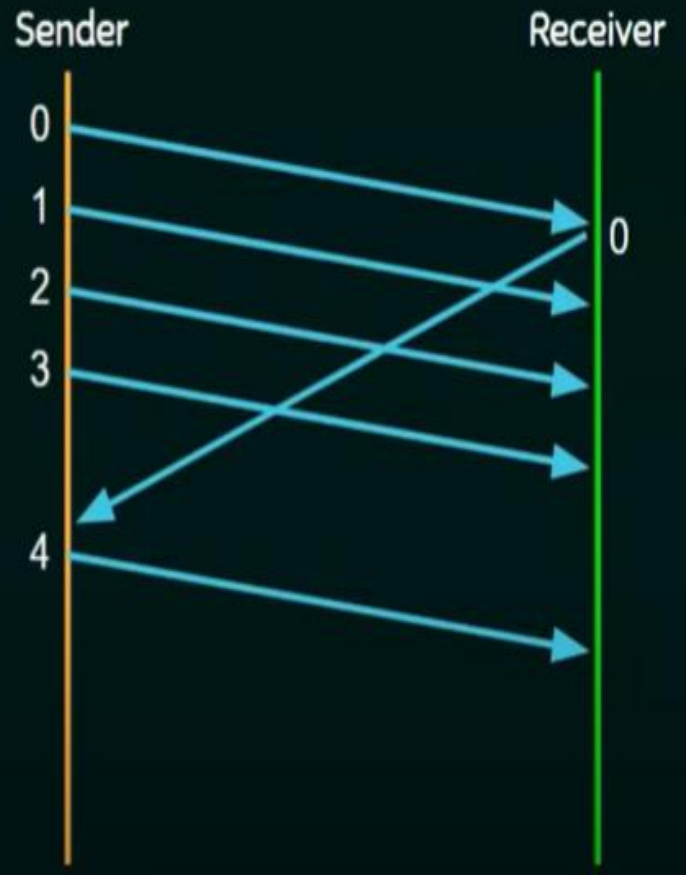
Working of Sliding Window Protocol

Window Size: 4



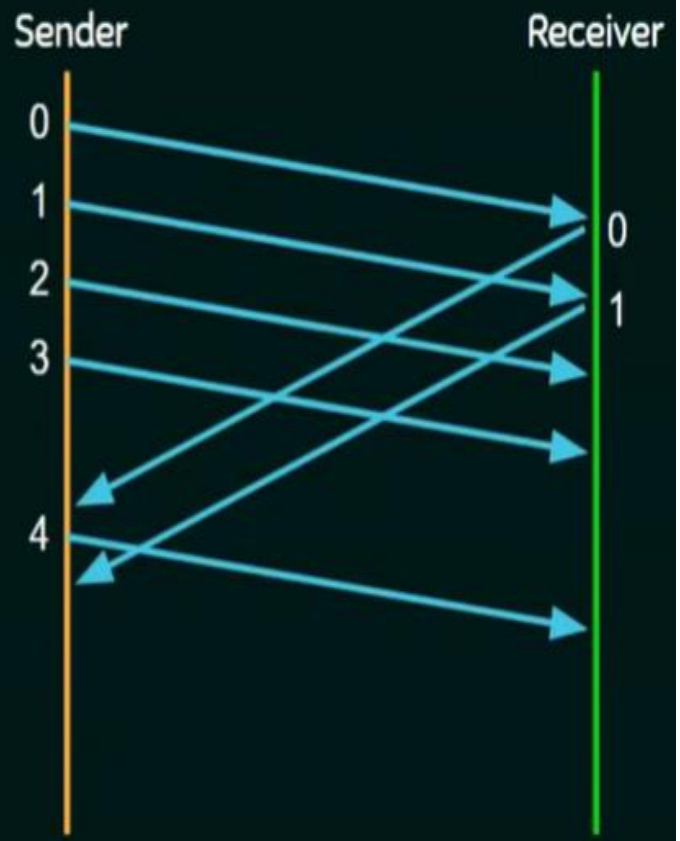
Working of Sliding Window Protocol

Window Size: 4



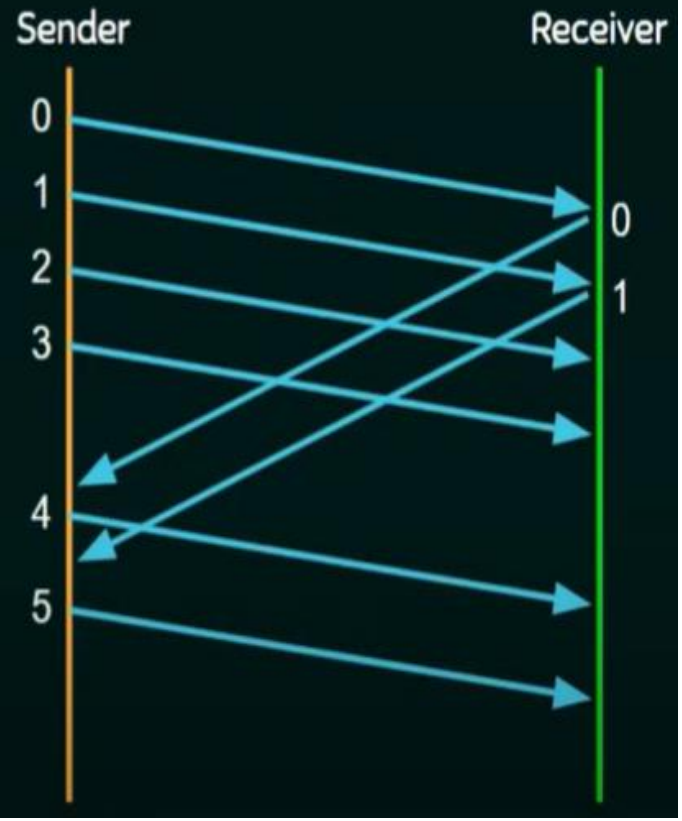
WORKING OF SLIDING WINDOW PROTOCOL

Window Size: 4



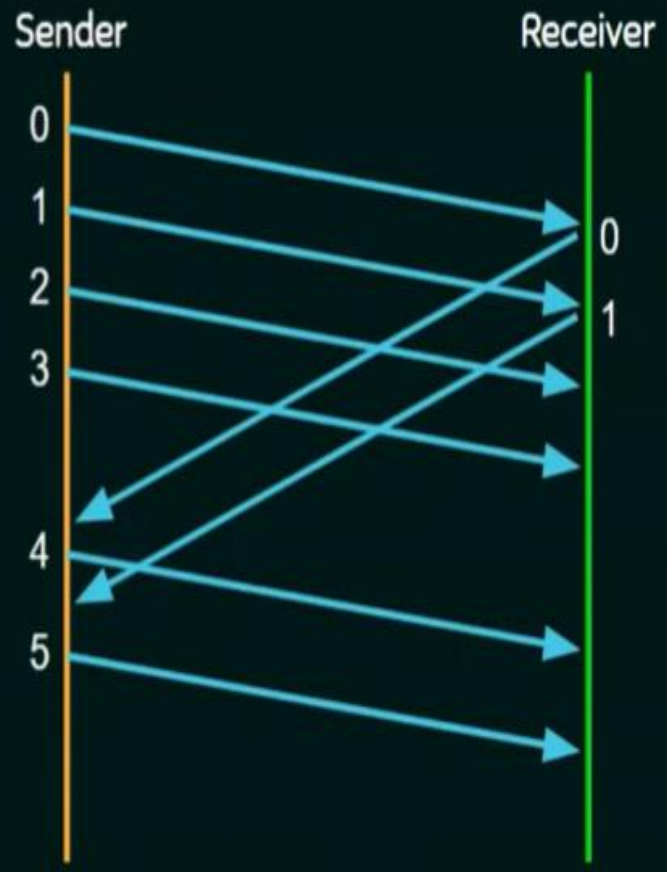
Working of Sliding Window Protocol

Window Size: 4



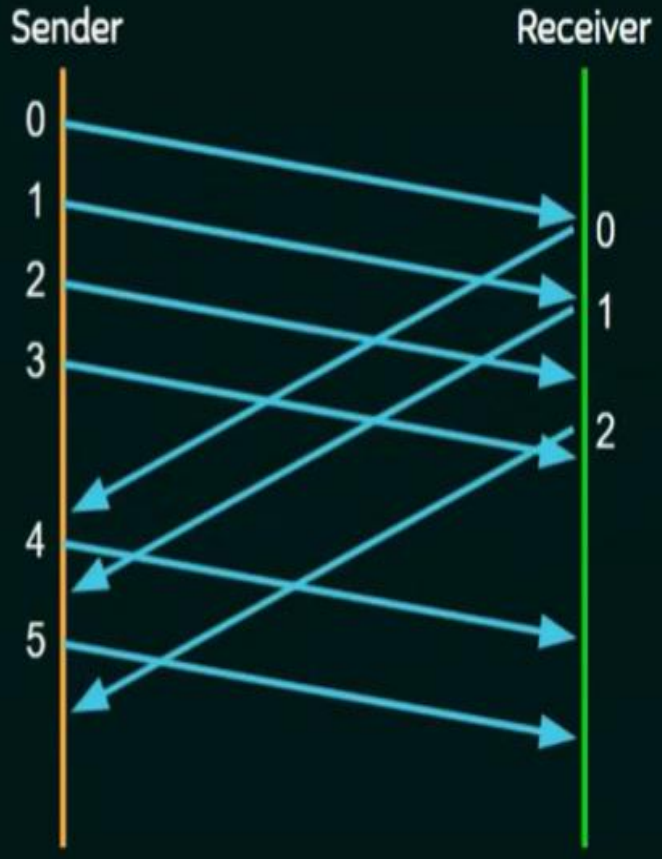
WORKING OF SLIDING WINDOW PROTOCOL

Window Size: 4



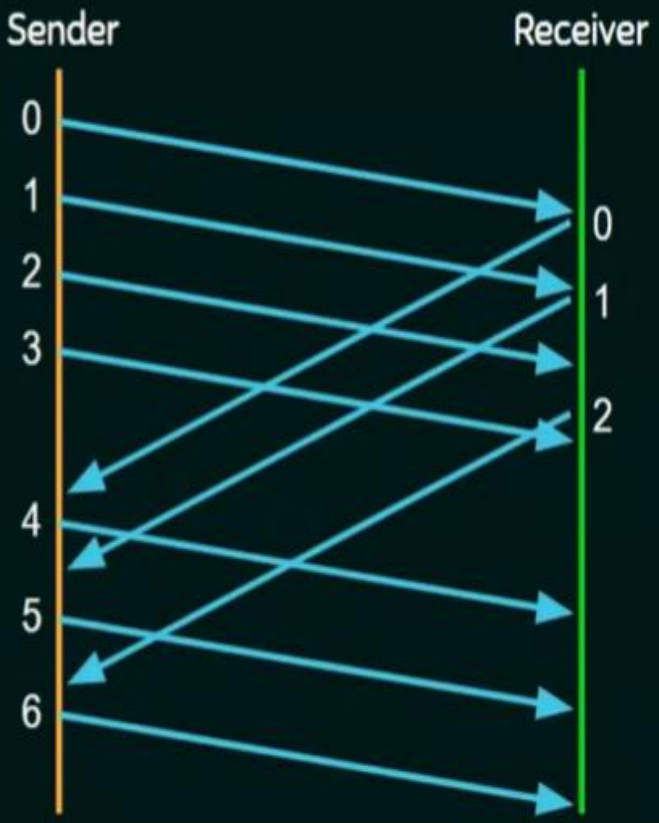
Working of Sliding Window Protocol

Window Size: 4

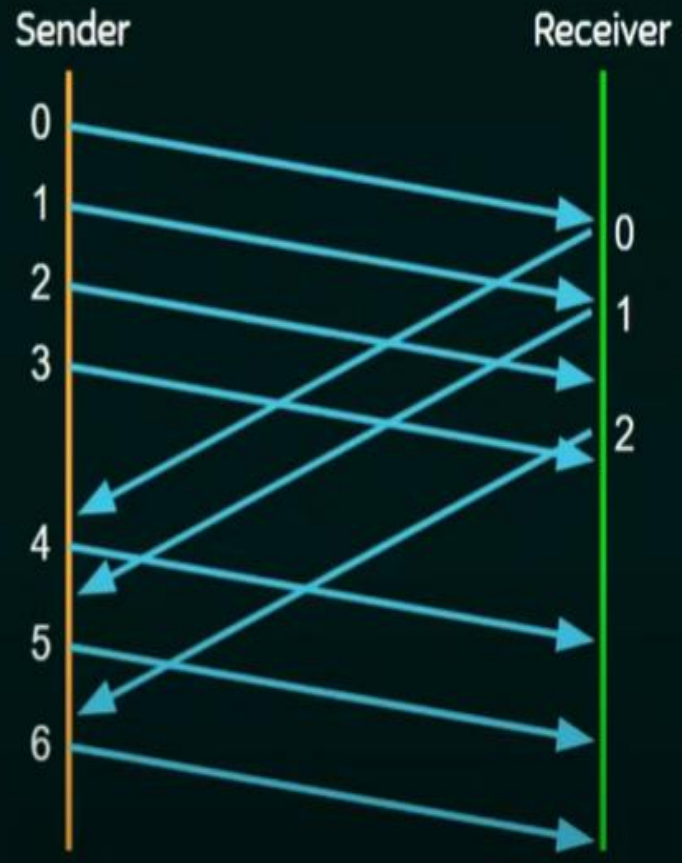


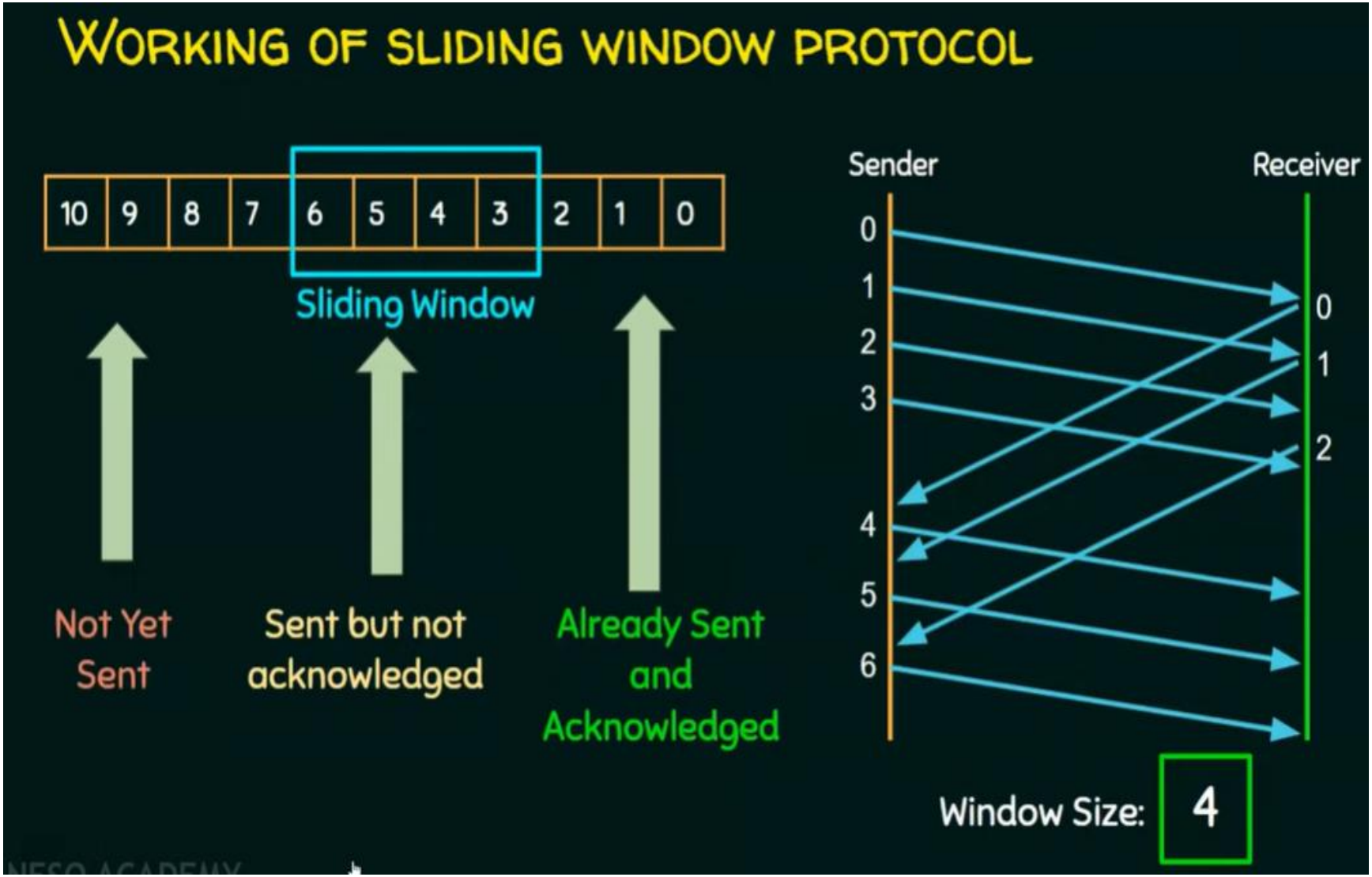
WORKING OF SLIDING WINDOW PROTOCOL

Window Size: 4



WORKING OF SLIDING WINDOW PROTOCOL




WORKING OF SLIDING WINDOW PROTOCOL
10 9 8 7 6 5 4 3 2 1 0
Sliding Window
Not Yet Sent
Sent but not acknowledged
Already Sent and Acknowledged
Sender
Receiver
0
1
2
3
4
5
6
0
1
2
Window Size: 4

Go-Back-N ARQ

- ★ Go - Back - N ARQ uses the concept of protocol pipelining i.e. the sender can send multiple frames before receiving the acknowledgment for the first frame.
- ★ There are finite number of frames and the frames are numbered in a sequential manner.
- ★ The number of frames that can be sent depends on the window size of the sender.
- ★ If the acknowledgment of a frame is not received within an agreed upon time period, all frames in the current window are transmitted.

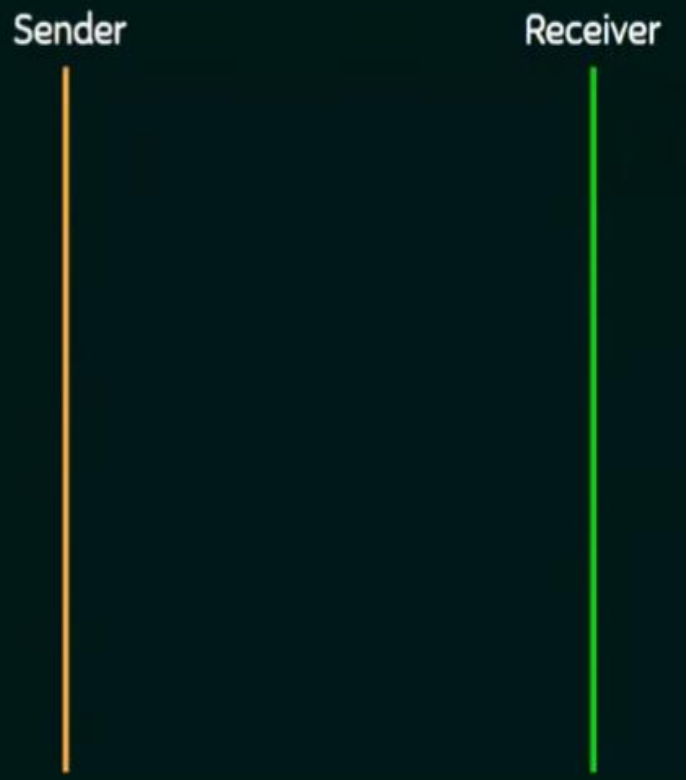
Go-Back-N ARQ

★ The size of the sending window determines the sequence number of the outbound frames.

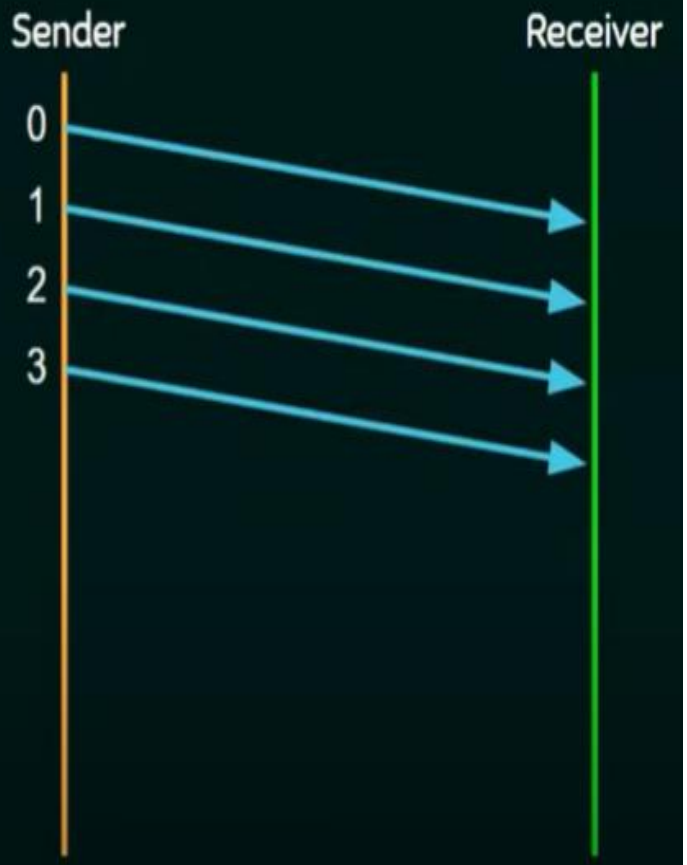
Go-Back-N ARQ

- ★ N - Sender's Window Size.
- ★ For example, if the sending window size is 4 (2^2), then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on.

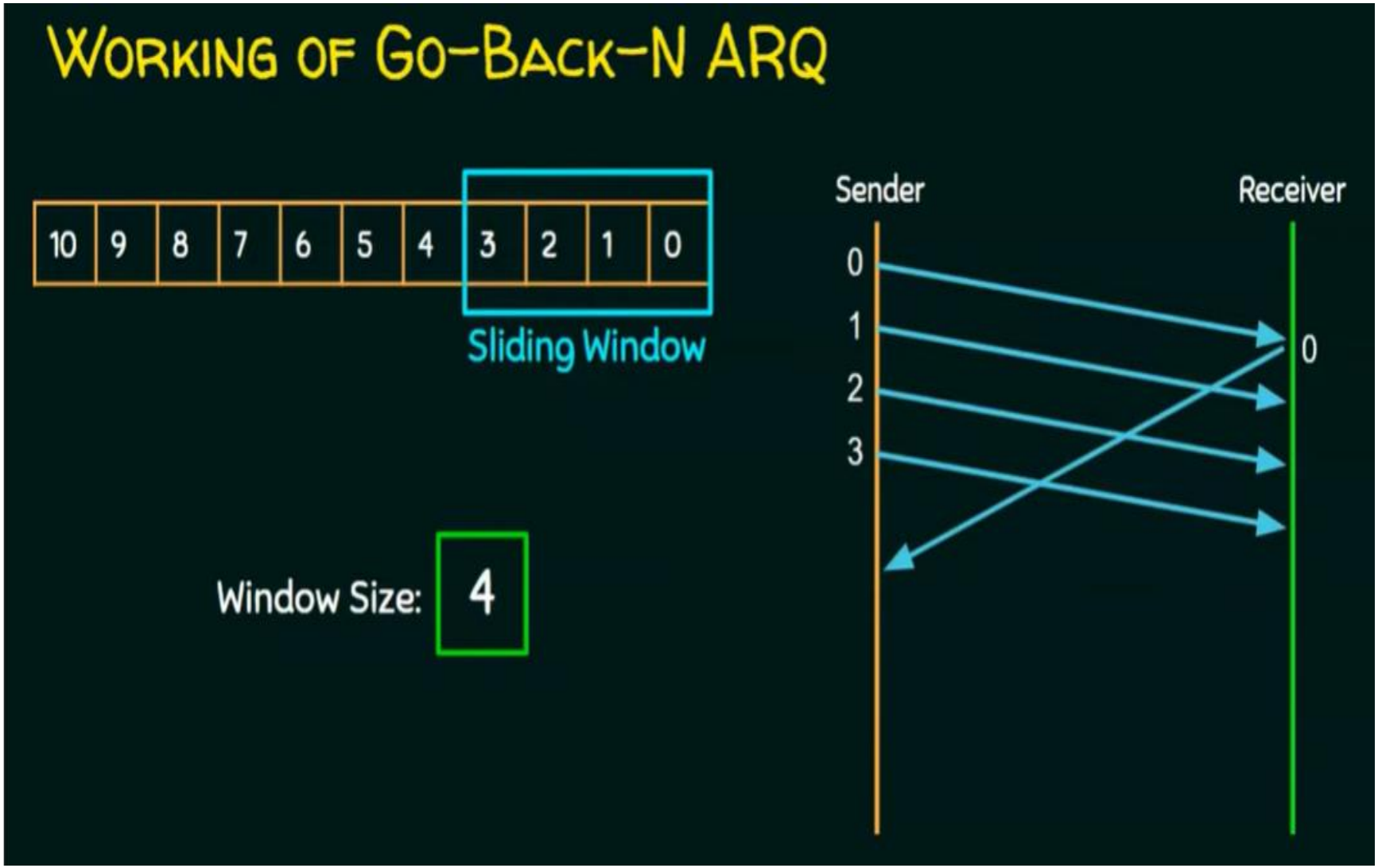
Working of Go-Back-N ARQ



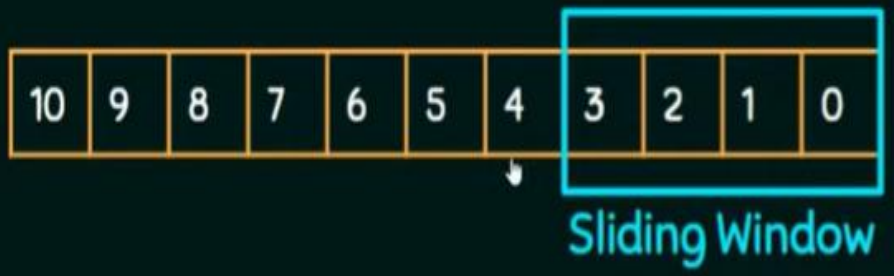
Working of Go-Back-N ARQ



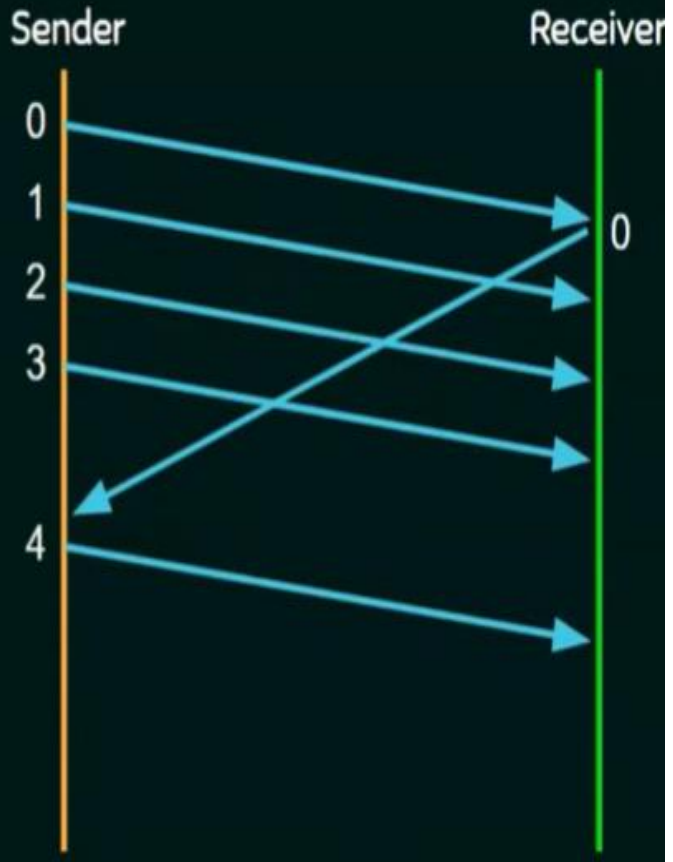
Window Size: 4


WORKING OF GO-BACK-N ARQ
10 9 8 7 6 5 4 3 2 1 0
Sliding Window
Window Size: 4
Sender
Receiver
0
1
2
3
0

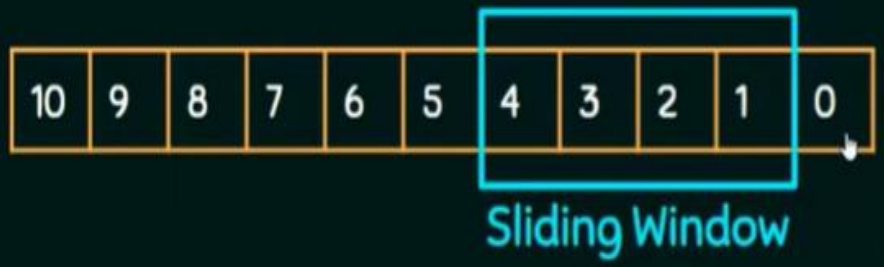
Working of Go-Back-N ARQ



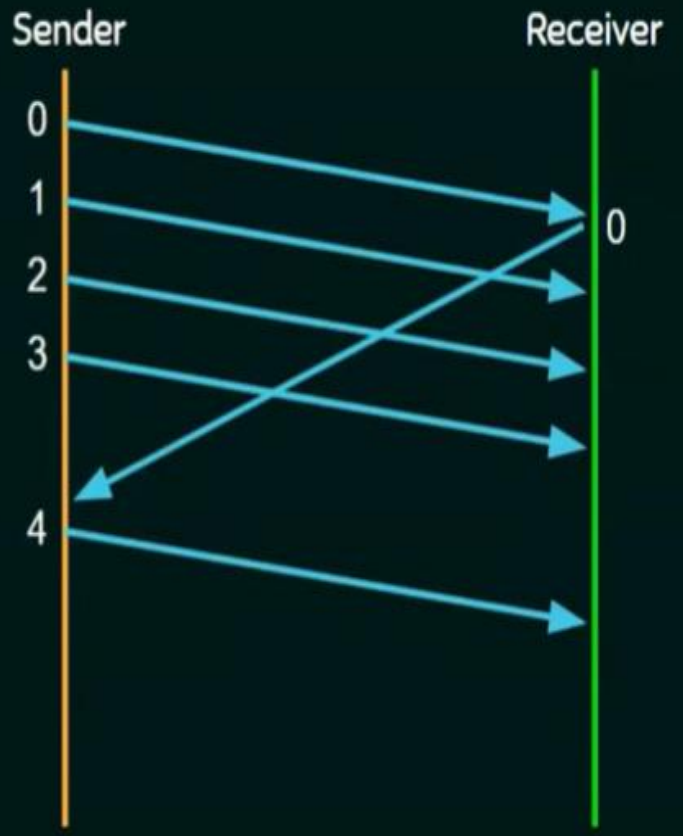
Window Size: 4



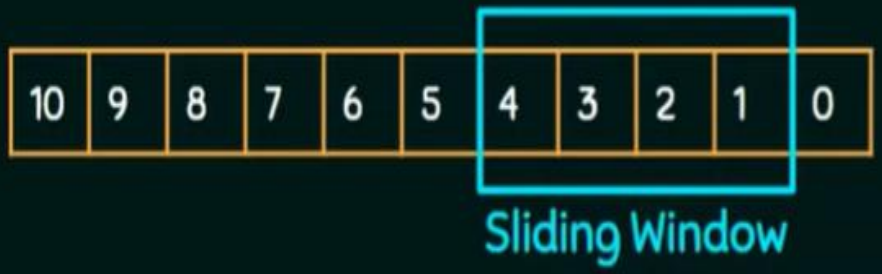
Working of Go-Back-N ARQ



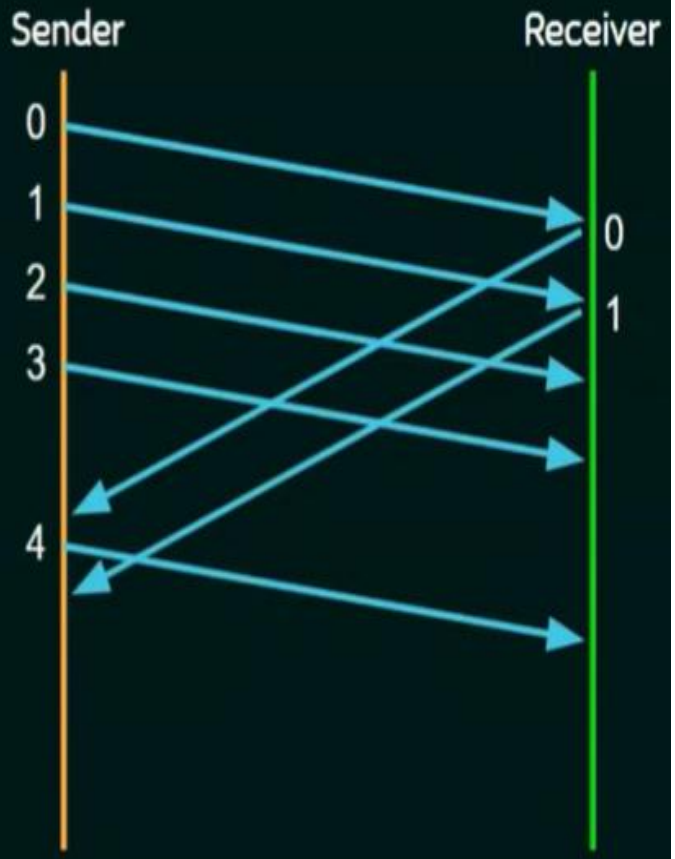
Window Size: 4



Working of Go-Back-N ARQ

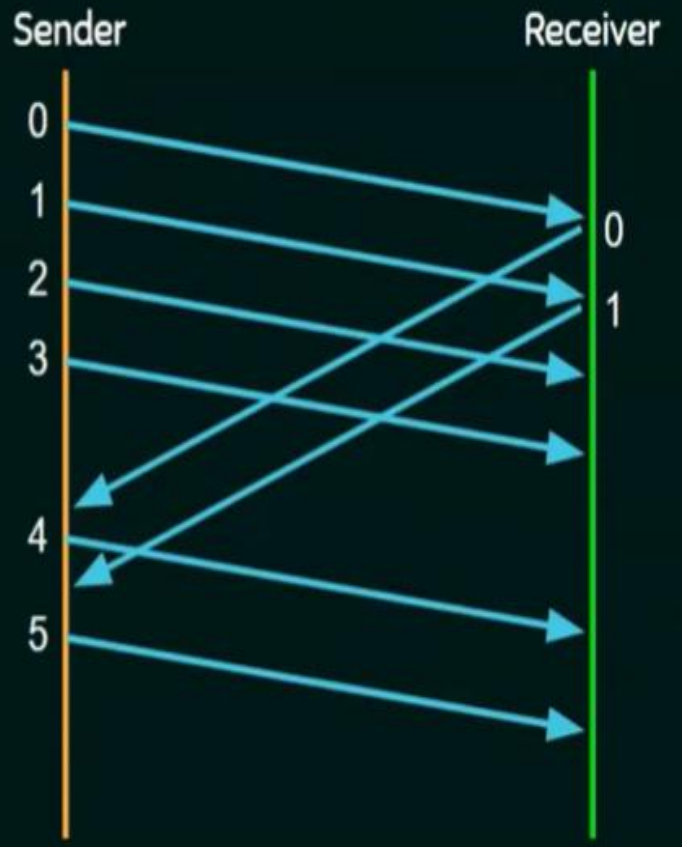


Window Size: 4

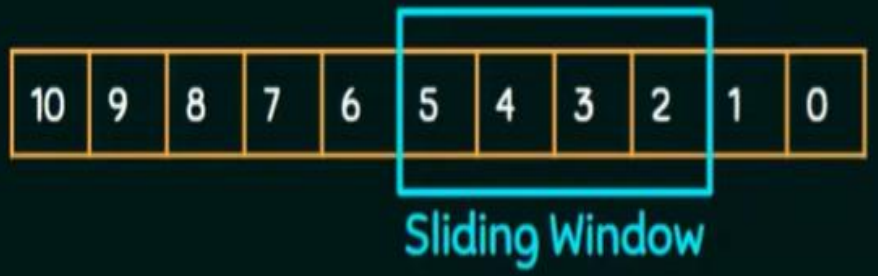


Working of Go-Back-N ARQ

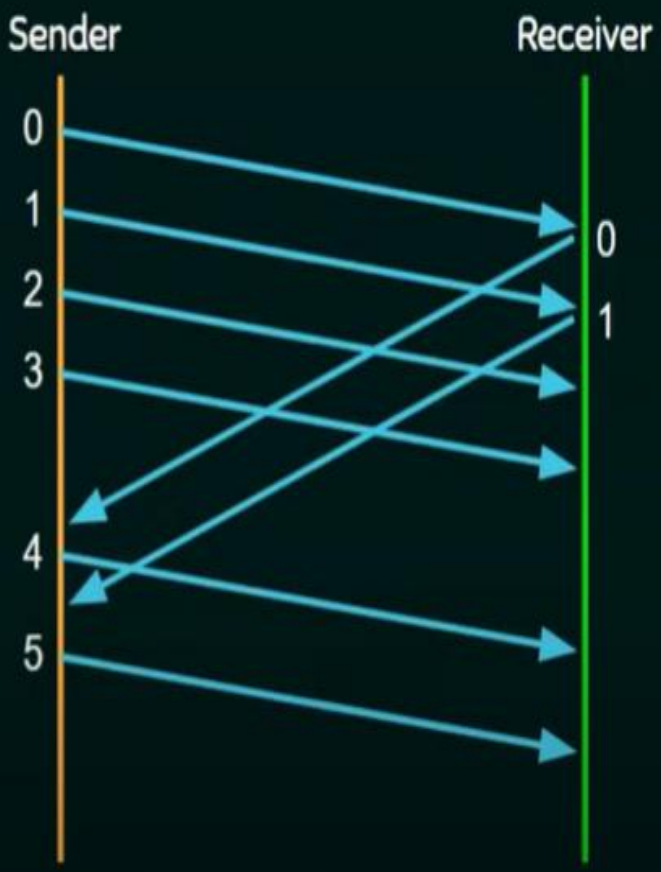
Window Size: 4



Working of Go-Back-N ARQ

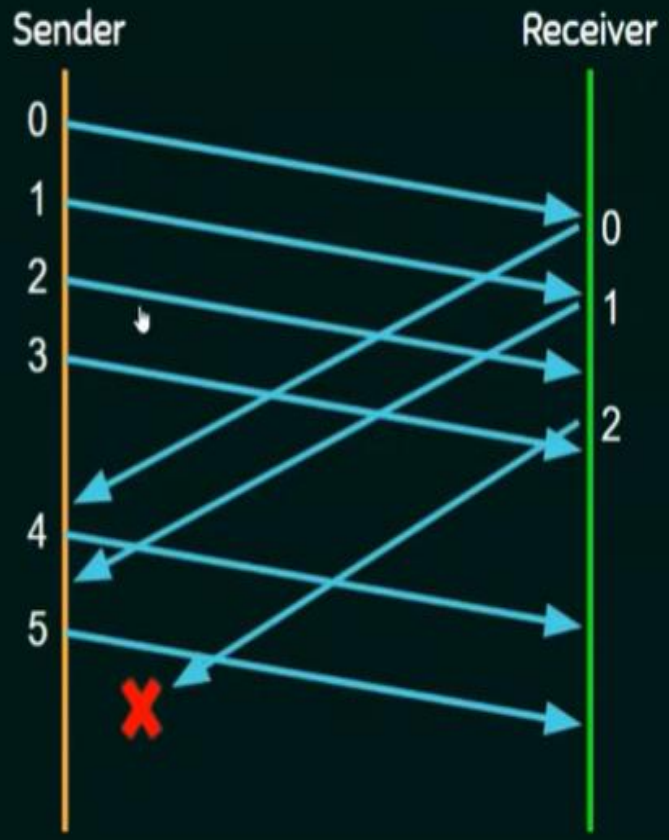


Window Size: 4

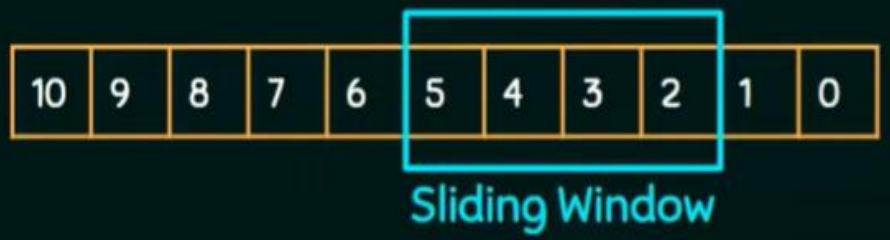


Working of Go-Back-N ARQ

Window Size: 4

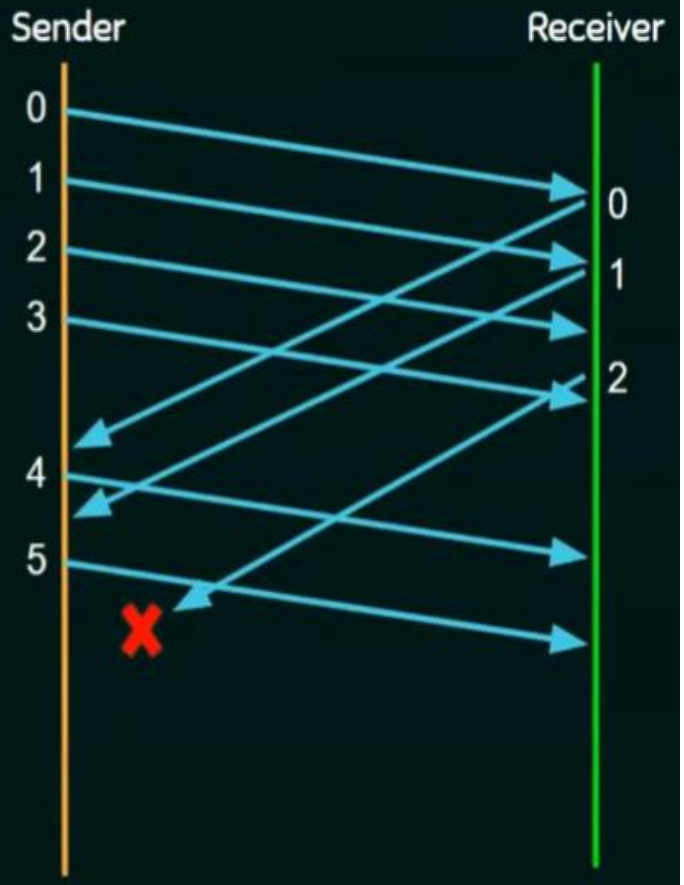


Working of Go-Back-N ARQ



Window Size: 4

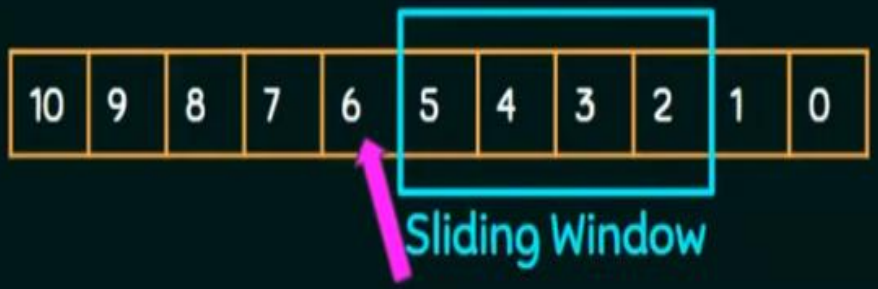
ACK not received in time. So Sender times out.



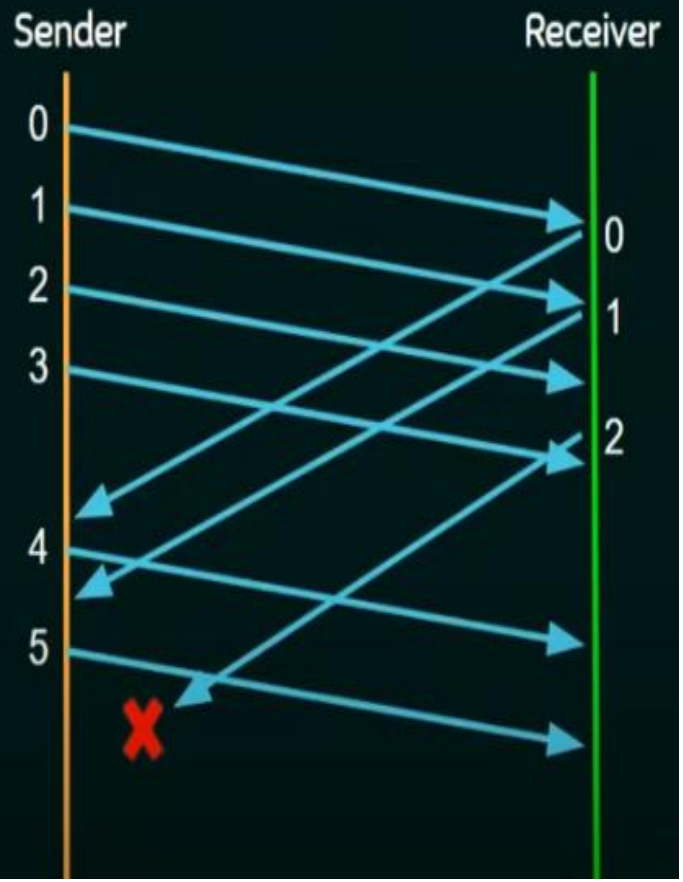
Note

- Sender side will go back to the unacknowledged frame and retransmit that frame, along with all the frames shared after that frame with the receiver.
- This represents the Go-Back-N ARQ protocol method.

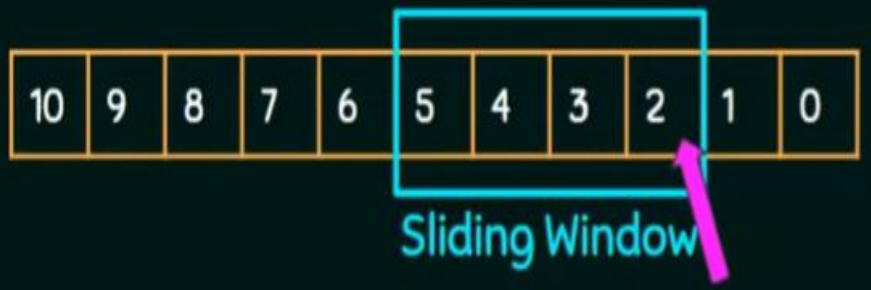
Working of Go-Back-N ARQ



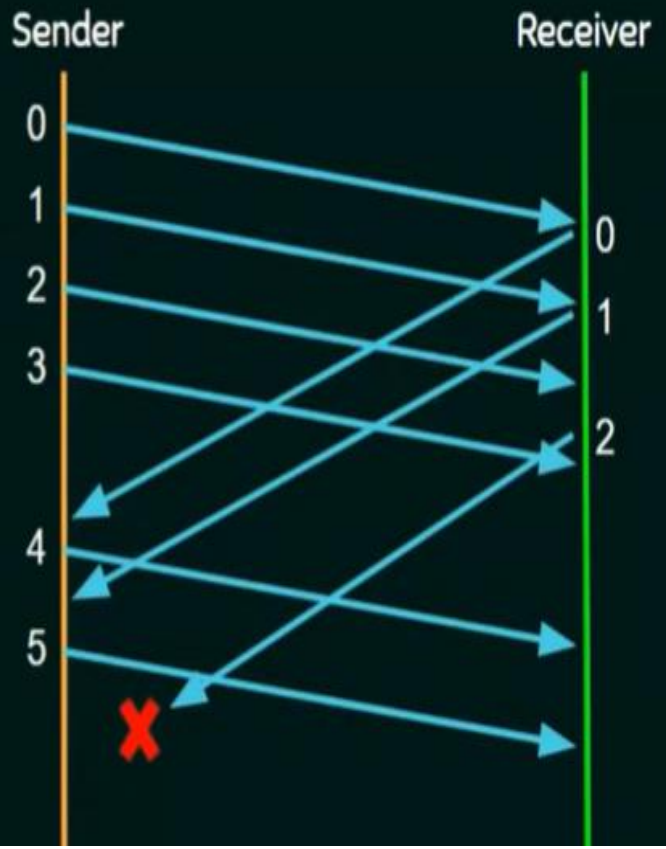
Window Size: 4



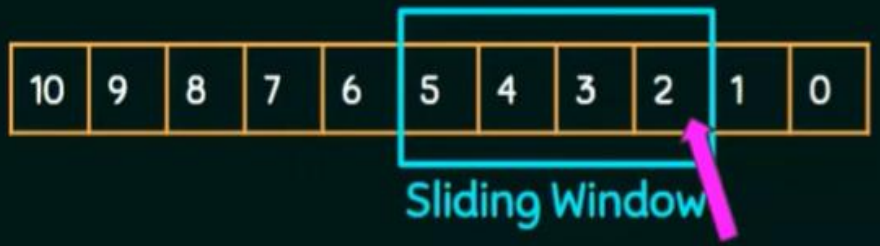
Working of Go-Back-N ARQ



Window Size: 4

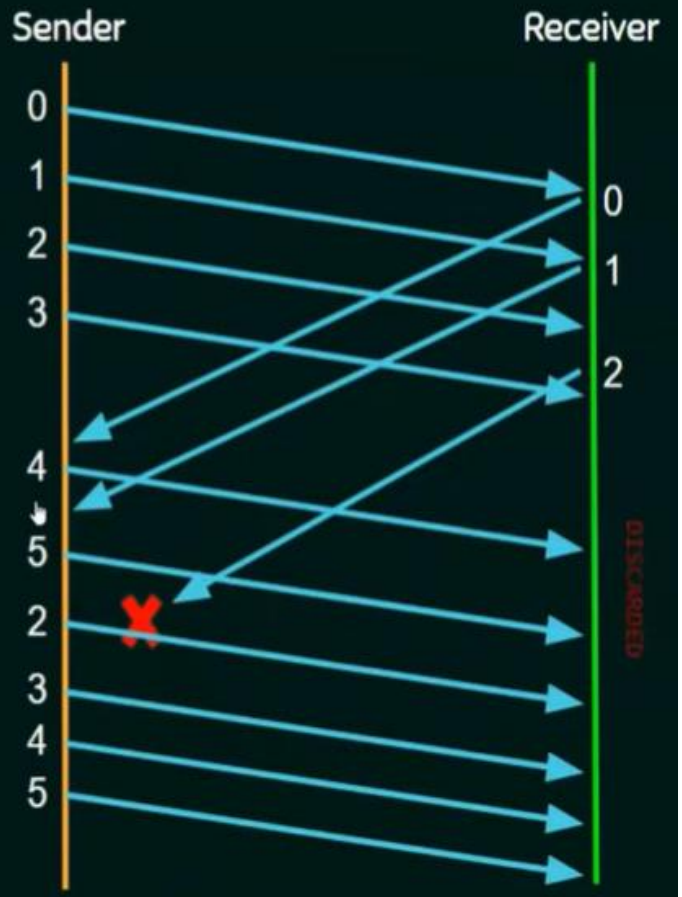


Working of Go-Back-N ARQ

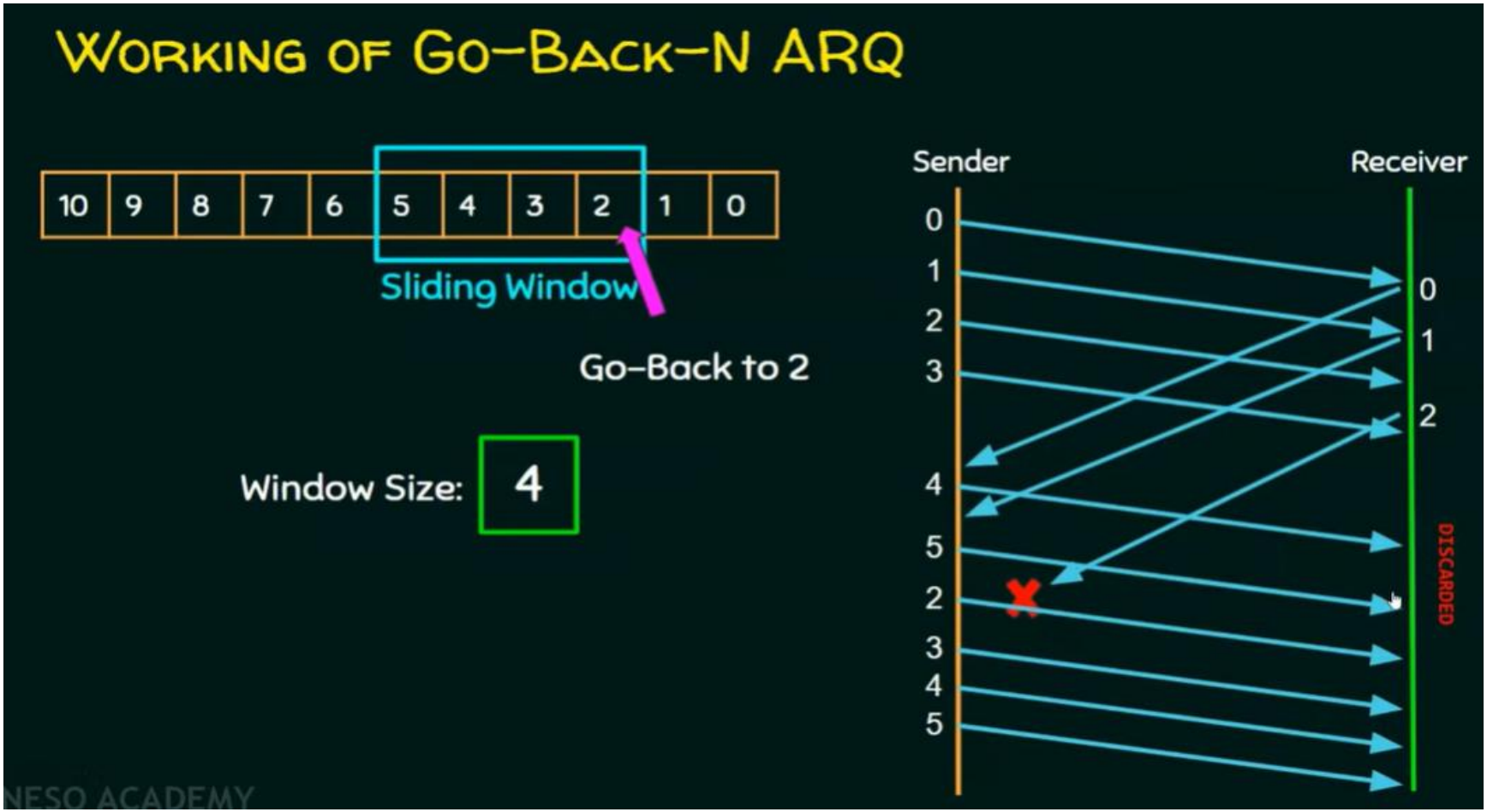


Go-Back to 2

Window Size: 4



Receiver discards all frames in receiver buffer



QUESTION

Station A needs to send a message consisting of 9 packets to station B using a sliding window (window size 3) and go-back-n error control strategy. All packets are ready and immediately available for transmission. If every 5th packet that A transmits gets lost (but no ACKs from B ever get lost), then what is the number of packets that A will transmit for sending the message to B? [GATE CS 2006]

(A) 12

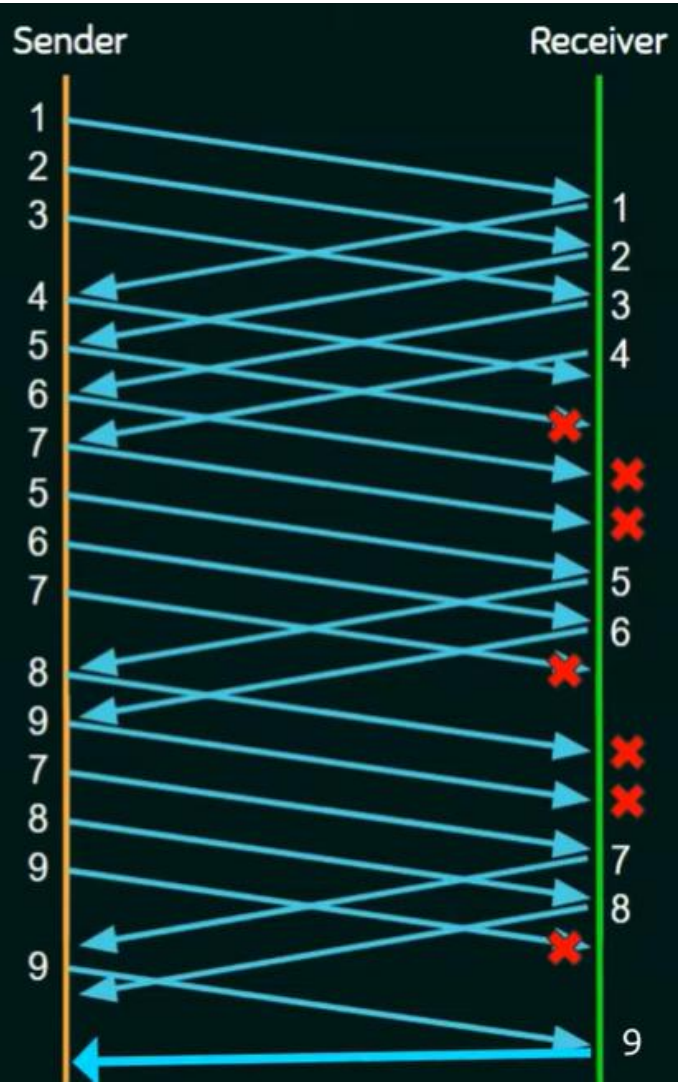
(B) 14

(C) 16

(D) 18

Solution

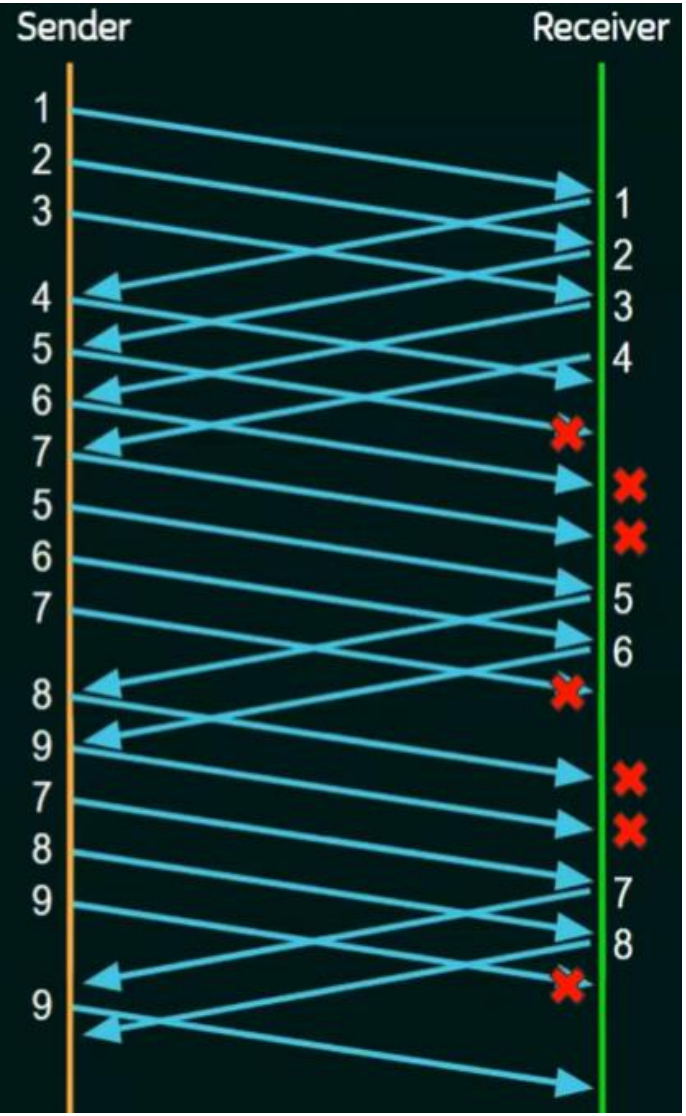
Window Size: 3



SOLUTION

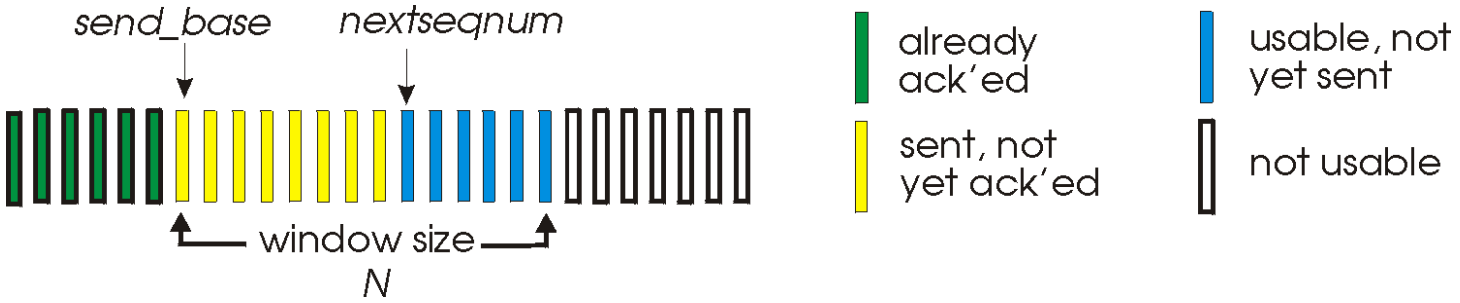
Window Size: 3

No. of packets transmitted by A (sender) 16



Go-Back-N(GBN): sender

- Window slides forward over the sequence number space
- **N – Window size**
- GBN – sliding window protocol



- **b**ase – Seq# of the oldest unacknowledged packet
- **nextseqnum – smallest unused sequence number**
- **[0, base-1]**
- **[base, nextseqnum-1]**
- **[nextseqnum, base+N-1]**
- **Seq#>= base+N**

Important things

- A packet's sequence number is carried in a fixed length field in the packet header.
- If k is the number of bits in the packet sequence number field, the range of sequence numbers is **[0 , 2^k-1]**
- With a finite range of Seq numbers, all arithmetic involving sequence numbers must be done using modulo **2^k** arithmetic
- Can be thought of as a ring of size **2^k** where seq# **2^k-1** is immediately followed by 0

Go Back N Receiver

- Suppose packet n is expected but packet n+1 arrives…What happens?

Go-Back-N: receiver

- If packet k has been received and delivered then all packets with a seq# lower than k have also been delivered.
- Use of cumulative acknowledgements is a natural choice for GBN
- Receiver discards *out-of-order* packets
- Suppose packet n is expected but packet n+1 arrives…What happens?
- If packet n is lost ,both n and n+1 will be retransmitted
- Receiver discards packet n+1
- Adv : Simplicity of buffering – Receiver need not buffer any out-of-order packets.
- Disadv : Retransmissions

Characteristics of Go-Back-N ARQ

- The size of the sender window in Go Back N ARQ is equal to **N.**
- The size of the receiver window in Go Back N ARQ is equal to **1**.
- When the acknowledgment for one frame is not received by the sender or the frames received by the receiver are out of order, then the whole window starting from the corrupted frame is retransmitted.
- Go-Back-N ARQ follows the principle of pipelining i.e. a frame can be sent by the sender before receiving the acknowledgment of the previously sent frame.

Advantages of Go-Back-N ARQ

- It can send multiple frames at once.
- Pipelining is present in the Go-Back-N ARQ i.e. a frame can be sent by the sender before receiving the acknowledgment of the previously sent frame. This results in a shorter waiting time for the frame.
- It handles corrupted as well as out-of-order frames which result in minimal frame loss.

Disadvantages of Go-Back-N ARQ

- If acknowledgment for a frame is not received, the whole window of frames is retransmitted instead of just the corrupted frame. This makes the Go Back N ARQ protocol inefficient.
- Retransmission of all the frames on detecting a corrupted frame increases channel congestion and also increases the bandwidth requirement.
- It is more time-consuming because while retransmitting the frames on detecting a corrupted frame, the error-free frames are also retransmitted.

Selective repeat(SR): the approach

- *pipelining*: *multiple* packets in flight(Window size is Large)
- In GBN – a single packet error will cause GBN to retransmit a large no. of packets ,many unnecessarily.
- As the Prob. Of channel error increases, pipeline can become filled with unnecessary retransmissions.
- **SR avoids unnecessary retransmissions by having the sender retransmit only those packets in error.**
- Unlike GBN, the sender will have already received ACKs for some of the packets in the window.

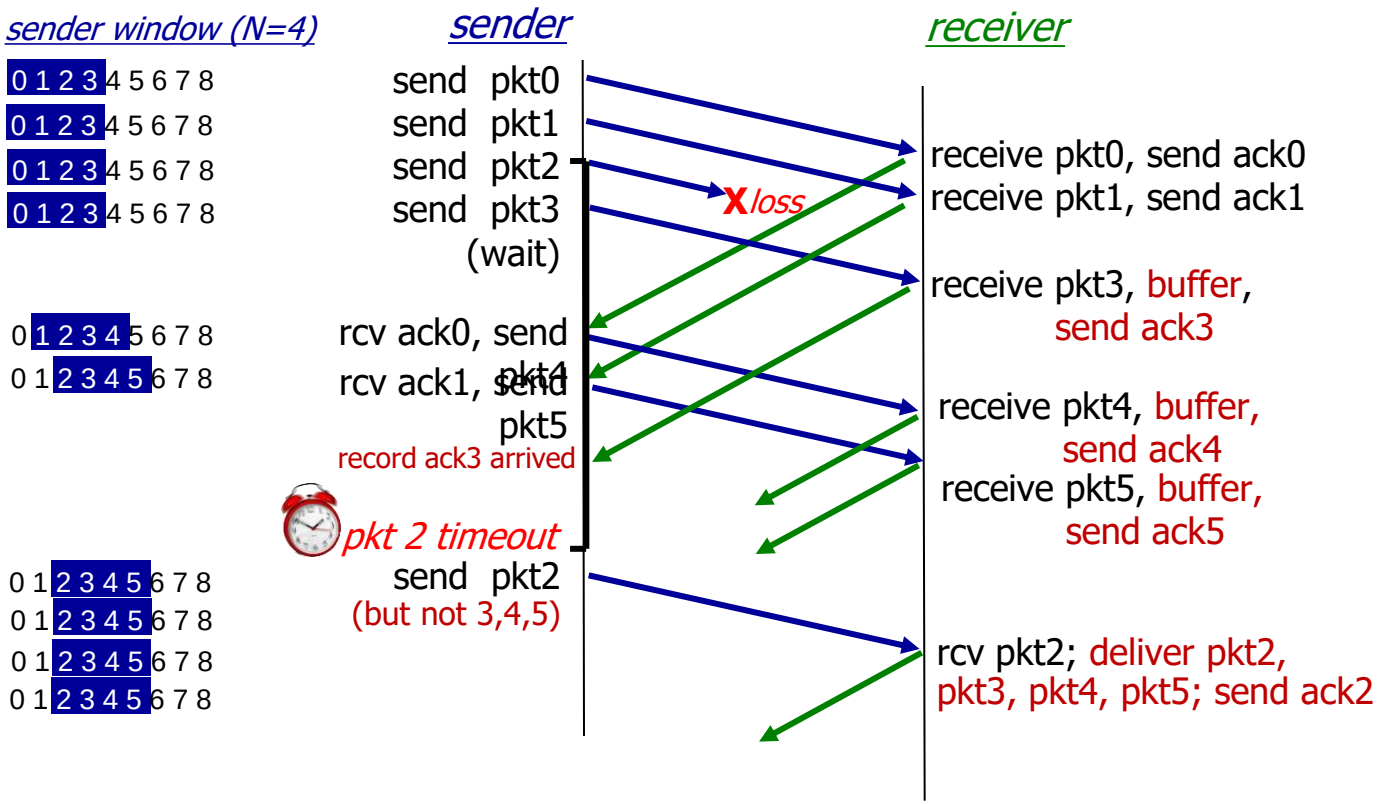
The selective repeat protocol can perform the following actions

- The receiver is capable of sorting the frame in a proper sequence, as it receives the retransmitted frame whose sequence is out of order of the receiving frame.
- The sender must be capable of searching the frame for which the NAK has been received.
- The receiver must contain the buffer to store all the previously received frame on hold till the retransmitted frame is sorted and placed in a proper sequence.

Note

- SR Receiver will acknowledge a correctly received packet whether or not it is in order.
- Out-of-order packets are buffered until any missing packets are received at which point a batch of packets can be delivered in order to the upper layer.

Selective Repeat in action



Q: what happens when ack2 arrives?

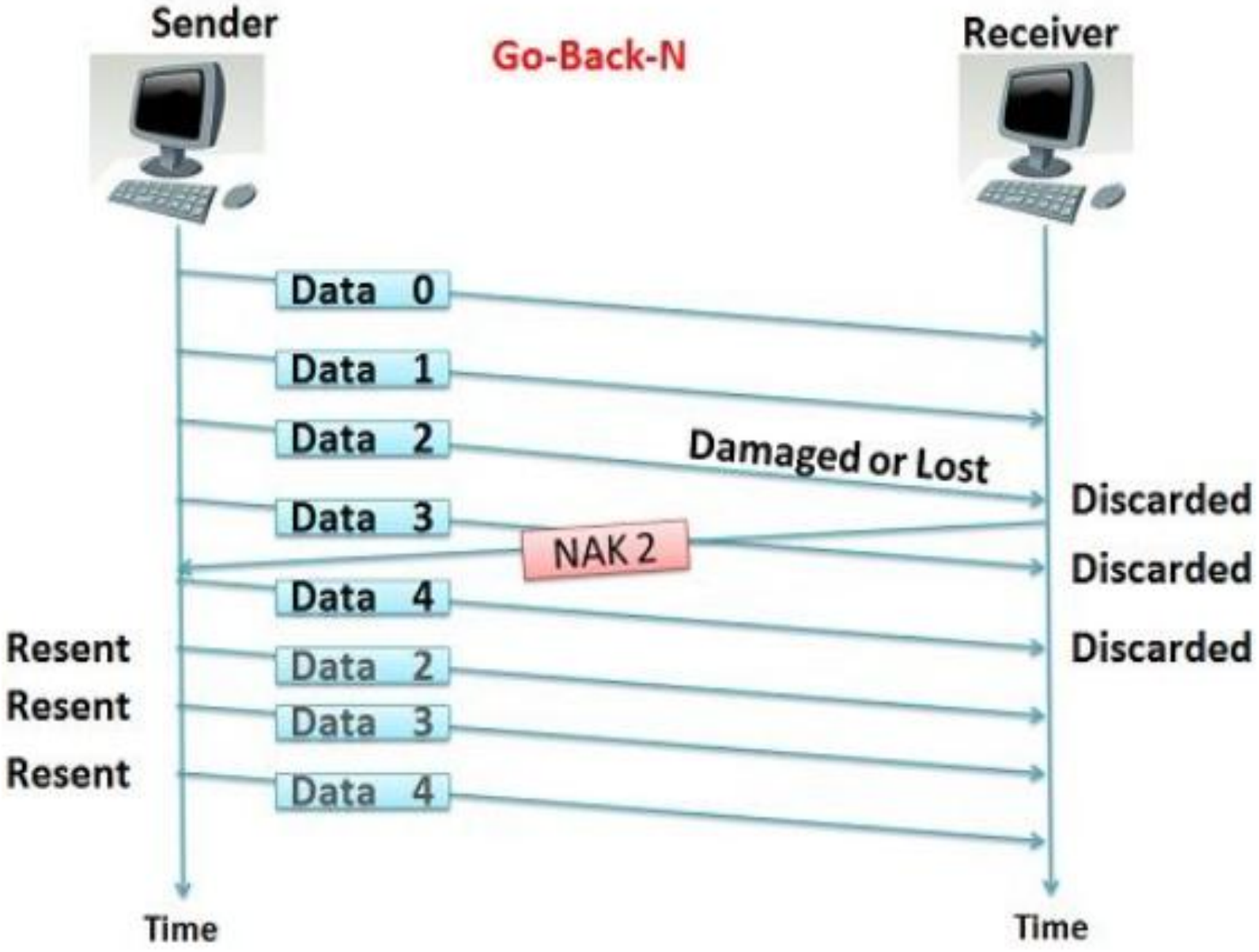
Example:

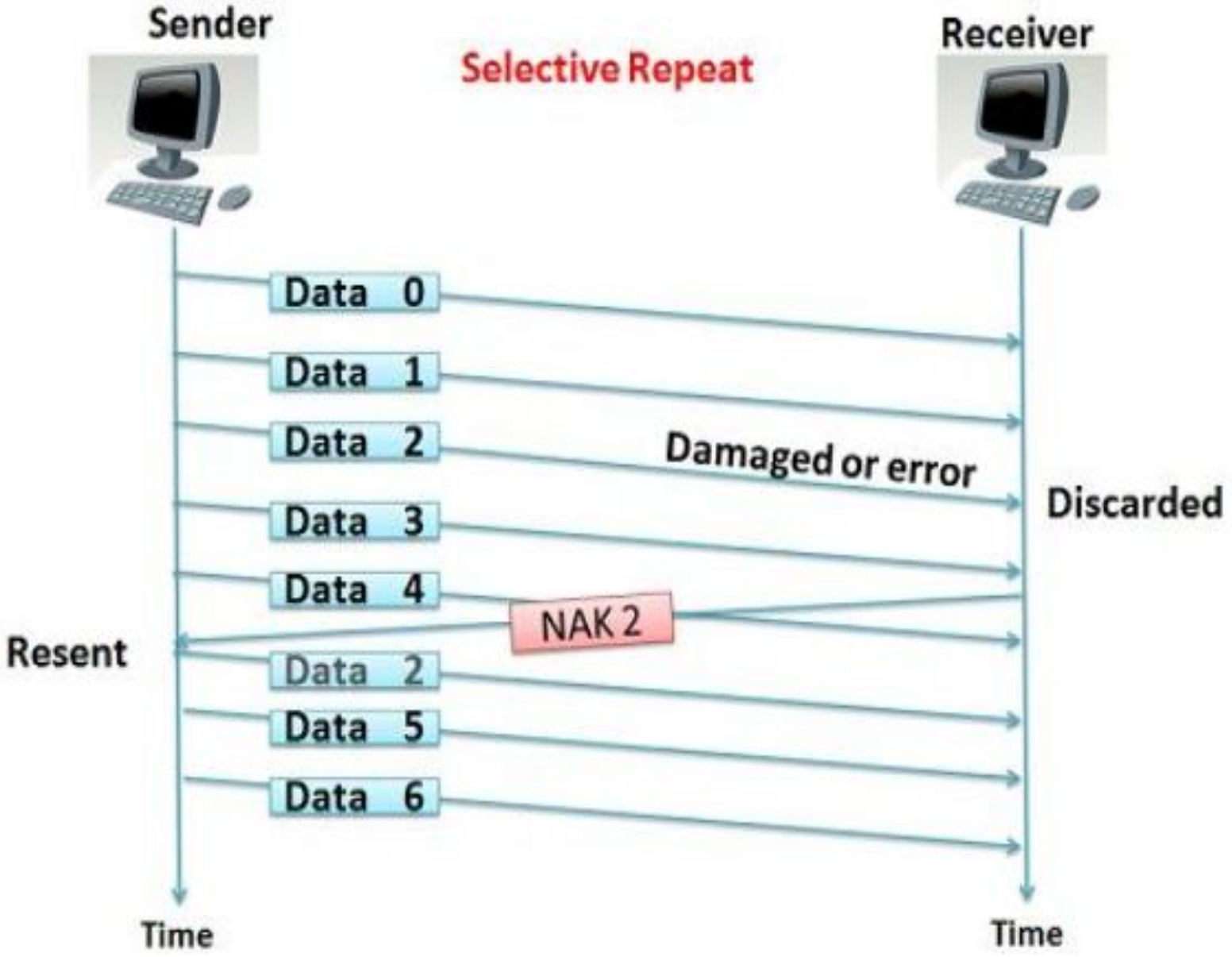
- Suppose the sender transmits frames 0, 1, 2, 3, and 4.
- Frame 1 is lost, but frames 2, 3, and 4 are received successfully.
- What will the receiver do now?

The receiver:

- Buffers frames 2, 3, and 4.
- Sends ACKs for frames 2, 3, and 4.
- The sender retransmits frame 1 upon timeout.
- After receiving frame 1, the receiver delivers frames 1, 2, 3, and 4 to the application layer in order.

The receiver will always send ACKs for any correctly received packets, even if they arrive after a lost frame. This helps the sender avoid retransmitting unnecessary frames and improves the protocol's efficiency.


Sender
Go-Back-N
Receiver
Data 0
Data 1
Data 2
Damaged or Lost
Discarded
Data 3
NAK 2
Discarded
Data 4
Discarded
Resent
Data 2
Resent
Data 3
Resent
Data 4
Time
Time


Sender
Receiver
Selective Repeat
Data 0
Data 1
Data 2
Damaged or error
Discarded
Data 3
Data 4
NAK 2
Resent
Data 2
Data 5
Data 6
Time
Time

Stop and Wait	GoBackN	Selective Repeat
In the Stop and Wait protocol, the sender sends one frame and then waits for the receiver to acknowledge it.	Any frames that were sent after the frame that is believed to have been damaged or lost are resent.	Only the suspected damaged or lost frames are retransmitted.
In the Stop and Wait protocol, the Sender window size is 1.	In the Go-Back-N protocol, the Sender window size is N.	The Selective Repeat approach has a Sender window size of N.
In the Stop and Wait protocol, the receiver window size is 1.	In the Go-Back-N protocol, the receiver window size is 1.	The Selective Repeat approach has a receiver window size of N.
No particular sequence needs to be followed at the receiver end of the Stop and Wait protocol.	The Go-Back-N protocol's receiver end will only accept the correct sequence.	Out-of-order deliveries can also be accepted at the receiving end of the Selective Repeat technique.
In the Stop and Wait protocol, the transmission type is Half duplex.	In the Go-Back-N protocol, the transmission type is full duplex.	In the Selective Repeat protocol, the transmission type is full duplex.
Efficiency formula for the Stop and Wait protocol is 1/(1+2*a), where a is the ratio of propagation delay to transmission delay.	Efficiency formula for the Go-Back-N protocol is N/(1+2*a), where a is the proportion of propagation delay to transmission delay and N is the number of packets transmitted.	Efficiency formula for the Selective Repeat protocol is N/(1+2*a), where a is the ratio of propagation delay to transmission delay and N is the number of packets transmitted.

IMPORTANT FORMULAS

- To achieve 100% efficiency, the optimal Sender window size should be **1+2a**

- In Go-back-N

Calculating Efficiency:

Efficiency (η) = N / (1+2a)

Note

- Max no of frames that can be sent in a window is : **1+2a**
- Min no of seq numbers required is : **1+2a**
- To have 1+2a seq numbers, the min no of bits required in seq no field is : **$[\log_2(1+2a)]$**

NOTE-

- **When minimum number of bits is asked, we take the ceil.**
- **When maximum number of bits is asked, we take the floor.**

- Number of bits available in the sequence number field also bounds the sender window size.
- If sequence number field contains n bits, then 2^n sequence numbers are possible.
- Thus, maximum number of frames that can be sent in one window = 2^n.

For n bits in sequence number field, Sender Window Size = min (1+2a , 2^n)

PROBLEM #1

- A 3000 km long trunk operates at 1.536 Mbps and is used to transmit 64 byte frames and uses sliding window protocol. If the propagation speed is 6 μsec / km, how many bits should the sequence number field be?

Solution

Given-

- Distance = 3000 km
- Bandwidth = 1.536 Mbps
- Packet size = 64 bytes
- Propagation speed = 6 μsec / km

Transmission delay (Tt)

- = Packet size / Bandwidth
- = 64 bytes / 1.536 Mbps
- = (64 x 8 bits) / (1.536 x 10^6 bits per sec) = 333.33 µsec

For 1 km, propagation delay = 6 µsec

- For 3000 km, propagation delay = 3000 x 6 µsec = 18000 µsec

a = Tp / Tt

- a = 18000 µsec / 333.33 µsec
- a = 54

Bits required in sequence number field

- $= \lceil \log_2(1+2a) \rceil$
- $= \lceil \log_2(1 + 2 \times 54) \rceil$

- $= \lceil \log_2(109) \rceil = \lceil 6.76 \rceil = 7$ bits

- **Minimum number of bits required in sequence number field = 7**
- With 7 bits, number of sequence numbers possible = 128
- We use only (1+2a) = 109 sequence numbers and rest remains unused.

PROBLEM #2

- Station A uses 32 byte packets to transmit messages to station B using a sliding window protocol. The round trip delay between A and B is 80 msec and the bottleneck bandwidth on the path between A and B is 128 Kbps. What is the optimal window size that A should use?

Solution

- Given-

- Packet size = 32 bytes
- Round Trip Time = 80 msec
- Bandwidth = 128 Kbps

Transmission delay (Tt)

- = Packet size / Bandwidth
- = 32 bytes / 128 Kbps
- = (32 x 8 bits) / (128 x 10^3 bits per sec)
- = 2 msec

Propagation delay (Tp)

- = Round Trip Time / 2
- = 80 msec / 2
- = 40 msec

- a = Tp / Tt
- a = 40 msec / 2 msec
- a = 20

Calculating Optimal Window Size-

- Optimal window size = 1 + 2a
- = 1 + 2 x 20
- = 41

PROBLEM #3

- If transmission delay and propagation delay in a sliding window protocol are 1 msec and 49.5 msec respectively, then:

1- What should be the sender window size to get the maximum efficiency?

2- What is the minimum number of bits required in the sequence number field?

3- If only 6 bits are reserved for sequence numbers, then what will be the efficiency?

SOLUTION

Given

- Transmission delay = 1 msec.
- Propagation delay = 49.5 msec.
- Calculating sender window size to get the maximum efficiency:
- To get the maximum efficiency, sender window size
- = 1 + 2a
- = 1 + 2 x (Tp / Tt)
- = 1 + 2 x (49.5 msec / 1 msec)
- = 1 + 2 x 49.5
- = 100

a) For maximum efficiency, **sender window size = 100.**

b) Calculating the minimum number of bits required in the sequence number field:

- Minimum number of bits required in the sequence number field
- = ⌈log2(1+2a) ⌉
- = ⌈log2(100) ⌉
- = ⌈6.8⌉
- = 7
- Minimum number of bits required in the sequence number field = 7.

c) Calculating the efficiency according to sequence numbers:

- If only 6 bits are reserved in the sequence number field, then Maximum sequence numbers possible = 2^6 = 64
- *Efficiency = Sender window size in the protocol / Optimal sender window size*
- = 64 / 100 = 0.64 = **64%**

PROBLEM #4

- A 20 Kbps satellite link has a propagation delay of 400 ms, the transmitter employs the "Go back N" ARQ" scheme with N set to 10.Assuming that each frame is 100 bytes long, what is the maximum data rate possible?

SOLUTION

Given:

- Bandwidth = 20 Kbps
- Propagation delay (Tp) = 400 msec.
- Frame size = 100 bytes
- Go back N is used where N = 10.

Transmission delay (Tt) = Frame size / Bandwidth

- = 100 bytes / 20 Kbps
- = (100 x 8 bits) / (20 x 10^3 bits per sec)
- = 0.04 sec
- = 40 msec.

Calculating Value of 'a':

- a = Tp / Tt
- a = 400 msec. / 40 msec.
- **a = 10**

Calculating Efficiency:

- Efficiency (η) = N / (1+2a)
- = 10 / (1 + 2 x 10)
- = 10 / 21
- = 0.476
- = 47.6 %.

Calculating Maximum Data Rate Possible:

- Maximum data rate possible or Throughput = Efficiency x Bandwidth
- = 0.476 x 20 Kbps
- = 9.52 Kbps $\cong$ **10 Kbps**

PROBLEM #5

- A 1 Mbps satellite link connects two ground stations. The altitude of the satellite is 36504 km and speed of the signal is 3 x 10^8 m/sec. What should be the packet size for a channel utilization of 25% for a satellite link using go back 127 sliding window protocol?

SOLUTION

Given

- Bandwidth = 1 Mbps
- Distance = 2 x 36504 km = 73008 km
- Propagation speed = 3 x 10^8 m/sec
- Efficiency = 25% = 1/4
- Go back N is used where N = 127
- Let the packet size be L bits.

Calculating Transmission Delay (Tt) = Packet size / Bandwidth

- = L bits / 1 Mbps
- = L μsec.

Calculating Propagation Delay (Tp) = Distance / Speed

- = (73008 x 10^3 m) / (3 x 10^8 m/sec)
- = 24336 x 10^-5 sec = **243360 μsec.**

Calculating Value of 'a':

- a = Tp / Tt
- a = 243360 μsec / L μsec
- a = 243360 / L

Calculating Packet Size:

- **Efficiency (η) = N / (1+2a)**
- Substituting the values, we get
- 1/4 = 127 / (1 + 2 x 243360 / L)
- 1/4 = 127 x L / (L + 486720)
- L + 486720 = 508 x L

$508 \times L - L = (508 - 1) \times L = 507 \times L$

First, let's get all the terms involving L on one side by subtracting L from both sides:

$$L - L + 486720 = 508 \times L - L$$

- 507 x L = 486720 ……So, L = **960 bits**.
- From here, packet size = **960 bits or 120 bytes.**

Go-Back-N & SR Animation

- https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/

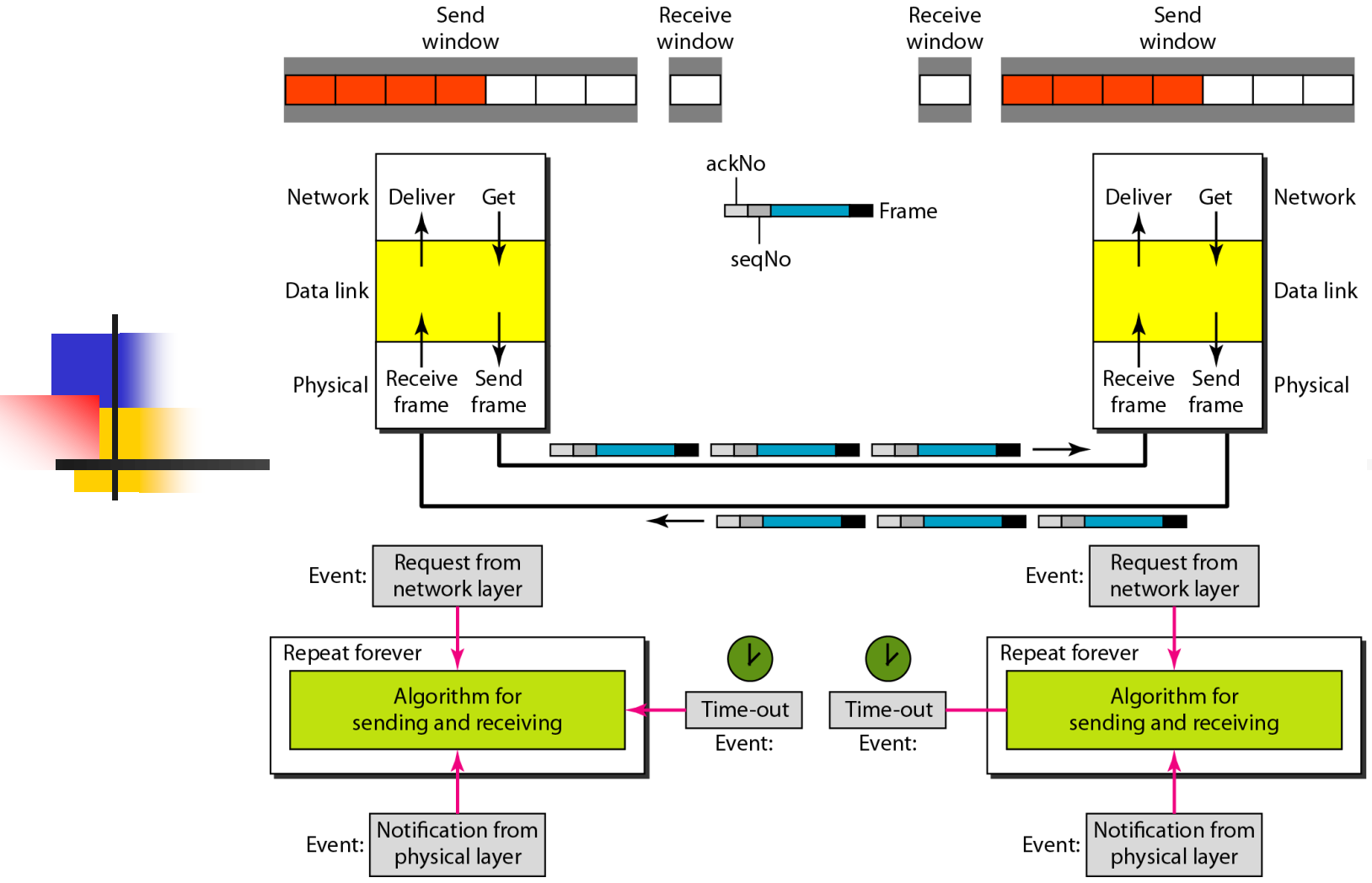
Piggybacking

- Piggybacking is a process of attaching the acknowledgment with the data packet to be sent.
- Piggybacking reduces the bandwidth utilization of the network.
- Piggybacking acknowledgment also contains the data while pure acknowledgment only contains the acknowledgment.
- For example, when a device sends data to another, it can piggyback its acknowledgment message onto the same data packet, thereby reducing the number of packets exchanged over the network.

Piggybacking

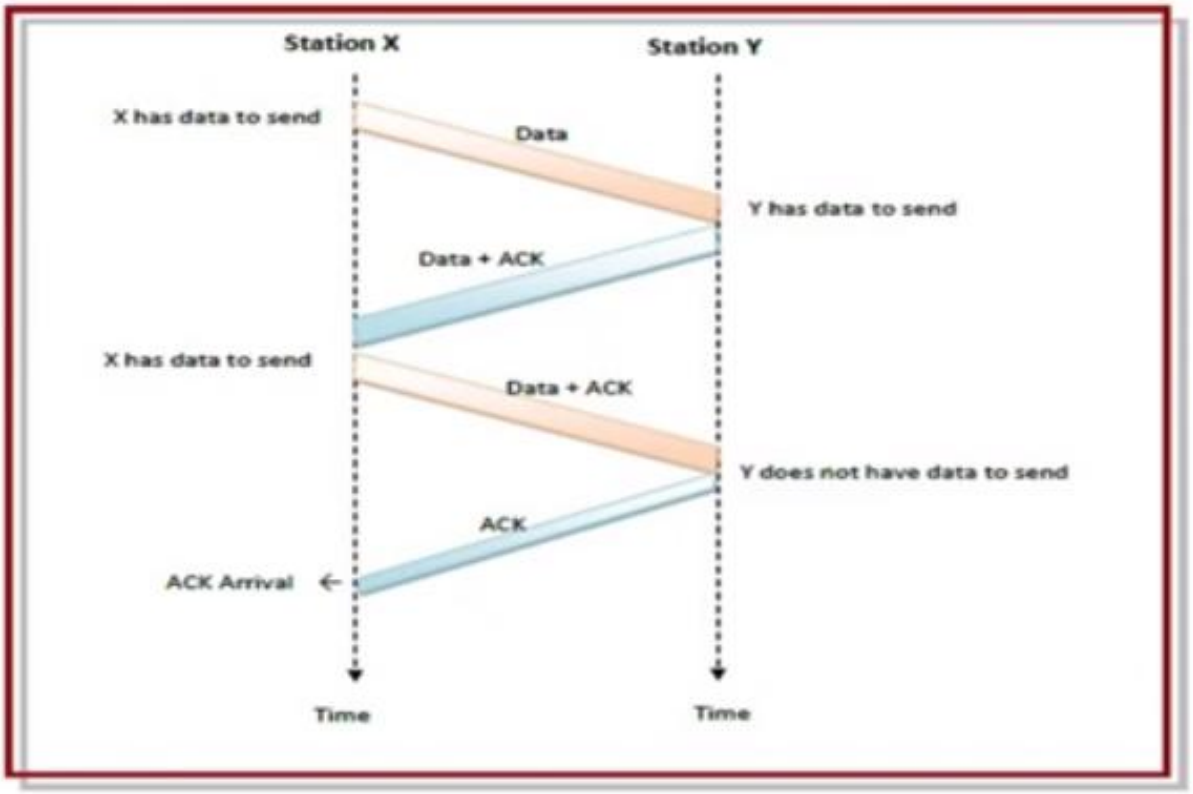
- In Stop and Wait ARQ, Go-Back N ARQ and Selective Repeat ARQ, data frames flow in only one direction.
- Control information such as ACK and NAK frames can travel in the other direction.
- Data frames are normally flowing in both directions from node A to node B and from node B to node A.
- This means that the control information also needs to flow in both directions.

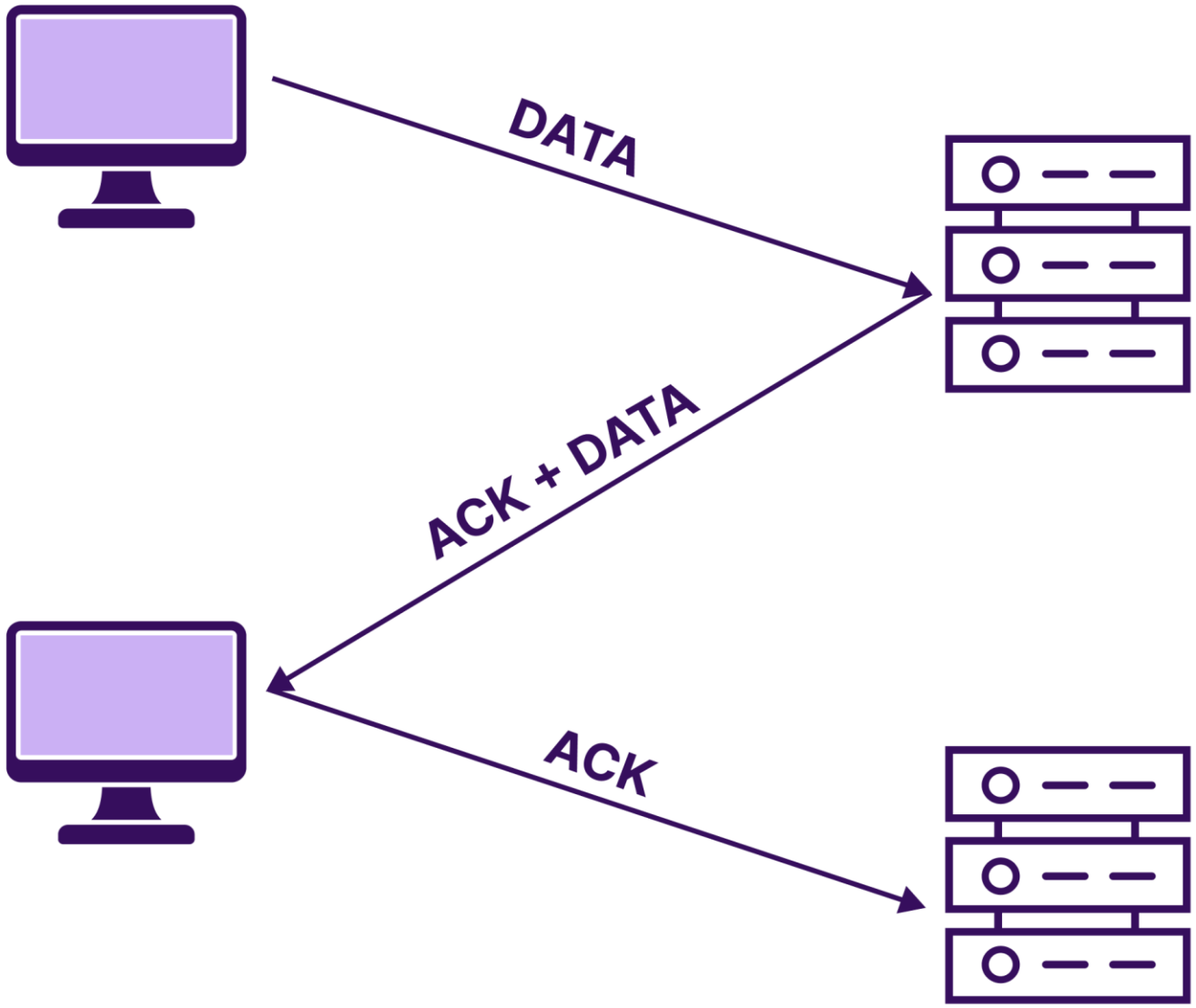
- A technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols.
- When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.



Working Principle

- If station X has both data and acknowledgment to send, it sends a data frame with the *ack* field containing the sequence number of the frame to be acknowledged.
- If station X has only an acknowledgment to send, it waits for a finite period of time to see whether a data frame is available to be sent. If a data frame becomes available, then it piggybacks the acknowledgment with it. Otherwise, it sends an ACK frame.
- If station X has only a data frame to send, it adds the last acknowledgment with it. The station Y discards all duplicate acknowledgments. Alternatively, station X may send the data frame with the *ack* field containing a bit combination denoting no acknowledgment.




DATA
ACK + DATA
ACK

Advantages of Piggybacking

- Efficient use of available channel bandwidth.
- Reduces communication delays, benefiting real-time apps.
- Enhances flow control in sliding window protocols.

Disadvantages of Piggybacking

- There may be a possibility of delay in the transfer of the acknowledgment message ..So if the sender does not get the acknowledgment in a fixed time, then the sender needs to resend the data.
- This technique requires additional complexity for its implementation.

UNIT - II

- **Data Link Layer:** Fundamentals of Error Detection and Error Correction, Block coding, Hamming Distance, CRC;
- Flow Control and Error control protocols - Stop and Wait, Goback-N ARQ, Selective Repeat ARQ, Sliding Window, Piggybacking,
- Random Access,Multiple access protocols - Pure ALOHA, Slotted ALOHA, CSMA/CD, CDMA/CA

Random Access Protocols

- A transmitting node always transmits at the full rate of the channel R bps .
- When there is a collision, each node involved in the collision repeatedly retransmits its frame until its frame gets through without a collision.
- It waits a random delay before retransmitting the frame.
- random access protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - ALOHA, slotted ALOHA
  - CSMA, CSMA/CD

ALOHA

- The ALOHA protocol was one of the earliest networking protocols designed to allow multiple devices to share a common communication channel.
- Originally developed at the University of Hawaii in the 1970s.
- First used for satellite and radio communications in Hawaii.
- It introduced the concept of random access, where devices could transmit data packets whenever they wanted, and retransmit in case of collisions.
- This laid the foundation for many modern wireless communication systems like Ethernet and Wi-Fi.

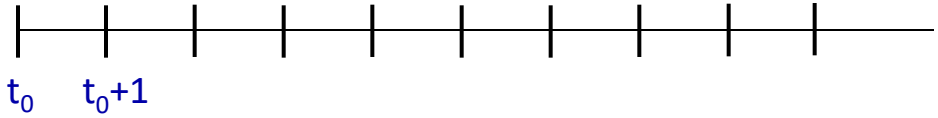
Types of ALOHA:

1. **Pure ALOHA**:
   - In Pure ALOHA, devices transmit data whenever they have data to send without checking if the channel is busy. This leads to collisions when multiple devices send data simultaneously.
   - After sending, the sender waits for an acknowledgment. If none is received (due to a collision), the sender waits for a random amount of time before retransmitting.
   - **Efficiency:** The maximum throughput is around **18.4%**, meaning only about 18% of the available bandwidth is effectively used due to collisions.
2. **Slotted ALOHA**:
   - Slotted ALOHA introduces time slots, where devices can only begin transmitting at the start of a slot. This reduces the chance of collisions since transmissions are more organized.
   - If a collision occurs, retransmission still follows a random delay, but because of time-slotting, the efficiency is improved compared to Pure ALOHA.
   - **Efficiency:** The maximum throughput of Slotted ALOHA is around **36.8%.**

UNSLOTTED ALOHA/PURE ALOHA

- Unslotted, fully decentralized
- In pure ALOHA, when a frame first arrives the node immediately transmits the frame in its entirety into the broadcast channel.
- If a transmitted frame experiences a collision with one or more other transmissions, the node will then immediately retransmit the frame with a prob **p**.
- Otherwise the node waits for a frame transmission time.
- After this wait, it then transmits the frame with prob p or waits for another frame time with prob **1-p.**

Slotted ALOHA



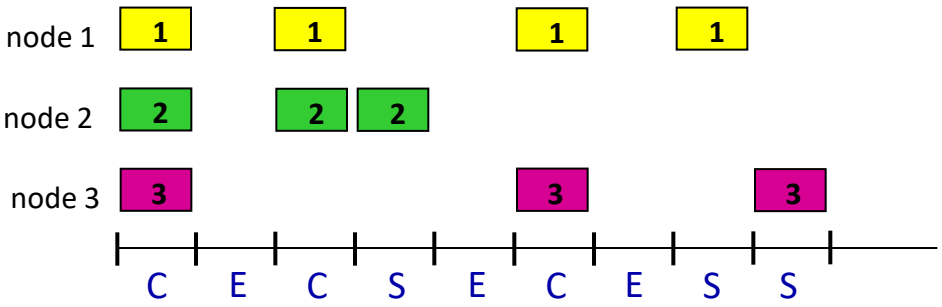
Assumptions:

- all frames consist of exactly L bits
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit frame only at the beginning of a slot
- nodes are synchronized
- if 2 or more frames collide in a slot, all nodes detect collision before the slot ends.

Operation:

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with probability *p* until success

Slotted ALOHA



C: collision
S: success
E: empty

Pros:

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Slotted ALOHA: efficiency

efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

- *suppose:* N nodes with many frames to send, each transmits in slot with probability p
- *Prob that a given node transmits is* $\boldsymbol{p}$
- *Prob that remaining nodes do not transmit* $\boldsymbol{(1-p)^{N-1}}$
  - prob that given node has success in a slot = $\boldsymbol{p(1-p)^{N-1}}$
  - prob that *any* node has a success = $\boldsymbol{Np(1-p)^{N-1}}$
  - *max efficiency* $= 1/e = .37$
- *at best:* channel used for useful transmissions **37%** of time!

Note

- Allows a node to transmit at the full rate R bps when that node is the only active node.
- Highly decentralized – each node detects collisions but decides independently when to retransmit
- Extremely simple
- *A slot in which exactly one node transmits is said to be successful slot.*

Observation

- In both slotted and pure ALOHA, a node's decision to transmit is made independent of the activity of the other nodes attached to the broadcast channel.
- A node neither pays attention to whether another node happens to be transmitting when it begins to transmit nor stops transmitting if another node begins to interfere with its transmission.

SOLVE??

- There are 5 stations in slotted LAN. Each station attempts to transmit with a probability P=0.2 in each time slot. What is the probability that ONLY one station transmits in a given time slot?

For $N = 5$ stations and $P = 0.2$:

$$P_{\text{only-one}} = 5 \cdot 0.2 \cdot (1 - 0.2)^{5-1} = 5 \cdot 0.2 \cdot (0.8)^4 = 5 \cdot 0.2 \cdot 0.4096 = 0.4096$$

Applications of ALOHA:

- **Ethernet**: The CSMA/CD (Carrier Sense Multiple Access/Collision Detection) protocol used in Ethernet networks builds on ALOHA by adding mechanisms to detect and avoid collisions.
- **Wi-Fi**: The CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) used in Wi-Fi networks also inherits concepts from ALOHA, particularly in how devices handle contention for a shared medium.

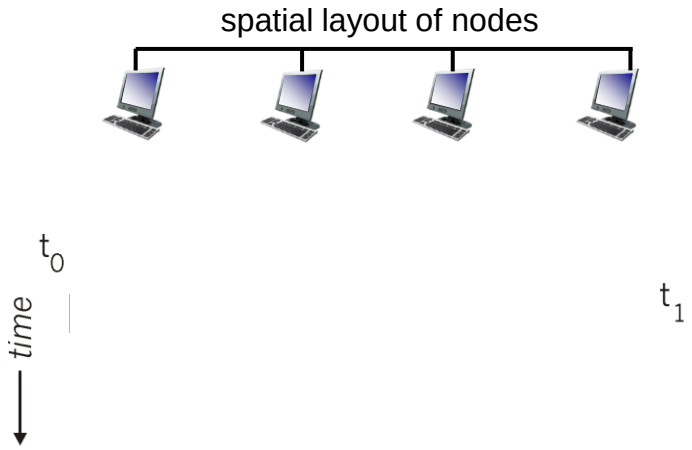
CSMA (carrier sense multiple access)

- Two important rules for polite human conversation

1. Listen before speaking – Carrier Sensing(a node listens to the channel before transmitting)
2. If someone else begins talking at the same time, stop talking – Collision detection(If a node detects that another node is transmitting an interfering frame, it stops transmitting and wait a random amount of time before repeating the sense-and-transmit-when-idle-cycle.

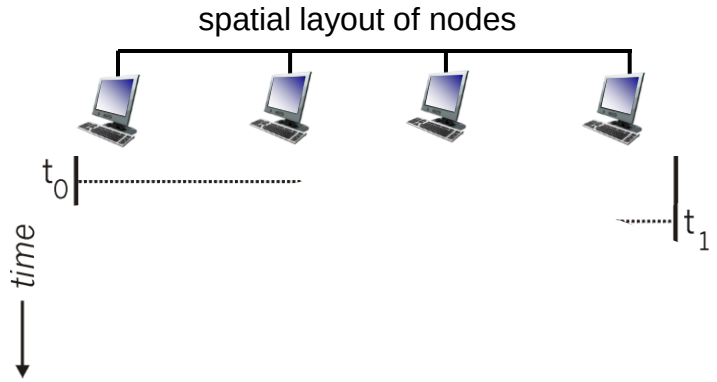
CSMA: collisions

- collisions can *still* occur with carrier sensing:
  - propagation delay means two nodes may not hear each other's just-started transmission
- collision: entire packet transmission time wasted
  - distance & propagation delay play role in determining collision probability



CSMA/CD:

- CSMA/CD reduces the amount of time wasted in collisions
  - transmission aborted on collision detection



Ethernet CSMA/CD algorithm

1. Ethernet receives datagram from network layer, creates frame
2. If **Ethernet** senses channel:
   if idle: start frame transmission.
   if busy: wait until channel idle, then transmit
3. If entire frame transmitted without collision - done!
4. If another transmission detected while sending: abort, send jam signal
5. After aborting, enter *binary (exponential) backoff:*
   - After nth collision, a node chooses *K* at random from $\{0, 1, 2, \ldots, 2^n-1\}$.
   - **For Ethernet, the actual amount of time a node waits is *K**512** bit times(K times the amount of time needed to transmit 512 bits into Ethernet), returns to Step 2
   - Max value that **n** can take is capped at **10.**
   - The more the collisions, the longer the backoff interval

Binary (Exponential) Backoff

- **Binary (Exponential) Backoff** is an algorithm used primarily in network communication protocols like Ethernet (CSMA/CD) and Wi-Fi (CSMA/CA) to handle collisions when multiple devices try to send data simultaneously.
- The backoff mechanism plays a critical role in regulating access to a shared communication medium, ensuring that devices do not continuously collide and that the network remains efficient.

How it Works

- It is used to determine the wait time before retransmitting a packet after a collision occurs.

1. **Initial Transmission**: When a device wants to transmit data, it sends the packet.
2. **Collision Detection**: If two devices transmit simultaneously, a collision occurs, and the devices detect the collision.
3. **Backoff Procedure**:
   - ✓ Each device involved in the collision waits for a random period before attempting to retransmit.
   - ✓ The wait time is calculated based on the number of collisions encountered.
   - ✓ The backoff time increases exponentially with each subsequent collision.
4. **Exponential Growth**: The backoff time is chosen from a random range that doubles with each collision. Specifically, if the number of collisions is k, the backoff time is chosen from the interval [0, 2^k−1] slots.

Example Scenario

Step 1: Initial Attempt

Both **Device A** and **Device B** want to send data at the same time.

They both transmit simultaneously.

Step 2: Collision Detected

A collision occurs because both devices are trying to transmit at the same time.

Step 3: First Backoff

Each device now needs to wait a random amount of time before trying to send again.

Since this is the first collision (k=1), each device chooses a random wait time from the range

[0, 2^1 - 1] = [0, 1]

- **Device A** randomly chooses **0** (waits 0 time slots).
- **Device B** randomly chooses **1** (waits 1 time slot).

Step 4: Second Attempt

Device A sends its data immediately (after waiting 0 slots).

Device B waits for 1 time slot, then attempts to send.

- If **Device A** successfully sends its packet without any further collisions, **Device B** will not encounter a collision. However, if they collide again, they will follow the next steps.

Contd..

Step 5: Second Collision (if it occurs)

Suppose they collide again. Now both devices need to back off again.

Since this is the second collision (k=2), they choose a random wait time from [0, 2^2 - 1] = [0, 3].

- **Device A** randomly chooses **2** (waits 2 time slots).
- **Device B** randomly chooses **1** (waits 1 time slot).

Step 6: Third Attempt

Device A waits for 2 slots and then tries to send.

Device B waits for 1 slot and then tries to send.

Depending on the timing, they may or may not collide again.

This process continues, with the range of backoff times increasing exponentially after each collision, helping to minimize the likelihood of repeated collisions.

Problem

- In a CSMA/CD network running at 1 Gbps over 2 km cable with no repeaters, the signal speed in the cable is 400000 km/sec. What is minimum frame size?

Solution

Step 1: Calculate the propagation delay

Propagation delay (Tp) Tp=Cable Length/Propagation Speed=2000 m / 4×10^8 m/s=5 μs

This is the one-way propagation delay.

For CSMA/CD, we need the round-trip time (RTT)

RTT=2×Tp=2×5 μs=10 μs

Step 2: Calculate the minimum frame size

To detect a collision, the transmission time of the frame must be greater than or equal to the round-trip time.

The transmission time Tt**=Frame Size (in bits) / Data Rate (bps)**

To ensure collision detection: $Tt \geq RTT$

Frame Size/10^9 ≥ 10×10^{-6}

Frame Size ≥ $10^9\times10\times10^{-6}$=10,000 bits

Step 3: Convert bits to bytes

10,000 bits=10,000/8=**1250 bytes**

How Binary (Exponential) Backoff Works:

1. **Initial Attempt**: When a device wants to send data, it listens to the communication medium to check if it is idle.
    1. In **CSMA/CD** (Ethernet), the device will detect a collision during transmission.
    2. In **CSMA/CA** (Wi-Fi), the device uses a contention mechanism to avoid collisions.
2. **Collision Occurs**: If two or more devices transmit simultaneously, a collision occurs. The devices stop transmitting and enter a backoff period to avoid another immediate collision.
3. **Random Backoff Time**: Each device chooses a random backoff time within a range (based on a contention window), before retrying to send the data.
4. **Exponential Increase**: If another collision occurs, the **backoff time increases exponentially**. This is where the "binary exponential backoff" comes in.
    1. For the **n-th retransmission attempt**, the backoff time is chosen randomly from the range [0, 2^n - 1].
    2. For example:
        1. On the first collision (n = 1), the backoff time is randomly chosen from [0, 1].
        2. On the second collision (n = 2), the backoff time is randomly chosen from [0, 3].
        3. On the third collision (n = 3), the backoff time is randomly chosen from [0, 7].
    3. The process repeats with the backoff range doubling each time until it reaches a predefined maximum (usually to avoid excessively long delays).
5. **Collision Avoidance**: By using exponentially larger backoff times, the algorithm reduces the probability of repeated collisions, giving each device a better chance of accessing the medium.
6. **Success**: Once the device successfully sends its data, the backoff counter resets, and the process starts anew if more data needs to be transmitted.

Example in CSMA/CD (Ethernet):

- If a collision occurs, each device waits for a random period, increasing the wait time exponentially with each failed attempt, until a maximum number of retries is reached.
- After the maximum retries, the transmission is aborted.

CSMA/CA

- CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) is a network protocol that is primarily used in wireless communication (such as Wi-Fi).
- It is designed to avoid collisions before they occur, unlike CSMA/CD (used in Ethernet), which deals with collisions after they happen.

Key Concepts of CSMA/CA:

1. **Carrier Sense**: Before transmitting, a device listens to the communication channel to check if it's free (no ongoing transmission). This ensures that the device does not interrupt other ongoing transmissions.
2. **Collision Avoidance**: If the channel is found to be busy, the device does not immediately retry to transmit. Instead, it waits for a random backoff time to avoid collisions that might occur if multiple devices attempt to transmit simultaneously when the channel becomes free.
3. **Acknowledgment**: After a successful transmission, the receiving device sends an acknowledgment (ACK) to the sender. If the sender does not receive the ACK within a certain period, it assumes a collision or error occurred, and it will try retransmitting the data.

CSMA/CA algorithm:

Step 1: **Carrier Sensing (Channel Listening)**
The device first senses the communication channel to check if it is idle or busy:

1. If **idle**, the device proceeds to transmit data.
2. If **busy**, the device waits for the channel to become idle before proceeding.

Step 2: **Wait for Interframe Space (IFS)**
After the channel is detected to be idle, the device does not immediately transmit. It waits for a specified period called the **Interframe Space (IFS)**. This waiting period is to ensure the channel remains idle and to give priority to high-priority frames (if any).

Step 3: **Random Backoff Time**
After the IFS, the device waits for a random amount of time known as the **Backoff Time**. This random waiting helps avoid multiple devices transmitting simultaneously when the channel becomes free (collision avoidance).

1. The backoff time is calculated based on the **Contention Window (CW)**. The CW is a range of time slots, and the device randomly selects a slot within this range. The value of the CW increases exponentially with each failed transmission attempt (this is called **Binary Exponential Backoff**).

Step 4: **Transmit Data**
After the backoff time has passed and if the channel is still idle, the device proceeds to transmit its data.

Step 5: **Acknowledgment**
After the data is successfully transmitted, the receiver sends an ACK back to the sender to confirm successful reception. If the sender does not receive an ACK within a certain time frame, it assumes that the transmission failed (possibly due to a hidden node problem or other interference) and repeats the process.

Key Aspects of CSMA/CA:

- **Hidden Node Problem**: In wireless networks, some devices may be out of range of each other, so they cannot detect each other's transmissions. CSMA/CA helps mitigate this issue through the use of control packets (like RTS/CTS).
- **RTS/CTS Mechanism** (Optional):
  - **Request to Send (RTS)**: The sender sends an RTS message to the intended receiver, asking for permission to transmit.
  - **Clear to Send (CTS)**: If the receiver is ready, it responds with a CTS message, granting permission to transmit.

Summary of MAC protocols

- **Random Access (dynamic),**
  - **ALOHA, S-ALOHA, CSMA, CSMA/CD**
  - **carrier sensing: easy in some technologies (wire), hard in others (wireless)**
  - **CSMA/CD used in Ethernet**
  - **CSMA/CA used in 802.11**